第3章 需求确定

3.1 项目导引

项目组启动了一个新的项目——招聘管理系统，目前已得到项目标书，负责用例编写的小王开始忙碌了起来。小张知道小王曾经有过用户需求的经验，所以这些天除了完成老李分配给自己的工作外，他一直围在小王身边问东问西。小张说："怎样才能获得和理解用户的真实需求，为后续的分析工作打好基础呢？软件开发人员分明是按照需求开发出软件，客户为什么仍然不满意?"小王说："何止啊，客户也总是在困惑为什么软件和自己想要的差距会那么大？人们常常错误地认为，在需求分析阶段，开发者必须确定客户想要什么样的软件。事实上，许多项目开发时，客户可能不很明确他们到底需要什么，即使一个客户对所需要的东西有一个好的想法，他也可能难于准确地将其表达给开发者，因为大多数客户的计算机知识比软件开发小组成员来讲要少得多。"

3.2 项目分析

所有的系统需求都有可能时刻在发生改变，负责开发的软件工程人员随着项目的进展对软件的理解不断地加深，购买软件的客户其本身的组织结构可能发生变化，系统的硬件、软件和组织的环境随着时间的推移也会变化。因此，软件开发应从考察需求获取开始。软件需求分析过程不仅需要获得最终用户的需求，更需要不断地与用户沟通、提取需求、验证需求、管理需求，最终才有可能取得用户的满意。

将一个软件产品及时而又不超出预算地开发出来的机会经常会很小，除非软件开发小组成员对软件产品将做什么的理解非常准确且一致，同时开发过程的组织也非常有效。目前，软件工程的焦点正从编写可靠的大型软件转移到确保所设计的软件能够满足用户需要。调查研究和描述用户变化的需求，连同确定该需求所蕴涵的系统特性并编写文档，正是需求分析阶段需要完成的工作。

3.3 需求阶段的任务和目标

需求阶段要解决的问题，是让用户和开发者共同明确将要开发的是一个什么样的系统。具体而言，需求分析主要有两个任务：第一个是通过对问题及其环境的理解、分析，建立分析模型；第二个是在完全弄清用户对软件系统的确切要求的基础上，通过编写需求文档把用户的需求表达出来。

1. 建立分析模型

一般地说，现实世界中的系统不论表面上怎样杂乱无章，总可以通过分析与归纳从中找出一些规律，再通过“抽象”建立起系统的模型。分析模型是描述软件需求的一组模型。由于用户群体中的各个用户往往会从不同的角度阐述他们对原始问题的理解和对目标软件的需求，因此，有必要为原始问题及其目标软件系统建立模型。这种模型一方面用于精确地记录用户对原始问题和目标软件的描述；另一方面，它也将帮助分析人员发现用户需求中的不一致性，排除不合理的部分，挖掘潜在的用户需求。这种模型往往包含问题及其环境所涉及的信息流、处理功能、用户界面、行为模型及设计约束等。它是形成需求说明、进行软件设计与实现的基础。

2. 编写需求说明文档

需求说明文档应该具有准确性和一致性。因为它是连接计划时期和开发时期的桥梁，也是软件分析与设计的依据。任何含混不清、前后矛盾或者一个微小的错漏，都可能导致误解或铸成系统的大错，在纠正时付出巨大的代价。需求说明文档应该具有清晰性且没有二义性。因为它是沟通用户和系统分析员思想的媒介，双方要用它来表达对于需要计算机解决的问题的共同理解。如果在需求说明中使用了用户不易理解的专门术语，或用户与分析员对要求的内容可以做出不同的解释，便可能导致系统的失败。“需求说明”应该直观、易读和易于修改。为此应尽量采用标准的图形、表格和简单的符号来表示，使不熟悉计算机的用户也能一目了然。

3.4 基本概念

什么是需求？到目前为止还没有公认的定义，比较权威的是 IEEE 软件工程标准词汇表中的需求定义：用户解决问题或达到目标所需要的条件或权能。系统或系统部件要满足合同、标准、规范或其他正式规定文档所要具有的条件或权能。反映上面两条的文档说明。

IEEE 公布的需求定义分别从用户和开发者的角度阐述了什么是需求，以需求文档的方式一方面反映了系统的外部行为，另一方面也反映了系统的内部特性。比较通俗的需求定义如下：需求是指明系统必须实现什么的规约，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。

需求工程是指系统分析人员通过细致的调研分析，准确地理解用户的需求，将不规范的

需求陈述转化为完整的需求定义，再将需求定义写成需求规约的过程。需求工程包含需求开发和需求管理两部分。

软件需求是软件工程过程中的重要一环，是软件设计的基础，也是用户和软件工程人员之间的桥梁。简单地说，软件需求就是确定系统需要做什么；严格意义上，软件需求是系统或软件必须达到的目标与能力。软件需求在软件项目中占有重要地位，是软件设计和实现的基础。需求的改变将导致其后一系列过程的更改，因而软件需求是软件开发成功的关键因素。

3.4.1 功能需求

简单地说，功能需求描述系统所应提供的功能和服务，包括系统应该提供的服务、对输入如何响应及特定条件下系统行为的描述。对于用户需求，用较为一般的描述给出；对于功能性的系统需求，需要详细地描述系统功能、输入和输出、异常等，这些需求是从系统的用户需求文档中摘取出来的，往往可以按许多不同的方式来描述。有时，功能需求还包括系统不应该做的事情。功能需求取决于软件的类型、软件的用户及系统的类型等。

理论上，系统的功能需求应该具有全面性和一致性。全面性意即应该对用户所需要的所有服务进行描述，而一致性则指需求的描述不能前后自相矛盾。在实际过程中，对于大型的复杂系统来说，要做到全面和一致几乎是不可能的。原因有二，其一是系统本身固有的复杂性；其二是用户和开发人员站在不同的立场上，导致他们对需求的理解有偏颇，甚至出现矛盾。有些需求在描述的时候，其中存在的矛盾并不明显，但在深入分析之后问题就会显露出来。为保证软件项目的成功，不管是在需求评审阶段，还是在随后的软件生命周期阶段，只要发现问题，都必须修正需求文档。

3.4.2 非功能需求

作为功能需求的补充，非功能需求是指那些不直接与系统的具体功能相关的一类需求，但它们与系统的总体特性相关，如可靠性、响应时间、存储空间等。非功能需求定义了对系统提供的服务或功能的约束，包括时间约束、空间约束、开发过程约束及应遵循的标准等。它源于用户的限制，包括预算的约束、机构政策、与其他软硬件系统间的互操作，以及如安全规章、隐私权保护的立法等外部因素。

与关心系统个别特性的功能需求相比，非功能需求关心的是系统的整体特性，因而对于系统来说，非功能需求更关键。一个功能需求得不到满足会降低系统的能力，但一个非功能需求得不到满足则有可能使系统无法运行。

非功能需求不仅与软件系统本身有关，还与系统的开发过程有关。与开发过程相关的需求包括：对在软件过程中必须要使用的质量标准的描述、设计中必须使用的 CASE 工具集的描述以及软件过程所必须遵守的原则等。

按照非功能需求的起源，可将其分为三大类：产品需求、机构需求、外部需求。进而还可以细分。产品需求对产品的行为进行描述；机构需求描述用户与开发人员所在机构的政策和规定；外部需求范围比较广，包括系统的所有外部因素和开发过程。非功能需求的分类如表 3-1 所示。

表 3-1 非功能需求的类别

非功能需求	产品需求	可用性需求	
		效率需求	性能需求
			空间需求
		可靠性需求	
		可移植性需求	
	机构需求	交付需求	
		实现需求	
		标准需求	
	外部需求	互操作需求	
		道德需求	
		立法需求	隐私需求
			安全性需求

非功能需求检验起来非常困难。这些非功能需求可能来自于系统的易用性、可恢复性和对用户输入的快速反应性能的要求，同时需求描述的不详细和不确定也会给开发者带来许多困难。虽然理论上非功能需求能够量化，通过一些可用来指定非功能性系统特性的度量(如表 3-2 所示)的测试可使其验证更为客观，但在实际过程中，对需求描述进行量化是很困难的。这种困难性体现为客户没有能力把目标需求进行量化的同时，有些目标(如可维护性)本身也没有度量可供使用。因此，在需求文档中的目标陈述中，开发者应该明确用户对需求的优先顺序，同时也要让用户知道一些目标的模糊性和无法客观验证性。

表 3-2 指定非功能需求的度量方法

非功能需求	可使用的度量	度 量 方 法
性能	对用户输入的响应时间 每秒处理的事务数	用户/事件响应时间
		屏幕刷新时间
规模	系统最大的尺寸	KB
		RAM 芯片数
易用性	学习 75%的用户功能所需要的时间 在给定时间内，由用户引起的错误的平均值	培训时间
		帮助画面数
可靠性	出错时间 错误发生率	失败平均时间
		无效的概率
		失败发生率
鲁棒性/健壮性	系统出错后重新启动的时间	失败之后的重启次数
		事件引起失败的百分比
		失败中数据崩溃的可能性
可移植性	目标系统数	依赖于目标的语句百分比
有效性	请求后出错的可能性	
完整性	系统出错时，允许丢失数据的最大限度	

