

理解需求

要点浏览

概念：在开始任何技术工作之前，关注于一系列需求工程任务是个好主意。这些任务有助于你理解软件将如何影响业务、客户想要什么以及最终用户将如何和软件交互。

人员：软件工程师（在IT界有时指系统工程师或分析员）和与项目利益相关的其他人员（项目经理、客户、最终用户）都将参与需求工程。

重要性：在设计和开发某个优秀的计算机软件时，如果软件解决的问题是错误的，那么即使软件再精巧也满足不了任何人的要求。这就是在设计和开发一个基于计算机的系统之前理解客户需求的重要性所在。

步骤：需求工程首先从“起始”开始定义将要解决的问题的范围和性质；然后是

导出，帮助利益相关者定义需要什么；接下来是精化，精确定义和修改基本需求。当利益相关者提出问题后，就要进行谈判：如何定义优先次序？什么是必需的？何时需要？最后，以某种形式明确说明问题，再经过评审或确认以保证我们和利益相关者对于问题的理解是一致的。

工作产品：需求工程的目的在于为各方提供关于问题的一个书面理解。不过依然可以得到工作产品：用户场景、功能和特征列表、需求模型或规格说明。

质量保证措施：利益相关者评审需求工程的工作产品，以确保你所理解的正是他们真正想要的。需要提醒大家的是：即使参与各方均认可，事情也会有变化，而且变化可能贯穿整个项目实施过程。

关键概念

分析模型

分析模式

协作

精化

导出

起始

谈判

质量功能部署

需求工程

需求收集

需求管理

规格说明

利益相关者

用例

确认需求

确认

视点

工作产品

理解问题的需求是软件工程师所面对的最困难的任务之一。第一次考虑开发一个清晰易于理解的需求的时候，可能看起来并不困难。试问，客户难道不知道需要什么？最终用户难道对将给他们带来实际收益的性能和功能没有清楚的认识？不可思议的是，很多情况下的确是这样的。甚至即使用户和最终用户清楚地知道他们的要求，这些要求也会在项目的实施过程中改变。


在Ralph Young[You01]的一本关于有效需求实践的书的前言中我写道：

这是最恐怖的噩梦：一个客户走进你的办公室，坐下，正视着你，然后说“我知道你认为你理解我说的是，但你并不理解的是：我所说的并不是我想要的”。这种情况总是在项目后期出现，而当时的情况通常是：已经作出交付期限的承诺，声誉悬于一线并且已经投入大量资金。

我们这些已经在系统和软件业中工作多年的人就生活这样的噩梦中，而且目前还都不知道该怎么摆脱。我们努力从客户那里推导出需求，但是难以理解获取的信息。我们通常采用混乱的方式记录需求，而且总是花费极少的时间检验到底记录了什么。我们容忍变更控制我们自己而不是建立机制去控制变更。总之，我们未能为系统或软件奠定坚实的基础。这些问题中的每一个都是极富挑战性的，当这些问题集中在一起时，即使是最有经验的管理人员和实际工作人员也会感到头痛。但是，确实存在解决方法。


把需求工程称做以上所述挑战的“解决方案”可能并不合理，但需求工程确实为我们提供了解决这些挑战的可靠途径。

5.1 需求工程

 “构建一个软件系统最困难的部分是确定构建什么。其他部分工作不会像这部分工作一样，在出错之后会如此严重地影响随后实现的系统，并且以后修补定会如此的困难。”——Fred Brooks

 **KEY POINT**
需求工程为设计和构造奠定了坚实的基础。如果没有需求工程，那么实现的软件很有可能无法满足客户的需求。

 **ADVICE**
可以在需求阶段作一些设计工作，在设计阶段作一些需求工作。

 “大多数软件灾难的种子通常都是在软件项目开始的头三个月内种下的。”——Capers Jones

设计和编写软件富有挑战性、创造性和趣味性。事实上，编写软件是如此吸引人，以至于很多软件开发人员在清楚地了解用户需要什么之前就迫切地投入到编写工作中。开发人员认为：在编写的过程中事情总是会变得清晰；只有在检验了软件的早期版本后项目利益相关者才能够更好地理解要求；事情变化太快，以至于理解需求细节是在浪费时间；最终要做的是开发一个可运行的程序，其他都是次要的。构成这些论点的原因在于其中也包含了部分真实情况^①，但是这中间的每个论点都存在一些小问题，汇集在一起就可能导致软件项目的失败。

需求工程 (Requirement Engineering, RE) 是指致力于不断理解需求的大量任务和技术。从软件过程的角度来看，需求工程是一个软件工程动作，开始于沟通活动并持续到建模活动。它必须适应于过程、项目、产品和人员工作的需要。

需求工程在设计和构造之间建立起联系的桥梁。桥梁源自何处？有人可能认为开始于项目利益相关者（如项目经理、客户、最终用户），也就是在他们那里定义业务需求，刻画用户场景，描述功能和特性，识别项目约束条件。其他人可能会建议从宽泛的系统定义开始，此时软件只是更大的系统范围中的一个构件。但是不管起始点在哪里，横跨这座桥梁将把我们带到项目之上更高的层次：允许由软件团队检查将要进行的软件工作的内容；必须提交设计和构建的特定要求；完成指导工作顺序的优先级定义；以及将深切影响随后设计的信息、功能和行为。

需求工程为以下工作提供了良好的机制：理解客户需要什么、分析要求、评估可行性、协商合理的方案、无歧义地详细说明方案、确认规格说明、管理需求以至将这些需求转化为可运行系统[Tha97]。需求工程过程通过执行七个不同的活动来实现：起始、导出、精化、协商、规格说明，确认和管理。重要的是注意到，某些需求工程任务并行发生且要全部适应项目的要求。

起始

如何开始一个软件项目？有没有一个独立的事件成为新的基于计算机的系统或产品的催化剂？或是要求随时间流逝而发展吗？这些问题没有确定的答案。某些情况下，一个偶然的交谈就可能必须投入大量的软件工程项目。但是多数项目都是当确定了商业要求或是发现了潜在的新市场、新服务时才开始。业务领域的利益相关者（如业务管理人员、市场人员、产品管理人员）定义业务用例，确定市场的宽度和深度，进行粗略的可行性分析，并确定项目范围的工作说明。所有这些信

① 对小项目（不超过一个月）和只涉及简单的软件工作的更小项目，这确实是正确的。随着软件规模和复杂性的增加，这些论点就开始出问题了。


② 如果要开发一个基于计算机的系统，讨论将从系统工程开始，涉及关乎各领域的全局以及系统所在的领域活动。请访问本书所列的公司网站获取更多系统工程的详细讨论。

在项目起始阶段中^①，要建立基本的理解，包括对问题、谁需要解决方案、所期望解决方案的性质、与项目利益相关者和开发人员之间达成初步交流合作的效果。

导出

询问客户、用户和其他人，系统或产品的目标是什么，想要实现什么，系统和产品如何满足业务的要求，最终系统或产品如何用于日常工作，这些问题是非常简单的。但是，实际上并非如此简单——这非常困难。

Christel和Kang[Cri92]指出了一系列问题，可以帮助理解为什么导出需求这么困难。

 为什么对客户的要求要获得清晰的理解会非常困难？



精化是件好事，但你必须知道何时可以停止精化。关键是能采用为设计建立一个坚实的基础的方式说明问题。如果超出这个点就是在做设计。



在有效的协商中没有赢家也没有输家，而是双赢。这是因为双方合作才是“交易”的坚实基础。



规格说明的形式和格式随着待开发软件的规模和复杂度的不同而变化。

- **范围问题**：系统的边界不清楚，或是客户或用户的说明带有不必要的技术细节，这些细节可能会导致混淆而不是澄清系统的整体目标。
- **理解问题**：客户或用户并不能完全确定需要什么，他们对其计算环境的能力和限制所知甚少，对问题域没有完整的认识，与系统工程师在沟通上存在问题，省略那些他们认为是“明显的”信息，确定的需求和其他客户或用户的需求相冲突，需求说明有歧义或不可测试。
- **易变问题**：需求随时间变化。

