China-pub.com
下载

第8章　聆听客户的需求

需求获取(requirement elicitation)是需求工程的主体。对于所建议的软件产品，获取需求是一个确定和理解不同用户类的需要和限制的过程。获取用户需求位于软件需求三层结构的中间一层。来自项目视图和范围文档的业务需求决定用户需求，它描述了用户利用系统需要完成的任务。从这些任务中，分析者能获得用于描述系统活动的特定的软件功能需求，这些系统活动有助于用户执行他们的任务。本章将介绍需求获取原理，并把重点放在“使用实例”方法的应用，这种方法是用于获取产品的用户需求。

需求获取是在问题及其最终解决方案之间架设桥梁的第一步。获取需求的一个必不可少的结果是对项目中描述的客户需求的普遍理解。一旦理解了需求，分析者、开发者和客户就能探索出描述这些需求的多种解决方案。参与需求获取者只有在他们理解了问题之后才能开始设计系统，否则，对需求定义的任何改进，设计上都必须大量的返工。把需求获取集中在用户任务上——而不是集中在用户接口上——有助于防止开发组由于草率处理设计问题而造成的失误。

需求获取、分析、编写需求规格说明和验证并不遵循线性的顺序，这些活动是相互隔开、增量和反复的。当你和客户合作时，你就将会问一些问题，并且取得他们所提供的信息（需求获取）。同时，你将处理这些信息以理解它们，并把它们分成不同的类别，还要把客户需求同可能的软件需求相联系（分析）。然后，你可以使客户信息结构化，并编写成文档和示意图（说明）。下一步，就可以让客户代表评审文档并纠正存在的错误（验证）。这四个过程贯穿着需求开发的整个阶段。

由于软件开发项目和组织文化的不同，对于需求开发没有一个简单的、公式化的途径。下面列出了14个步骤，你可以利用它们指导你的需求开发活动。对于需求的任何子集，一旦你完成了第十三步，那么你就可以很有信心地继续进行系统的每一部分的设计、构造，因为你将开发出一个好的产品。

8.1　需求获取的指导方针

需求获取可能是软件开发中最困难、最关键、最易出错及最需要交流的方面。需求获取只有通过有效的客户——开发者的合作才能成功。分析者必须建立一个对问题进行彻底探讨的环境，而这些问题与产品有关。为了方便清晰地进行交流，就要列出重要的小组，而不是假想所有的参与者都持有相同的看法。对需求问题的全面考察需要一种技术，利用这种技术不但考虑了问题的功能需求方面，还可讨论项目的非功能需求。确定用户已经理解：对于某些功能的讨论并不意味着即将在产品中实现它。对于想到的需求必须集中处理并设定优先级，以避免一个不能带来任何益处的无限大的项目。

需求获取是一个需要高度合作的活动，而并不是客户所说的需求的简单誊本。作为一个分析者，你必须透过客户所提出的表面需求理解他们的真正需求。询问一个可扩充（open-ended）的问题有助于你更好地理解用户目前的业务过程并且知道新系统如何帮助或改进他们

China-pub.com
下载

的工作。调查用户任务可能遇到的变更，或者用户需要使用系统其它可能的方式。想像你自己在学习用户的工作，你需要完成什么任务？你有什么问题？从这一角度来指导需求的开发和利用。

还有，探讨例外的情况：什么会妨碍用户顺利完成任务？对系统错误情况的反映，用户是如何想的？询问问题时，以“还有什么能……”，”当……时，将会发生什么“”你有没有曾经想过……”,“有没有人曾经……”为开头。记下每一个需求的来源，这样向下跟踪直到发现特定的客户。

建议需求开发过程

1. 定义项目的视图和范围
2. 确定用户类
3. 在每个用户类中确定适当的代表
4. 确定需求决策者和他们的决策过程
5. 选择你所用的需求获取技术
6. 运用需求获取技术对作为系统一部分的使用实例进行开发并设置优先级
7. 从用户那里收集质量属性的信息和其它非功能需求
8. 详细拟订使用实例使其融合到必要的功能需求中
9. 评审使用实例的描述和功能需求
10. 如果有必要，就要开发分析模型用以澄清需求获取的参与者对需求的理解
11. 开发并评估用户界面原型以助想像还未理解的需求
12. 从使用实例中开发出概念测试用例
13. 用测试用例来论证使用实例、功能需求、分析模型和原型
14. 在继续进行设计和构造系统每一部分之前，重复 6~13步

尽量把客户所持的假设解释清楚，特别是那些发生冲突的部分。从字里行间去理解以明确客户没有表达清楚但又想加入的特性或特征。 Gause 和Weinberg（1989）提出使用“上下文无关问题”—— 这是一个高层次的问题，它可以获取业务问题和可能的解决方案的全部信息。客户对这些问题的回答诸如“产品要求怎样的精确度”或“你能帮我解释一下你为什么不同意某人的回答吗？”这些回答可以更直接地认识问题，而这是封闭（ close-end）问题所不能做到的。

