第一部分　软件需求：是什么和为什么

第1章　基本的软件需求

“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，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。“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。我不记得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。

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问题，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？“

“这并不是我的错！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）。“我还有别的事。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。因此你不能因我未猜出你的想法（需求）就责备我。”

Maria不得不愤怒地屈从：“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？”

如果作为客户有过类似的经验，你一定知道：一个不能进行一项基本操作的软件产品是多么令人烦恼。尽管开发者最终会满足你的要求，你也不会感谢他。但从开发者角度来看，在整个系统已经完成后，用户再提出对功能的进一步要求是多么烦人的事。同时，修改系统的请求迫使你放下当前的项目，而且往往修改请求还要求你优先处理，也是令人很不愉快的。

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，

他们都明白完工以后的修改会造成损失，以及变更细节的危害性。然而，涉及到软件开发，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）——开发者开发的与用户所想得到的软件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本书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。

1.1 软件需求的定义

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，不同层次的需求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。

1.1.1 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。

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所有的开发者，客户被迫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，现在我要的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。所以如果只有一堆邮件、贴条、会谈过几次或一些零碎的对话，你就确信你已明白用户的需求，那完全是自欺欺人。

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、功能及属性等 ”。这些定义强调的是产品是什么样的，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 and Sawyer 1997）：

> 需求是……指明必须实现什么的规格说明。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。

1.1.2 需求的层次

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。

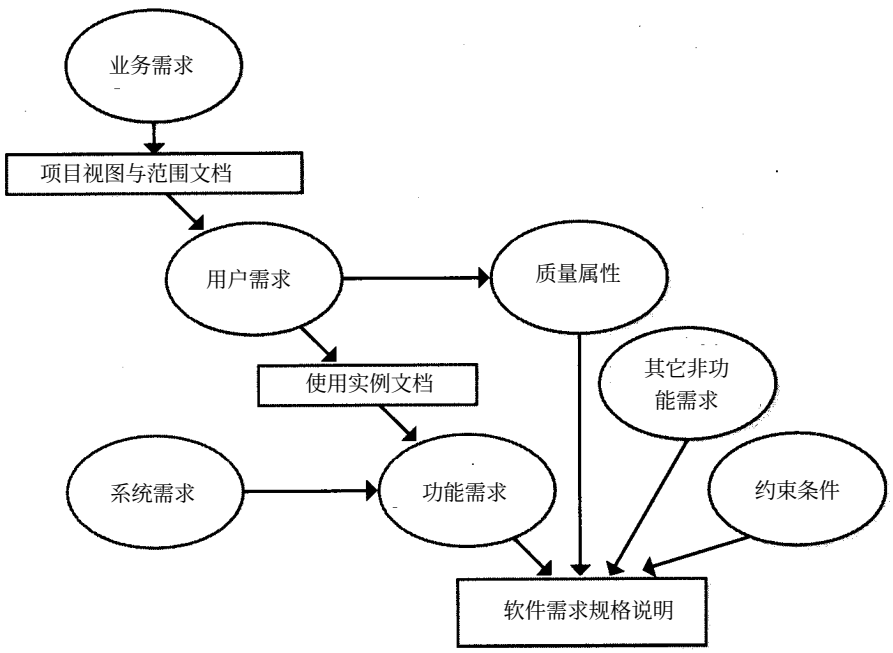

图1-1 软件需求各组成部分之间的关系

在软件需求规格说明（software requirements specification，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的限制。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是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，这使公司运作更加高效（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。用户需求使需求分析者能从中总结出功能需求以满足用户对产品的要求从而完成其任务，而开发人员则根据功能需求来设计软件以实现必须的功能。

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。

1.2 每个项目都有需求

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，将最终会给系统带来极大损害的部分，并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是重要的，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、组件和工具这些供开发小组内部使用的软件。当然你可能偶尔勿需文档说明就能与其他人意见较为一致，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。这说明那怕需求明确无误并构思准确，如果我们没有编写文档，软件达不到期望目标也只能是咎由自取了。

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简单的一张需求清单竟是如此有用。我依据需求对系统进行测试时，此系统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。

1.3 什么情况将会导致好的群体发生不合格的需求说明

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、及时限在功能、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防止与需求有关的风险的出现。

不适当的需求过程所引起的一些风险

- 用户不多导致产品无法被接受。
- 用户需求的增加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的浪费和返工。
- 用户增加一些不必要的特性和开发人员画蛇添足 (gold-plating)。
- 过分简略的需求说明以致遗漏某些关键需求。
- 忽略某类用户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无法准确进行。

1. 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的参与。究其原因：一是因为与用户合作不如编写代码有意思；二是因为开发人员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，而客户也不太明白自己的真正需求。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。

2. 用户需求的不断增加

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及预算范围。计划并不总是与项目需求规模与复杂性、风险、开发生产率及需求变更实际情况相一致，这使得问题更难解决。实际上，问题根源在于用户需求的改变和开发者对新需求所作的修改。