为了帮助解决这些问题，需求工程师必须以有组织的方式开展需求收集活动。

精化

在起始和导出阶段获得的信息将在精化阶段进行扩展和提炼。该任务集中于开发一个精确的需求模型（第6、7章），用以说明软件的功能、特征和信息的各个方面。

精化是由一系列的用户场景建模和求精任务驱动的。这些用户场景描述了如何让最终用户和其他参与者与系统进行交互。解析每个用户场景以便提取分析类——最终用户可见的业务域实体。应该定义每个分析类的属性，确定每个类所需要的服务^②，确定类之间的关联和协作关系，并完成各种补充图。

协商

只给定了有限的业务资源，而客户和用户提出了过高的要求，这是常有的事。另一个相当常见的现象是，不同的客户或用户提出了相互冲突的需求，并坚持“我们的特殊要求是至关重要的”。

需求工程师必须通过协商过程来调解这些冲突。应该让客户、用户和其他利益相关者对各自的需求排序，然后按优先级讨论冲突。使用迭代的方法给需求排序，评估每项需求对项目产生的成本和风险，表述内部冲突，删除、组合和（或）修改需求，以便参与各方均能达到一定的满意度。

规格说明

在基于计算机的系统（和软件）的环境下，术语规格说明对不同的人有不同的含义。一个规格说明可以是一份写好的文档、一套图形化的模型、一个形式化的数学模型、一组使用场景、一个原型或上述各项的任意组合。

有人建议应该开发一个“标准模板”[Som97]并用于规格说明，他们认为这样将促使以一致的也更易于理解的方式来表示需求。然而，在开发规格说明时保持灵活性有时是必要的，对大型系统而言，文档最好采用自然语言描述和图形化

① 应该记得第2章统一过程定义了更全面的“起始阶段”，包括本章所讨论的起始、导出和精化工作。

② 服务通过对类的封装操作数据，也可使用术语“操作”和“方法”。如果你不熟悉面向对象的概念，附录2有基本的入门指导。

模型来编写。而对于技术环节明确的较小产品或系统，使用场景可能就足够了。

INFO

软件需求规格说明模板

软件需求规格说明 (SRS) 是在项目商业化之前必须建立详细描述软件各个方面的文档。值得注意的是常常并没有写正规的SRS。事实上很多实例表明花在软件需求规格说明的工作量还不如投入到其他软件工程活动。然而，当软件由第三方开发时，当缺少规格说明导致严重业务问题时，或是当系统非常复杂或涉及十分重要的业务时，才能表明需求规格说明是非常必要的。

Process Impact公司的Karl Wiegers[Wie03]开发了一套完整的模板(参考www.processimpact.com/process_assets/srs_template.doc)能为那些必须建立完整需求规格说明书的人提供指导。大纲如下：

目录

版本历史

1 引言

1.1 目的

1.2 文档约定

1.3 适用人群和阅读建议

1.4 项目范围

1.5 参考文献

2 总体描述

2.1 产品愿景

2.2 产品特性

2.3 用户类型和特征

2.4 操作环境

2.5 设计和实现约束

2.6 用户文档

2.7 假设和依赖

3 系统特性

3.1 系统特性1

3.2 系统特性2 (等等)

4 外部接口需求

4.1 用户接口

4.2 硬件接口

4.3 软件接口

4.4 通信接口

5 其他非功能需求

5.1 性能需求

5.2 安全需求

5.3 保密需求

5.4 软件质量属性

6 其他需求

附录A: 术语表

附录B: 分析模型

附录C: 问题列表

每个需求规格说明主题的详细描述可以从前面所提的URL下载SRS模板来得到。

确认



需求确认时的一个重要问题是一致性。使用分析模型可以保证需求说明保持一致性。

在确认这一步将对需求工程的工作产品进行质量评估。需求确认要检查规格说明^①以保证：已无歧义地说明了所有的系统需求；已检测出不一致性、疏忽和错误并得到纠正；工作产品符合为过程、项目和产品建立的标准。

正式的技术评审(第15章)是最主要的需求确认机制。确认需求的评审小组包括软件工程师、客户、用户和其他利益相关者，他们检查系统规格说明，查找内容或解释上的错误，以及可能需要进一步解释澄清的地方、丢失的信息、不一致性(这是建造大型产品或系统时遇到的主要问题)、冲突的需求或是不可实现的(不能达到的)需求。

① 应该记得每个项目有不同的规格说明特性。在某些情况下，规格说明是指收集到的用户场景和其他一些事物。在另一些情况下，规格说明可以包括用户场景、模型和写成说明书的文档。

INFO

需求确认检查表：

通常，按照检查表上的一系列问题检查每项需求是非常有用的。这里列出其中部分可能会问到的问题：

- 需求说明清晰吗？有没有可能造成误解？
- 需求的来源（如人员、规则、文档）弄清了吗？需求的最终说明是否已经根据或对照最初来源检查过？
- 需求是否用定量的术语界定？
- 其他哪些需求和此需求相关？是否已经使用交叉索引矩阵或其他机制清楚地加以说明？
- 需求是否违背某个系统领域的约束？
- 需求是否可测试？如果可以，能否说明检验需求的测试（有时称为确认准则）？
- 对已经创建的任何系统模型，需求是否可跟踪？
- 对整体系统/产品目标，需求是否可跟踪？
- 规格说明的构造方式是否有助于理解、轻松引用和翻译成更技术性的工作产品？
- 对已创建的规格说明是否建立了索引？
- 与系统性能、行为及运行特征相关需求的说明是否清楚？哪些需求是隐含出现的？

需求管理

基于计算机的系统其需求会变更，而且变更的要求贯穿于系统的整个生存期。需求管理是用于帮助项目组在项目进展中标识、控制和跟踪需求以及需求变更的一组活动^①。这类活动中的大部分内容和第22章中讨论的软件配置管理（SCM）技术是相同的。

SOFTWARE TOOLS

需求工程

目的：需求工程工具有助于需求收集、需求建模、需求管理和需求确认。

工作机制：工具的工作机制多种多样。通常，需求工程工具创建大量的图形化（例如UML）模型来说明系统的信息、功能和行为。这些模型构成了软件过程中其他所有活动的基础。

代表性工具：^②

Volere需求资源网站（www.volere.co.uk/tools.htm）提供了一个相当全面（也是最新的）列表。我们在第6、7章将讨论需求建模工具。下面提到的工具主要侧重于需求管理。

EasyRM：由Cybernetic Intelligence GmbH（www.easy-rm.com）开发，构建项目专用的字典或术语，包含详细的需求说明和属性。

Rational RequisitePro：由Rational Software（www-306.ibm.com/software/awdtools/reqpro/）开发，允许用户构建需求数据库，描述需求之间的关联，组织、优先级排序以及跟踪需求。

从前面所提的Volere网站和WWW.jiludwig.com/requirements_management_tools.html，还有另外一些的需求管理工具。

① 正规的需求管理只适用于具有数百个可确认需求的大型项目。对于小项目，该需求工程工作可以适当的裁减，一定程度的不正规也是可以接受的。

② 这里提到的工具只是此类工具的例子，并不代表本书支持采用这些工具。在大多数情况下，工具名称被各自的开发者注册为商标。

5.2 建立根基

KEY POINT

利益相关者是那些对将要开发的系统有直接的兴趣或直接从中获益的人。

在理想情况下，利益相关者和软件工程师在同一个小组中工作^①。在这种情况下，需求工程就只是和组里熟悉的同事进行有意义的交谈。但实际情况往往不是这样。

客户或最终用户可能位于不同的城市或国家，对于想要什么可能仅有模糊的想法，对于将要构建的系统可能存在有冲突的意见，他们的技术知识可能很有限，可能仅有有限的时间与需求工程师沟通。这些事情都是不希望遇到的，但却又是十分常见的，软件开发团队经常被迫在这种环境的限制下工作。

在下面的小节中，将要讨论启动需求工程所必需的步骤，以便理解软件需求。这是项目自始至终走向成功解决方案的方向。

5.2.1 确认利益相关者

Sommerville和Sawyer[Som97]把利益相关者定义为“直接或间接地从正在开发的系统中获益的人”。可以确定如下几个容易理解的利益相关者：业务运行管理人员、产品管理人员、市场销售人员、内部和外部客户、最终用户、顾问、产品工程师、软件工程师、支持和维护工程师以及其他人员。每个利益相关者对系统都有不同的考虑，当系统成功开发后所能获得的利益也不相同，同样地，当系统开发失败时所面临的风险也是不同的。