3.5 需求获取方法

为了获取正确的需求信息，可以使用一些基本的需求获取方法和技术。下面介绍几种常用的需求获取方法。

3.5.1 建立联合分析小组

系统开始开发时，系统分析员往往对用户的业务过程和术语不熟悉，用户也不熟悉计算机的处理过程。因此用户提供的需求信息，在系统分析员看来往往是零散和片面的，需要由一个领域专家来沟通。因而，建立一个由用户、系统分析员和领域专家参加的联合分析小组，对开发人员与用户之间的交流和需求的获取将非常有用。通过联合分析小组的工作，可极大地方便系统开发人员和用户沟通。有些学者也将这种面向联合开发小组的需求收集方法称为“便利的应用规范技术”(Facilitated Application Specification Techniques，FAST)。有人主张，在参加 FAST 小组的人员中，用户方的业务人员应该是系统开发的主体，是“演员”和“主角”；系统分析员作为高层技术人员，应成为开发工作的“导演”；其他的与会开发人员是理所当然的“配角”。切忌在需求获取阶段忽视用户业务人员的作用，由系统开发人员越俎代庖。

3.5.2 客户访谈

为了获取全面的用户需求，光靠联合分析小组中的用户代表是不够的，系统分析员还必须深入现场，同用户方的业务人员进行多次交流。根据用户将来使用软件产品的功能、频率、优先等级、熟练程度等方面的差异，将他们分成不同的类别，然后分别对每一类用户通过现场参观、个别座谈或小组会议等形式，了解他们对现有系统的问题和新功能等方面的看法。

客户访谈是一个直接与客户交流的过程，既可了解高层用户对软件的要求，也可以听取直接用户的呼声。由于是与用户面对面的交流，如果系统分析员没有充分的准备，也容易引起用户的反感，从而产生隔阂，所以分析员必须在这个过程中尽快找到与用户的“共同语言”，进行愉快的交谈。在与用户接触之前，先要进行充分的准备：首先，必须对问题的背景和问题所在系统的环境有全面的了解；其次，尽可能了解将要会谈用户的个性特点及任务状况；最后，事先准备一些问题。在与用户交流时，应遵循循序渐进的原则，切不可急于求成，否则欲速则不达。

3.5.3 问卷调查

所谓“问卷调查法”，是指开发方就用户需求中的一些个性化的、需要进一步明确的需求(或问题)，通过采用向用户发问卷调查表的方式，达到彻底弄清项目需求的一种需求获取方法。这种方法适合于开发方和用户方都清楚项目需求的情况。因为开发方和建设方都清楚项目的需求，则需要双方进一步沟通的需求(或问题)就比较少，通过采用这种简单的问卷调查方法就能使问题得到较好的解决。

3.5.4 问题分析与确认

不要期望用户在一两次交谈中，就会对目标软件的要求阐述清楚，也不能限制用户在回答问题过程中的自由发挥。在每次访谈之后，要及时进行整理，分析用户提供的信息，去掉错误的、无关的部分，整理有用的内容，以便在下一次与用户见面时由用户确认。同时，准备下一次访谈时的进一步更细节的问题。如此循环，一般需要 2～5 次。

3.5.5 快速原型法

通常，原型是指模拟某种产品的原始模型。在软件开发中，原型是软件的一个早期可运行的版本，它反映最终系统的部分重要特性。如果在获得一组基本需求说明后，通过快速分析构造出一个小型的软件系统，满足用户的基本要求，就使得用户可在试用原型系统的过程中得到亲身感受并受到启发，做出反应和评价，然后开发者根据用户的意见对原型加以改进。随着不断试验、纠错、使用、评价和修改，获得新的原型版本。如此周而复始，逐步减少分析和通信中的误解，弥补不足之处，进一步确定各种需求细节，适应需求的变更，从而提高最终产品的质量。

作为开发人员和用户的交流手段，快速原型可以获取两个层次上的需求。第一层包括设计界面。这一层的目的，是确定用户界面风格及报表的版式和内容。第二层是第一层的扩展，用于模拟系统的外部特征，包括引用了数据库的交互作用及数据操作，执行系统关键区域的操作等。此时用户可以输入成组的事务数据，执行这些数据处理的模拟过程，包括出错处理。

在需求分析阶段采用快速原型法，一般可按照以下步骤进行。

(1) 利用各种分析技术和方法，生成一个简化的需求规约。

(2) 对需求规约进行必要的检查和修改后，确定原型的软件结构、用户界面和数据结构等。

(3) 在现有的工具和环境的帮助下快速生成可运行的软件原型并进行测试、改进。

(4) 将原型提交给用户评估并征求用户的修改意见。

(5) 重复上述过程，直到原型得到用户的认可。

表 3-3 中总结出了使用原型实现方法的优点和缺点。

表 3-3 原型实现方法的优缺点

编号	优　　点	缺　　点
1	开发者与用户充分交流，可以澄清模糊需求，需求定义比其他模型好得多	开发者在不熟悉的领域中不易分清主次，原型不切题
2	开发过程与用户培训过程同步	产品原型在一定程度上限制了开发人员的创新
3	为用户需求的改变提供了充分的余地	随着更改次数的增多，次要部分越来越大，“淹没”了主要部分
4	开发风险低，产品柔性好	原型过快收敛于需求集合，而忽略了一些基本点
5	开发费用低，时间短	资源规划和管理较为困难，随时更新文档也带来麻烦
6	系统易维护，对用户更友好	只注意原型是否满意，忽略了原型环境与用户环境的差异

由于开发一个原型需要花费一定的人力、物力、财力和时间，而且用于确定需求的原型在完成使命后一般就被丢弃，因此，是否使用快速原型法必须考虑软件系统的特点、可用的开发技术和工具等方面。如表 3-4 所示的 6 个问题可用来帮助判断是否要选择原型法。

表 3-4 原型实现方法的选择

问 题	废弃型原型法	演化型原型法	其他预备性工作
应用领域已被理解吗？	是	是	否
问题可以被建模吗？	是	是	否
客户能够确定基本需求吗？	是	否	否
需求已被建立而且稳定吗？	否	是	是
有模糊不清的需求吗？	是	否	是
需求中有矛盾吗？	是	否	是

先进的快速开发技术和工具是快速原型法的基础。如果为了演示一个系统功能，需要手工编写数千行甚至数万行代码，那么采用快速原型法的代价就太大，变得没有现实意义了。为了快速开发出系统原型，必须充分利用快速开发技术和复用软件构件技术。

1984 年，Boar 提出一系列选择原型化方法的因素，包括应用领域、应用复杂性、客户特征以及项目特征。如果是在需求分析阶段使用原型化方法，必须从系统结构、逻辑结构、用户特征、应用约束、项目管理和项目环境等多方面来考虑，以决定是否采用原型化方法。

(1) 系统结构。联机事务处理系统、相互关联的应用系统适合于使用原型化方法，而批处理、批修改等结构不适宜用原型化方法。

(2) 逻辑结构。有结构的系统，如操作支持系统、管理信息系统、记录管理系统等适合于用原型化方法，而基于大量算法的系统不适宜用原型化方法。

(3) 用户特征。不满足于预先做系统定义说明、愿意为定义和修改原型投资、不易肯定详细需求、愿意承担决策的责任、准备积极参与的用户是适合于使用原型的用户。

(4) 应用约束。对已经运行系统的补充，不能用原型化方法。

(5) 项目管理。只有项目负责人愿意使用原型化方法，才适合于用原型化的方法。

(6) 项目环境。需求说明技术应该根据每个项目的实际环境来选择。

当系统规模很大、要求复杂、系统服务不清晰时，在需求分析阶段先开发一个系统原型是很值得的。特别是当性能要求比较高时，在系统原型上先做一些试验也是很必要的。

为了有效实现软件原型，必须快速开发原型，以使得客户可以评估其结果并及时变更。可以使用三类方法和工具来进行快速原型实现。

(1) 第 4 代技术(4GT)。第 4 代技术包含广泛的数据库查询和报表语言、程序和应用生成器以及其他很高级的非过程语言。4GT 使软件工程师能快速生成可执行代码，因此它们是理想的快速原型实现工具。

(2) 可复用软件构件。结合原型实现方法和程序构件复用只能在一个库系统已经被开发以便存在可以被分类和检索的构件的情况下，才可以有效地工作。特殊的是，现有的软件产品可被用做“新的、改进的”替代产品的原型，这在某种意义下也是一种软件原型实现的复用形式。

(3) 形式化规约和原型实现环境。过去 20 年中已经开发出了一系列的形式化规约语

言和工具，来替代自然语言规约技术。现在，正在继续开发交互式的环境，以便于分析员能够交互地创建基于语言的系统或者软件规约；激活自动工具把基于语言的规约翻译成可执行代码；使得客户可以使用原型可执行代码去精化形式化需求。

3.6 RUP 中需求的特点

在 RUP 中，项目的初始阶段是展开一系列需求讨论会，并尽快配合具有产品品质的编程和测试，此时不需要进行彻底的分析和需求编写。根据需求易于变化的特点，需要尽早地在开发中获得用户的反馈，并用于精化规格说明。在初始阶段，需要确定大部分需求的名称，即确定用例的名称，同时详细分析 10%具有高优先级的用例。对于其他部分的需求描述将在细化阶段进行，这种需求随着项目的不断进展而不断完善的过程被称为进化式的需求。对于需求的演进过程见表 3-5。