要想把需求变更范围控制到最小，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，有助于所有风险承担者明白业务决策的合理性，即为何进行某些变更，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不完善的话，收回变更和删除特性会带来问题。如果你尽早地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，同时也有利于减少因变更导致质量的下降。

3. 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并重做许多测试。

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 40%，而70%～85%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，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，但每个评审人员都真正了解需求文档，这样二义性就不会直到项目后期才被发现，那时再发现的话会使得更正代价很大。其它检测模棱两可需求的技术由 Gause 和Weinberg（1989）给予介绍，本章的后面也有所涉及。

4. 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。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并为他们提供一些具有创新意识的思路，具体提供哪些功能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，开发人员应努力使功能简单易用，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，而实现这些功能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。

5. 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，于是只作一份简略之至的规格说明，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（McConnell 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）。

6. 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。

7. 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）。

对不准确的要求所提问题的正确响应是“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对需求不成熟的估计很容易为一些因素左右。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）。未经准备的估计通常是作为一种猜测给出的，听者却认为是一种承诺。因此我们要尽力给出可达到的目标并坚持完成它。

1.4 高质量的需求过程带来的好处

实行有效的需求工程管理的组织能获得多方面的好处。最大的好处是在开发后期和整个维护阶段的重做的工作大大减少了。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要多付出68倍的成本。近来很多研究表明这种错误导致成本放大因子可以高达200倍。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。传统的质量成本角度分析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风险承担者的积极努力。收集需求能使开发小组更好地了解市场，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。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负面影响。最后，将需求编写成清晰、无二义性的文档将会极大地有利于系统测试，确保产品质量，以使所有风险承担者感到满意。

1.5 优秀需求具有的特性

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，就会写出更好的（尽管并不十分完美）需求文档，同时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。

1.5.1 需求说明的特征

1. 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。

2. 正确性

每一项需求都必须准确地陈述其要开发的功能。做出正确判断的参考是需求的来源，如用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此类说法：“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。

3. 可行性

每一项需求都必须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，由他负责检查技术可行性。

4. 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。

5. 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。

6. 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。

7. 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。

1.5.2 需求规格说明的特点

1. 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。

2. 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，才能知道某一项需求是否确实正确。

3. 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。

4. 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。

1.6 需求的开发和管理

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 ”，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：

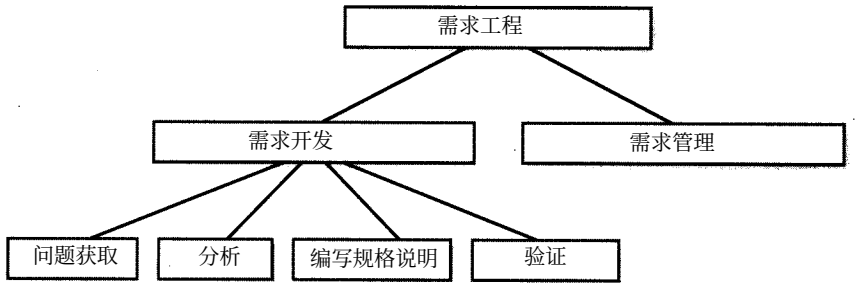

图1-2 需求工程域的层次分解示意图

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 and 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。
- 估计变更需求所产生影响并在此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：

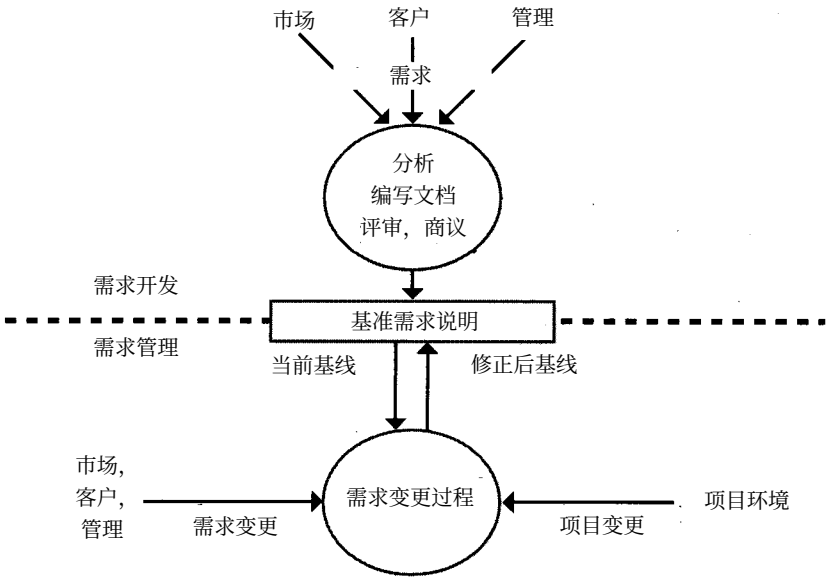

图1-3 需求开发与需求管理之间的界限

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。指明每一个问题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，有利于相互交流沟通与协作。