在开始阶段，需求工程师应该创建一个人员列表，列出那些有助于诱导出需求的人员(5.3节)。最初的人员列表将随着接触的利益相关者人数增多而增加，因为每个利益相关者都将被询问“你认为我还应该和谁交谈”。

5.2.2 识别多重观点

把三个利益相关者请进一个房间，然后问他们想要什么样的系统，你很可能得到四个或更多的不同观点。”
——作者不详

因为存在很多不同的利益相关者，所以系统需求调研也将从很多不同的视角开展。例如，市场销售部门关心能激发潜在市场的、有助于新系统更容易销售的功能和特点；业务经理关注应该在预算内实现特点，并且这些特点应该满足已规定的市场限制；最终用户可能希望系统的功能是他们所熟悉的并且易于学习和使用；软件工程师可能关注哪些非技术背景的利益相关者看不到的软件基础设施能够支持更多的适于销售的功能和特点；支持工程师可能关注软件的可维护性。

这些参与者（以及其他人员）中的每一个人都将为需求工程过程贡献信息。当从多个角度收集信息时，所形成的需求可能存在不一致性或是相互矛盾。需求工程师的工作就是把所有的利益相关者提供的信息（包括不一致或是矛盾的需求）分类，分类的方法应该便于决策制定者为系统选择一个内部一致的需求集合。

5.2.3 协同合作

假设一个项目中有五个利益相关者，那么对一套需求就会有五种或更多的正确观点。我们提醒客户（和其他利益相关者）之间应团结协作（避免内讧），并和软件工程师团结协作，才能成功实现预定的系统。但是如何实现协作？

需求工程师的工作是标识公共区域（即所有利益相关者都同意的需求）和矛盾区域或不一致区域（即某个利益相关者提出的需求和其他利益相关者的需求相矛盾）。很明显，后一种分类更有挑战性。

^① 推荐所有项目都使用该方法，而且该方法是敏捷软件开发方法的主要部分。


使用“优先点”：

有一个方法能够解决相互冲突的需求，同时更好地理解所有需求的相对重要性，那就是使用基于优先点的“投票”方案。所有的利益相关者都会分配到一定数量的优先点，这些优先点可以适用于很多需求。每个利益相关者在一个需求列表上，通过向每个需求分配一个或多个优先点来表明（从他或她的个人观点）该需求的相对重要性。优先点用过之后就不能再次使用，一旦某个利益相关者的优先点用完，他就不能再对需求实施进一步的操作。所有利益相关者在每项需求上的优先点总数显示了该需求的综合重要性。

协作并不意味着必须由委员会定义需求。在很多情况下，利益相关者的协作是提供他们各自关于需求的观点，而一个强有力的“项目领导者”（例如业务经理或高级技术员）可能要对删减哪些需求做出最终决策。

5.2.4 首次提问


在项目需求导出时的提问应该是“与环境无关的”[Gau89]。第一组与环境无关的问题集中于客户和其他利益相关者、整体目标、收益。例如，需求工程师可能会问：

 “知道问题比知道所有的答案更好。”——James Thurber

- 谁是这项工作的最初请求者？
- 谁将使用该解决方案？
- 成功的解决方案将带来什么样的经济收益？
- 对于这个解决方案你还需要其他资源吗？


这些问题有助于识别所有对构建软件感兴趣的利益相关者。此外，问题还确认了某个成功实现的可度量收益以及定制软件开发的可选方案。

下一组问题有助于软件开发组更好地理解问题，并允许客户表达其对解决方案的看法：

 什么问题有助于你获得对问题的初步认识？

- 如何描述由某成功的解决方案产生的“良好”输出的特征？
- 该解决方案强调解决了什么问题？
- 能向我们展示（或描述）解决方案使用的商业环境吗？
- 存在将影响解决方案中特殊的性能问题或约束吗？

最后一组问题关注于沟通活动本身的效率。Gause 和 Weinberg [Gau89] 称之为“元问题”。下面并给出了元问题的简单列表：

 “问问题的人是五分钟的傻瓜，而不问问题的人将永远是傻瓜。”——中国谚语

- 你是回答这些问题的合适人选吗？你的回答是“正式的”吗？

- 我的提问和你想解决的问题相关吗？
- 我的问题是否太多了？
- 还有其他人员可以提供更多的信息吗？
- 还有我应该问的其他问题吗？

这些问题（和其他问题）将有助于“打破坚冰”，并有助于交流的开始，而且这样的交流对需求导出的成功至关重要。但是，会议形式的问与答（Q&A）并非是一定会取得成功的好方法。事实上，Q&A会议应该仅仅用于首次接触，然后应该用需求诱导方式包括问题求解、协商和规格说明取代。在5.3节中将介绍这类方法。


5.3 导出需求


导出需求（又称为需求收集）是与问题求解、精化、谈判和规格说明等方面的元素结合在

一起的。为了鼓励合作，一个包括利益相关者和开发人员的团队共同完成如下任务：确认问题，为解决方案的要素提供建议，商讨不同的方法并描述初步的需求解决方案 [Zah90]^①。

5.3.1 协作收集需求

关于需求收集，现在已经提出了很多不同的协同需求收集方法，各种方法适用于稍有不同的场景，而且所有这些均是在下面的基本原则之上作了某些改动：

 主持一个协作的需求收集会议的基本原则是什么？

 “我们大量的时间（项目的主要工作量）不是花在实现或测试上，而是花在决定构造什么上。”——
Brian Lawrence

- 会议由软件工程师和其他的利益相关者共同举办和参与。
- 制定筹备和参与会议的规则。
- 建议拟定一个会议议程，这个议程既要足够正式，使其涵盖所有的重点；但也不能太正式，以鼓励思想的自由交流。
- 由一个“调解人”（可以是客户、开发人员或其他人）控制会议。
- 采用“方案论证手段”（可以是工作表、活动挂图、不干胶贴纸或电子公告牌、聊天室或虚拟论坛）。

目的是识别问题，提出解决方案的要素，协商不同的方法以及在有利于完成目标的氛围中确定一套解决需求问题的初步方案。为了更好地理解在会议召开时不断发生的事件流，我们提出了一个概略的场景，用其表述在需求收集会议中及会议后将发生的一些事件。

在需求的起始阶段（5.2节），基本问题和问题的答案确定了问题的范围和对解决方案的整体理解。除了这些启动会议之外，开发人员和客户要写下一个1~2页的“产品要求”。

此外还要选择会议地点、时间和日期，并选择调解人。来自开发组和其他利益相关者组织的代表被邀请出席会议，在会议召开之前应将产品要求分发给所有的与会者。

举一个例子^②，考虑SafeHome项目中的一个市场营销人员撰写的产品要求。此人对SafeHome项目的住宅安全功能叙述如下：

我们的研究表明，住宅管理系统市场以每年40%的速度增长。我们推向市场的首个SafeHome功能将是住宅安全功能，因为多数人都熟悉“报警系统”，所以这将更容易销售。

住宅安全功能应该为防止和（或）识别各种不希望出现的“情况”提供保护，如非法入侵、火灾、漏水、一氧化碳浓度超标等。该功能将使用无线传感器监控每种情况，户主可以编程控制，并且在发现情况时自动电话联系监控部门。

事实上，其他人在需求收集会议中将补充大量的信息。但是，即使有了补充信息，仍有可能存在歧义性和疏漏，也有可能发生错误。但对此时而论，上面的“功能描述”是足够了。

在召开会议评审产品要求的前几天，要求每个与会者列出构成系统周围环境的对象、由系统产生的其他对象以及系统用来完成功能的对象。此外，要求每个与会者列出服务操作或与对象交互的服务（过程或功能）列表。最后，

WebRef

联合应用程序开发（JAD）是需求收集普遍采用的技术。可以从以下地址找到详细的说明：www.carolla.com/wp-jad.htm。

ADVICE

如果一个系统或产品将要为很多用户提供服务，必须绝对保证需求是从所有用户的代表群中提取的。如果只有一个用户定义所有的需求，那么接收风险将会非常高。

① 这种方法有时称为协助应用规格说明技术（Facilitated Application Specification Technique, FAST）

② SafeHome例子（有一定的扩展和改动）在下面很多章节中用于说明软件工程的重要方法，作为一个练习，为该例子举行你自己的需求收集会议并为其开发一组列表是值得的。

