第7章.　基于用例/场景模型展开用户需求获取

7.1. 用户需求获取活动的展开

7.1.1.　展开用户需求获取活动时的注意事项

确定项目的前景与范围之后，需求工程就进入了后期阶段，就可以在前景与范围的指导下展开用户需求获取活动了。

用户需求的获取要时刻检查项目边界，在范围内的不要遗漏，在范围外的坚决排除，必要时维护项目边界。可以围绕系统边界计划获取活动，在面向对象方法中，以用例为显示逐一展开获取过程，在结构化方法中，以系统与外界的输入/输出流为线索逐一展开获取过程。除了以系统边界为线索之外，还可以辅之以业务需求、系统特性、目标模型、活动图等前景与范围阶段的工作成果，以更好地时刻把握系统边界。如果在用户需求获取中，发现前期阶段的前景与范围定义地不准确，可以在确认后修正项目的边界，并以新边界为线索展开用户需求获取活动。

用户需求获取的成功依赖于合适需求获取方法的选择与应用。需求工程前期阶段的需求获取也需要选择和应用合适的需求获取方法，但相对而言，需求工程后期阶段需要获取的内容更多、情况更复杂、要求更细致，所以需求工程后期阶段尤其需要正确选择与应用合适的需求获取方法。需求工程的前期阶段更为关键的是需求工程师的创造性——建立解决方案的创造力，它的需求获取多数情况下可以只依赖面谈一种方法完成。到了需求工程后期阶段，需求工程就要综合应用面谈、头脑风暴、原型、观察、硬数据抽取等多种方法才能有效地完成用户需求获取任务。在明确具体主题后，需求工程师要综合考虑涉众特征、环境特征、主题成熟度、主题稳定性等多种因素来最终确定应该使用的方法，具体细节请参见第8～10章。

要准备使用多次“获取→分析”的迭代过程最终完成用户需求获取。每个迭代可以进行多次具体的获取活动，每次获取都可以得到更多的信息，包括更完备的问题域信息、更具体的用户要求、更细化的解决方案细节等。每次获取后的分析是为了检查获取结果是否正确，以及指导下一个迭代中获取的内容与方向。

要及时地将每次获取的内容组织起来。传统上一直使用获取笔录组织获取内容，但获取笔录只是各种资料的简单堆积，是比较散乱的。1990s之后，人们更愿意使用用例/场景模型组织获取到的内容。用例/场景模型以用例/场景为基本单位，既能够以任务方式清晰、条理地展现各部分内聚的已获取内容，又能够为需求分析界定合理范围，还能够综合所有用例/场景结构化地展现整个项目范围下的用户需求获取进展情况。

7.1.2.　用户需求获取活动的主线索——用例/场景模型

用例/场景模型的作用如图7-1所示。在三个典型的需求层次上：

(1).　目标模型用于组织系统的目标、特性、任务等与业务需求相关的内容，目标分析过程是建立目标模型并验证其正确性、完备性、一致性的过程。

(2).　用例/场景模型用于组织用户需求的相关内容，用例/场景分析是建立用例/场景模型的过程，但用例/场景分析无法完成对用户需求相关内容正确性、完备性、一致性的验证。

(3).　面向对象分析模型或结构化分析模型用于描述软件解决方案的细节知识，组织和指导系统级需求的建立。面向对象分析或结构化分析是建立面向对象分析模型或结构化分析模型的过程，同时还能够验证用户需求相关内容的正确性、完备性和一致性。

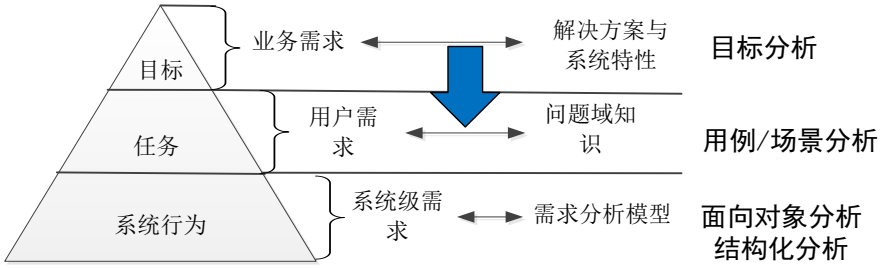


图7-1 用户需求需求活动展开示意图

比较三个层次可以发现，用例/场景模型不能说完全没有保证用户需求相关内容正确性、完备性、一致性的分析作用，但至少不是完成用户需求相关内容分析的主要手段。用例/场景分析更多地是组织用户需求相关内容，并将其组织结果提供给面向对象分析或结构化分析，让面向对象分析或结构化分析进行地更顺利和更有目的性。

总的来说，用例/场景模型能够及时地将每次需求获取活动的进展组织起来，展现、提供给分析活动，并在得到分析结果后进一步指导后续获取活动，所以，用例/场景模型在用户需求获取活动中有着主线索的作用[Maiden2005]。

7.2. 用例与场景

7.2.1. 什么是用例/场景

用例/场景虽然用例在前，场景在后，但事实上场景是更为基本的元素，用例只是一种特殊的场景，是需求工程师在组织需求时更喜欢使用的场景类型。

[Zorman1995]将场景定义为对系统和环境行为的局部描述。[Plihon1998]将场景定义为对行为或者事件序列的描述，序列中的行为和事件是系统需要完成的一个任务的特殊示例。[Jarke1996]认为场景包含有行为序列和行为发生的环境，环境描述了行为的主体、客体和上下文设置。实际上，以上的描述都不足以作为场景的准确定义，人们也很难给场景下一个非常准确的定义[Rolland1998]。可以明确的是场景具有重点描述真实世界的特征，它利用情景、行为者之间的交互、事件随时间的演化等方式来叙述性地描述系统的使用。从宽泛的意义上讲，示例、情景、上下文环境的叙述性描述、原型、序列图、脚本等都是场景常见的表现形式，如图7-2所示。

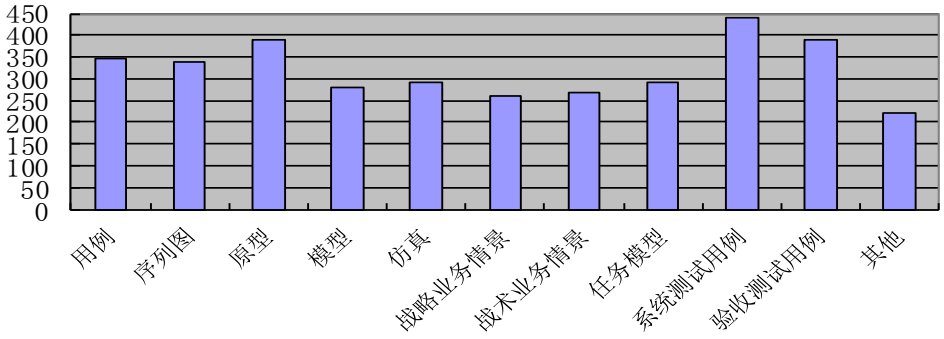


图7-2 场景类型及其使用程度分布，源自[Jarke1997]

用例是[Jacobson1992]最先在Objectory方法中提出的，用于描述电话通讯中的信息交换序列——对话过程。后来人们开始使用用例描述系统与外界交互的行为序列——软件功能的执行场景，并得到越来越多的关注与应用。统一建模语言UML也将用例和用例模型看作是整体中的一个重要组成部分，UML对用例/场景的定义成为人们事实上的使用标准。

共同前提要求：已插入银行卡并验证密码通过
共同结果要求：保持储户账户的数据一致性
场景1：顺利取款
1.储户选择取款任务
2.系统允许储户输入取款额度
3.储户输入取款额度
4.系统验证额度，通过后吐出现金
5.储户拿走现金
6.系统更新储户账户
场景2：额度不足，未能取款
1.储户选择取款任务
2.系统允许储户输入取款额度
3.储户输入取款额度
4.系统验证额度，额度高于账户可用余额，提示额度超支，不能取款
场景3：中途取消，未取款
1.储户选择取款任务
2.系统允许储户输入取款额度
3.储户请求取消取款任务
4.系统取消取款任务
场景4：现金未拿走异常，未能取款
1.储户选择取款任务
2.系统允许储户输入取款额度
3.储户输入取款额度
4.系统验证额度，通过后吐出现金
5. 1分钟后储户仍然没有拿走现金
6.系统收回现金，提示取款失败

用例：取款
前置条件：储户已通过登录验证并得到授权
后置条件：保持储户账户的数据一致性
正常流程：
1.储户选择取款任务
2.系统允许储户输入取款额度
3.储户输入取款额度
4.系统验证额度，通过后吐出现金
5.储户拿走现金
6.系统更新储户账户
异常流程：
3a.储户请求取消取款任务
1. 系统结束取款任务
4a.系统验证额度，额度高于账户可用余额，
1. 提示额度超支，不能取款
5a. 1分钟后储户仍然没有拿走现金
1.系统收回现金，提示取款失败

图7-3 用例与场景的关系示例

UML将用例定义为“在系统（或者子系统或者类）和外部对象的交互当中所执行的行为序列的描述，包括各种不同的序列和错误的序列，它们能够联合提供一种有价值的服务”[Rumbaugh2004]。[Cockburn2001]认为用例描述了在不同条件下系统对某一用户的请求所做出的响应。根据用户请求和请求时的系统条件，系统将执行不同的行为序列，每一个行为序列被称为一个场景。一个用例是多个场景的集合。