表 3-5 跨越早期迭代的需求工作任务示例

科目	制品	初始(1 周)	细化 1(4 周)	细化 2(4 周)	细化 3(3 周)	细化 4(3 周)
需求	用例模型	两天的需求讨论会。定义大多数用例的名称，并附以简短文字摘要 从高阶列表中选择 10%的需求加以分析并详细编写。这 10%的用例应具有重要的架构意义、风险和高业务价值	在本次迭代接近结束时，举行两天的需求讨论会。 从实践工作中获取理解和反馈，然后完成 30%的详细用例	在本次迭代接近结束时，举行两天的需求讨论会。 从实践工作中获取理解和反馈，然后完成 50%的详细用例	重复，详细完成 70%的用例	重复，确定 80%～90%的详细用例，并详细编写。其中只有一小部分在细化阶段构建，其余在构造阶段实现
设计	设计模型	无	对一组高风险的、具有重要架构意义的需求进行设计	重复	重复	重复。高风险和重要架构意义的方面现在应该稳定化
实现	实现模型(代码等)	无	实现之	重复，构建了 5%的最终系统	重复，构建了 10%的最终系统	重复，构建了 15%的最终系统
项目管理	软件开发计划	十分粗略地估计整体工作量	预算开始成形	少许改进……	少许改进……	现在可以提交合理的总体项目进程、主要里程碑、工作量、成本预算

注意，当仅定义约 10%的需求时，项目小组就开始构建系统的产品化核心了。因为这 10%用例的确定是根据优先级的高低确定的，即具有高业务价值、高架构意义、高风险的用例。此时，项目小组需要刻意推进进一步的深入需求工作，直到第一次迭代接近尾声时为止。这样的做法有助于将本次迭代所获得的反馈增加到下一次迭代过程中，也有助于对其他需求的进一步认识和理解。通过细化阶段，给予对部分系统增量构建的反馈、调整，其他

需求将更为清晰并且可以将补充性信息记录在补充性规格说明中。在细化阶段结束时，就可以完成并提交用例、补充性规格说明和设想了。因为此时，这些文档能够合理地反映系统的主要特性和其他需求。通过构造阶段，主要需求（包括功能性需求和其他需求）已经基本稳定下来了，虽然还不是终结，但是已经可以专注于次要的微扰事务了。因此，在该阶段补充性规格说明和设想都不必进行大量改动。

在 RUP 最佳实践的需求管理中有这样一段描述："采用一种系统的方法来寻找、记录、组织和跟踪系统不断变更的需求。"定义中的"不断变更"表明 RUP 能够包容需求中的变更，并将其作为项目的基本驱动力。另一个重要的词是"寻找"。也就是说，RUP 中提倡使用一些有效的技巧以获得启示，例如，与用户一起编写用例，开发者和客户共同参加需求讨论会、请客户代表参加项目小组的讨论以及向客户演示每次迭代的成果以便获得反馈。

在 RUP 或其他进化式方法中，具有产品品质的编程和测试要远早于大多数需求的分析和规格化——或许当时只完成了 10%～20%的需求规格说明，这些需求都具有重要的架构意义、存在风险以及具有高业务价值。

在此阶段中系统分析人员将要关注的重要需求制品包括：用例模型和补充性规格说明。

3.7 用例模型

用例模型是所有书面用例的集合，同时也是系统功能性和环境的模型。用例模型中可包括 UML 用例图，以显示用例和参与者的名称及其关系。UML 用例图可以为系统及其环境提供良好的语境图，也为按名称列出用例提供了快捷方式。下面介绍一下用例模型中的两个重要的概念：参与者和用例。

1. 参与者

参与者（或称为执行者）是任何具有行为的人或事物。参与者和用例通信并且期待它的反馈——一个有价值或可觉察的结果。主要参与者和协助参与者会出现在用例文本的活动步骤中。参与者不仅是人所扮演的角色，也可以是组织、软件和计算机。它们必须能刺激系统部分并接收返回。

在某些组织中很可能有许多参与者实例（例如有很多个销售员），但就该系统而言，他们均起着同一种作用，扮演着相同的角色，所以用一个参与者表示。一个用户也可以扮演多种角色。例如，一个高级营销人员既可以是贸易经理，也可以是普通的营销人员；一个营销人员也可以是售货员。在处理参与者时，应考虑其作用，而不是人或工作名称，这一点是很重要的。参与者触发用例，并与用例进行信息交换。

通常有以下三种类型的参与者。

1）主要参与者

具有用户目标，并通过使用当前系统的服务完成，例如，收银员。他们是发现驱动用例的用户目标。

2）协助参与者

为当前系统提供服务，例如，自动付费授权服务。协助参与者通常是计算机系统，但也

可以是组织或人。通过协助参与者可以明确外部接口和协议。

3）幕后参与者

在用例行为中具有影响或利益，但不是主要或协助参与者，例如，政府税收机关。幕后参与者的确定，确保确定并满足所有必要的重要事务。如果不明确地对幕后参与者进行命名，则有时很容易忽略其影响或利益。

2．用例

在 RUP 中，用例被定义为一组用例的实例，其中每个实例都是系统执行的一系列活动，这些活动产生了对某个参与者而言可观察的返回值。用例的含义可从下面几个方面进行解释。

1）用例是一个自包含的单元

用例与行为相关意味着用例所包含的交互在整体上组成一个自包含的单元，它以自身为结果，而无须有业务规定时间延迟。

2）用例必须由参与者发起并监控

用例必须由参与者发起，由参与者监控，直至用例完成。

3）用例必须完成一个特定目标

可观察的返回值意味着用例必须完成一个特定的业务目标。如果用例找不到与业务相关的目标，则应该重新考虑该用例。

用例是面向目标的，这一点很关键，它们表示系统需要做什么，而不是怎么做。用例还是中立于技术的，因此它们可以应用于任何应用程序体系结构或过程中。

4）用例应该使系统保持在稳定状态

用例应该使系统保持在稳定状态下，它不能只完成一部分，得不到系统处理的最终结果。一个完整的用例必须描述系统在执行了一系列操作之后所达到的某种状态，而这种状态不至于触发其他动作的执行。

用例描述了当参与者给系统特定的刺激时系统的活动。也描述了触发用例的刺激本质，包括输入、输出到其他参与者，转换输入到输出的活动。用例文本通常也描述每一个活动在特殊的活动路线时可能的错误和系统应采取的补救措施。

这样说可能会非常复杂，其实一个用例就是描述了系统和一个参与者的交互顺序。用例被定义成系统执行的一系列动作，动作执行的结果能被指定的参与者察觉到。用例可以捕获某些用户可见的需求，实现一个具体的用户目标。用例由参与者激活，并由系统提供确切的可观察的值给参与者。

在具体的需求过程中，有大的用例（业务用例），也有小的用例。主要是由用例的范围决定的。用例像是一个黑盒，它没有包括任何和实现有关或是内部的一些信息。它很容易就被用户（也包括开发者）所理解（简单的谓词短语）。如果用例不足以表达足够的信息来支持系统的开发，就有必要把用例黑盒打开，审视其内部的结构，找出黑盒内部的参与者和用例。就这样通过不断地打开黑盒，分析黑盒，再打开新的黑盒，直到整个系统可以被清晰地了解为止。采用这种不同层次来描述信息，主要有以下几点原因。

（1）需求并不是在项目一开始就很明确，往往是随着项目的推进逐渐细化。

（2）人的认知往往具有层次的特性，从粗到细、从一般到特殊。采用不同的层次来描

述，适于认知的过程。

黑盒用例是最常用和推荐使用的类型，它不对系统内部工作、构建或设计进行描述。反之，它通过职责来描述系统，这是面向对象思想中普遍使用的隐喻主题——软件元素具有职责，并且与其他具有职责的元素进行协作。

在需求分析中避免进行“如何”的决策，而是规定系统的外部行为，就像黑盒一样。此后，在设计过程中创建满足该规格说明的解决方案。表 3-6 是对用例的黑盒风格和非黑盒风格的一个比较。

表 3-6 用例的黑盒风格与非黑盒风格的比较

黑 盒 风 格	非黑盒风格
系统记录订单信息	系统将录入订单信息写入数据库。……或者(更糟糕的描述) 系统对录入订单信息生成 SQL INSERT 语句……

用例不是面向对象的，编写用例时不会进行面向对象分析。但这并不妨碍其有效性，用例可以被广泛应用。也就是说，用例是经典面向对象分析与设计的关键需求输入。

3.7.1 用例的描述形式

用例是一种编写形式，它可用于多种形式，例如，用来描述一个业务工作过程，或用来集中讨论未来系统的需求问题。用例作为系统的功能性需求将系统分析结果文档化，可能被应用在小型的、集中的工作组中，也可能被应用在大型的、分散的工作组中。每种情况下提倡的编写风格都会有所差异。项目开始阶段中识别出的各个事件必须由用例来满足。一个用例可以满足许多事件，因此一个用例可能有多个路径。路径是为满足参与者的目标而必须进行的步骤的集合。收集有关用例的高层信息，这里所包含的大部分内容都是资料性的，描绘了用例的总体目标。

用例文档是一个按照项目开发者提前定义的格式来创建的文档。有很多格式模板。模板将文档分为几部分，并且引入其他写作惯例。用例文档就是用户需求。

用例有以下三种常用形式，它能够以不同的形式化程度或格式进行编写。

1. 摘要

简洁的一段式概要描述，通常用于主成功场景。在早期需求分析过程中，为了快速了解主题和范围，经常使用摘要式用例描述。可能需要花费几分钟编写即可完成。

2. 非正式

非正式的段落格式。用几个段落覆盖不同场景。一般也是在需求分析早期进行使用。

3. 详述

详细编写所有步骤及各种变化，同时具有补充部分，如前置条件和成功保证。确定并以摘要形式编写了大量用例后，在第一次需求讨论会中，详细地编写其中少量的具有重要架构意义和高价值的用例。表 3-7 给出了详述形式的用例模板中所包含的主要内容。