“事实不会因忽视它们而不存在。”——
Aldous Huxley

Advice
一定要避免攻击客户意见“太昂贵”或“不实际”这样的严厉谴责。这里建议通过协商形成一个大家均接受的列表。为了达到这个目标，必须保持一个开放的思想。

还要开发约束列表（如成本、规模大小、业务规则）和性能标准（如速度、精确度）。告诉与会者，这些列表不要求完备无缺，但要反映每个人对系统的理解。

SafeHome描述的对象可能包括：一个控制面板、若干烟感器、若干门窗传感器、若干动态检测器、一个警报器、一个事件（一个已被激活的传感器）、一个显示器、一台计算机、若干电话号码、一个电话等。服务列表可能包括：配置系统、设置警报器、监测传感器、电话拨号、控制面板编程以及读显示器（注意，服务作用于对象）。采用类似的方法，每个与会者都将开发约束列表（例如，当传感器不工作时系统必须能够识别，必须是用户友好的，必须能够和标准电话线直接连接）和性能标准列表（例如，一个传感器事件应在一秒内被识别，应实施事件优先级方案）。

这些对象列表可以用大的纸张钉在房间的墙上或用便签纸贴在墙上或写在墙板上。或者，列表也可以贴在内部网站的电子公告牌上或聊天室中，便于会议前的评审。理想情况下，应该能够分别操作每个列表项，以便于合并列表、删除项以及加入新项。在本阶段，严禁批评和争论。

当某一专题的各个列表被提出后，小组将生成一个组合列表。该组合列表将删除冗余项，并加入在讨论过程中出现的一些新的想法，但是不删除任何东西。在所有话题域的组合列表都生成后，由调解人主持开始讨论协调。组合列表可能会缩短、加长或重新措词，以求更恰当地反映即将开发的产品或系统。目标是为所开发系统的对象、服务、约束和性能提交一个意见一致的列表。

在很多情况下，列表所描述的对象或服务需要更多的解释。为了完成这一任务，利益相关者为列表中的条目编写小规格说明（mini-specification）^①。每个小规格说明是对包含在列表中的对象和服务的精练，例如，对SafeHome对象控制面板的小规格说明如下：

控制面板是一个安装在墙上的装置，尺寸大概是9×5英寸；控制面板和传感器、计算机之间是无线连接；通过一个12键的键盘与用户交互，通过一个3×3的LCD彩色显示器为用户提供反馈信息；软件将提供交互提示、回显以及类似的功能。

将每个小规格说明提交给所有的与会者讨论，进行添加、删除和进一步细化等工作。在某些情况下，编写小规格说明可能会发现新的对象、服务、约束或性能需求，可以将这些新发现加入到原始列表中。在所有的讨论过程中，团队可能会提出某些在会议中不能解决的问题，将这些问题列表保留起来以便这些意见在以后的工作中发挥作用。

SAFEHOME

召开需求收集会议

[场景] 一间会议室，进行首次需求收集会议。

[人物] Jamie Lazar、Vinod Raman和Ed Robbins是软件团队成员；Doug Miller是软件工程师；三个市场营销人员；一个产品工程代表；一个会议主持人。

[对话]

主持人（指向白板）：这是目前住宅安全功能的对象和服务列表。

营销人员：从我们的观点看差不多覆盖了需求。

Vinod：没有人提到他们希望通过Internet访问所有的SafeHome功能吗？这应该包括住宅安全功能，不是吗？

① 除了创建小规格说明，很多软件团队选择创建用户场景，即所谓的用例（use-case）。这些在5.4节和第6章中有详细说明。

营销人员: 是的, 这很正确……我们必须加上这个功能以及合适的对象。

主持人: 这还需要加上一些限制吗?

Jamie: 肯定, 包括技术上的和法律上的。

产品代表: 什么意思?

Jamie: 我们必须确保外人不能非法侵入系统、使系统失效、抢劫甚至更糟。我们的责任非常重。

Doug: 非常正确。

营销人员: 但我们确实需要……只是保证能够制止外人接入。

Ed: 说比做起来容易, 而且……

主持人 (打断): 我现在不想讨论这个问题。我们把它作为动作项记录下来, 然后继续讨论 (Doug作为会议的记录者记下合适的内容)。

主持人: 我有种感觉, 这儿仍存在很多需要考虑的问题。

(小组接下来花费20分钟提炼并扩展住宅安全功能的细节。)

5.3.2 质量功能部署

KEY POINT

QFD以最大限度地满足客户的方式来定义需求。

ADVICE

每个人都希望实现大量的“令人兴奋的需求”, 但是必须小心, 那是“需求蔓延”开始的原因。然而另一方面, 经常是令人兴奋的需求才最终导致突破性的产品!

WebRef

有关QFD的更多信息可从以下地址获得:
www.qfdi.org。

质量功能部署 (Quality Function Deployment, QFD) 是一种将客户要求转化成软件技术需求的质量管理技术。QFD “目的是最大限度地让客户从软件工程过程中感到满意[Zul92]”。为了达到这个目标, QFD强调整理 “什么是对客户有价值的”, 然后在整个工程活动中部署这些价值。QFD确认了三类需求[Zul92]:

正常需求: 这些需求反映了在和客户开会时确定的针对某产品或系统的目标。如果实现了这些需求, 将满足客户。这方面的例子如: 所要求的图形显示类型、特定的系统功能以及已定义的性能级别。

期望需求: 这些需求隐含在产品或系统中, 并且可能是非常基础的以至于客户没有显式地说明, 但是缺少这些将导致客户非常不满。这方面的例子如: 人机交互的容易性、整体的运行正确性和可靠性以及软件安装的简易性。

令人兴奋的需求: 这些需求反映了客户期望之外的特点, 但是如果实现这些特点的话将会使客户非常满意。例如, 新移动电话的软件来自标准特性, 但关联了一些超出期望的能力 (例如多重触控技术的触摸屏, 可视语音邮箱), 这些能力让产品的用户很欣喜。

虽然QFD概念可应用于整个软件过程[Par96], 但是特定的QFD技术可应用于需求提取活动。QFD通过客户访谈和观察、调查以及检查历史数据 (如问题报告) 为需求收集活动获取原始数据。然后把这些数据翻译成需求表——称为客户意见表, 并由客户和利益相关者评审。接下来使用各种图表、矩阵和评估方法抽取期望的需求并尽可能导出令人兴奋的需求[Aka04]。

5.3.3 用户场景

当收集需求时, 系统功能和特性的整体愿景开始具体化。但是在软件团队弄清楚不同类型的最终用户如何使用这些功能和特性之前, 很难转移到更技术化的软件工程活动。为实现这一点, 开发人员和用户可以创建一系列的场景—场景可以识别对将要构建系统的使用线索。场景通常称为用例[Jac92], 它提供了将如何使用系统的描述。在5.4节中将更详细地讨论用例。

开发一个初步的用户场景

[场景] 一间会议室，继续首次需求收集会议。

[人物] Jamie Lazar、Vinod Raman和Ed Robbins是软件团队成员；Doug Miller是软件工程师；三个市场营销人员；一个产品工程代表；一个会议主持人。

[对话]

主持人：我们已经讨论过通过Internet访问SafeHome功能的安全性，我想考虑得再多些。下面我们开发一个使用住宅安全功能的用户场景。

Jamie：怎么做？

主持人：我们可以采用两种不同的方法完成这个工作，但是现在，我想不要太正式吧。请问（他指向一个市场人员），你设想该如何使用这样的系统。

营销人员：嗯……好的，如果我出门在外，我想我能做的就是必须让某个没有安全码的人比如管家或修理工进入我家。

主持人（微笑）：这就是你的原因……告诉我们实际上你怎么做？

营销人员：嗯……首先是我需要一台电脑，登录为所有SafeHome用户提供维护的Web网站，提供我的用户身份证号……

Vinod（打断）：Web页面必须是安全的、加密的，以确保我们安全……

主持人（打断）：这是个有用信息，Vinod，但这太技术性了，我们还是只关注最终用户将如何使用该功能，好吗？

Vinod：没问题。

营销人员：那么，就像我所说的，我会登录一个Web网站并提供我的用户身份证号和两级密码。

Jamie：如果我忘记密码怎么办？

主持人（打断）：好想法，Jamie。但是我们先不谈这个。我们先把这种情况作为“异常”记录下来。我确信还有其他的异常。

营销人员：在我输入密码后，屏幕将显示所有的SafeHome功能。我选择住宅安全功能，系统可能会要求我确认我是谁，要求我的地址或电话号码或其他什么，然后显示一张图片，包括安全系统控制面板和我能执行的功能列表——安装系统、解除系统、解除一个或多个传感器。我猜还可能会允许我重新配置安全区域和其他类似的东西，但是我不确定。