换句话说，如图7-3所示，每个用例是对相关场景集合——同一个目标下的多个场景的叙述性的文本描述，这些场景是用户和系统之间的交互行为序列，互有重合、互为补充共同实现用户的目的。更精确地说，一个用例承载了所有和用户某一个目标相关的成功和失败场景的集合。用例是一个理想的容器，以外部视图和描述系统可观察行为的方式记录系统的功能需求。

7.2.2. 用例/场景的组织特点

本质上，用例/场景是对用户需求及相关内容的组织[Wiegers2010]，如图7-4所示。图7-4的左右两侧分别是两种组织方式，左侧是用例/场景的组织方式，它将原本独立的多个需求组织成一个个故事，让用户、客户等应用领域中的涉众看起来更容易理解和接受。右侧是以各自独立的方式组织所有需求，每一条需求都独立于其他需求，这更符合开发者的视角，可以让开发者可以集中精力对付每一条需求而不受其他需求内容的干扰。

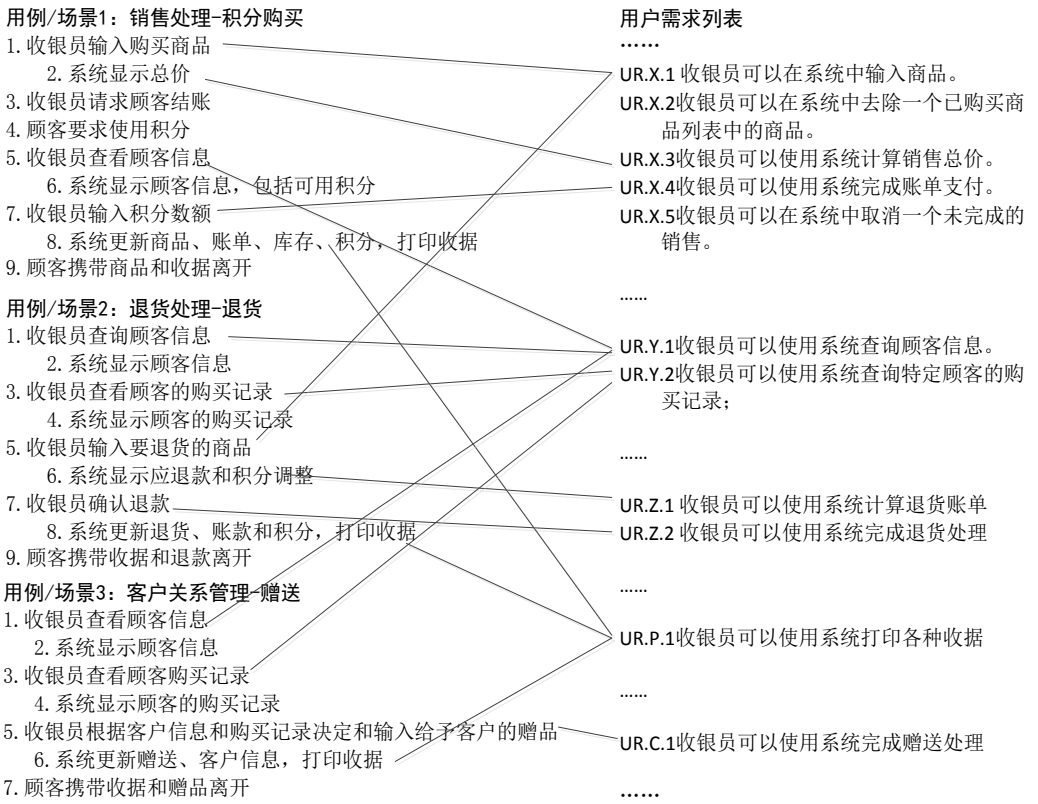


图7-4 用例/场景的组织作用示例

传统上，人们使用的是用户需求列表的方式，因为需求内容的组织工作本来就是开发者负责的，但这无疑会使得应用领域的涉众难以阅读，而且其分散性也不利于需求工程师执行获取、分析与验证任务。用例/场景的出现使得人们认识到还有一种更能为涉众所接受的需求组织方式，它不仅可以将需求组织为易于理解的故事，而且其内聚性还有助于需求工程师执行获取、分析与验证任务。

当然，用例/场景的组织方式虽然有助于需求理解，但不利于设计师、程序员、测试工程师等开发者的后续开发工作，因为开发工作希望分解复杂度而不是增加复杂度，喜欢独立处理各条需求而不是一次性满足

很多需求，尤其是不同用例/场景中会出现的重复部分是开发者最不喜欢的。所以，只要不是受到成本或进度限制，人们还是希望既建立需求的用例/场景组织方式以利于需求阶段的开发工作，又建立需求的列表组织方式以利于后续开发工作[Wiegers2010]。一个广泛的做法是在用户需求获取展开阶段建立用例/场景的组织方式，等到所有的获取、分析工作都结束之后，再为系统级需求建立列表方式并文档化传递给后续开发者。

如模版7-1所示，用例/场景不仅可以将多个独立的功能需求组织为故事，还能够以功能为中心，将涉众及目标、问题域知识（例如业务规则）、质量需求（特殊需求部分）、对外接口（特殊需求部分）、假设与依赖等众多的相关内容也组织在一起[Cockburn2001]。

ID:	用例的标识，通常会结合用例的层次结构使用X.Y.Z的方式
名称:	对用例内容的精确描述，体现了用例所描述的任务，通常是“动词＋名词”
用例属性	包括创建者、创建日期、更新历史等
参与者:	描述系统的主参与者、辅助参与者和每个参与者的目标
描述:	简要描述用例产生的原因，大概过程和输出结果
优先级:	用例所描述的需求的优先级
触发条件:	标识启动用例的事件，可能是系统外部的事件，也可能是系统内部的事件，还可能是正常流程的第一个步骤
前置条件:	用例能够正常启动和工作的系统状态条件
后置条件:	用例执行完成后的系统状态条件
正常流程:	在常见和符合预期的条件下，系统与外界的行为交互序列
分支流程:	用例中可能发生的非常见的其他合理场景（该段经常与异常流程合并为扩展流程）
异常流程:	在非预期的错误条件发生时，系统对外界进行响应的交互行为序列
相关用例:	记录和该用例存在关系的其他用例。
业务规则:	可能会影响用例执行的业务规则
特殊需求:	和用例相关的其他特殊需求，尤其是非功能性需求
假设:	在建立用例时所做的假设
待确定问题:	一些当前的用例描述还没有解决的问题

模版7-1用例描述格式

当然，模版7-1也仅仅是一个参考模版，具体的用例描述方式还需要参考用例的内容。如果用例是需求工程前期的一些抽象的描述，那么可能就仅仅是一段概括性的描述而已。如果用例的内容是专门针对特殊情况的，例如工作流、规则断言、异常流程等，那么可能就需要采用特殊的结构化文本方式[Hurlbut1997]，甚至可能会采用形式化文本的方式。

用例/场景以内聚的功能为中心组织各种知识，这取得了易于理解（各种知识内聚）的效果，也产生了弱点：①它只考虑其他内容与功能需求之间的联系，却无法描述其他内容相互之间的联系，例如质量需求的相互依赖、界面需求的跳转、对外接口需求与质量需求的联系……②它只考虑了存在联系的事实，却无法分

析联系的合理性，例如有无遗漏功能需求、数据需求及业务规则是否充分、质量需求是否可行……所以，虽然用例/场景的优点非常明显，但它毕竟只是一种组织形式，不能寄希望于单凭用例/场景模型解决所有问题[Gottesdiener2002]，目标模型、面向对象分析模型或结构化模型等其他的模型形式仍然是必要的。

7.2.3. 用例/场景的层次性

用例/场景是对需求的组织，需求是有层次性的，那么用例/场景自然也是有层次性的[Cockburn2001, Maiden2005]。

用例/场景可以用于组织业务需求内容，如图7-5所示，它的场景描述可以只是一段抽象的文字描述，也可以是对业务过程的描述。

ID	3	名称	商品库存管理	优先级	高
参与者及目标	（主参与者）总经理：库存分析，减少商品积压、缺货和报废 （辅助参与者）收银员：记录销售及退货中的商品出入库情况 （辅助参与者）业务经理：记录商品的批量入库与出库。				
用例描述	系统准确记录商品的入库、出库、销售及退货信息，并以此为基础掌握库存实时数据，分析和预测未来商品出库量，发现未来可能出现的积压、缺货和报废，提醒总经理进行处理。				
主流程	1.业务经理：商品入库 2.收银员：销售处理，售出的商品出库 3.总经理：库存分析，发现商品积压、缺货和报废				
分支流程	2a、收银员：退货处理，退回的商品入库 2b、业务经理：商品批量出库				

图7-5 抽象用例示例

用例/场景也可以用于组织用户需求的内容，如图7-6所示。用户需求级别的场景描述由用户需求连接而成，每个步骤都是一个用户任务。为了让故事连贯起来，场景描述中也经常会添加一些不属于用户需求的内容，例如两个涉众之间的外部互动。在扩展流程中，只要不明确指明后续的步骤或者结束，故障（错误、异常）扩展流程自动回到主流程中的扩展步骤，分支流程则自动回到主流程中扩展步骤的下一个步骤。