表 3-7 用例模板内容

用例的不同部分	注 释
用例名称	以动词开始
范围	要设计的系统
级别	“用户目标”或是“子功能”
主要参与者	调用系统，使之交付服务
涉众及其关注点	关注该用例的人及其需要
前置条件	值得告知读者的，开始前必须为真的条件
成功保证	值得告知读者的，成功完成必须满足的条件
基本流程	典型的、无条件的、理想方式的成功场景
分支流程	成功或失败的替代场景
特殊需求	相关的非功能性需求
技术和数据变元表	不同的 I/O 方法和数据格式
发生频率	影响对实现的调查、测试和时间安排
杂项	例如未解决问题

下面对表 3-7 中各部分的含义做一简单的解释。

1. 范围

范围界定了所要设计的系统。通常，用例描述的是对一个软件系统（或硬件加软件）的使用，这种情况下称之为系统用例。在更广义的范围上，用例也能描述顾客和有关人员如何使用业务。这种企业级的过程描述被称为业务用例。

2. 级别

用户目标级别是通常所使用的级别，描述了实现主要参与者目标的场景，该级别大致相当于业务流程工程中的基本业务流程。子功能级别用例描述支持用户目标所需的子步骤，当若干常规用例共享重复的子步骤时，则将其分离出来，创建为子功能级别用例，以避免重复公共的文本。

3. 主要参与者

调用系统服务来完成目标的主要参与者。

4. 涉众及其关注点

它建议并界定了系统必须完成的工作。用例应该包含满足所有涉众关注点的事务。在编写用例其余部分之前就确定涉众及其关注点，能够使项目小组更加清楚地了解详细的系统职责。下面给出一个涉众及其关注点的例子。

涉众及其关注点

收银员：希望能够准确、快速地输入，并且没有支付错误。因为，如果少收货款将从其薪水中扣除。

售货员：希望自动更新销售提成。

5. 前置条件和后置条件(成功保障)

首先,不要被前置条件和后置条件所烦扰,除非要对某些不明显却值得重视的事务进行陈述时,以帮助读者增强理解,不要给需求文档增加无意义的干扰。也就是说,在绝大部分情况下,不需要对这部分内容进行描述。

前置条件:给出在用例场景开始之前,必须永远为真的条件。在用例中不会检查前置条件,前置条件总是被假设为真。通常,前置条件隐含已经成功完成的其他用例场景,例如"登录"。要注意的是,有些条件也必须为真,但是不值得编写出来,例如"系统有电力供应"。前置条件传达的是应该引起读者警惕的那些值得注意的假设。

后置条件:给出用例成功结束后必须为真的事务,包括主成功场景及其替代路径。该保证应该满足所有涉众的需求。

下面给出一个前置条件和后置条件的例子。

> **前置条件**:收银员必须经过确认和认证。
> **后置条件**:存储销售信息。准确计算税金。更新账务和库存信息。记录提成。生成票据。

6. 基本流程(主成功场景和步骤)

用例中最常用的路径。该路径中所有内容都产生正面结果。

7. 分支流程(扩展)

分支流程描述了其他所有场景和分支,包括成功和失败路径。成功的分支路径产生正面结果,但发生的频率低于主路径。

8. 特殊需求

如果有与用例相关的非功能性需求、质量属性或约束,那么应该将其写入用例。其中包含需要考虑的和必须包含在内的质量属性(如性能、可靠性和可用性)和设计约束(通常用于I/O 设备)。下面是一个特殊需求的例子。

> **特殊需求**:
> 使用大尺寸平面显示器触摸屏 UI。文本信息可见性为 1m。
> 90%的信用卡授权响应时间应小于 30s。
> 支持文本显示语言的国际化。
> 在步骤 3 和步骤 7 之间能够加入可插拔的业务规则。

9. 技术和数据变元表

需求分析中通常会发现一些技术变元,这些变元是关于如何实现系统的,而非实现系统的哪些功能。这些变元需要记录在用例中。常见的例子是,涉众指定了关于输入或输出技

术的约束。例如，涉众可能要求“POS 系统中必须使用读卡器和键盘来支持信用卡账户”。要注意的是，以上都是在项目早期进行的设计决策或约束。一般来说，应该避免早期不成熟的设计决策，但有时候这些决策是明显的或不可避免的，特别是关于输入/输出技术的决策。

虽然在详述过程中，系统分析人员需要描述用例的许多要素，但是，还应该清楚什么是用例最根本的东西，什么是使用用例最根本的目的，这就是对场景的描述。实际中使用的要素要看项目的大小而定。把大量的时间花在用例的描述上是没有意义的。用户需要的是一个软件系统，并不是一大堆的用例说明。而且(在有些情况下)，单独列取的用例要素内容实际上已经包含在用例文本描述中了。例如，用例的后置条件实际上已经隐含在用例基本路径中对系统响应的描述中了。因此，用例的描述形式在大多数情况下，采用非正式形式足以表达意图，除非有重要的必须要记录的约束内容，将以详述的形式出现。

3.7.2 用例图

用例图是一种 UML 技术，可用于可视化用例、参与者以及它们之间的联系。可视化图形可以帮助系统分析人员和用户简化对用例的理解，也可以采用流程图、序列图、Petri 网或程序设计语言来表示用例。但从根本上说，用例是文本形式的。所以需要强调的是，用例不是图形，而是文本。用例建模主要是编写文本的活动，而非制图。初学者常见的错误就是注重于次要的 UML 用例图，而非重要的用例文本。

用例图是一种优秀的系统语境图。也就是说，用例图能够展现系统边界、位于边界之外的事务以及系统如何被使用。用例图可以作为沟通工具，用以概括系统及其参与者的行为。图 3-1 展示了某系统绘制的简单的部分用例语境图。本书使用不带箭头的线段将角色与用例连接到一起，表示两者之间交换信息，称为通信联系。参与者之间可以通过泛化关系分类。

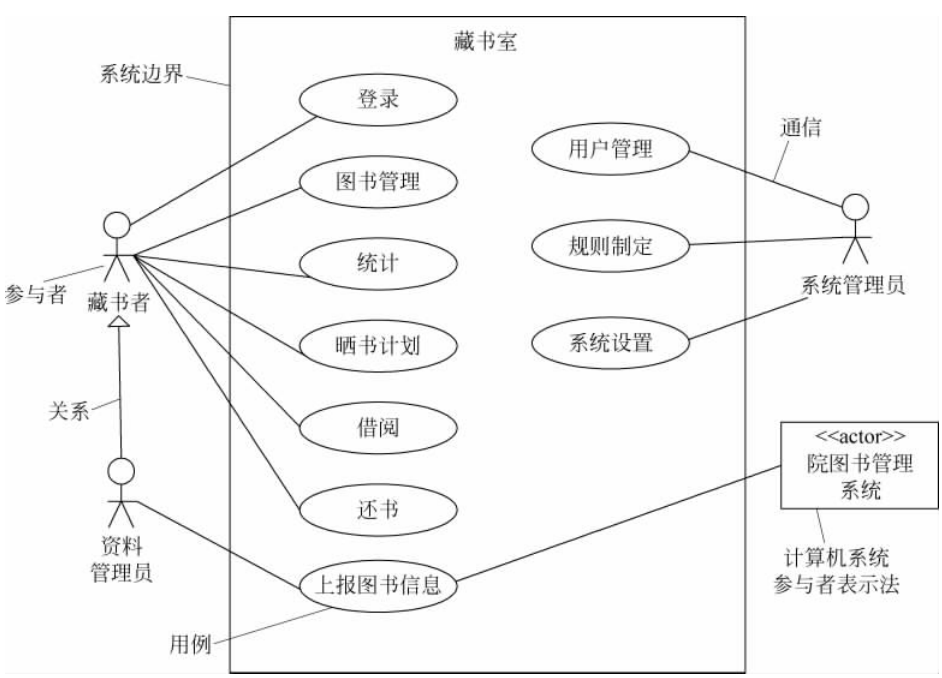

图 3-1 部分用例语境图

用例可以通过关联和泛化关系或者两个构造型关系<<include>>和<<extend>>连接在一起。<<include>>关系表示一个业务用例的执行总是涵盖所包含业务用例(子用例)的所有功能。也就是说,该业务的执行依赖于子用例的执行,如果没有子用例则业务用例的执行是不完整的。例如,用例“预订航班”与子用例“支付费用”,如果用户要预订航班则必须支付预订费用以保证航班预订的有效性,否则预订航班的过程不完整,不能够最终完成预订的功能。<<extend>>联系表示一个业务用例的执行有时需要对用例的功能进行扩展。也就是说,这个业务用例的执行不依赖于该子用例,没有子用例的存在该业务用例仍然能够达成自己的任务目标。例如,子用例“为飞行常客预订航班”扩展了用例“预订航班”。用户在正常情况下使用“预订航班”用例就可以完成自己的目标。在特殊情况下,如果用户是经常乘坐飞机的乘客,则可以使用“为飞行常客预订航班”用例,可以使用专门针对于飞行常客的一些优惠政策,完成特殊情况的处理。<<include>>和<<extend>>联系总是单向的,以指出包含和扩展的方向。<<include>>中被包含的子用例指向父用例。<<extend>>中父用例指向扩展的子用例。图 3-2 给出了 UML 图中对用例的描述形式。