（当市场营销人员继续讨论时，Doug记录下大量内容。这些构成了最初非正式的用例场景基础。另一种方法是让市场营销人员写下场景，但这应该在会议之外进行。）

5.3.4 导出工作产品

什么样的信息是需求收集产生的？

根据将要构建的系统或产品的规模不同，需求导出后产生的工作产品也不同。对于大多数系统而言，工作产品包括：

- 要求和可行性陈述。
- 系统或产品范围的界限说明。
- 参与需求导出的客户、用户和其他利益相关者的名单。
- 系统技术环境的说明。
- 需求列表（最好按照功能加以组织）以及每个需求适用的领域限制。
- 一系列使用场景，有助于深入了解系统或产品在不同运行环境下的使用。

- 任何能够更好地定义需求的原型。

所有参与需求导出的人员需要评审以上的每一个工作产品。

5.4 开发用例

在一本讨论如何编写有效用例的书中，Alistair Cockburn[Coc01b]写道：“一个用例捕获一个合同……即说明当对某个利益相关者的请求响应时，在各种条件下系统的行为”。本质上，用例讲述了能表达主体场景的故事：最终用户（扮演多种可能角色中的一个）如何在一定环境下和系统交互。这个故事可以是叙述性的文本、任务或交互的概要、基于模板的说明或图形表示。不管其形式如何，用例从最终用户的角度描述了软件或系统。

KEY POINT

从参与者的角度定义用例。参与者是人员（用户）或设备在和软件交互时所扮演的角色。

WebRef

可以从以下地址下载一篇非常好的关于用例的论文：www.ibm.com/developerworks/webservices/library/codesign7.html

为了开发有效的用例，我需要知道什么？

撰写用例的第一步是确定故事中所包含的“参与者”。参与者是在将要说明的功能和行为环境内使用系统或产品的各类人员（或设备）。参与者代表了系统运行时人（或设备）所扮演的角色，更为正式的定义就是：参与者是任何与系统或产品通信的事物，且对系统本身而言参与者是外部的。当使用系统时，每个参与者都有一个或多个目标。

要注意的是，参与者和最终用户并非一回事。典型的用户可能在使用系统时扮演了许多不同的角色，而参与者表示了一类外部实体（经常是人员，但并不总是如此），在用例中他们仅扮演一种角色。例如，考虑一个机床操作员（一个用户），他和生产车间（其中装有许多机器人和数控机床）内的某个控制计算机交互。在仔细考察需求后，控制计算机软件需要4种不同的交互模式（角色）：编程模式、测试模式、监测模式和故障检查模式。因此，4个参与者可定义为：程序员、测试员、监控员和故障检修员。有些情况下，机床操作员可以扮演所有这些角色，而有些情况下，每个参与者的角色可能由不同的人员扮演。

因为需求导出是一个逐步演化的活动，因此在第一次迭代中并不能确认所有的参与者。在第一次迭代中有可能识别主要的参与者[Jac92]，而对系统了解更多之后，才能识别出次要的参与者。主要参与者直接且经常使用软件是要获取所需的系统功能并从系统得到预期收益。次要参与者支持系统，以便主要参与者能够完成他们的工作。

一旦确认了参与者，就可以开发用例。对于应该由用例回答的问题，Jacobson[Jac92]提出了以下建议^①：

- 谁是主要参与者、次要参与者？
- 参与者的目标是什么？
- 故事开始前有什么前提条件？
- 参与者完成的主要工作或功能是什么？
- 按照故事所描述的还可能需要考虑什么异常？
- 参与者的交互中有什么可能的变化？
- 参与者将获得、产生或改变哪些系统信息？
- 参与者必须通知系统有关外部环境的改变吗？
- 参与者希望从系统获取什么信息？

① Jacobson的问题已经被扩展到为用例场景提供更复杂的视图。

- 参与者希望得知会有意料之外的变更吗？

回顾基本的SafeHome需求，我们定义了4个参与者：房主（用户）、配置管理人员（很可能就是房主，但扮演不同的角色）、传感器（附属于系统的设备）和监测子系统（监测SafeHome房间安全功能的中央站）。仅从该例子的目的来看，我们只考虑了房主这个参与者。房主通过使用报警控制面板或计算机等多种方式和住宅安全功能交互：

- 输入密码以便能进行其他交互操作。
- 查询安全区的状态。
- 查询传感器的状态。
- 在紧急情况时按下应急按钮。
- 激活或关闭安全系统。

考虑房主使用控制面板的情况，系统激活的基本用例如下^①：

1. 房主观察SafeHome控制面板（图5-1），以确定系统是否已准备好接收输入。如果系统尚未就绪，“not ready”消息将显示在LCD显示器上，房主必须亲自动手关闭窗户或门以使得“not ready”消息消失。（not ready消息暗示某个传感器是开着的，即某个门或窗户是开着的。）
2. 房主使用键盘键入4位密码，该密码和系统中存储的有效密码相比较。如果密码不正确，控制面板将鸣叫一声并自动复位以等待再次输入。如果密码正确，控制面板等待进一步的操作。
3. 房主选择键入“stay”或“away”（见图5-1）启动系统。“stay”只激活外部传感器（内部的运行监测传感器是关闭的）。“away”激活所有的传感器。
4. 当激活时，房主可以看到一个红色的警报灯。

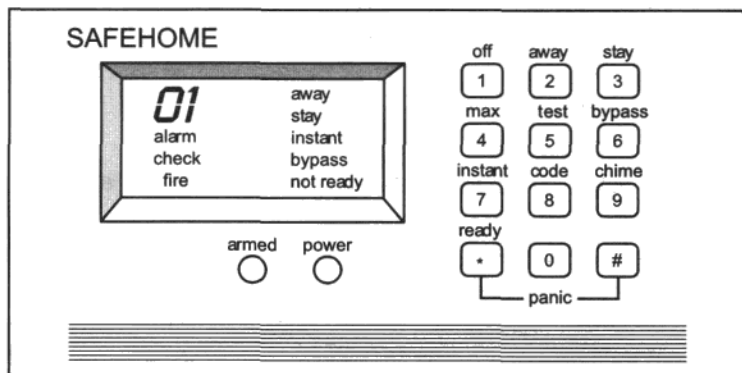


图5-1 SafeHome 控制面板



用例通常写得不规范，但是使用这里介绍的模板可以确保你已经说明了所有关键问题。

基本的用例从较高层次上给出参与者和系统之间交互的故事。

在很多情况下，进一步细化用例以便为交互提供更详细的说明。例如，Cockburn[Coc01]建议使用如下模板详细说明用例：

用例：初始化监测

主要参与者：房主

目标：设置系统在房主离开住宅或留在房间内时监测传感器。

前提条件：系统已经输入密码并识别各种传感器。

^① 注意，该用例和通过Internet访问系统的情形不同，该用例的环境是通过控制面板交互而不是使用计算机所提供的图形用户接口GUI。

触发器: 房主决定“设置”系统,即打开警报功能。

场景:

1. 房主: 观察控制面板。
2. 房主: 输入密码。
3. 房主: 选择“stay”或“away”。
4. 房主: 观察红色报警灯显示SafeHome已经被打开。

异常:

1. 控制面板没有准备就绪: 房主检查所有的传感器,确定哪些是开着的(即门窗是开着的),并将其关闭。

2. 密码不正确(控制面板鸣叫一声): 房主重新输入正确的密码。
3. 密码不识别: 必须对监测和响应子系统重新设置密码。
4. 选择stay: 控制面板鸣叫两声而且stay灯点亮;激活边界传感器。
5. 选择away: 控制面板鸣叫三声并且away灯点亮;激活所有传感器。

优先级: 必须实现。

何时可用: 第一个增量。

使用频率: 每天多次。

使用方式: 通过控制面板接口。

次要参与者: 技术支持人员, 传感器。

次要参与者使用方式:

技术支持人员: 电话线。

传感器: 有线或无线接口。

未解决的问题:

1. 是否还应该有不使用密码或使用缩略密码激活系统的方式?
2. 控制面板是否还应显示附加的文字信息?
3. 房主输入密码时,从按下第一个按键开始必须在多长时间内输入密码?
4. 在系统真正激活之前有没有办法关闭系统?