ID	1.1	名称	销售处理	优先级	高
参与者	收银员，目标是快速、正确地完成商品销售，尤其不要出现支付错误。				
触发条件	顾客携带商品到达销售点				
前置条件	收银员开始一个新的销售				
后置条件	准确完成支付过程，记录销售过程				
正常流程	1.收银员输入销售商品，系统记录并显示商品列表 2.收银员结束销售，系统计算和显示总价 3.收银员输入支付现金，系统计算并显示找零 4.收银员完成支付，系统记录销售信息，并打印收据				

扩展流程	1-2a收银员可以删除一个已经输入的商品 1.系统将该商品信息从记录中删除 1-3a收银员可以取消销售过程 1.系统放弃之前工作，结束销售处理
业务规则	总价=∑商品单价×数量 找零=支付数额-总价 商品时的条形码符合……
特殊需求	商品列表、总价、找零信息的显示要1米外可见。 输入商品可以使用键盘，也可以使用扫描仪。

图7-6 用户用例示例

用例/场景还可以用于组织系统级需求的内容，如图7-7所示。系统级需求级别的用例/场景描述由系统级需求连接而成，每个步骤都是一次外界与系统的交互。图7-7中主流程步骤5、扩展流程6a-1就是为了让故事连贯起来而添加的内容。1a、1-5a-1-1a是异常扩展流程，所以执行完扩展流程后分别回到原扩展步骤1、1-5a-1-。其他扩展流程都是分支流程，所以执行完扩展流程后回到各自扩展步骤的下一步步骤。

ID	1.1	名称	销售处理	优先级	高
参与者及目标	收银员，目标是快速、正确地完成商品销售，尤其不要出现支付错误。				
触发条件	顾客携带商品到达销售点				
前置条件	收银员开始一个新的销售。				
后置条件	存储销售记录，包括购买记录、商品清单和付款信息；更新库存；打印收据。				
正常流程	1.收银员输入商品标识 2.系统记录商品，并显示商品信息，商品信息包括商品标识、描述、数量、价格、特价（如果有商品特价策略的话）和本项商品总价 3.0.5秒后系统显示已购入的商品清单，商品清单包括商品标识、描述、数量、价格、特价、各项商品总价和所有商品总价 收银员重复1-3步，直到完成所有商品的输入 4.收银员结束输入，系统计算并显示总价 5.收银员请顾客支付账单 6.顾客支付，收银员输入收取的现金数额 7.系统给出应找的余额，收银员找零 8.系统记录销售信息、商品清单和账单信息，并更新库存，打印收据				

扩展流程	1a、非法标识： 1.系统提示错误并拒绝输入 1b、有多个具有相同商品类别的商品（如5把相同的雨伞） 1.收银员可以手工输入商品标识和数量 1-5a、顾客要求收银员从已输入的商品中去掉一个商品： 1.收银员输入商品标识并将其删除 1a、非法标识 1、系统显示错误并拒绝输入 2.返回正常流程第3步 1-5b、顾客要求收银员取消交易 1.系统放弃之前处理，结束销售任务 6a、顾客请求信用卡支付 1.收银员请求顾客使用信用卡付款器付款 2.信用卡付款成功后，收银员在系统中确认信用卡付款成功 3.转到主流程第8步 6b、顾客请求积分支付 1.收银员请求系统使用积分支付 2.系统使用积分支付方式，允许收银员输入会员编号 3.收银员输入会员编号 4.系统显示应付积分额和可用积分额 5.收银员确认使用积分付款 6.系统更新会员积分，付款成功。 7.转到主流程第8步 8a、如果顾客是VIP会员并且没有使用积分支付 1.系统记录销售信息、商品清单和账单信息，并更新库存，增加会员积分
业务规则	总价=∑商品单价×数量 找零=支付数额-总价 商品时的条形码符合…… 会员积分=现金账单/10
特殊需求	1、系统显示的信息要在1米之外能看清 2、输入商品可以使用键盘，也可以使用扫描仪。扫描仪的接口是…… 3、如果在一个销售任务在第8步更新数据过程中发生机器故障，系统的数据要能够恢复到该销售任务之前的状态

图7-7 系统用例示例

业务需求、用户需求、系统级需求只是需求的三个典型层次而已，在实际复杂系统中可能还存在其他的

需求层次(复杂系统中 ,目标可以像目标模型那样再分层 ,任务也可以按照工作分解结构(Work Breakdown Structure，简称WBS) 再分层) ，所以用例/场景描述的内容详略程度会有很大的差异性，一般在需求工程的早期阶段建立最为概要的用例/场景描述，在需求工程的中期阶段 (需求获取中...) 建立用户任务层次的用例/场景描述，在需求工程的后期阶段 (需求分析后) 建立系统交互层次的用例/场景描述。

7.2.4. 基于用例/场景进行软件开发

用例/场景是需求的组织 ,所以只要需求起作用的地方用例/场景就可以起作用。需求能够驱动项目管理、设计、构造、集成、测试、维护等后续的软件开发活动，用例/场景也能驱动它们，所以用例驱动的软件开发[Jacobson2004]某种程度上就是需求驱动的软件开发。

除了能够组织需求之后，用例/场景也可以用来组织其他内容，例如软件设计中的对象协作过程、软件构造中的算法执行过程、软件测试中的测试用例执行过程……理论上说，只要是具有叙述性、行为序列等特点的内容都可以用用例/场景来组织。

所以，虽然用例/场景较多地用在需求工程，尤其是用户需求获取阶段，但其他的软件开发阶段也会广泛使用用例/场景，如图7-8所示。“场景在实践 (尤其是需求工程实践) 中的应用情况比学术界所能想象的要丰富得多[Weidenhaupt1998]”。

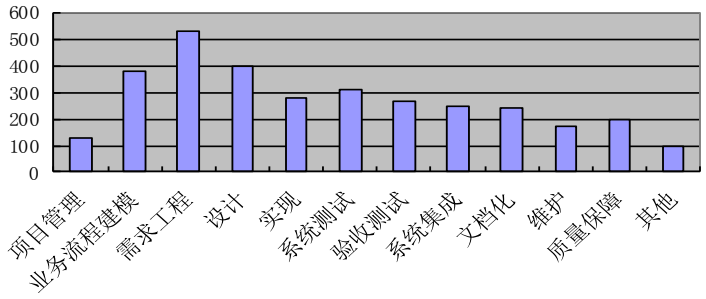


图7-8 场景在软件工程不同阶段的应用分布，源自[Jarke1997]

本章所要描述和强调的是在用户需求获取阶段使用的用例/场景，下面 (7.3) 将详细分析和限定该阶段用例/场景的特征，以保证能够基于用例/场景模型成功完成用户需求获取活动。

7.3. 用例/场景模型

用例/场景更多的只是需求内容的组织方式，不是基于形式化理论的分析技术 (虽然有不少方法试图为其建立形式化基础，但毕竟形式化的用例/场景会影响它被涉众理解的能力，所以没有在实践中得到广泛认同) ，所以用例/场景一直没有形成严谨、准确的语法、语义和语用体系，只有一些实践中总结出来的原则与经验。

本书不准备也不可能给出用例/场景模型的形式化体系，只是要汇总实践中总结出来的原则与经验，以更好地定位、理解和使用用户需求获取展开活动中的用例/场景。

7.3.1. 场景的定位

实践中，场景的使用差异性表现在很多方面[Jarke1997, Arnold1998, Weidenhaupt1998]，要更好地定位场景的特征就需要理解所有这些差异方面[Muller1999]。[Rolland1998]将实践调查中发现的差异性总结为如图7-9所示的场景分类框架，很好地描述了场景的差异特征。

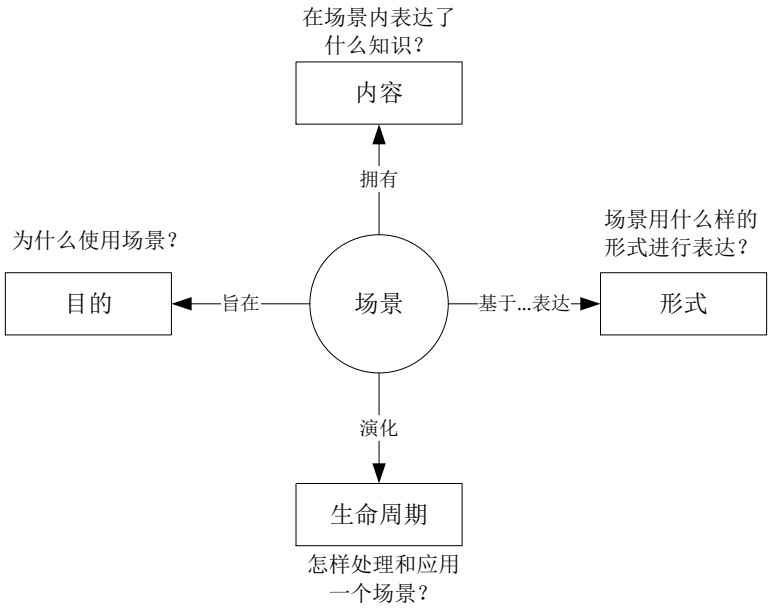


图7-9 场景的四个方面特征

在[Rolland1998]的分类框架中，场景在形式、目的、内容和生命周期四个方面都有差异。

1.形式

场景的形式是指场景的表达模式。人们使用多种方案来描述场景，每种方案或多或少的会涉及一些正式定义的表示法。而且，场景被表现出来的方式也是不一样的，有静态的图片和文本，也有能够支持用户动态交互的动态展示。