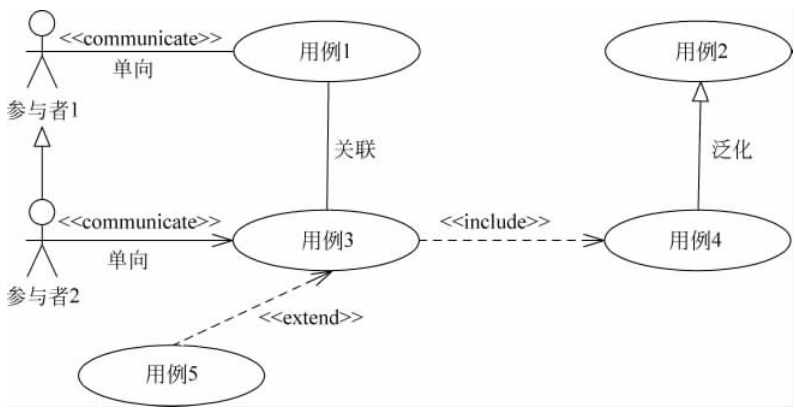

图 3-2 用例模型类元之间的联系

用例是相互连接的,因为它们需要通过合作来完成系统任务。但是如果标出所有的联系将扰乱模型的意图。因此,用例在绘制时可以设定一定的层次。例如,高阶用例用来描述系统的基本用例,使用用户级用例来描述分解后用例之间的关系,但这种联系仍然是有限的。只标出一些选定的联系将带来一个问题,为什么这些联系比其他的联系更重要呢? 这就是系统分析人员必须明确的用例的最根本描述形式是用例文本,图形是次要的辅助手段。要谨慎使用联系。

3.8 用例产生的过程

用例代表了用户的需求,在构建的系统中,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。用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。系统分析人员和用户必须检查每一个用例,在把它们纳入需求之前确定其是否在项目所定义的范围内。在理论上,用例的结果集将包括所有合理的系统功能。

当使用用例进行需求获取时,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，它们会给随后的设计过程带来不必要的限制。

产生用例最常用的方法是使用事件表。系统分析员以系统特性或业务流程为基础，捕获由此产生的事件，记录事件清单。在对事件清单中的事件补充相应的细节后，分析表中的每一个事件，以决定系统支持这个事件的方式、初始化这个事件的参与者，以及由于这个事件而可能触发的其他用例，并将其整理概括成用例。图 3-3 简单描述了用例的产生过程。

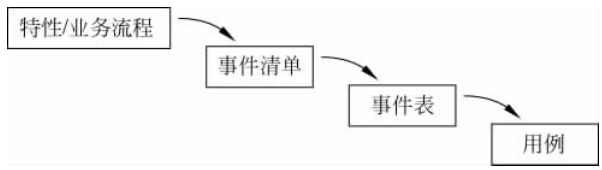

图 3-3 用例产生的过程

3.8.1 事件清单和事件表

实际上，所有的系统开发方法都是以时间概念开始建模的。事件发生在某一特定的时间和地点，可描述并且软件系统可以参与，并且需要记录下来。系统的所有处理过程都是由事件驱动或触发的。因此当定义系统需求时，把所有事件罗列出来并加以分析，是系统分析人员对系统着手分析的入手点。在有些参考书中事件清单和整理好的事件表被称为初始问题陈述。叫法虽然不一样，但是实质上都是对项目中出现的典型事件进行记录、总结的结果。无论是数据流图技术还是用例技术，列出事件清单都是一种快捷的抓住系统主要需求的方法。事件清单的构成就是将在与系统活动相关的行为描述抽取出来后形成一个列表的过程。

当一个系统定义需求时，先调查清楚能对该系统产生影响的事件是十分有用的。更准确地说，需要明确什么事件发生时需要系统参与并做出响应。事件的发现，可以借助于项目前景中系统特性的描述，将重点集中在对用户具有重要价值的核心目标上。在此基础之上询问对系统产生影响的事件。

通过询问对系统产生影响的事件，系统分析人员可以将注意力集中在外部环境上，遵循需求分析的目的是“做什么”，避免陷入“怎么做”的细节之中，应该把整个系统看成一个黑盒。最初的调查帮助系统分析人员主要从高层次上全面考察系统，而不是集中在系统内部工作上。最终用户，即真正使用系统的人，也习惯于按照那些影响他们工作的事件来描述系统需求。因此，当用户使用系统时，把重点集中在事件上也是合情合理的。最后，把重点集中在事件上也提供了一种划分（或分解）系统需求的方法，这样系统分析人员就可以针对不同场合下出现的相关事件而分别研究各个部分了。复杂的系统需要分解成易处理并能更好理解的小单元，而按照事件来划分系统是实现这种分解的一种方法。

1. 事件的类型

系统分析人员在记录和抽取事件的过程中，分清事件的类别有助于更好地理解系统需要做出的响应和系统的职责。

事件分为外部事件、内部事件两类。系统分析人员开始工作时识别并列出尽可能多的

事件，在与系统用户的交谈中不断细化这些事件列表。示例项目的外部事件与内部事件的对比如图 3-4 所示。

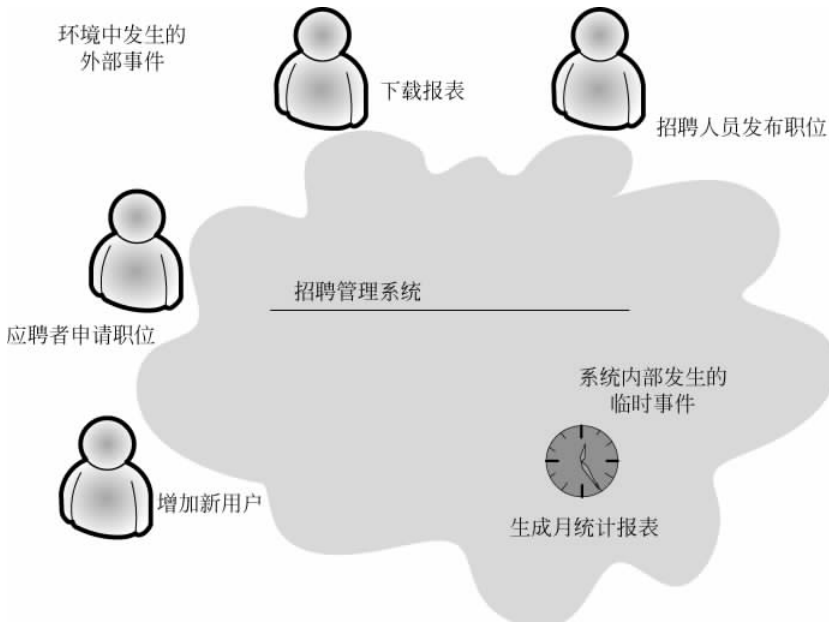

图 3-4　影响系统的事件

1）外部事件

外部事件是系统之外发生的事件，通常都是由外部实体或动作参与者触发的。外部实体（或动作参与者）是一个人或组织单位，它为系统提供数据或从系统获取数据。为了识别关键的外部事件，系统分析人员首先要确定所有可能需要从系统获取信息的参与者。例如，在示例项目中应聘者就是一个典型的参与者，他通过系统获得相关的职位信息，当他通过平台决定申请某一职位时，系统需要根据他提供的应聘者信息将职位信息与应聘者关联起来，记录应聘信息。由参与者发起的这些外部事件促使系统必须处理这些重要的事务。

当描述外部事件时，需要给事件命名，这样参与者才能明确触发的任务。同时将参与者需要进行的处理工作也包括进来。例如，“应聘者申请职位”描述了一个外部参与者（应聘者）以及这个参与者想做的事情（申请职位），这一事件将直接影响系统需要完成的任务。

下面的描述有助于帮助系统分析人员把握事件抽取的情形。当这些情景发生时，就产生了外部事件，系统分析人员就需要对这些情景进行记录。

（1）参与者需要触发一个事务处理（过程）；

（2）参与者想获取某些信息；

（3）数据发生改变后，需要更新这些数据，以备其他相关人员使用；

（4）管理部门想获取某些信息。

2）内部事件

内部事件是由于达到某一时刻时所发生的事件。许多信息需要系统预设在特定时间间隔内产生一些输出结果。例如，工资系统每两周（或每月）生成工资清单，每个月的 1 日需要话费管理系统自动产生话费清单。有时输出结果是管理部门需要定期获得的报表，例如业

绩报告、销售统计报表。这些是系统自动产生所需要的输出结果而不需要用户进行操作和干预。也就是没有外部动作参与者下达命令，系统就会在需要的时候（用户指定的时间点上）自己自动产生所需的信息或其他输出。

下面的描述有助于系统分析人员提取要记录的内部事件。

（1）所需的内部输出结果

如管理部门报表（汇总或异常报表），操作报表（详细的事务处理），综述、状况报表。

（2）所需的外部输出结果

如结算单、状况报表、账单、备忘录。

2. 示例中的事件

系统分析人员在获取事件的过程中，如果对于项目背景有些了解的话，可以利用原有的业务领域知识来引出一些典型事件，也可沿着典型的业务处理流程来逐个提取业务事件。针对于招聘管理系统，一方面参照用户提供的项目说明（参见 3.10.2 节的项目说明）抽取里面出现的事件，一方面可以根据企业较为成型的招聘管理流程进行分析。现在以通常情况下招聘流程为例，描述一下事件提取过程，请参见图 3-5。从这里，系统分析人员可以提取这样几个事件。