需求获取利用了所有可用的信息来源，这些信息描述了问题域或在软件解决方案中合理的特性。第7章曾列出了软件需求的来源。一个研究表明：比起不成功的项目，一个成功的项目在开发者和客户之间采用了更多的交流方式（ Kiel and Carmel 1995）。与单个客户或潜在的用户组一起座谈，对于业务软件包或信息管理系统（ MIS）的应用来说是一种传统的需求来源。（如何与用户座谈方面的指导，见 Beyer and Holtzblatt [1998]，Wood and Silver[1995]，McGraw and Harbison[1997]）。直接聘请用户进行获取需求的过程是为项目获得支持和买入（buy-in）的一种方式。

在每一次座谈讨论之后，记下所讨论的条目 (item)，并请参与讨论的用户评论并更正。及早并经常进行座谈讨论是需求获取成功的一个关键途径，因为只有提供需求的人才能确定是否真正获取需求。进行深入收集和分析以消除任何冲突或不一致性。

尽量理解用户用于表述他们需求的思维过程。充分研究用户执行任务时作出决策的过程，并提取出潜在的逻辑关系。流程图和决策树是描述这些逻辑决策途径的好方法。

China-pub.com 下载

当进行需求获取时，应避免受不成熟的细节的影响。在对切合的客户任务取得共识之前，用户能很容易地在一个报表或对话框中列出每一项的精确设计。如果这些细节都作为需求记录下来，他们会给随后的设计过程带来不必要的限制。你可能要周期性地检查需求获取，以确保用户参与者将注意力集中在与今天所讨论的话题适合的抽象层上。向他们保证在开发过程中，将会详尽阐述他们的需求。

在一个逐次详细描述过程中，重复地详述需求，以确定用户目标和任务，并作为使用实例。然后，把任务描述成功能需求，这些功能需求可以使用户完成其任务，也可以把它们描述成非功能需求，这些非功能需求描述了系统的限制和用户对质量的期望。虽然最初的屏幕构思有助于描述你对需求的理解，但是你必须细化用户界面设计。

8.2 基于使用实例的方法

多年来，分析者总是利用情节或经历来描述用户和软件系统的交互方式，从而获取需求（McGraw and Harbison 1997）。最近，Ivar Jacobson （1992）和其他人把这种看法系统地阐述成用使用实例(use-case)的方法进行需求获取和建模。虽然使用实例来源于面向对象的开发环境，但是它也能应用在具有许多开发方法的项目中，因为用户并不关心你是怎样开发你的软件。一些实际方法包括使用实例模型概念（ Regnell, Kimbler and Wesslen 1995; Booth, Rumbaugh and Jacobson 1999）。然而，使用实例的观点和思维过程带给需求开发的改变比起是否画正式的使用实例图显得更为重要。注意用户要利用系统做什么远远强于询问用户希望系统为他们做什么这一传统方法。

一个使用实例描述了系统和一个外部“执行者”(actor)的交互顺序，这体现执行者完成一项任务并给某人带来益处。执行者是指一个人，或另一个软件应用，或一个硬件，或其它一些与系统交互以实现某些目标的实体（ Cockburn 1997a，b）。执行者可以映像到一个或多个可以操作的用户类的角色。例如，“化学制品跟踪系统”中的“请求一种化学制品”使用实例包括一个名为“请求者”的执行者。在“化学制品跟踪系统”中设有一个客户类叫请求者，但是药剂师和化学制品仓库的人员均可充当这一角色。

使用实例为表达用户需求提供了一种方法，而这一方法必须与系统的业务需求相一致。分析者和用户必须检查每一个使用实例，在把它们纳入需求之前决定其是否在项目所定义的范围内。基于“使用实例”方法进行需求获取的目的在于：描述用户需要使用系统完成的所有任务。在理论上，使用实例的结果集将包括所有合理的系统功能。在现实中，你不可能获得完全包容，但是比起我所知道的其它获取方法，基于使用实例的方法可以为你带来更好的效果。

8.2.1 使用实例和用法说明

一个单一的使用实例可能包括完成某项任务的许多逻辑相关任务和交互顺序。因此，一个使用实例是相关的用法说明的集合，并且一个说明（ scenario）是使用实例的例子。在使用实例中，一个说明被视为事件的普通过程(normal course)，也叫作主过程、基本过程，普通流，或“满意之路”(happy path)。在描述普通过程时列出执行者和系统之间相互交互或对话的顺序。当这种对话结束时，执行者也达到了预期的目的。“请求一种化学制品”的使用实例的普通过程导致一个用户请求，要求从外界供应商订购化学制品。