在场景的形式上，又分为两个方面：

（1）描述（Description）

这个方面的第一个要点是描述场景所使用的表示法的正规性，分别可能为非形式化语言（完全自由，没有任何规则）、半形式化语言（有一定的规则但不严谨）和形式化语言（有形式化体系，有完备的语法、语义和语用）。第二个要点是描述场景时所使用的媒介形式（Medium）。在实践中有下面这些常见的媒介形式（如图7-10 所示）：叙述性的自由文本、结构化文本、强限制文本、表格、图表、图像等。

在用户需求获取中，建议使用表格、结构化文本、模版等半形式化语言。

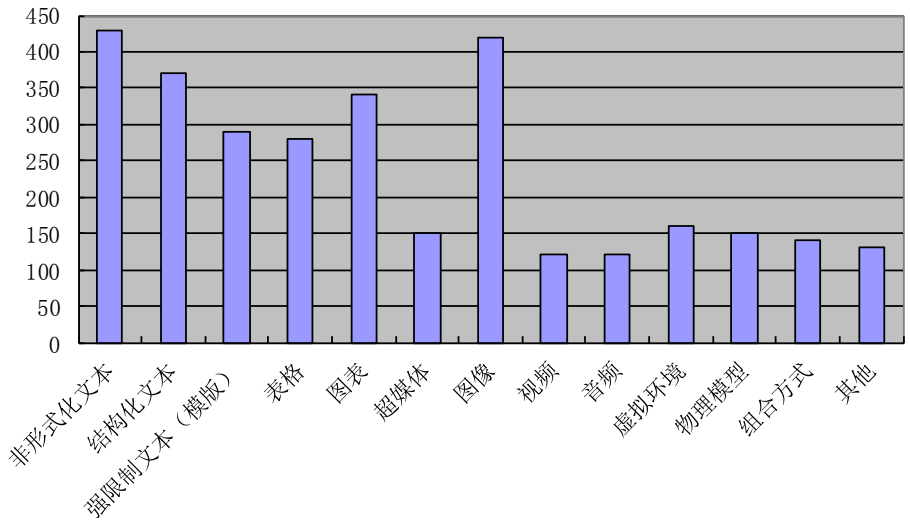


图7-10 场景的不同媒介形式在实践中的应用程度，源自[Jarke1997]

（2）外观（Presentation）

外观是指场景被表达出来时的效果，主要有静态、动态和交互三种类型。静态外观的场景被展现为一个或者数个描述性的文本或者图片。动态外观的场景会被以动态的方式展现出来，读者可能会要求按时序向前或者向后浏览场景，也可能会要求跳转到场景的某一个时刻进行观察。交互外观的场景提供交互性，它允许用户在一定程度上控制和改变场景的变化时序或者效果。

在用户需求获取中，建议以静态的场景外观为主。

2. 内容

场景的内容是指场景所表达的知识类型。它又被分为六个不同的方面：

（1）主要关注点

场景内包含的知识可能是关于现在的，也可能是关于未来的。实践中的场景可能会被用来描述当前的系统状况，也可能会被用来描述能够解决当前问题的未来系统的期待方案，还有可能被用来描述一个各项决策都已明确的系统的实际运行情况。

在用户需求获取中，建议关注于期待系统的解决方案。

（2）上下文环境

在描述行为时，场景内可能会包括下列内容：发生在系统内部的行为细节，系统和应用环境的交互，以及完全是外部环境的交互。在需求工程当中，考虑到需求处理的需要，常见的场景内容应该是对系统与环境交互行为的描述，实践中也确是如此。同时，人们越来越提倡将组织背景、文化背景、目标等环境上下文信息的描述包括在场景的内容当中[Pol1997]。

在项目前景和范围定义时，可以适当使用描述外部环境交互的场景形式。

在用户需求获取中，建议使用描述系统与外部环境交互的场景形式。必要的时候（解决方案细节较为复杂，无法仅仅通过描述其与外部的交互来限定），可以适当使用描述系统内部行为细节的场景形式（即详细的顺序图，参见第14章）。

（3）抽象层次

场景的内容可能是具体的、抽象的或者抽象与具体的混合。具体场景，又称为实例场景（Instance Scenario），是对个别行为者、事件、情节的细节描述（例如张三到××ATM点取1000块钱），很少或者完全没有抽象内容。抽象场景，又称为类型场景（Type Scenario），是以经验中的类别和抽象概念来描述事实（例如储户在ATM上取钱）。在场景包含的复杂内容当中，可能一部分已经非常具体，但其他部分仍然比较抽象，这是混合场景（例如储户要从ATM中取1000块钱）。

在用户需求获取中，建议使用抽象场景形式。

（4）覆盖范围

需求既有功能需求也有非功能需求，场景的覆盖范围就是指它对功能需求和非功能需求的覆盖情况。实践表明，场景对功能需求的覆盖情况较好，无论是静态结构还是动态行为，场景的内容都有所包含。同时，场景的内容能够反映一定的非功能需求，但是比起功能需求仍然不足。

在用户需求获取中，场景覆盖应该以功能需求为主，依赖于功能需求覆盖其必需的非功能需求。

（5）粒度

场景的描述可以在不同的粒度层次上进行。实践发现它有三个常见的描述粒度：描述整个业务过程；描述某个任务完成过程；描述某个交互行为详细处理步骤。

在这三种形式都会在用户需求获取中得到体现，分别用于其早期、中期和后期阶段。

（6）示例类型

在使用场景描述示例时，可能是描述正常流程下的示例，也可能是描述异常流程下的示例。很多学者认为场景在描述异常示例时具有一定的优势，但实践的情况并没有证实这个看法。在实践当中，虽然正常流程示例和异常流程示例的描述都得到了应用，但是描述正常流程示例的场景被应用的更为广泛一些。

用户需求获取中，正常流程、异常流程两种场景形式都需要得到应用，而且最好将它们联合起来应用。

3.目的

目的是指场景在使用时打算扮演的角色，也就说是为什么使用场景。为了不同的目的，对场景的描述、解释和使用也会有所不同。在理论上，需求工程利用场景的目的可能有三种：描述（descriptive）、探索（exploratory）和解释（explanatory）。

描述性场景的目的是为了记录已经得到的需求，也就是整理每次需求获取行为中得到的信息。记录下来的内容可以更好地进行交流，也可以成为开发者和涉众之间达成协议的依据。也就是说，描述性场景可以用来进行需求的文档化，或者为软件开发各方的协商提供基础。

探索性场景可以用于两种目的：一是以需求为关注点进行探索，可以作为需求获取的一种行为手段；二是以解决方案为关注点进行探索，发现能够满足需求的可行方案。也就说探索性场景可以用来进行需求获取和需求建模与分析。

解释性场景是为了解释某个主题和疑问，利用示例来说明原因或可行性。解释性场景可以在需求分析时用于降低模型的复杂性，或者被用于进行需求的验证。

场景在需求工程中的应用目的分布如图7-11所示。

在用户需求获取中，主要使用场景的探索目的。在获取基本结束时，再使用场景的描述目的（用例文档）。

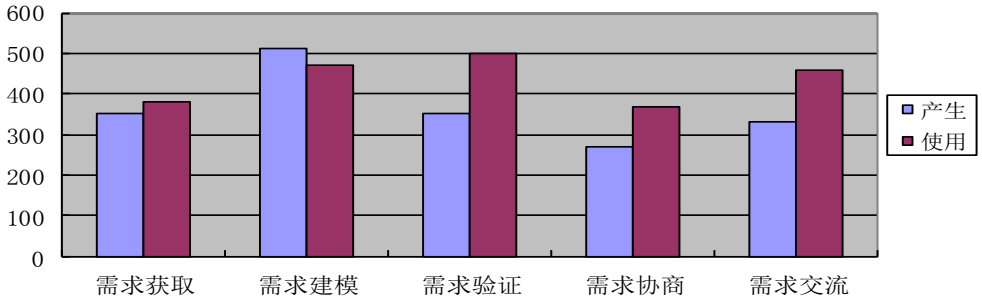


图7-11 场景在需求工程中的应用目的分布，源自[Jarke1997]

4.生命周期

场景的生命周期关注场景的处理和应用，也就是关注场景在整个需求工程当中是如何被捕获、修改和演化的。

实践中发现的场景应用和处理可以概括为6种情况（如图7-12所示）：

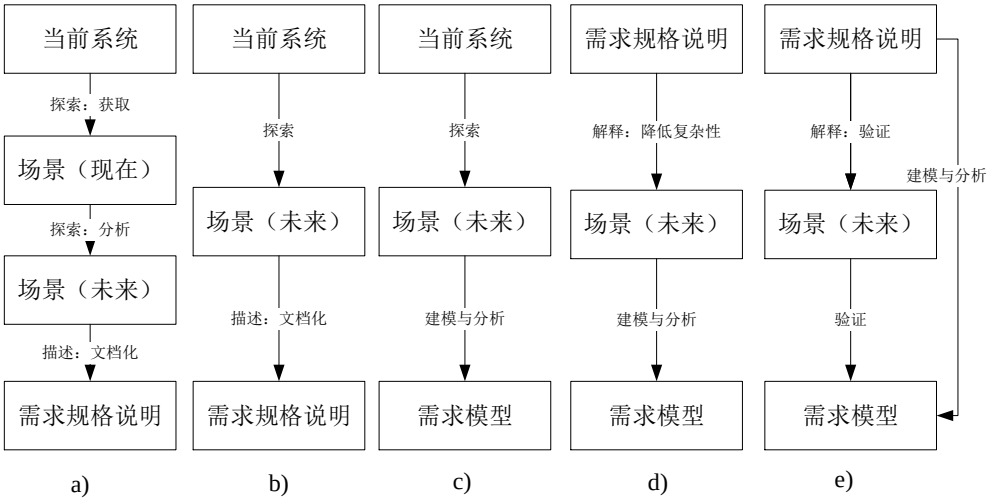


图7-12 场景的应用和处理

（1）从当前系统中捕获和建立关于现在的场景，它们描述问题域的状态和问题。对现在的场景做进一步的分析，转化产生关于未来的场景，描述期待中系统的解决方案。将关于未来的场景进行文档化，产生系统的需求规格说明。