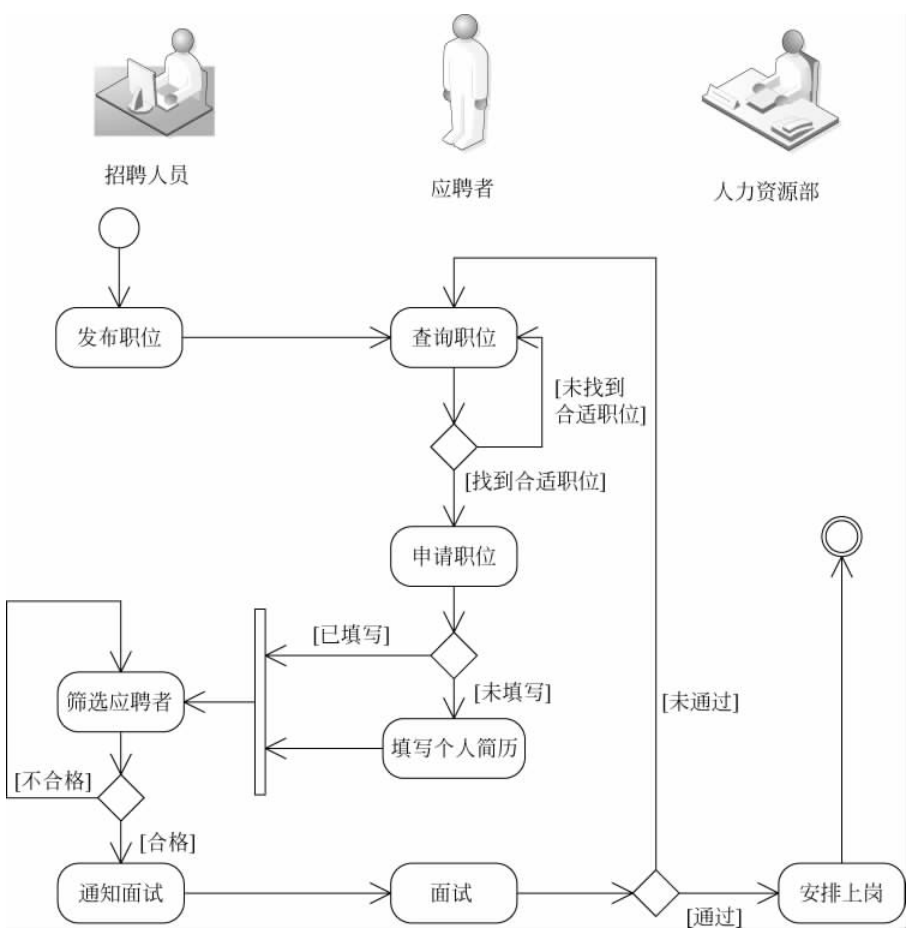

图 3-5 招聘业务流程

首先，根据企业各部门招聘的要求，需要对外发布相应职位人才需求的信息，按照招聘管理的要求，需对职位信息进行登记，此时的工作需要招聘人员来完成。因此，相对于招聘人员来讲就会产生与系统业务相关的活动——发布职位，通过这件事情的发生，大家可以观察到系统将会对这件事件的结果进行处理。也就是说，以后系统的构建应该能够提供相应的功能以帮助招聘人员响应这个事件的发生。

应聘者通过一些渠道了解到企业招聘信息，应聘者根据自己的意向查找相应的职位，如果有满足要求的职位，则提出职位的申请请求。那么，应聘者职位申请信息的上报，就成为企业招聘流程中的一个环节。这个事件的发生，对整个系统的行为将会产生影响。因此，将应聘者申请职位作为一个事件进行提取。

招聘人员对提交申请的应聘者信息进行筛选，确定符合要求的人选。对于符合要求的应聘者可以集中发送面试通知。在这里，筛选应聘者和通知面试都将会对业务执行产生影响，这些都可作为事件进行提取。

后续的应聘者的面试过程及人力资源部对招聘进来的新员工安排岗位等活动，都会对业务数据产生影响，都可以作为事件进行提取，这里不一一进行讲解了。

另外，还可以从事件的分类上去获得相关事件。例如，前面所提取的事件都有相应的参与者参与，外界可以观察到的，这些都是外部事件。那么，是否存在需要根据时间触发的任务呢，也就是所谓的内部事件呢？这需要从业务规则入手，参照前面提到的内部事件内容列表，可以发现招聘管理业务实施过程中，需要每间隔一个固定的时间，要求相关人员提供信息统计报表，用于招聘团队的绩效考核。通过系统可以自动在指定时间完成报表的自动上报就成为系统内部的一个事件。另外，招聘人员在筛选应聘者后将会向符合职位需求的应聘者在某一特定时刻统一由系统发送面试通知，也可以成为系统的一个内部事件。

综合整理这些内部事件和外部事件，系统分析人员将获得招聘管理系统中的很多事件，并采用主语＋谓语(动词)＋宾语的形式对事件进行描述。其中，主语是前述过程提到的参与人员，例如招聘人员、应聘者。谓语给出需要进行的操作，如发布职位、申请职位等。宾语是由谓语定义的操作的对象。例如，应聘者应聘职位中的职位。

将这些事件按照上述规则做以简单的整理后，形成如下所示的外部事件清单和内部事件清单。

外部事件清单：

(1) 应聘者申请职位；

(2) 应聘者查找职位；

(3) 应聘者填写简历；

(4) 招聘人员筛选应聘者；

(5) 招聘人员发布职位；

(6) 招聘人员更新职位；

(7) 招聘人员删除职位；

(8) 人力资源部制订招聘流程；

(9) 人力资源部安排新入职员工上岗；

(10) 管理人员生成常用报表；

(11) 招聘人员浏览应聘者信息；

(12) 招聘人员面试应聘者；
(13) 管理人员设定权限；
(14) 管理人员分配角色。
内部事件清单：
(1) 系统生成统计报表；
(2) 系统发送面试通知。

3. 关注每个事件

事件的提取并不只停留在表述的这一个层面上。系统分析人员还需要对事件做进一步的分析。当项目小组在一次会议中提出事件后，需要尽快地在另一次会议中添加事件的相关信息，并将事件放置到事件表中。

事件表是确定项目前景中第一批非常关键的图表之一。事件表识别出事件的附加信息，可能为构建系统的其他领域的内容提供重要的输入。一个事件表包括行和列，事件表中的每一行记录了一个事件的详细信息。表中的每列代表了事件的一个关键信息。被填入事件表中的信息包括主语、谓语、宾语、到达方式、响应。其中，主语描述的是事件的发起者是谁；谓语描述的是引发事件的动作；宾语描述的是事件发生所针对的对象；事件的到达方式则描述了事件发生的频率是有规律的(周期式)还是随机的(阵发式)；响应描述的是事件发生时，系统为应对发生的事件所必须做的相应处理。

表 3-8 列出了示例项目中的事件信息。

表 3-8 示例项目中的事件信息

主　语	谓语	宾　语	到达方式	响　应
应聘者	申请	职位	阵发式	系统告知招聘人员应聘信息并进行记录
应聘者	查找	职位	阵发式	系统根据应聘者查找的条件提取职位信息列表
应聘者	填写	简历	阵发式	系统编辑并保存应聘者的简历信息
招聘人员	筛选	简历	阵发式	系统根据招聘人员的筛选条件提取应聘者信息
招聘人员	发布	职位	阵发式	系统编辑职位信息并保存在系统中
招聘人员	更新	职位	阵发式	系统修改职位信息，并更新原有信息
招聘人员	删除	职位	阵发式	系统删除对应职位信息
人力资源部	制订	招聘流程	阵发式	系统编辑并设定招聘的执行流程并进行保存
人力资源部	安排	上岗	阵发式	系统登记新员工信息，并分配上岗部门
管理人员	生成	常用报表	阵发式	系统根据统计条件提取数据信息并形成报表
招聘人员	浏览	应聘者信息	阵发式	系统提取应聘者简历信息
招聘人员	面试	应聘者	阵发式	系统保存面试结果信息
管理人员	设定	权限	阵发式	系统编辑并保存权限信息
管理人员	分配	角色	阵发式	系统将用户对应的角色信息保存在系统中
系统	生成	统计报表	周期式	系统产生统计报表
系统	发送	面试通知	周期式	系统根据筛选结果统一发送面试通知邮件

事件表的获得使得系统分析人员对于系统需要提供的功能有了一个初步的了解。但并不能说每个事件就对应一个功能，或者说是每个事件对应一个用例，系统分析人员还需要进一步对事件进行分析和归类。对于如何从事件表向用例模型进行转换，将在 3.8.2 节中介绍。

4. 业务规则的识别和分类

在事件搜集的过程期间，业务规则经常会被提出来，它将始终贯穿于整个项目的始终，鉴于它的重要性，业务规则将被单独列出，那么如何对业务规则进行识别呢？下列规则的分类有助于系统分析人员识别规则。

业务规则分类大致如下。

1）结构性事实

在业务背景下，必须具有的业务信息的组成形式为结构性事实。要求这些事实或条件为真的情况下，才能进行下一步的工作。例如，每个订单都必须有一个订单处理日期，否则该订单不能称为有效订单。

2）限制性操作

根据其条件来禁止一个或多个操作。例如，进行注册时，通过员工号来确定用户身份，也就是说，非公司员工不允许使用本系统。

3）触发操作

在其条件为真时触发一个或多个操作。例如，对于图书馆借阅时间的监控，如果某个读者的借阅时间距离规定期限还有三天，系统将自动生成催还通知单；逾期未还，则系统连续三天再次发送催还通知单，如若其间归还图书或三天后归还，则结束发送。

4）推论

如果某些事实为真，则推出一个结论或了解一些新的知识。例如，如果新员工进行登记注册后，则拥有了默认的访问网站的用户名和密码。

5）推算

即进行计算的公式。例如，为了鼓励大家进行图书共享，制定优惠政策奖励提供共享图书者。默认情况下，借阅图书数量最多不超过两本。如果该读者共享图书一本或两本，则可多借一本，如果共享图书 3 本或 4 本，则可多借两本，以此类推。