图8-1描述了“请求化学制品”的使用实例图，并运用了统一建模语言（ UML）概念

China-pub.com
下载

（Booth, Rumbaugh and Jacobson1999 ⊖）。执行者（请求者）用一个简单的分叉状图形表示，并在执行者和他所需要的每个使用实例（以椭圆形所示）之间画一条线。在这幅图中，主要的使用实例是“请求一种化学制品”。其它的使用实例：“查看仓库中可用的化学制品容器”和“输入货物编号”，描述了两种可能的使用实例中《extend》和《include》的关系，这将在这一章的后面部分介绍。使用实例图提供了一个高级别的用户需求的可视化表示。

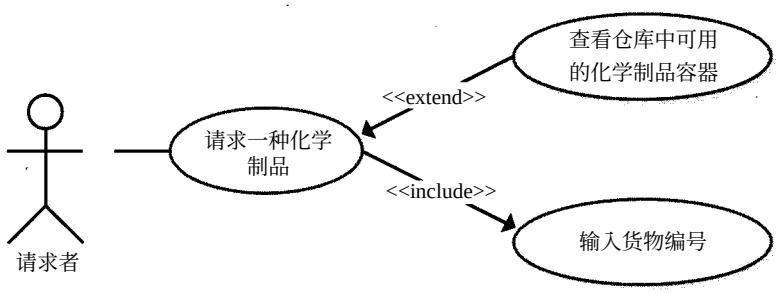

图8-1 来自“化学制品跟踪系统”的“请求一种化学制品”使用实例

在使用实例中的其它说明可以描述为可选过程 (alternative coruse)。可选过程也可促进成功地完成任务，但它们代表了任务的细节或用于完成任务的途径的变化部分。在对话序列中，普通过程可以在一些决策点上分解成可选过程，然后再重新汇成一个普通过程。在“请求一种化学制品”的使用实例中，“向化学制品仓库请求一种化学制品”是一个可选过程。虽然用户在普通过程和可选过程中请求一种化学制品。但是用户和系统之间的详细交互在许多方面存在差异。在可选过程中，用户仍必须确定合适的化学制品，但也许可以从化学制品仓库里现存的多种容器中选择。

可选过程中的一些步骤与在普通过程中的步骤相同，但是需要一些唯一的动作来完成可选路径（见图 8-2）。有时，通过在流中插入一个定义可选过程的分离的使用实例可以很方便地扩充(extend)普通过程。被扩充的使用实例必须是一个完整的使用实例，它可以独立地运行。图8-1表示：使用实例“查看仓库中可用的化学制品容器”扩充了“请求一种化学制品”使用实例。这一扩充形成了“从化学制品仓库中请求一种化学制品”的可选过程。

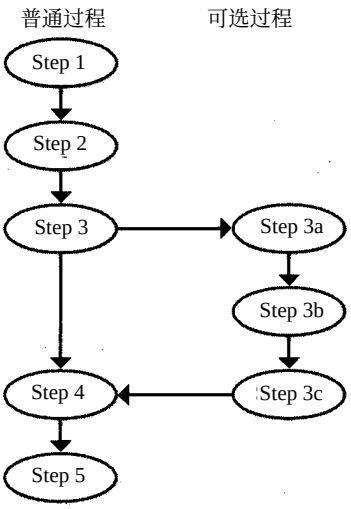

图8-2 使用实例中普通过程和可选过程的对话流

许多使用实例可能共享一些公共函数。为了避免重复，你可以定义一个独立的使用实例，这一实例包含这个公共函数，并指定其它使用实例必须包括这个公共使用实例。这与在编程语言中的所谓“公共子过程”在逻辑上是等价的。被包括的使用实例对于任务的完成是必不可少的，但一个扩充其它实例的使用实例则是可选的，因为普通过程可以完全替代它（Rumbaugh 1994）。举一个例子：在图 8-1中，“请求一种化学制品”实例是那些包含一个称为“输入货物编号”独立实例的许多使用实例中的一种。

⊖ 发展中的软件设计语言的规则或术语，如UML在不断改变，因此这里所用的表示方法在将来可能会过时。本章所介绍的使用实例的概念比起用于表示它们的形式更为重要。

China-pub.com 下载

引起任务不能顺序完成的情况称为例外（exception），在某些时候它可以视为可选过程。

在定义使用实例时，描述例外路径是很重要的，因为它们描述了在特定条件下用户对系统如何工作的看法。“请求一种化学制品”使用实例中的一个例外是不存在业务上可用的化学制品。如果你没有将例外记录在文档上，那么开发者可能在设计和构造阶段忽视这些可能性。此时，当系统遇到一个例外条件时，就会发生系统崩溃。

8.2.2 确定使用实例并编写使用实例文档

可以使用多种方法来确定使用实例（Ham1998; Larman 1998）：

- 首先明确执行者和他们的角色，然后确定业务过程，在这一过程中每一个参与者都在为确定使用实例而努力。
- 确定系统所能反映的外部事件，然后把这些事件与参与的执行者和特定的使用实例联系起来。
- 以特定的说明形式表达业务过程或日常行为，从这些说明中获得使用实例，并确定参与到使用实例中的执行者，
- 有可能从现在的功能需求说明中获得使用实例。如果有些需求与使用实例不一致，就应考虑是否真的需要它们。