可以使用类似的方法开发其他房主的交互用例。重要的是必须认真评审每个用例。如果某些交互元素模糊不清,用例评审将解决这些问题。

SAFEHOME

开发高级的用例图

[场景] 会议室,继续需求收集会议。

[人物] Jamie Lazar、Vinod Raman和Ed Robbins是软件团队成员; Doug Miller是软件工程师;三个市场营销人员;一个产品工程代表;一个会议主持人。

[对话]

主持人: 我们已经花费了相当多的时间讨论SafeHome住宅安全功能。在休息时间我画了一个用例图,用它来概括重要的场景,这些场景是该功能的一部分。大家看一下。

(所有的与会者注视图5-2。)

Jamie: 我恰好刚开始学习UML符号^①。住宅安全功能是由中间包含若干椭圆的大方框表示吗?而且这些椭圆代表我们已经用文字写下的用例,对吗?

^① 对于不熟悉UML符号的读者请参考附录1介绍UML的基本指南。

主持人：是这样的。而且棍型小人代表参与者——和系统交互的人或东西，如同用例中所描述的……哦，我使用作了标记的矩形表示在这个用例中那些不是人而是传感器的参与者。

Doug：这在UML中合法吗？

主持人：合法性不是问题，重点是交流信息。我认为使用类似人的棍型小人代表设备可能会产生误导，因此我做了一些改变。我认为这不会产生什么问题。

Vinod：好的。这样我们就为每个椭圆进行了用例说明，还需要生成更详细的基于模板的说明吗？我们已经阅读过那些说明了。

主持人：有可能，但这可以等到考虑完其他的SafeHome功能之后。

营销人员：等一下，我已经看过这幅图，突然间我意识到我们遗漏了什么。

主持人：哦，是吗。告诉我们遗漏了什么。

（会议继续进行。）

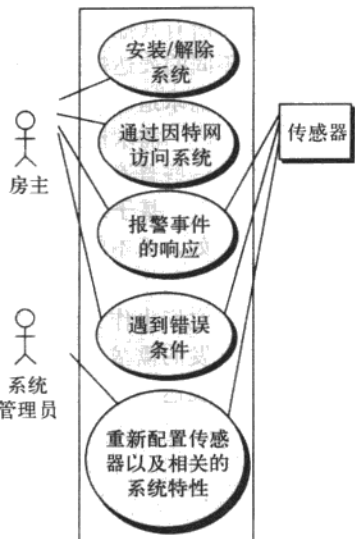


图5-2 住宅安全系统功能的UML用例图

SOFTWARE TOOLS

用例开发

目的：通过使用能提高访问透明性和一致性的自动化模板和机制，帮助开发用例。

机制：工具的原理各不相同。通常，用例工具为创建有效用例提供填空式的模板。大多数用例功能能嵌入到一系列更宽泛的需求工程功能中。

代表性工具：⊖

大量的分析建模工具（多数基于UML）：为用例开发和建模提供文字和图形化支持。

Objects by Design: (www.objectsbydesign.com/tools/umltools_byCompany.html) 提供对该类工具的全面链接。

5.5 构建需求模型[⊖]

分析模型的目的是为基于计算机的系统提供必要的信息、功能和行为域的说明。随着软件工程师更多地了解将要实现的系统以及其他利益相关者更多地了解他们到底需要什么，模型将动态的变更。因此，分析模型是任意给定时刻的需求快照，我们对这种变更更应有思想准备。

当需求模型演化时，某些元素将变得相对稳定，为后续设计任务提供稳固的基础。但是，有些模型元素可能是不稳定的，这表明利益相关者仍然没有完全理解系统的需求。分析模型及其构建方法将在第6、7章详细说明，下面小节仅提供简要的概述。

⊖ 这里提到的工具只是此类工具的例子，并不代表本书支持采用这些工具。在大多数情况下，工具名称被各自的开发者注册为商标。

⊖ 本书通篇把分析模型和需求模型作为同义词使用。它们都是用于描述信息、功能和行为领域问题的需求。

5.5.1 需求模型的元素

有很多不同的方法考察计算机系统的需求。某些软件人员坚持最好选择一个表达模式（例如用例）并排斥所有其他的模式。有些专业人士则相信使用许多不同的表现模式来描述需求模型是值得的，不同的表达模式促使软件团队从不同的角度考虑需求——一种方法更有可能造成需求遗漏、不一致性和歧义性。



把利益相关者包括进来通常是个好主意。做到这一点最好的方法之一是让每个利益相关者写下描述将如何使用软件的用户例。

需求模型的特定元素取决于将要使用的分析建模方法（第6、7章）。但是，一些普遍的元素对大多数分析模型来说都是通用的。

基于场景的元素：使用基于场景的方法可以从用户的视角描述系统。例如，基本的用例（5.4节）及其相应的用例图（图5-2）演化成更精细的基于模板的用例。需求模型基于场景的元素通常是正在开发模型的第一部分。同样，它们也作为创建其他建模元素时的输入。例如，图5-3描绘了一张使用用例引发的需求并表达它们的UML活动图^①。给出了最终基于场景表示的三层详细表达。

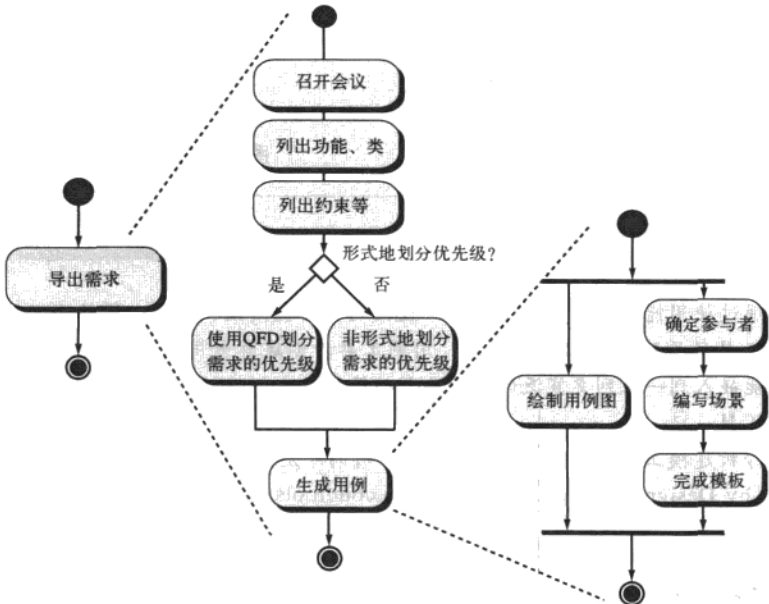


图5-3 导出需求的UML活动图



一种分离类的方法是查找用例脚本中的叙述性名词。至少某些名词将是候选类。第8章中将有更多内容说明这一点。

基于类的元素：每个使用场景都暗示着当一个参与者和系统交互时所操作的一组对象，这些对象被分成类——具有相似属性和共同行为的事物集合。例如，可以用UML类图描绘SafeHome安全功能的传感器Sensor类如图5-4所示。注意，UML类图列出了传感器的属性（如name, type）和可以用于修改这些属性的操作（例如identify、enable）。除了类图，其他分析建模元素描绘了类

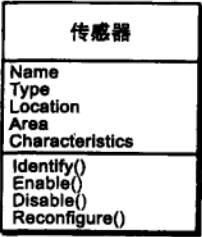


图5-4 传感器Sensor的类图

① 对于不熟悉UML符号的读者请参考附录1介绍UML的简要指南。

KEY POINT

状态是外部可观察到的行为模式，外部激励导致状态间的转换。

相互之间的协作以及类之间的关联和交互。在第7章中将有更详细的讨论。

行为元素：基于计算机系统的行为能够对所选择的设计和所采用的实现方法产生深远的影响。因此，需求分析模型必须提供描述行为的建模元素。

状态图是一种表现系统行为的方法，该方法描绘系统状态以及导致系统改变状态的事件。状态是任何可以观察到的行为模式。另外，状态图还指明了在某个特殊事件后采取什么动作（例如激活处理）。