这些分类不是一成不变的，但它们提供了一种收集业务规则的方法。如果需要，系统分析人员还可以添加额外的分类，以便提供更细的粒度。

在整理和认识业务规则的同时，还应注意到一种情况，那就是业务规则的实施往往依附于业务本身的流程。而这种业务流程在有了计算机系统的参与之后可能会发生变化，也就是说，原有系统的业务流程有可能与当前系统的业务流程不一致，这就是所谓的业务流程再造。

5. 业务流程再造

业务流程再造已经成为许多新的信息系统的来源。在许多系统开发项目中，通过提高新的自动化水平来支持现有的商业过程。而在其他的一些项目中，新系统支持一种新的商业服务或业务。

系统分析人员会发现有时他们必须也要成为商业分析员，并且需要根据新的、创新的信息系统来帮助客户重建所有内部商业过程。必须牢记一点，在项目进行期间有可能发现改进商业过程的机会，也有可能对商业过程的某些领域进行彻底地重新评估。作为一个系统分析员，经常有责任在项目进行期间识别和建议这样一些类型的改进方法。系统分析人员可以对公司的不断成功发挥巨大影响，甚至使得公司在激烈的竞争中生存下来。

在这方面，一个典型的案例就是网上拍卖系统，它使得传统的拍卖过程由于信息化的参与完全改变了其原有的业务方式。但是其间所固有的业务规则却并没有改变。这使得拍卖这个领域迅速扩大和发展起来。一些不便于在拍卖交易所进行的民间自发组织的日常用品的交易得以实现。

3.8.2 从事件表转换成用例

下面将着重介绍如何从事件表转换成用例。事件表，按照主语(参与者)进行分类，整理后获得的部分事件表如表 3-9 所示。系统分析员通过查找在事件表中的主语所在列，分析谁或什么在使用这个系统以及它的用途是什么，可以建立一个参与者列表。注意系统中同一个人可能充当多个特定的角色。

一般情况下，一个事件对应一个用例，但注意表中有些事件具有群集的趋向，例如，处理职位信息维护的一些事件，包括发布职位、删除职位、查询职位等。但有时，一个事件可能产生多个用例，或多个事件表示同一个用例。

系统分析人员应该保持这些自然的分组，并各加一个简略的描述性短语，再提出下列问题做以检查(验证群集分类的正确性)。

(1) 这些事件的共同点是什么？

(2) 这些事件有相同的最终目标吗？倘若有，目标是什么？

表 3-9 示例项目部分事件表

主 语	谓语	宾 语	到达方式	响 应
应聘者	申请	职位	阵发式	系统告知招聘人员应聘信息并进行记录
应聘者	查找	职位	阵发式	系统根据应聘者查找的条件提取职位信息列表
应聘者	填写	简历	阵发式	系统编辑并保存应聘者的简历信息
招聘人员	筛选	简历	阵发式	系统根据招聘人员的筛选条件提取应聘者信息
招聘人员	发布	职位	阵发式	系统编辑职位信息并保存在系统中
招聘人员	更新	职位	阵发式	系统修改职位信息，并更新原有信息
招聘人员	删除	职位	阵发式	系统删除对应职位信息
招聘人员	浏览	应聘者信息	阵发式	系统提取应聘者简历信息
招聘人员	面试	应聘者	阵发式	系统保存面试结果信息
人力资源部	制订	招聘流程	阵发式	系统编辑并设定招聘的执行流程并进行保存
人力资源部	安排	上岗	阵发式	系统登记新员工信息，并分配上岗部门
管理人员	生成	常用报表	阵发式	系统根据统计条件提取数据信息并形成报表
管理人员	设定	权限	阵发式	系统编辑并保存权限信息
管理人员	分配	角色	阵发式	系统将用户对应的角色信息保存在系统中
系统	生成	统计报表	周期式	系统产生统计报表
系统	发送	面试通知	周期式	系统根据筛选结果统一发送面试通知邮件

通过表 3-9，可以很容易根据参与者将事件分为 5 大类：应聘者、招聘人员、人力资源部、管理人员、系统。应聘者的操作大多数集中在职位信息的查找和申请上，而招聘人员则集中在职位信息的维护和筛选应聘者上。同时，管理人员对权限进行管理，还需要获得一些统计信息。系统作为当前正在研究的对象，不适合充当用例模型中的参与者。分析其产生

事件的特点，将会发现这些事件都是由特定时间触发的内部事件。因此，在这里应将“系统”替换为“时间”作为参与者。

接下来，将这些描述性的短语放在一个椭圆中，并加上相关的参与者，这就形成了初步的用例图，如图 3-6 所示。

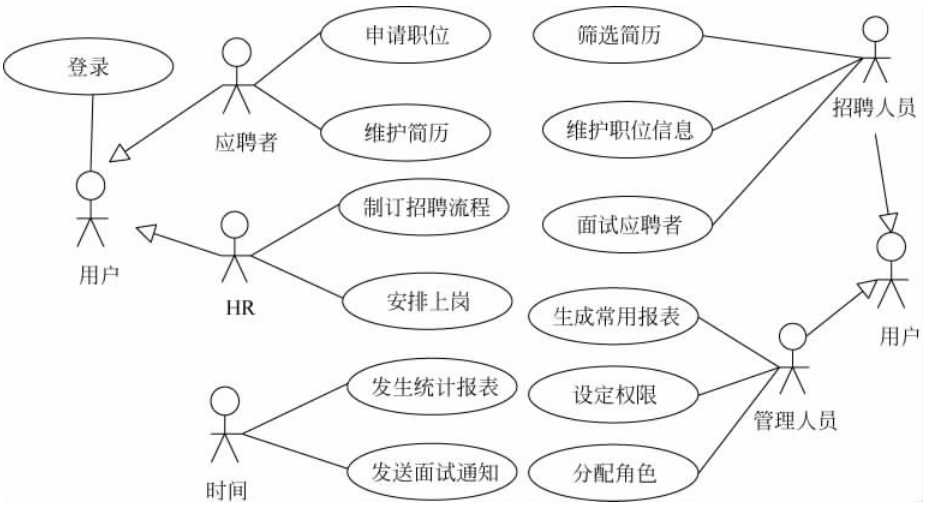

图 3-6 初步用例图

在将事件表中的事件转换成用例的时候，一定要注意用例的特征，即是否达成了用户的一个使用目标。如果只是达成目标中的一个步骤，则不能将其视为一个用例。

概要用例图的绘制，只是为了方便对用例的识别和整理。下一步就需要根据事件表中响应所在列的内容，采用文本形式对识别出来的用例进行摘要式的描述，从而获得对用例的进一步认识。示例项目中的摘要式描述如下所示。

登录：设定使用权限。用户提供用户名和密码，系统根据注册信息进行验证，通过后根据用户权限显示主界面。

维护职位信息：对职位信息进行管理和维护。包括对发布职位、更新职位和删除职位。

申请职位：应聘者根据自己的需求查询职位信息。

筛选简历：招聘人员根据职位的要求，检索符合条件的应聘者信息。为了能够更多地了解应聘者，也可以查看其详细的个人信息。对于符合条件的应聘者，可以将其放入候选人员名单中，并在指定的时间向其发送面试通知邮件。

制定招聘流程：人力资源部对于不同岗位的人才需要制订不同的招聘流程。例如技术人员的招聘，需要在招聘流程中添加技术面试的环节；根据岗位需求的不同，决定是否需要添加背景调查环节。

安排上岗：人力资源部录入新员工信息，并根据招聘意向将新员工分配到相应的部门，并通知相关人员。

生成常用报表：常用报表包括候选人情况一览、招聘状态一览。

生成统计报表：每个固定周期，将会由系统自动招聘统计信息，用于招聘团队的绩效考核。

发送面试通知：在指定的固定时间将会向候选列表中的应聘者发送面试通知的电子邮件。目的是减少筛选过程中单独发送面试通知占用的时间。

初始阶段不需要以详述形式编写所有用例，在获得初步的系统用例之后，需要对这些用例进行优先级的排序。以此作为确定增量的基础。用例的优先级确定的标准，主要从业务价值、风险、架构意义这三个方面进行考虑。

首先是高业务价值的用例。用户应当列出用例的业务优先级。一般分为三级，首先要实现的、短期内可以没有、长期内可以没有的。首先要实现的这部分用例的需求稳定性相对较高，因为是用户业务中的核心功能，用户对于要完成的目标比较明确。例如，职位的发布和申请就是当前系统的核心业务。

其次，具有高架构意义的用例。高架构意义是指被其他的用例多次引用的用例。对于每一个用例，开发人员都应考虑体系结构(或技术上)的风险。具有架构意义的用例往往处在系统的核心环节当中。对这部分用进行细致分析将有助于为系统获得解决方案提供帮助。例如，为了能够在海量信息中做到精确定位、快速检索，筛选简历用例就是解决这个问题的关键。筛选简历的过程和形式确定下来之后，整个系统就以此为基础，架构的形式就基本明晰了。

最后是高风险的用例。风险性越高，说明风险可能发生的概率就越高，马上着手对这部分内容进行分析可以及早解决问题，减弱对系统造成的可能影响。在示例项目中，统计信息的实现策略是使用图表的形式进行实现，那么如果选用 JFreeChart 作为实现技术，则必须考虑其可能存在的风险，尽快做出一个原型则是化解风险的一种手段。