由于使用实例代表了用户的需求，在你的系统中，应该直接从不同用户类的代表或至少应从代理那里收集需求。在化学制品跟踪系统项目中，分析者召集一系列 2到3小时的使用实例收集工作专题研习会 (Workshop)，会议隔天举行一次。在客户和开发者的交流方面，召集与组织小组是一种最有效的技术之一（kiel and Carmel 1995）。每一个研习会的参与者包括代表特定用户类的产品代表者，其它选出的用户代表，还有一个或多个开发者。当用户提出不可行的需求时，开发者总是慎重考虑这些需求的实现问题，参与需求获取的工作有助于使他们能及早认识所要开发的产品。“化学制品跟踪系统”中的不同用户类成员参加到不同的研习会，这一切顺利进展是由于对于多个客户类只有少数几个使用实例是通用的。

在举行研习会内的讨论之前，每一分析者总是要求用户考虑他们使用新系统需要完成的任务或业务过程。每一个任务成为一个候选的使用实例，分析者用任务的简明说明对这些使用实例进行编号和命名，例如“请求一种化学制品”或“为化学制品打印安全数据单”。当小组在研习会中探讨使用实例时， 他们发现，一些实例代表了相关的说明，这些说明可以合并到一个简单的抽象使用实例中。抛弃一些建议的使用实例，因为它们超出了项目的范围。小组成员还发现了他们所定义的最初集合以外的附加使用实例。

用户总是趋向于首先确定最重要的使用实例，所以可以根据发现的顺序来确定优先级。另一种确定优先级的方法是当提出候选的使用实例时，给每个实例写 2到3句说明。确定候选的使用实例的优先级，首先为最高优先级的实例填写细节部分，然后为余下的候选使用实例重新评估其优先级。

需求获取的每一次讨论会都可以探索出多个使用实例，并根据标准化模板把它们编写成文档。图8-3描述了简化的“请求一种化学制品”使用实例的模板。首先，参与者将第一个从使用实例中获益的人定为操作者；接下来，他们定义任何满足使用实例运行的先决条件及在完成使用实例后描述系统的后续条件。预先估计的使用频度为并行使用和性能需求提供了一个早期指示。下一步，分析者询问参与者想怎样与系统交互来完成任务。操作者动作的最终

China-pub.com
下载

对话顺序和系统响应构成一个流，这个流可视为事件的普通过程。虽然每个参与者对真正的用户接口和交互机制有不同的想法，但是在执行者 —— 系统对话中，开发组可以在这些步骤上达成共识。

分析者总是在注释中获取单个执行者动作和系统响应，这些注释位于每一个使用实例的流程单上。另一种指导工作的方法是在计算机中生成一个使用实例模板，并在讨论过程中完善模板内容。由于对话的顺序包含了复杂的逻辑或符合式决策，用诸如流程图等图形表达比用一系列标明数字的步骤表达更有启发性。

使用实例标识号	CU —— 5		
使用实例名称	**请求一种化学制品**		
创建者	Tim	最后一次更新者	Janice
创建时间	10/4	最后一次更新时间	10/27
执行者	请求者		
说明	求者通过输入化学制品的 ID或输入化学制品的结构来指定对化学制品的请求。系统则提供给请求者一个来自化学制品仓库的新的或已用过的化学制品容器，或者让请求者向外界供货商订货		
先决条件	1. 用户的身份被证实 2. 具有在线的化学制品存货清单数据库		
请求结果	1. 将完成的请求存入“ 化学制品跟踪系统 ” 2. 通过电子邮件把请求发往化学制品仓库或完成采购		
优先级	高		
使用频度	对于每个药剂师大约每周 5次，对于化学制品仓库的每个成员每周 100次		
普通过程	5.0**从供应商那里请求一种化学制品** **执行者行为** 1. 输入化学制品的ID或者包含化学制品结构的文件名 4. 确定供应商（待续）或化学制品仓库（可选过程5.1）	**系统响应** 2. 验证化学制品的ID是否合法 3. 询问请求者需要一个新的供应商订单或一个来自化学制品仓库的容器 5.<继续对话，直到请求完成>	
可选过程	5.1 **从化学制品仓库中请求一种化学制品（** 5.0.4**之后的分枝）** **执行者行为** 2. 可选择地查看任何容器的历史 3. 选择一个特定的容器或发出供应商订单	**系统响应** 1. 显示出化学制品仓库中现存的所要求的化学容器的列表	
例外	5.E.1**化学制品在业务上不可用** **执行者行为** 3. 请求另一种化学制品	**系统响应** 1. 显示消息：不存在供应商 2. 询问请求者是否要请求另一种化学制品或退出 4. 普通过程结束	
包括	UC-12**输入货物编号**		
特定需求	系统必须可以按标准的编码形式输入化学制品的结构，这一标准来自对不同化学制品图形包的支持		
假设	输入的化学制品结构被认为是合法的		
注释和问题	Tim将查明：在危险列表的第一级上请求一种化学制品是否需要管理层的批准。预期时间1/4		