（2）在当前的系统中分析问题和期望，捕获、分析和建立关于未来的场景。然后再将关于未来的场景进行文档化，产生系统的需求规格说明。

（3）在当前的系统中分析问题和期望，捕获、分析和建立关于未来的场景，并依据场景对解决方案的描述，建立需求模型。在这种情况下，需求工程除了场景之外不会再产生专门的需求规格说明，而是以场景作为需求规格说明的替代物。

（4）依据已经建立的需求规格说明，解释和建立关于未来的场景。然后为场景中描述的解决方案建立

需求模型。

（5）依据需求规格说明所描述的解决方案，建立需求模型。同时建立能够验证解决方案的场景。最后，使用场景来验证需求模型的正确性。

实践中还发现，场景信息的捕获主要是利用面谈、原型、观察等基础需求获取方法得到的。在对场景的处理中，微软的office套件（MS Word、MS Power point）是人们利用的主要工具。

在用户需求获取中，主要按照c)方式使用场景。在面对非常简单的系统时（不需要进行需求分析就能够保证需求完备、正确和一致），可以按照b）方式使用场景。在面对问题域非常复杂的系统时（问题域非常复杂、不易理解），不妨按照a）方式使用场景，虽然耗费了工作量（两次描述场景），但分解了需要处理的复杂度（一次是理解问题域，一次是描述用户需求内容）。如果获取源中有旧系统、竞争系统或相关系统的需求规格说明，可以按照d）方式使用场景，完成需求逆向工作。e）方式主要用在需求验证阶段，在用户需求获取阶段基本不使用。

7.3.2. 用例的定位

用例是场景方法中的一种，在场景的分类框架中，用例的定位是：

- 用例是静态的结构化文本描述。
- 用例的内容可以是①对当前世界的描述；②对将来确定的解系统的内部行为描述；③对期待的解决方案的描述。需求工程中的用例描述倾向于采取最后一种方式，需求获取当中可能会使用第一种方式。第二种方式基本不会在需求工程当中使用。
- 用例可能会被用于描述系统内部的交互，也可能被用于描述系统和环境的交互，还可能会被用于描述行为的环境和背景。需求工程倾向于第二种方式的用例描述，也可能会包含有第三方式的用例描述。
- 用例是类型层次的事件描述，主要用来描述功能需求，可以围绕功能需求组织其他需求内容。
- 用例可以是比较抽象的，用于描述整个业务过程；也可以是比较具体的，用于描述任务完成过程；还可以是非常具体的，描述交互行为详细步骤。在需求工程的前期，会产生第一种和第二种用例描述，但最终都需要细化为最后一种形式的用例描述。
- 用例的内容既包含有正常流程，又包含有异常流程。
- 用例可以用于各种目的的应用，包括描述、探索和解释（explanatory）。需求获取和需求验证是它在需求工程中的主要应用阶段，它也可以用于需求的建模、交流和协商。
- 场景的各种生命周期特征、应用和处理过程都适用于用例。需要强调指出的是，用例是在对现实世界的探索当中或者是在对需求规格说明的解释当中产生的，而不是通过功能分解的方式创建的。至少，在高层的功能需求获取完备之前，在用例的产生方式当中是不允许使用功能分解方式的[Chalkiadakis2001]。

除了使用场景分类框架定位用例特征之外，其他一些概念对理解准确用例的含义和用法也是非常重要的，分别是主参与者（Primary Actor）、辅助参与者（Secondary Actor）、目标（Goal）、职责（Responsibility）、行为（Action）和交互（Interaction）等概念。

如图7-13所示，提出请求的用户被称为主参与者，他有一个需要在系统协助下才能得以实现的目标。在实现主参与者目标的过程当中，系统可能无法独自完成任务，它可能需要请求其他参与者或者其他系统的协助，那么这些被系统请求外部对象就被称为辅助参与者。系统本身也被看作是一个参与者，一个需要被实现的系统参与者（System Under Design）。每一个行为者都有一些需要完成的职责，表现为一系列需要达到的目标。为了达到目标，参与者会执行一些行为。参与者执行的行为会触发自己与其他参与者之间的交互，在交互当中其他参与者履行自己的某些职责，满足发起行为的参与者的目标。在简单的情况下，参与者之间的交互仅仅是一次消息的传递。在复杂的情况下，参与者之间的交互是一系列消息的传递。在一次交互当中所传递的准确的消息内容、顺序和过程因系统之前、现在和将来状态的变化而变化。在系统每一种确定状态下发生的交互和消息传递序列就是一个场景。用例描述了在交互中所有可能发生的场景的集合。

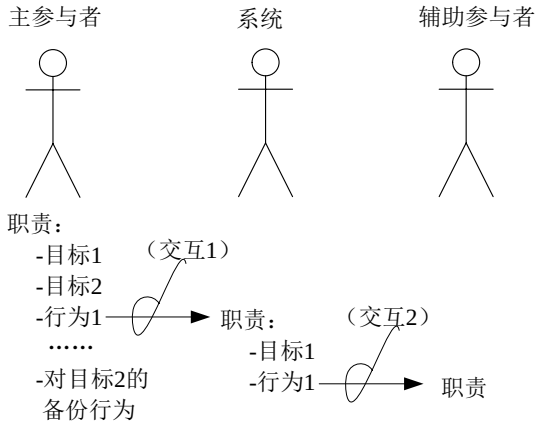


图7-13 用例的交流模型，源自[Cockburn1997]

7.3.3. 用例图

用例的定位只是说明了单个用例的内容描述特点，要将多个用例联系起来，共同表达系统某一部分甚至整个系统的功能，还需要使用UML[Rumbaugh2004]的用例图。

用例图是以用例、参与者（Actor）为基本元素，描述系统功能的静态视图。要注意区分用例和用例图，[Sinnig2005]在实践调查当中发现了将其二者混淆的现象。用例是一种文本方式的需求描述手段，而用例图是将获取得到的用例进行集中展示的图形表示法。用例的目的是描述业务的细节，用例图的目的是以用例为单位将系统的功能和行为展示出来。

用例图的基本元素有四种：用例（Use Case）、参与者、关系（Relationship）和系统边界（System Boundary）。

1.用例

用例是用例图最重要的元素，是对业务工作的描述，或者说是对系统功能的陈述。

在用例图当中，使用一个水平的椭圆来表示用例，如图7-14所示。需要注意的是，在用例图中的椭圆并不是目的，更细节的用例文本描述才是真正有价值的东西。“和用例图的图示相比，用例的文本描述是更加重要的工作，实践当中很多围绕图示法（主要是对用例关系的处理）的争论是一种舍本逐末的行为”[Larman2002]。

图7-14 用例图示

2.参与者

发起或触发用例的外部用户以及其他软件系统等角色被称为参与者。它的图示是一个小人，如图7-15所示。

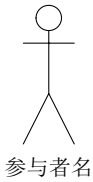


图7-15 参与者图示

参与者代表的是同系统进行交互的角色，不是一个人或者工作职位。一个实际用户可能对应系统的多个参与者。不同的用户也可以只对应一个执行者。事实上，参与者也不必非得是一个实际用户，它也可以是一个组织、另一个系统、外部设备、时间概念等等。

3.关系

用例图中的关系有以下几种：

（1）关联（association）

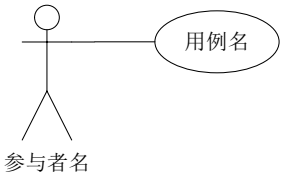


图7-16 关联图示

关联是用例和参与者之间的关系，描述了用例和参与者之间的交互。如果一个参与者参与了一个用例（不论是主参与者还是辅助参与者），那么该参与者和用例之间就存在一个关联。关联是用一条连接参与者和用例的实线来表示的，如图7-16所示。

有些开发者建议将主参与者描述在用例的左边，将辅助参与者描述在用例的右边。当然，这并不是一种标准，适当使用可以让用例图的描述更清晰。

（2）包含（include）

在多个用例当中常常会发生同样的行为，这些行为跨越了多个用例。与其重复书写这些共同部分，不如将这些共同部分抽取出来，形成一个抽象用例（Abstract Use Case），然后原有的用例通过使用新建立的抽象用例来减少用例描述的冗余。原有用例和新建立的抽象用例的关系即为包含关系。需要注意的是抽象用例是不能被实例化的，它必须被包含在其他用例中才能得以执行。