为了更好地说明状态图的使用，考虑将软件嵌入到SafeHome的控制面板负责读取用户的输入信息。简化的UML状态图如图5-5。

另外作为一个整体的系统行为表达也能够建模于各个类的行为之上。第7章将有更多关于行为建模的讨论。

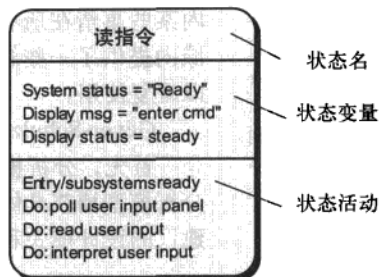


图5-5 UML状态图表示

SAFEHOME

初步的行为建模

[场景] 会议室，继续需求会议。

[人物] Jamie Lazar、Vinod Raman和Ed Robbins是软件团队成员；Doug Miller是软件工程师；三个市场营销人员；一个产品工程代表；一个会议主持人。

[对话]

主持人：我们刚才差不多已经讨论完SafeHome住宅安全功能。但是在结束之前，我希望讨论一下功能的行为。

营销人员：我不太理解你所说的行为意味着什么。

Ed（大笑）：那就是如果产品行为错误就让它“暂停”。

主持人：不太准确，让我解释一下。

（主持人向需求收集团队解释行为建模的基本知识。）

营销人员：这看起来有点技术性，我不敢确定能不能在这里帮上忙。

主持人：你当然可以。你从用户的角度观察到什么行为？

营销人员：嗯……好的，系统将监测传感器，将从房主那里读指令，将显示其状态。

主持人：看到了吧，你是可以帮上忙的。

Jamie：还应该使用计算机确定是否有任何输入，例如基于Internet的访问或配置信息。

Vinod：是的，实际上，配置系统是其权利内的一个状态。

Doug：你这家伙开始转过弯儿了，让我们多想一些……有方法把这个画出来吗？

主持人：有方法，但让我们等到会后再开始吧。

面向数据流的元素：信息在基于计算机的系统中流动时会被转换，系统接受多种形式的输入，使用函数将其转换；生成多种形式的输出。输入可以是由变频器发送的控制信号，可以是操作人员输入的数字，也可以是网络上传送的信息包或从备份存储器取回的庞大数据文件。转换可能由单一的逻辑比较、复杂的数值算法或专家系统的规则推理方法构成。输出可能是点亮一个发光二极管或是生成一份200页的报告。实际上，我们可以为任何基于计算机的系统（不管其规模大小和复杂度）创建数据流模型。第7章将更详细地讨论流的建模。

5.5.2 分析模式

“折衷是这样一种艺术：采用某种方式分蛋糕，让每个人感觉自己得到了最大的一块。”——
Ludwig Erhard

任何有一些软件项目需求工程经验的人都开始注意到，在特定的应用领域内某些事情在所有的项目中重复发生^①。这些分析模式[Fow97]在特定应用领域内提供了一些解决方案（如类、功能、行为），在为许多应用项目建模时可以重复使用。

Geyer-Schulz和Hahsler[Gey01]提出了使用分析模式的两个优点：

首先，分析模式提高了抽象分析模型的开发速度，通过提供可重复使用的分析模型捕获具体问题的主要需求，例如关于优点和约束的说明。其次，通过建议的设计模式和可靠的通用问题解决方案，分析模式有利于把分析模型转变到设计模型。

通过参照模式名把分析模式整合到分析模型中。同时还将存储在仓库中以便需求工程师能通过搜索工具发现并应用。在标准模板[Gey01]^②中会提供关于分析模式（和其他类型模式）的信息，更多细节在第12章讨论。分析模式的样例和有关这一论题更多的讨论在第7章中。

5.6 协商需求

在一个理想的需求工程情境中，起始、导出和精工工作能确保得到足够详细的客户需求，以开展后续的软件工程步骤。但遗憾的是，这几乎不可能发生。实际中，一个或多个利益相关者恐怕得进入到协商的过程中，在多数情况下要让利益相关者以成本和产品投放市场的时间为背景，平衡功能、性能和其他的产品或系统特性。这个协调过程的目的是保证所开发的项目计划，在满足利益相关者要求的同时反映软件团队所处真实世界的限制（如时间、人员、预算）。

WebRef

可以从以下地址下载一篇关于软件需求协商的简短文章：www.alexanderegied.com/publications/Software_Requirements_Negotiation_Some_Lessons_Learned.html。

最好的协商是争取“双赢”的结果^③，即利益相关者的“赢”在于获得满足客户大多数需要的系统或产品，而作为软件团队一员的“赢”在于按照实际情况、在可实现的预算和时间期限内完成工作。

Boehm[Boe98]定义了每个软件过程迭代启动时的一系列协商活动。不是定义单一的客户交流活动，而是定义了如下系列活动：

1. 识别系统或子系统关键的利益相关者。
2. 确认利益相关者“赢”的条件。
3. 就利益相关者“赢”的条件进行协商，以便使其与所有涉及人（包括软件团队）的一些双赢条件一致。

这些初期步骤的成功实施可以达到双赢的结果，这是继续开展后续软件工程施工活动的关键。

INFO

协商的艺术

学习如何有效地协商可以帮助你更好地度过人生或技术生涯。如下指导原则非常值得考虑：

1. 认识到这不是竞争。为了成功，为了获得双赢，双方将不得不妥协。

① 在某些情况下，事情重复发生而不管应用领域是什么。例如，不管所考虑的应用领域是什么，用户接口的特点和功能都是共有的。

② 文献中已经提出了各种各样的模式模板，感兴趣的读者可以参阅[Fow97]、[Bus07]、[Yac03]和[Gam95]。

③ 有许多关于谈判技巧的书籍（如[Lew06]、[Rai06]、[Fis06]）。这是一个应该学习的许多重要技巧之一。读一本吧。

2. 制定策略。决定你希望得到什么；对方希望得到什么；你将如何行动以使得这两方面的希望都能实现。

3. 主动地听。不要当对方说了之后你才做出程式化的响应。听，是为了获取信息，这些信息有助于在磋商中更好地说明你的立场。

4. 关注对方的兴趣。如果想避开冲突，就不要太过于坚持自己的立场。

5. 不要进行人身攻击。应集中于需要解决的问题。

6. 要有创新性。当处于僵局时不要害怕而应考虑如何摆脱困境。

7. 随时准备作出承诺。一旦已经达成一致，不要闲聊胡扯，马上作出承诺然后继续进行。

SAFEHOME

开始协商

[场景] Lisa Perez的办公室，在第一次需求收集会议之后。

[人物] Doug Miller，软件工程师；Lisa Perez，市场营销经理。

[对话]

Lisa: 我听说第一次会议进行得很好。

Doug: 确实是这样，你派了几个有经验的人参加会议……他们确实有很多贡献了。

Lisa (微笑): 是的，他们的确告诉我他们融入了会议，而且会议卓有成效。

Doug (大笑): 下次再见面时我一定要脱帽致敬……看，Lisa，我想在你们主管所说的日期内获取所有的住宅安全功能可能会有问题。我知道现在还早，但是我们已经有一些落后于原定计划并且……

Lisa (皱眉): 我们必须在那个时间获得产品，Doug。你说的是什么功能？

Doug: 我认为我们可以在截止日期前完成所有的住宅安全功能，但是必须把Internet访问功能推迟到第二次发布的产品中加以考虑。

Lisa: Doug，Internet访问是SafeHome最引人注目之处，我们正在围绕这一点开发我们整个的营销活动。我们必须拥有它！

Doug: 我理解你的处境，我确实理解。问题在于为了向你提供Internet访问，我们将需要一整套Web站点安全防护措施，这将花费时间和人力。我们还必须在首次发布的产品中开发很多的附加功能……我认为我们不能在现有资源下完成这些。

Lisa (皱眉): 我知道，但你必须找到实现方法，Internet访问对住宅安全功能非常关键，对其他功能也很关键……其他功能可以等到下一次发布的产品再予考虑……我同意这样。