图8-3 “ 化学制品跟踪系统 ”中“ 请求一种化学制品 ”使用实例的部分描述

China-pub.com 下载

需求获取小组为可选的过程确定出相似的对话并确定例外情况。当分析者问一些诸如“那时，倘若数据库没有联机，将会发生什么情况？”或“当化学制品在业务上不可用时该怎么办？”这类问题时，将会发现许多例外条件。在每个使用实例完全研究清楚后，并且没有提出附加的变化、例外或细节时，研习会的参与者就可以转入到另一个使用实例中去。他们并不尽力在马拉松式的会议中挖掘所有的使用实例。相反，他们计划渐增地挖掘使用实例，然后就不断评价和提炼他们，详尽阐述功能需求的细节和可能的用户接口。

图8-4展示了与使用实例获取研习会及后继活动相关的事件的顺序。在每一个研习会之后，分析者得到使用实例的说明并开始从说明中获取功能需求。这其中有一些是显而易见的，例如“使用一个唯一的、系统赋予的序列号来标识每一个已提交的请求”。其它的则比较微妙，代表了开发者将要开发出的功能，这些功能可以使用户在与系统的交互过程中执行那些步骤。分析者以分级结构化形式编写这些功能需求文档，这种结构化形式适合于写入软件需求规格说明中。

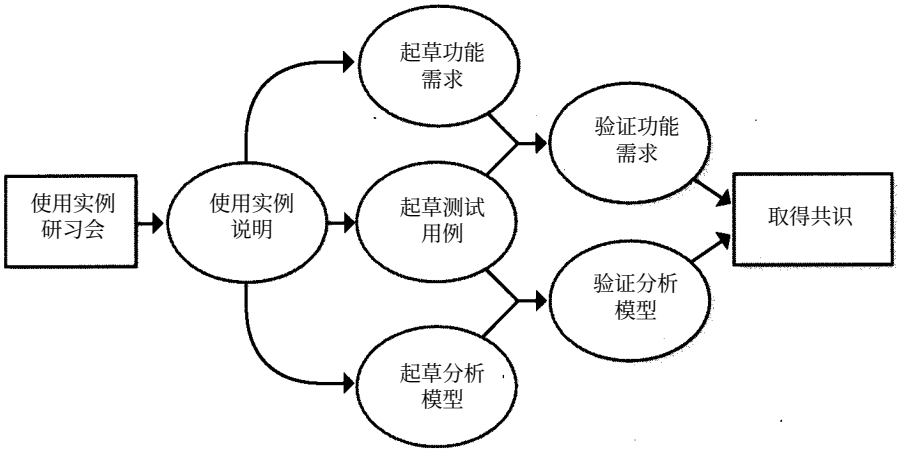

图8-4 使用实例获取方法

分析者发现对于一些复杂的使用实例，画出图形分析模型是有益的，这些模型包括数据流程图、实体关系图、状态转化图、对象类和联系图 (interaction diagram)，还有可能的用户接口机制的对话映像。这些模型描述了需求的不同观点，这样可以经常发现在文档中不易被发现的遗漏、模糊和不一致性的地方。第 10章将为“化学制品跟踪系统”图示多种这样的分析模型。

一天之中，在每一个专题研习会之后，分析者给出使用实例的说明和与研习会参与者的有关的功能需求，这些参与者在把他们的工作成果送至下一个研习会之前都要检查这些功能需求。这些经常性的、不正式的检查可以发现许多错误，例如先前没有发现的可选过程和例外、不正确的功能需求及在执行者—系统 (actor-syetem)对话中遗失的步骤。使用实例方法通过这样渐增式的检查为提高需求质量提供了有力手段。

对于主持高效的获取使用实例的研习会的频繁程度有一个限制。那些每天都主持研习会的分析者发现：参与者难于在他们所检查的文档中发现错误，因为在他们的思维中这些信息太生疏了。从紧张的思维行为中脱离出来，让头脑休息一两天，可以使你从一个新的角度来看待早先的工作。如果你要从检查文档中得益，那么一周最好只主持二或三个研习会。

China-pub.com
下载

早在需求开发阶段，测试者就可以从使用实例中为“化学制品跟踪系统”开发出概念性测试用例。这些测试用例有助于开发组编写清晰的文档，并且在针对特定的使用说明、系统如何运作方面享有共同的见解。测试用例可以使分析者验证他们是否获得了在测试用例中规定的用于产生系统行为的所有功能需求。他们也可以用测试用例来判断分析方法是否完善、正确，并且与功能需求是否一致。第14章更详细地讨论了从需求中生成测试用例。

大系统可以有成百上千个使用实例，因此需要花费相当的时间来确定、详述、编写文档并审核他们。然而，通过使用实例研习会来定义系统预期的功能，对于建立一个高质量软件系统，并且可以满足不同用户需要方面是一个极好的投资 (investment)。