例如，在图书管理系统当中，执行借书和续借时都需要验证读者的身份，因此就可以从“借书”和“续借”两个用例中抽取出附加用例“身份验证”，建立如图7 - 17所示的用例图。

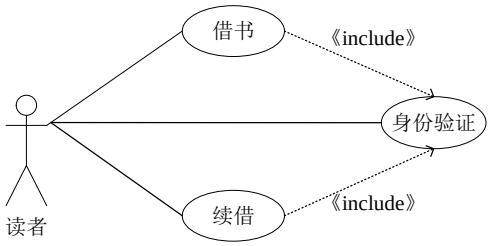


图7-17 用例的包含关系示例

（3）扩展（Extend）

在需求开发当中，随着理解的深入，经常会需要依据新的需求扩充原有的用例文本描述，增加新的异常处理流程和场景。但在有些情况下，一些原因（例如建立基线的要求）使得原有的用例文本不能被直接修改。这时，可以建立一个新需求的附加用例（Additional Use Case），然后使用新的附加用例扩展原有用例。

新的附加用例会描述对新需求的处理流程，并定义新的处理流程在原有用例流程中的扩展点和触发条件。在执行新的附加用例时，新的附加用例会首先执行原有用例的流程，而且可能会按照原有用例的流程执行整个过程。只有在到达新流程在原有用例流程中的扩展点并且条件满足触发条件时，才可能执行附加用例

的新流程。

例如，在一个正常的商品销售用例“销售处理”中，如果顾客要求采用礼券的支付方式，那么就可能会引发一个新的扩展流程。为了在不修改“销售处理”的情况下满足礼券支付的新流程要求，可以定义一个扩展用例“礼券式付费处理”来加以实现。扩展后的用例关系如图7-18所示。

UC1: 销售处理 (基用例)

…

扩展点: VIP 顾客, 步骤 1； 付费, 步骤 7.

主成功场景:

1.顾客抵达销售终端POS，为要购买的商品或服务买单.

…

7.顾客付费，系统进行付费处理.

…

UC15: 礼券式付费处理(扩展用例)

…

触发原因: 顾客想要用礼券付费.

扩展点: 销售处理中的付费.

主成功场景:

1. 顾客向收银员递交礼券.

2. 收银员输入礼券的标识ID.

…

图7-18 用例的扩展关系示例

在有些用例过于复杂时，为了降低复杂度，也可以使用扩展关系，将原有用例的一些复杂处理行为扩展为附加用例。这种用法非常普遍和有效。例如，如果处理商品销售的用例“销售处理”在顾客采用现金支付、信用卡支付和礼券支付三种方式下都会发生比较复杂的处理，那么就可以通过图7-19所示的方式降低用例“销售处理”本身的复杂性。

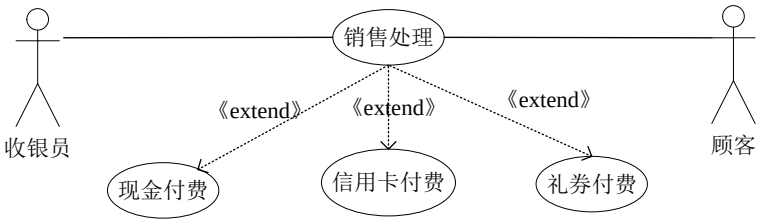
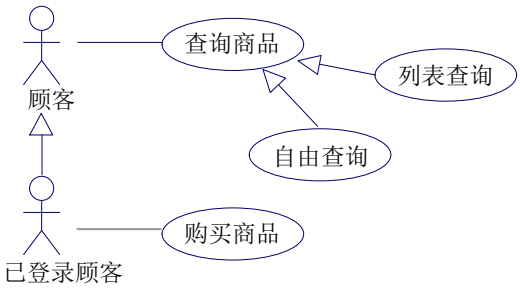


图7-19 利用扩展关系简化用例复杂度示例

（4）（用例）泛化（Generalization）

用例间的泛化关系是指子用例继承了父用例的特征并增加了新的特征，如图7-20所示。



7-20 用例的泛化关系示例

（5）（参与者）泛化（Generalization）

用例间的泛化关系是指子参与者继承了父参与者的特征并增加了新的特征，如图7-20所示。

再次强调：用例文本描述比用例图要重要得多，所以不要为了推敲用例图的各种关系而煞费苦心和争论不休，为了减少用例图的复杂度，不提倡在用例图中较多使用复杂的用例间关系（尤其是泛化关系）[Cockburn2001, Larman2002, Gottesdiener2002, Sninig2005]。

4.系统边界

系统边界是指一个系统所包含的系统成份与系统外事物的分界线。用例图使用一个矩形框来表示系统边界，以显示系统的上下文环境。

一个系统边界的简单示例如图7-21所示。所有的用例都是系统的内部功能，介于系统边界之内。所有的参与者都是系统外的交互对象，介于系统边界之外。

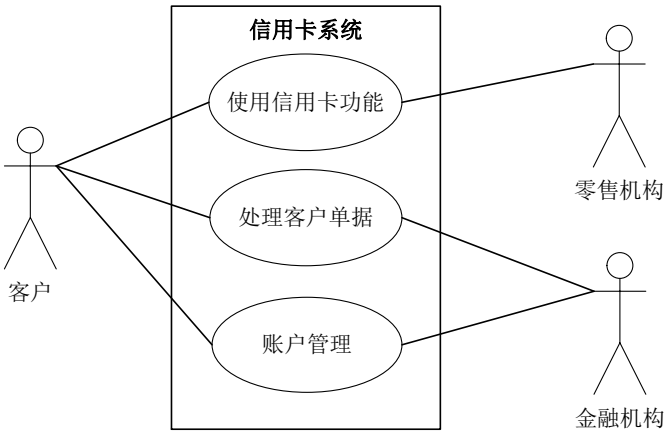


图7-21 系统边界示例

7.4. 以用例/场景模型为主线索开展用户需求获取

以用例/场景模型为主线索，展开用户需求获取的过程如图7-22所示。下面分别描述其中的各个步骤。

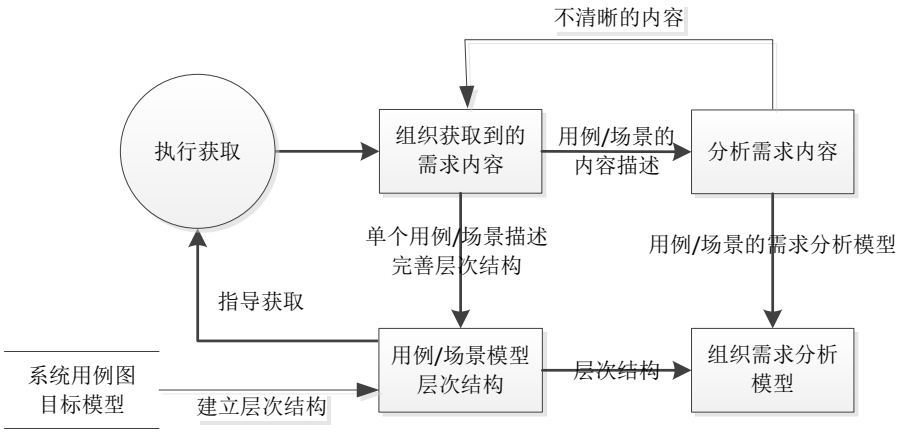


图7-22以用例/场景模型为主线索开展用户需求获取过程

7.4.1. 依据系统用例图、目标模型建立初始用例/场景模型

系统用例图是前景与范围阶段建立的系统边界，它的主要元素就是用例，这就是最初始的用例/场景。

系统用例图中的用例通常是平等的。但如果有目标模型为参考，就可以依据目标Cover链接的场景，建立具有层次结构的用例/场景模型，如图7-23所示。

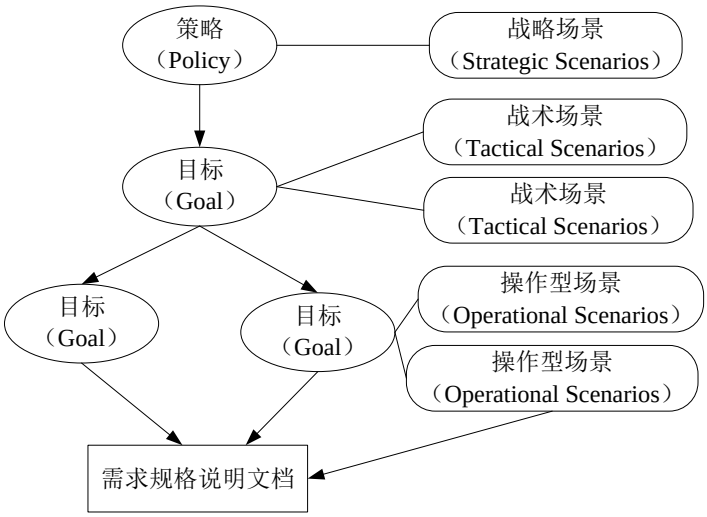


图7-23 利用目标结构组织用例/场景层次结构，修改自[Jarke1998]

7.4.2. 根据用例/场景模型指导获取，完善层次结构

用例/场景模型是开展用户需求获取的主线索，具体包括：

- 初始系统用例涉及的主题需要获取。
- 概要用例描述中发现的新主题需要获取。例如，分析图7-26所示的概要用例描述时，可以发现新主题，建立新的具体用例如图7-24所示。
- 具体用例中发现的模糊、不正确、不完备等细节内容需要再获取，具体示例参见7.4.6节。