很明显Lisa和Doug陷入了僵局，而且他们必须协商出一个解决办法。他们能够“双赢”吗？如果你扮演调解人的角色，有什么建议？

5.7 确认需求

当需求模型的每个元素都已创建时，需要检查一致性、是否有遗漏以及歧义性。模型所表现的需求由利益相关者划分优先级并组合成一个整体，该需求整体将以软件逐步加以实现。需求模型的评审将提出如下问题：

- 每项需求都和系统或产品的整体目标一致吗？
- 所有的需求都已经在相应的抽象层上说明了吗？换句话说，是否有一些需求是在技术细节过多的层次上提出的，并不适合当前的阶段。

当我评审
需求时，
我将提出什么
问题？

- 需求是真正必需的，还是另外加上去的，有可能不是系统目标所必需的特性吗？
- 每项需求都有界定且无歧义吗？
- 每项需求都有归属吗？换句话说，是否每个需求都标记了来源（通常是一个明确的个人）？
- 有需求和其他需求相冲突吗？
- 在系统或产品所处的技术环境下每个需求都能够实现吗？
- 一旦实现后，每项需求是可测试的吗？
- 需求模型恰当地反映了将要构建系统的信息、功能和行为吗？
- 需求模型是否已经使用合适的方式“分割”，能够逐步地揭示详细的系统信息吗？
- 已经使用了需求模式简化需求模型吗？所有的模式都被恰当地确认了吗？所有的模式都和客户的需求一致吗？

应当提出以上这些问题和其他一些问题，并回答问题，以确保需求模型精确地反映利益相关者的要求并为设计奠定坚实的基础。

5.8 小结

需求工程的任务是为设计和构建活动建立一个可靠坚固的基础。需求工程发生在与客户沟通活动和为一般的软件过程定义的建模活动过程中。软件团队成员要实施7个不同的需求工程职能：起始、导出、精化、协商、规格说明、确认和管理。

在项目起始阶段，项目利益相关者建立基本的问题需求，定义最重要的项目约束以及陈述主要的特征和功能，必须让系统表现出这些特征和功能以满足其目标。该信息在导出阶段得到提炼和延伸，在此阶段中利用有主持人的会议、QFD和使用场景的开发进行需求收集活动。

精化阶段进一步把需求扩展为分析模型—基于场景、基于类、行为和面向数据流的模型元素的集合。模型可能对分析模式和在不同的应用系统中重复出现分析问题的解决方案加以注解。

当确定需求并且创建分析模型时，软件团队和其他项目利益相关者协商优先级、可用性和每条需求的相对成本。协商的目标是开发一个现实可行的项目计划。此外，将按照客户需求确认每个需求和整个需求模型，以确认将要构建的系统对于客户的要求是正确的。

习题与思考题

- 5.1 为什么大量的软件开发人员没有足够重视需求工程？以前有没有什么情况让你可以跳过需求工程？
- 5.2 你负责从一个客户处导出需求，而他告诉你太忙了没时间见面，这时你该怎么做？
- 5.3 讨论一下当需求必须从3、4个不同的客户中提取时会发生什么问题。
- 5.4 为什么我们说需求模型表现了系统的时间快照？
- 5.5 让我们设想你已经说服客户（你是一个绝好的销售人员）同意你作为一个开发人员所提出来的每一个要求，这能够让你成为一个高明的协商人员吗？为什么？
- 5.6 想出三个以上在需求起始阶段可能要问利益相关者的“与情景无关的问题”。
- 5.7 开发一个促进需求收集的“工具包”。工具包应包含一系列的需求收集会议指导原则，用于促进列表创建的材料以及其它任何可能有助于定义需求的条款。
- 5.8 你的指导老师将把班级分成4或6人的小组，组中一半的同学扮演市场部的角色，另一半将扮演软件工程部的角色。你的工作是定义本章所介绍的SafeHome安全功能的需求，并使用本章所提出的指

导原则引导需求收集会议。

- 5.9 为如下活动之一开发一个完整的用例：
 - a. 在ATM取款。
 - b. 在餐厅使用信用卡付费。
 - c. 使用一个在线经纪人账户购买股票。
 - d. 使用在线书店搜索书（某个指定主题）。
 - e. 你的指导老师指定的一个活动。
- 5.10 用例“异常”代表什么？
- 5.11 用你自己的话描述一个分析模式。
- 5.12 使用5.5.2节描述的模板，为下列应用领域建议1~2个分析模式：
 - a. 会计软件系统
 - b. E-Mail电子邮件系统
 - c. 互联网浏览器
 - d. 字符处理系统
 - e. 网站生成系统
 - f. 由指导老师特别指定的应用领域
- 5.13 在需求工程活动的谈判情境中，“双赢”意味着什么？
- 5.14 你认为当需求确认揭示了一个错误时将发生什么？谁将参与修正错误？

推荐读物与阅读信息

因为需求工程是成功创建任何复杂的基于计算机系统的关键，所以大量的书籍都在讨论需求工程。Hood和他的同事（《Requirements Management》，Springer, 2007）讨论了大量需求工程关于横跨系统硬件和软件工程的问题。Young（《The Requirements Engineering Handbook》和Artech House publishers, 2007）对需求工程任务进行了更深层次的讨论。Wiegers（《More About Software Requirements》和Microsoft Press, 2006）提供了很多需求收集和需求管理的技术实践。Hull和她的同事（《Requirements Engineering》，Springer-Verlag, 2002）、Bray（《An Introduction to Requirements Engineering》，Addison-Wesley, 2002）、Arlow（《Requirements Engineering》，Addison-Wesley, 2001）、Gilb（《Requirements Engineering》，Addison-Wesley, 2000）、Graham（《Requirements Engineering and Rapid Development》，Addison-Wesley, 1999）、Sommerville和Kotonya（《Requirement Engineering: Processes and Techniques》，Wiley, 1998）等，这些书都讨论了该主题。Gottesdiener（《Requirements by Collaboration: Workshops for Defining Needs》，Addison-Wesley, 2002）为项目利益相关人员协同收集需求环境提供非常有用的指导。

Lauesen（《Software Requirements: Styles and Techniques》，Addison-Wesley, 2002）全面概述了需求分析方法和表示方法。Weigers（《Software Requirements》，Microsoft Press, 1999）、Leffingwell及其同事（《Managing Software Requirements: A Unified Approach》，2nd ed., Addison-Wesley, 2003）提出了一系列有用的需求最佳实践，并为需求工程过程的很多方面提供了实用指导原则。

Withall（《Software Requirement Patterns》，Microsoft Press, 2007）描述了基于模式视图的需求工程。Ploesch（《Assertions, Scenarios and Prototypes》，Springer-Verlag, 2003）讨论了开发软件需求的先进技术。Windle和Abreo（《Software Requirements Using the Unified Process》，Prentice-Hall, 2002）从统一过程和UML符号的角度讨论了需求工程。Alexander和Steven（《Writing Better Requirements》，Addison-Wesley, 2002）提出了一套简短的指导原则，目的是指导编写清楚的需求，使用场景表现需求并评审最终结果。

用例建模通常在创建分析模型的所有其他方面时使用,讨论该主题的资料有Rosenberg和Stephens (《Use Case Driven Object Modeling with UML:Theory and Practice》,Apress, 2007)、Denny (《Succeed with Use Cases:Working Smart to Deliver Quality》,Addison-Wesley, 2005)、Alexander 和Maiden (《Scenarios,Stories,Use Cases:Through the Systems Development Life-Cycle》,Wiley, 2004)、Leffingwell 和他的同事 (《Managing Software Requirements: A Use Case Approach》,2nd ed.,Addison-Wesley, 2003) 表述了非常有用的需求收集的最佳实践。在Bittner和Spence (《Use-Case Modeling》,Addison-Wesley, 2002)、Cockburn[COC01]、Armour和Miller (《Advanced Use-Case Modeling: Software Systems》,Addison-Wesley, 2000)、Kulak和同事 (《Use Cases: Requirements in Context》,Addison-Wesley, 2000) 讨论中强调使用用例模型进行需求收集。

在Internet上有大量的、丰富的关于需求工程和分析的信息资源。与需求工程和需求分析相关的最新的Web引用列表可以在SEPA的Web站点<http://www.mhhe.com/engcs/compsci/pressman/professional/olc/ser.htm>上找到。