8.2.3 使用实例和功能需求

使用实例的描述并不向开发者提供他们所要开发的功能的细节。如果你在用户需求阶段停止了需求开发，你将会发现在软件的构造阶段，开发者必须询问许多问题来弥补他们的信息空白。为了减少这种不确定性，你需要把每一个使用实例叙述成详细的功能需求（ Arlow 1998）。

每一个使用实例可引伸出多个功能需求，这将使执行者可以执行相关的任务；并且多个使用实例可能需要相同的功能需求。例如，如果有 5个使用实例需要进行用户身份的验证，你不必因此编写5个不同的代码块。可以用多种方法来编写与一个使用实例相联系的功能需求文档。采用何种方法取决于你是否希望开发组从使用实例文档、软件需求规模说明，或二者相结合来设计、构造并测试。这些方法都不完善，因此选择那些最适合你编写文档和管理项目的软件需求方法。避免在不同地方产生信息冗余，因为冗余将使需求管理变得更加困难。

1. 仅利用使用实例的方法

虽然你仍可能需要用一个独立的软件需求规格说明来记录与特定的使用实例无关的需求，但是一种可能的方法是把功能需求包括在每一个使用实例的说明之中。你必须交叉引用在多个使用实例中重复的那些功能需求。一种方法是可以通过先前讨论的“包括”的关系来阐述，在这一关系中一些公共功能（如用户身份验证）被分割到一个独立的，可重用的使用实例中。

2. 利用使用实例和SRS相结合的方法

另一种方法是把使用实例说明限制在抽象的用户需求级上，并且把从使用实例中获得的功能需求编入软件需求规格说明中。在这种方法中，你将需要在使用实例和与之相关的功能需求之间建立可跟踪性。最好的方法是把所有使用实例和功能需求存入数据库或者业务需求管理工具中，这将允许你定义这些跟踪性联系（见第 19章）。

3. 仅利用软件需求规格说明的方法

第三种方法是通过使用实例来组织软件需求规格说明，并且包括使用实例和功能需求的说明。采用这种方法，你无需独立编写详细的使用实例文档；然而，你需要确定冗余的功能需求，或者对每个功能需求仅陈述一次，并且无论需求是否重复出现在其它使用实例中，都要参考它的原始说明。

8.2.4 使用实例的益处

使用实例方法给需求获取带来的好处来自于该方法是以任务为中心和以用户为中心的观点。比起使用以功能为中心的方法，使用实例方法可以使用户更清楚地认识到新系统允许他

China-pub.com 下载

们做什么。在许多Web开发工程中，用户代表发现，使用实例的方法真正有助于他们弄清 Web 站点的访问者可以做什么。使用实例有助于分析者和开发者理解用户的业务和应用领域。认真思考执行者 —— 系统对话的顺序，使其可以在开发过程早期发现模糊性，也有助于从使用实例中生成测试用例。

如果开发者所编写的代码从未被使用过，这将是令人沮丧的。有了使用实例，所得到的功能需求明确规定了用户执行的特定任务。使用实例技术防止了“ 孤立 ”的功能 —— 这些功能在需求获取阶段似乎是一个好的见解，但没有人使用它们，因为它们并没有与用户任务真正地联系在一起。

在技术方面，使用实例的方法也带来了好处。使用实例观点揭示了域对象以及它们之间的责任。开发者运用面向对象的设计方法可以把使用实例转化为对象模型。进而，当业务过程随时间而变时，内嵌在特定的使用实例中的任务也会相应改变。如果你跟踪功能需求、设计、编码和测试以至到它们父类的使用实例 —— 用户意见 —— 那么这就很容易看出整个系统中业务过程的级联变化。

8.2.5 避免使用实例陷阱

在使用实例的方法中应注意如下的陷阱：

- 太多的使用实例　如果你发现自己陷入使用实例的爆炸之中，你就可能不能在一个合适的抽象级上为之编写文档。不要为每一个可能的说明编写单独的使用实例。而是把普通过程、可选过程以及例外集成起来写入一个简单的使用实例。也不要把交互顺序中的每个步骤看成一个单独的使用实例。每一个使用实例都必须描述一个单独的任务。你将获得比业务需求多得多的使用实例，但你的功能需求将比使用实例还多。每一个使用实例都描述了一个方法，用户可以利用这个方法与系统进行交互，从而达到特定的目标。
- 使用实例的冗余（duplication accross）　如果相同的函数出现在多个使用实例中，那么就有可能多次重写函数的实现部分。为了避免重复，可以使用“ 包括 ”关系，在这一关系中，将公共函数分离出来并写到一个单独的使用实例中，当其它使用实例需要该函数时，可以请求调用它。
- 使用实例中的用户界面的设计　使用实例应该把重点放在用户使用系统做什么，而不是关心屏幕上是怎么显示的。强调在执行者和系统之间概念性的对话的理解，而把用户界面的细节推迟到设计阶段。任何屏幕上的画面和用户界面机制的影响只是使执行者 —— 系统交互的可视化，而并不作为主要的设计说明。
- 使用实例中包括数据定义　我见过一些包括数据项和数据结构定义的使用实例，可以在该使用实例中操纵这些数据项和数据结构，包括数据类型、长度、格式和合法值。这个方法使项目的参与者难于找到他们所需要的定义，因为使用实例中说明它所包含的每一数据定义是不明显的。这也可导致冗余的定义，当一个使用实例改变时，其它的没有改变，但破坏了同步性。应该把数据定义集中在适用于整个项目范围的数据字典中（第 9 章讨论），以便在使用这些数据时减少不一致性。
- 试图把每一个需求与一个使用实例相联系　使用实例可有效地捕捉大多数所期望的系统行为，但是你可能有一些需求，这些需求与用户任务或其他执行者之间的交互没有特定的关系。这时你就需要一个独立的需求规格说明，在这个规格说明中可以编写非功能需