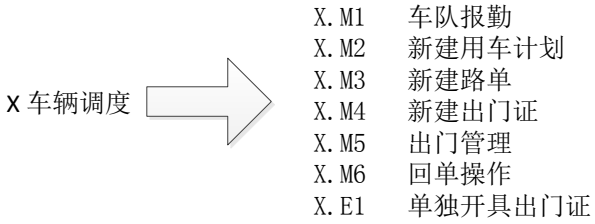


图7-24 使用概要用例描述发现新获取主题示例

7.4.3. 使用用例/场景组织获取内容

面谈、原型、头脑风暴、观察等每次需求获取活动都会得到一些需求内容，不要将这些内容散乱堆放，要及时地使用用例/场景将它们组织起来。

例如，图7-25是一次面谈中得到面谈报告，可以将这些内容组织为图7-26所示的用例描述。

谈话要点	被会见者观点
主要任务	主要的工作是车辆调度 包括：车队报勤、开用车计划、开路单、开出门证

具体流程	基本流程是 1.每天晚上，车队会报第二天的勤，包括人员报勤和车辆报勤 2.之后如果有新任务，会新建用车计划。 3.之后根据用车计划，开具路单，同时附带一张出门证。
分支流程	主要的分支流程 1没有用车计划，也有可能开路单。 2开路单的车辆选择也可能不算报勤车辆 3没有路单，也有可能单开出门证。

图7-25 获取内容示例——面谈报告

ID	X	名称	车辆调度	优先级	高
参与者及目标	（主参与者）调度室：安排车队的每日调度计划 （辅助参与者）车队领导：管理、批准调度计划 （辅助参与者）司机：上报自己车辆的安排				
触发条件	每天晚上				
主流程	1车队报勤，包括人员报勤和车辆报勤 2如果有新任务，新建用车计划。 3根据用车计划，开具路单 4为路单开具出门证。				
分支流程	3a 没有用车计划，也有可能开路单。 3b开路单的车辆选择也可能不算报勤车辆 4没有路单，也有可能单开出门证。				

图7-26 使用用例/场景组织需求获取内容示例

7.4.4. 用新组织或修正的用例/场景完善用例/场景模型

根据其主题发现点，可以将新组织的用例/场景整合入用例/场景模型的层次结构。如果用例/场景只是进行了修正和完善，那么也可以用它替代用例/场景模型中的原有用例/场景。

完善用例/场景模型的示意过程如图7-27所示。

7.4.5. 依据用例/场景模型组织需求分析模型

在复杂系统中，需求内容非常多，要为所有需求内容建立一个全局式的需求分析模型是基本不可能的——其复杂度会超出需求工程师的控制能力。所以，复杂系统建模时会先将系统分解为不同部分，每个部分的复杂度是可控的，这时再为这些部分分别建立需求分析模型。该种工作方式的关键是：①保证分析覆盖度，避免有需求被遗漏；②保证不同部分分析模型的一致性，要让他们可有效整合。

用例/场景模型可以帮助组织需求分析模型，如图7-28所示。

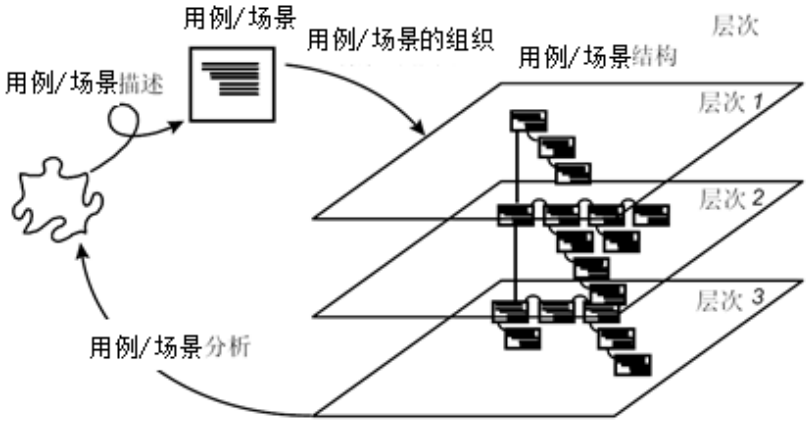


图7-27 用例/场景模型的完善，源自[Ben Achour1999]

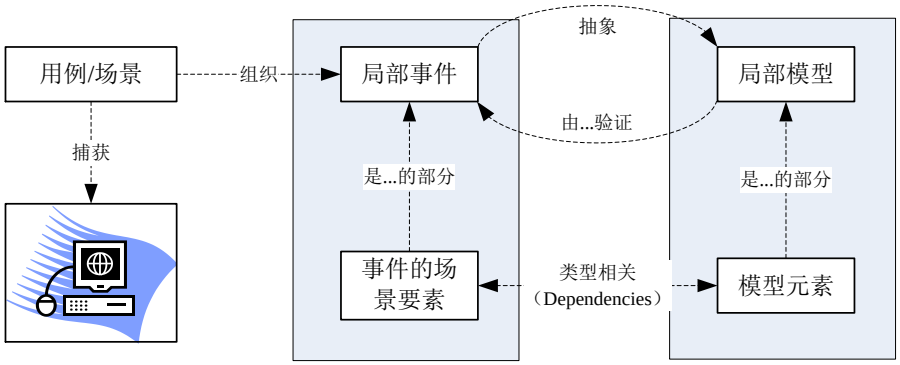


图7-28 利用场景帮助进行需求分析，修改自[Haumer1998]

用例/场景可以作为详细需求分析的信息基础，需求分析活动从用例/场景信息中抽象出需求模型。也就是说，可以仅仅针对局部事件的用例/场景信息进行需求分析，局部事件的用例/场景构成了局部需求分析的背景和上下文知识，在降低了局部需求分析复杂度的同时保证了正确性。

因为需求模型是对场景信息的抽象，所以需求模型中的元素都是来自于用例/场景中的信息要素。这样，通过遍历事件的用例/场景要素，就可以帮助更好、更快的建立需求模型。

局部事件的场景示例，还可以帮助验证需求模型的正确性。

7.4.6. 分析用例/场景发现仍需获取的需求内容

用例/场景没有验证内容正确性、完备性和一致性的能力，所以在建立比较详细的用例/场景描述之后，可以使用分析技术验证其内容的正确性、完备性和一致性[Some´2006, Whittle2006]，并将发现的信息缺失与不足交由下一次获取活动解决，逐步实现用例/场景的正确性、完备性和一致性[Gottesdiener2002]。

例如，假设销售用例描述如图7-29所示。

1.收银员输入会员编号；
2.收银员输入商品；
3.系统显示购买信息；
收银员重复 2-3 步，直至完成所有输入
4.系统显示总价和赠品信息；
5.顾客付款；
6.系统找零；
7.系统更新数据；
8.系统打印收据；
9.顾客离开

图7-29 销售用例的简单描述

为图7-29的用例建立系统顺序图（详细的系统顺序图技术参见第14章），可以发现（1）描述内容的交互性不足，即没有清晰的“外界请求→系统响应→外界再请求→系统再响应...”的过程；（2）顾客是无法直接与系统发生交互的，所以其步骤5、6、9都需要修正。

针对发现的问题，可以明确后续获取内容，并再次修正系统顺序图，最终可以将图7-29的销售用例细化和明确为图7-30所示，图7-30的用例描述是能通过系统顺序图验证。

1. 收银员输入会员编号；
2. 系统显示会员信息；
3. 收银员输入商品；
4. 系统显示输入商品的信息；
5. 系统显示所有已输入商品的信息；
收银员重复 3-5 步，直至完成所有输入
6. 收银员结束商品输入；
7. 系统显示总价和赠品信息；
8. 收银员请求顾客付款；
9. 顾客支付，收银员输入支付数额；
10.系统显示应找零数额，收银员找零；
11.收银员结束销售；
12.系统更新数据，并打印收据。

图7-30 销售用例的改进一

再为图7-30的用例建立概念类图（详细的类图技术参见第14章），可以发现其描述内容仍然不足，在问题域知识方面有着较大的欠缺：

- 部分信息的使用不准确，例如步骤2中输入的是商品标识，而不是商品，第5步显示的已输入商品列表信息和总价。
- 部分信息不明确，例如会员信息、商品信息、商品列表信息、赠品信息、更新的数据、收据等等各自的详细内容并没有描述。
- 遗漏了重要内容，例如总价的计算需要使用商品特价策略和总额特价策略，赠品的计算需要使用商品赠送策略和总额赠送策略。

上述问题也需求在后续获取活动中解决，直到能建立如图7-31所示的用例描述，才能通过类图验证。

1. 如果是会员，收银员输入客户编号
2. 系统显示会员信息，包括姓名与积分
3. 收银员输入商品标识
4. 系统记录并显示商品信息，商品信息包括商品标识、描述、数量、价格、特价（如果有商品特价策略的话）和本项商品总价
5. 系统显示已购入的商品清单，商品清单包括商品标识、描述、数量、价格、特价、各项商品总价和所有商品总价