China-pub.com
下载

求文档、外部接口需求文档以及一些不能由使用实例得到的功能需求支持。

8.3 对客户输入进行分类

不要指望你的客户会给需求分析者提供一个简洁、完整、组织良好的需求清单。分析者必须把代表客户需求的许多信息分成不同的类型，这样他们就能合理地编写信息文档并把它们用于最合理的方式上。图 8-5描述了许多这样的类型。那些不属于这些类型的信息可能代表一个非软件项目的需求，例如，为使用新系统而进行的用户培训或者它仅仅是不重要的信息。下面讨论在听取客户需求过程中的一些建议，这将有助于对信息进行分类处理。

图8-5 用户需求分类

1) 业务需求 描述客户或开发公司可以从产品中得到的资金、市场或其它业务利润的需求就是最可能的业务需求。再听听直接或间接的软件用户得到好处，例如“市场股票价格上升x%”，“每年节约$Y”，或“替代了维护费用高的老一代系统 Z”。

2) 使用实例或说明 有关利用系统执行的业务任务或达到用户目标的总的陈述可能就是使用实例，而特定的任务描述表示了用法说明。与客户一起商讨，把特定的任务概括成更广泛的使用实例。可以通过让客户描述他们的业务工作流活动来获取使用实例。

3) 业务规则 当一个客户说，一些活动只能在特定的条件下，由一些特定的人来完成时，该用户可能在描述一个业务规则，例如，“如果一个药剂师在危险化学制品培训方面是可靠的，那么他就可以在一级危险药品清单上订购化学制品”。业务规则是有关业务过程的操作原则。你可以用一些软件功能需求来加强规则，例如，让“化学制品跟踪系统”可以访问培训记录数据库。正如上面所说的那样，这里业务规则不是功能需求。

4) 功能需求 客户所说的诸如“用户应该能<执行某些功能>”或者“系统应该<具备某些行为〉”，这是最可能的功能需求。功能需求描述了系统所展示的可观察的行为，并且大多数是处于执行者 —— 系统响应顺序的环境中。功能需求定义了系统应该做什么，它们组成了软件需求规格说明的一部分。分析者应该明确，每个人应理解系统为什么“必须”执行某一功能。所提出的功能需求有时反映了过时的或无效的业务过程，而这些过程不能加入到新系统中。

5) 质量属性 对系统如何能很好地执行某些行为或让用户采取某一措施的陈述就是质量属性，这是一种非功能需求。听取那些描述合理特性的意见：快捷、简易、直觉性、用户友

China-pub.com 下载

好、健壮性、可靠性、安全性和高效性。你将要和用户一起商讨精确定义他们模糊的和主观言辞的真正含义，这将在第11章描述。

6) 外部接口需求　这类需求描述了系统与外部的联系。软件需求规格说明必须包括用户接口和通信机制、硬件和其它软件系统需求部分。客户描述外部接口需求包括如下习惯用语：

- “从<某些设备>读取信号”
- “给<一些其它系统>发送消息”
- “以<某种格式>读取文件”
- “能控制<一些硬件>”

7) 限制　限制是指一些合理限制设计者和程序员选择的条件。它们代表了另一种类型的非功能需求，你必须把这些需求写入软件需求规格说明。尽量防止客户施加不必要的限制，因为这将妨碍提出一个好的解决方案。不必要的限制将会降低利用现有商业化软件集成解决方案的能力。一定的限制有助于提高产品质量属性。只利用程序语言的标准命令而不允许使用供应商的扩展，可以提高可移植性。下面是客户描述限制的一些习惯用语：

- “必须使用<一个特定的数据库产品或语言>”
- “不能申请多于<一定数量的内存>”
- “操作必须与<其它系统>相同”
- “必须与<其它应用程序>一致”

8) 数据定义　当客户描述一个数据项或一个复杂的业务数据结构的格式、允许值或缺省值时，他们正在进行数据定义。例如，“邮政编码由5个数字组成，后跟一个可选的短划线或一个可选的四位数字，缺省为0000”就是一个数据定义。把这些集中在一个数据字典中，作为项目的参与者在整个项目的开发和维护中的主要参考文档。