收银员重复 3-5 步，直到完成所有商品的输入

6. 收银员结束输入，系统计算并显示总价，计算根据总额特价策略进行
7. 系统根据商品赠送策略和总额赠送策略计算并显示赠品清单，赠品清单包括各项赠品的标识、描述与数量
8. 收银员请顾客支付账单
9. 顾客支付，收银员输入收取的现金数额
10. 系统给出应找的余额，收银员找零
11. 收银员结束销售，系统记录销售信息、商品清单、赠品清单和账单信息，并更新库存
12. 系统打印收据（格式为...）

图7-31销售用例的改进二

如果一个用例/场景描述能够通过各种不同分析技术的验证，那么用例/场景就是正确、完备和一致的，就可以结束对其内容的需求获取活动了。

7.5. 用例文档

在用例驱动的软件开发当中，用例是整个软件开发过程的核心元素，因此将系统的所有用例都进行文档化是非常重要的，产生的结果被称为用例文档。用例文档将是进行项目交流的有效途径。

如3.1所述，在需求工程当中，主要产生三类重要的文档：项目前景和范围文档、用户需求文档以及需求规格说明。用例文档通常被用来代替用户需求文档，起到记录、交流领域信息和用户期望的作用。在特殊的情况下（例如市场和时间压力），用例文档可以用来替代需求规格说明，但总的来说这是一种并不值得提倡的方式。

用户需求文档的基本职责是把有关问题域的必要信息以及涉众的需求传达给解系统的设计者。它不涉及和解决方案直接相关的信息。除了内容上的区别之外，在文档的结构组织和细节写作上，用户需求文档和需求规格说明有着很大的相似性，所以关于用户需求文档的详细情况这里不再介绍，需要的读者请结合第15章的内容进行理解。

对期待中的解决方案的描述能够很好的反映涉众的期望和期望所依存的问题域信息，所以将这种描述方式的用例组织起来进行文档化，就可以用产生的用例文档来很好的替代用户需求文档。

一个用例文档的常见格式如模版7-2所示。

一、文档的信息

1、对文档本身特征的描述信息，例如文档的标题、作者、更新历史等；

2、为了方便读者阅读的导读性信息，例如写作的目的、主要内容概述、组织结构、文档约定、参考文献等。

二、用例图或者用例列表

使用一个和几个用例图来概括文档中出现的所有用例及用例间的关系。在文档内用例比较多的情况下，也可能使用一个列表来代替用例图，列表内逐一列出文档内所有用例的 ID、名称和其他需要的概括性信息。

三、用例描述

用例 1

对用例 1 的详细描述。

…

用例 n

模版7-2 用例文档模版

产生的用例文档一定要进行评审[Gottesdiener2002, Cox2004, Leite2005]，推荐使用图7-32所示的检查列表进行评审。

1.覆盖范围。

1.1 跨度：用例应该包含所有信息。

1.2 范围：用例应该只包含项目范围内的相关陈述细节。

2 正确性。

2.1 文本顺序正确：用例的描述应该有逻辑路径，尤其要注意错误事件描述的逻辑正确性。

2.2 依赖正确：用例应该是完备的（包括分支/异常流程），不能在未期待状态下终止。

2.3 合理：用例的逻辑描述应该是合理的，能够描述一个正确的解决方案，不能有不正确事件和认知错误。

3 抽象层次一致。用例的抽象水平应该是一致的，都应该是抽象的，以利于理解。

4 结构一致。

4.1 差异性：分支和异常流程应该被定义为主流程之外的单独部分。界面、质量、对外接口、业务规则、数据等非功能需求被定义为主流程之外的单独部分。

4.2 顺序：主流程的行为编号应该是一致的。

5 语言一致。应该使用一般现在时和简洁语句，避免使用副词、形容词、指代词、同义词。

6 分支流程异常流程。

6.1 可行：分支和异常流程应该是有意义和完备的。

6.2 编号：分支和异常流程的编号应该与主流程的编号相适配。

图7-32 用例评审检查列表

引用文献

[Arnold1998] Arnold, M., et al., Survey on the scenario use in twelve selected industrial projects. Aachener Informatik Berichte 98-7 (submitted for publication), 1998.

[Ben Achour1999] Ben Achour, C., Souveyet, C., and Tawbi, M., *Bridging the Gap between Users and Requirements Engineering: the Scenario-Based Approach,* International Journal of Computer Systems Science & Engineering, 14(6), 1999.

[Chalkiadakis2001] Chalkiadakis, G., *UML: A Survey Focused On Use Case Modeling*, available at citeseer.ist.psu.edu/ 497502.html, 2001.

[Cockburn2001] Cockburn, A., Writing Effective Use Cases, Addison-Wesley, 2001.

[Cox2004] Cox, K., Aurum, A., Jeffery, R., An Experiment in Inspecting the Quality of Use Case Descriptions, Journal of Research and Practice in Information Technology, Vol. 36, No. 4, November 2004.

[Gottesdiener2002] Gottesdiener, E., Top Ten Ways Project Teams Misuse Use Cases–and How to Correct Them, The Rational Edge, 2002.

[Haumer1998] Haumer, P., Pohl, K., Weidenhaupt, K., *Requirements Elicitation and Validation with Real World Scenes*. IEEE Transactions on Software Engineering, Special Issue on Scenario Management, M. Jarke, R. Kurki-Suonio (eds.), Vol.24, N.12, pp.11036-1054, 1998.

[Hurlbut1997] Hurlbut, R.R, *A survey of approaches for describing and formalizing use cases*. Technical Report XPT-TR-97-03, Expertech Ltd.,1997.

[Jacobson1992] Jacobson, I., Christerson, M., Jonsson, P., Oevergaard, G., *Object Oriented Software Engineering: a Use Case Driven Approach*, Addison-Wesley Publishing Company, 1992.

[Jacobson2004] Jacobson, I., *Use cases - Yesterday, today, and tomorrow*, In. Journal on Software and Systems Modeling, 3:210-220,. 2004.

[Jarke1996] Jarke, M., CREWS : Cooperative RE With Scenarios - Project Summary. http://SunSITE.Informatik.RWTH-Aachen.DE/CREWS/crews-sum.htm.

[Jarke1997] Jarke, M., Pohl, K., Haumer, P., Weidenhaupt, K., Dubois, E., Heymans, P., Rolland, C., Ben Achour, C., Cauvet, C., Ralyté, J., Sutcliffe, A., Maiden, N. A.M., Minocha, S., Scenario Use in European Software Organizations -- Results From Site Visits and Questionnaires. CREWS deliverable N°97-10, http:\\SUNSITE.informatik.rwthaachen.de\CREWS\, 1997.

[Jarke1998] Jarke, M., Bui, X.T., Carroll, J., *Scenario management – an interdisciplinary perspective*. Requirements Engineering Journal 3, 3/4, 1998.

[Jarke1999] JJarke, M., CREWS : Towards Systematic Usage of Scenarios, Use Cases and Scenes, WI

(Wirtschaftsinformatik) 99, Saarbrücken, , 1999.

[Larman2002] Larman, C., Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process, second edition, Prentice-Hall, 2002.

[Leite2005] Leite, P., Doorn, H., Hadad, S., Kaplan, N., Scenario inspections, Requirements Engineering, Volume 10 Issue 1, January 2005, Pages 1 - 21.

[Maiden2005] Maiden, N., Robertson, S., Developing Use Cases and Scenarios in the Requirements Process, ICSE'2005, St Louis, Missouri, United States, May 15-21, 2005.

[Plihon1998] Plihon, V., Ralyté, J., Benjamen, A., Maiden, N.A.M., Sutcliffe, A., Dubois, E., Heymans, P., A Reuse-Oriented Approach for the Construction of Scenario Based Methods. Proceedings of the International Software Process Association's 5th International Conference on Software Process (ICSP'98), Chicago, Illinois, USA, 14-17 June 1998.

[Rolland1998] Rolland, C., Ben Achour, C., Cauvet, C., Ralyté, J., Sutcliffe, A., Maiden, N. A.M., Jarke, M., Haumer, P., Pohl, K., Dubois, E., Heymans, P., A Proposal for a Scenario Classification Framework. RE, Vol.3, N°1, pp.23-47, 1998.

[Rumbaugh2004] Rumbaugh, J., Jacobson, I., Booch, G., The Unified Modeling Language Reference Manual (2nd Edition), Addison-Wesley Professional, 2004.

[Sinnig2005] Sinnig, D., Rioux, F. & Chalin, P., *Use Cases in Practice: A Survey*, in Proceedings of CUSEC 05, Ottawa , Canada , 2005.

[Some´2006] Some´, S. S., Supporting use case based requirements engineering, Information and Software Technology 2006, pp43–58.

[Weidenhaupt1998] Weidenhaupt, K., Pohl, K., Jarke, M., Haumer, P., CREWS Team, Scenario Usage in System Development: A Report on Current Practice. IEEE Software, March, 1998 and International RE Conference (ICRE'98), Colorado Springs, Colorado, USA, April 6-10, 1998.

[Whittle2006] Whittle, J., Specifying precise use cases with use case charts,　In: Bruel, J.-M. (ed.) MoDELS 2005. LNCS, vol. 3844, pp. 290–301. Springer, Heidelberg, 2006.

[Wiegers2010] Wiegers, K., More About Software Requirements: Thorny Issues and Practical Advice, Microsoft Press, November 12, 2010.

[Zorman1995] Zorman L. A., The Content and Composition of Scenarios, OOPSLA Workshop, Requirements Engineering: Use cases and. more, 1995.