9) 解决思想　如果一个客户描述了用户与系统交互的特定方法，以使系统产生一系列活动（例如：用户从下载清单中选择一个所需要的项），这时你正在听取建议性的解决方案，而不是需求。所建议的解决方案使获取需求小组成员在潜在的真正需求上分散精力。在获取需求时，应该把重点放在需要作什么而不是新系统应该如何设计和构造。探讨客户为什么提出一个特定的实现方法，因为这可以帮助你理解真正的需求和用户对如何构造系统的隐含的期望。

8.4 需求获取中的注意事项

在需求获取的过程中，你可能会发现对产品范围的定义存在误差，不是太大就是太小（Christer and Kang 1992）。如果范围太大，你将要收集比真正需要更多的需求，以传递足够的业务和客户的值，此时获取过程将会拖延。如果项目范围太小，那么客户将会提出很重要的但又在当前产品范围之外的需求。当前的范围太小，以致不能提供一个令人满意的产品。需求的获取将导致修改项目的范围和任务，但作出这样具有深远影响的改变，一定要小心谨慎。

正如经常所说的，需求主要是关于系统做什么，而解决方案如何实现是属于设计的范围。这样说虽然很简洁，但似乎过于简单化。需求的获取应该把重点放在“做什么”上，但在分析和设计之间还是存在一定的距离。你可以使用假设“怎么做”来分类并改善你对用户需求的理解。在需求的获取过程中，分析模型、屏幕图形和原型可以使概念表达得更加清楚，然后提供一个寻找错误和遗漏的办法。把你在需求开发阶段所形成的模型和屏幕效果看成是方便高效交流的概念性建议，而不应该看成是对设计者选择的一种限制。

China-pub.com
下载

需求获取研习会中如果参与者过多，就会减慢进度。我的同事 Debbie在她从事过Web开发项目中，由于使用实例研习会进展缓慢，曾一度感到灰心丧气。 12位参与者对不必要的细节进行激烈的讨论，并且在每个使用实例如何工作的问题上难以达成一致意见。当她把工作人员减少到只有6个人时，进度马上加快了，而这 6个人代表了客户、系统设计者、开发者和可视化设计者等主要工程角色。

相反地，从极少的代表那里收集信息或者只听到呼声最高、最有舆论影响的用户的声音，也会造成问题。这将导致忽视特定用户类的重要的需求，或者其需求不能代表绝大多数用户的需要。最好的权衡在于选择一些授权为他们的用户类发言的产品代表者，他们也被同组用户类的其它代表所支持。

8.5 如何知道你何时完成需求的获取

没有一个简单、清楚的信号暗示你什么时候已完成需求获取。当客户和开发者与他们的同事聊天、阅读工业和商业上的文献及在早上沐浴时思考时，他们都将对潜在产品产生新的构思。你不可能全面收集需求，但是下列的提示将会暗示你在需求获取的过程中的返回点。

- 如果用户不能想出更多的使用实例，也许你就完成了收集需求的工作。用户总是按其重要性的顺序来确定使用实例的。
- 如果用户提出新的使用实例，但你可以从其它使用实例的相关功能需求中获得这些新的使用实例，这时也许你就完成了收集需求的工作。这些新的使用实例可能是你已获取的其它使用实例的可选过程。
- 如果用户开始重复原先讨论过的问题，此时，也许你就完成了收集需求的工作。
- 如果所提出的新需求比你已确定的需求的优先级都低时，也许你就完成了收集需求的工作。
- 如果用户提出对将来产品的要求，而不是现在我们讨论的特定产品，也许你就完成了收集需求的工作。

在项目的早期阶段，列出系统可能包括的功能区。在考虑多个项目的情况下，你可能要列出公共函数的清单，例如：写错误日志、备份与恢复、报表、打印、预览功能、格式和用户最喜爱的特性。然后在需求开发阶段，把你所确定的功能与原始列表进行比较。如果没有发现不一致性，则你可能完成了收集需求的工作。

下一步：

- 在你的项目或软件需求规格说明中选择一部分已经写好的客户意见输入。如图 8-5所示，把包括在这一部分中的每一项进行分类。如果你发现有些项并不是功能需求中所要求的，就把它们放到软件需求规格说明中的正确位置上，或其它合适的文档中。
- 运用图8-3所示的使用实例模板为你的项目编写一个使用实例。包括所有可选过程和例外。定义功能需求，以使用户可以成功地完成这个使用实例。检查你目前的软件需求规格说明中是否包含了全部的功能需求。
- 列出在你当前项目中所用的所有需求获取的方法。那一个最有效？为什么？哪一个不可行？为什么？确定你认为最好的需求获取技术，并决定下次怎样应用它们。记住你在进行这些工作时遇到的障碍和在克服这些障碍时机智有效的办法。