到目前为止，大家已经将注意力集中到优先级最大的需求上，使得项目由粗略向精细演进。为此，根据上述三方面的考虑，此示例研究在该阶段的用例模型如表 3-10 所示。

表 3-10 用例模型的初步规划

摘 要 形 式	非正式形式	详 述 形 式
维护简历	制订招聘流程	申请职位
安排上岗	生成统计报表	发布职位
分配角色	发送面试通知	筛选简历
设定权限	生成常用报表	录入简历
管理用户	更新简历	面试应聘者
……	……	……

1. 非正式形式的样例项目用例

在进行需求讨论会时，对具有高优先级的用例进行讨论，需要尽快拿出非正式形式的用例描述，用以记录用户需求的核心内容。非正式形式的用例描述侧重于描述为了达到某个业务目标，用户与系统之间是如何进行交互的，需要用户提供的信息有哪些，用户希望系统做出什么样的处理，也就是系统的响应是什么。在这期间，还需要描述可能出现的其他情形，可能出现的必须处理的异常。

如下给出示例项目的非正式形式的用例描述的例子。

用例 UC1：登录

为了保证招聘工作的有效性和准确性，系统的使用需要进行用户身份验证，通过登录来验证系统使用者的身份合法性和相应的使用权限。

基本流程：

1. 用户在登录页面输入用户名密码并提交。
2. 系统检验该用户为系统有效用户。
3. 系统记录入口事件日志。
4. 系统认定用户的身份是应聘者，系统显示应聘主页面。

分支流程：

1.a 如果用户单击登录页面上的“注册”链接，则系统转入注册用例。

2.a 如果用户名正确但密码不正确，系统再次显示登录页面。

2.b 如果用户名不存在，系统将显示注册页面。

4.a 系统认定用户身份是招聘人员，系统显示招聘主页面。

4.b 系统认定用户身份是系统管理员，系统显示系统管理员主页面。

4.c 系统认定用户身份是 HR，系统显示人力资源主页面。

用例 UC5：申请职位

当应聘者获得了职位列表或是看到了职位的详细信息之后，应聘者可以申请该职位。

基本流程：

1. 应聘者在职位详细信息页面单击“申请”链接。
2. 系统确认应聘者已经填写过个人简历信息。
3. 系统显示申请职位页面。
4. 应聘者确认职位申请信息，提交申请请求。
5. 系统确认该应聘者没有申请过当前职位。
6. 系统保存应聘信息。
7. 系统将应聘者的状态从还未申请状态修改为还未面试状态。
8. 系统重新显示应聘者主页面。

分支流程：

2.a 如果应聘者没有填写过履历信息，则转入填写简历用例。

5.a 如果应聘者曾经应聘过该职位，但还没有被取消资格，即没有设定为取消状态，那么系统将会拒绝该应聘者的请求，并给出提示“您已经申请了该职位，请耐心等待！”

5.b 如果应聘者曾经应聘过该职位，并且没有通过面试，那么系统将会拒绝该应聘者的请求，并给出提示“您目前还不适合当前职位，请选择其他职位！”

备注：

应聘者在申请了一个职位之后必须等待该职位的处理结束后才能够应聘其他职位。这意味着应聘者在同一时间内只能申请一个职位。

2. 详述形式的样例项目用例

一般情况下，对于用例的描述采用非正式形式就可以了，但是在需要对一些细节问题进行记录和加以强调的时候就需要使用详述形式了。下面给出一个详述形式的例子，从中可注意到里面的核心内容就是前面非正式形式中给出的基本流程和分支流程。

用例 UC5：申请职位

参与者：应聘者

前置条件：应聘者已经拥有自己的注册账号，并已经通过身份认证。

后置条件：系统中增加一条职位申请信息。

用例概述：当应聘者获得了职位列表或是看到了职位的详细信息之后，应聘者可以申请该职位。

基本流程：

1. 应聘者在职位详细信息页面单击“申请”链接。
2. 系统确认应聘者已经填写过个人简历信息。
3. 系统显示申请职位页面。
4. 应聘者确认职位申请信息，提交申请请求。
5. 系统确认该应聘者没有申请过当前职位。
6. 系统保存应聘信息。
7. 系统将应聘者的状态从还未申请状态修改为还未面试状态。
8. 系统重新显示应聘者主页面。

分支流程：

2.a 如果应聘者没有填写过履历信息，则转入填写简历用例。

5.a 如果应聘者曾经应聘过该职位，但还没有被取消资格，即没有设定为取消状态，那么系统将会拒绝该应聘者的请求，并给出提示“您已经申请了该职位，请耐心等待！”

5.b 如果应聘者曾经应聘过该职位，并且没有通过面试，那么系统将会拒绝该应聘者的请求，并给出提示“您目前还不适合当前职位，请选择其他职位！”

备注：

应聘者在申请了一个职位之后必须等待该职位的处理结束后才能够应聘其他职位。这意味着应聘者在同一时间内只能申请一个职位。

特殊需求：

系统能够自动记录应聘者的申请时间。

发生频率：阵发式

3.9 补充性规格说明

用例不是需求的全部，补充性规格说明基本上是用例之外的所有内容。它主要用于非功能性需求，例如，性能、可支持性说明。该制品也用来记录没有表示（或不能表示）为用例

的功能特性，它是获取理解系统的必要内部行为所需的其他信息。下面从几个方面给出补充性规格说明的实例。

1. 功能性

1）日志和错误处理

在持久性存储中记录所有在运行期间系统捕获到的错误，同时记录系统的日常业务处理轨迹。

2）可扩展性

在几个用例的不同场景执行任意一组规则，以支持对系统功能的定制。

3）安全性

系统的任何使用都需要经过用户身份认证，根据授予的权限进行操作。

4）保留原有系统数据库信息

原有招聘管理系统的应聘人员信息、职位信息等数据能够平滑地导入到当前系统中，制定相应的导入机制和备份机制。

2. 实现约束

项目组的成员大都熟悉 Java 技术，而且该技术具有良好的可移植性。

3. 接口

1）重要的硬件接口

无。

2）软件接口

① 与各种职位发布渠道的接口。

② 公司内部人力资源管理系统的接口。

3.10 案例分析

项目小组的成员在拿到用户需求说明之后，开始着手理解需求，构建用例模型。

3.10.1 背景说明

斯科特信息技术有限公司（虚拟化名）创建于 1993 年，公司一直致力于为全球客户提供世界领先的信息技术、研发和业务流程外包服务，客户广泛分布在软件业、硬件业、金融业、通信业、医药和制造业等领域，重点集中在财富 500 强企业，在全球设有 50 个办公机构，共有员工两万五千多人。公司关注人力资源的发展，拥有一批极富潜力、训练有素的员工，为客户提供涵盖整个应用服务生命周期的服务，主要业务包括企业应用服务（应用开发与维护，质量测试），企业套装解决方案（Siebel 解决方案及支持，Oracle ERP 解决方案及支持服务）、产品工程服务（产品开发和测试，产品全球化服务），以及技术和解决方案服务（技术资源服务）。该公司希望信息化的管理手段使公司对人才的把握力度不断扩大，招聘职能人员

可以方便地掌握求职者信息，利用成熟的招聘流程，使招聘工作更加高效有序。

伴随着公司的快速发展、转型升级，人力资源的及时合理配备成为发展的重要因素；招聘部门在招聘过程中招聘渠道单一、分散，面对海量简历无从下手，难以从中找到合适的人才；人力资源信息安全滞后，没有建立专业的人才库，导致企业内部矛盾时常发生；人力资源管理流程不规范，成本不断上升，无法合理应用组织管理与运用人力。这些已存在或即将出现的问题促使公司急需开发出一套适用于外包企业发展的招聘管理系统。

国内当前使用比较广泛的招聘管理系统并不能适应于外包行业的发展特点，人是外包行业最大的财富，如何利用现有资源最大地节省招聘开支，用合理的招聘流程加快招聘进程，规范整合企业已有人力资源，缓解企业人才供需矛盾，成为企业发展的重要瓶颈。

3.10.2 项目说明

针对软件外包企业在招聘过程中遇到的问题，企业需要一套招聘软件去充分整合企业的资源，从而达到最大化地提升企业在激烈的人才市场中竞争力的目的，同时让求职者走近并且真实地了解企业，以此来帮助企业吸引更多的优秀人才。招聘软件除优化招聘环节外，还需要提升企业业务部门和招聘职能部门在不同地区间的互动与协作，提高招聘效率，降低招聘成本。要求系统涵盖从招聘需求产生，到招聘信息发布，招聘渠道管理，候选人交互，测评，甄选，录用的所有环节。

用户的期望如下。

1. 人才来源——多渠道、全方位

为最大限度地做好人才吸引工作，要求招聘辐射 HR 上传简历、公司官网、公司内部员工推荐、招聘门户网站（如前程无忧、智联招聘、中华英才等）等多种渠道。多渠道来源的简历纷繁多样，传统的招聘方法在收集简历、甄选简历环节耗费了大量的人力、物力。因此要求系统能够自动整合所有渠道的简历，将所有进入系统的简历进行标准化处理，将不同渠道不同形式的简历制作成统一的简历详细页面。

要求人才库中的对象是以人为单位，需要处理好候选人与简历、简历与职位之间的关系，候选人与简历形成一对多的关系。

需要开发企业外部简历投递页面，候选人将简历投递到本系统，并关联意愿职位；候选人填写必需的基本信息，如姓名、手机号码、邮箱地址等；候选人可以选择两种简历投递方式：提交附件，或者详细填写个人工作经验、项目经验等详细信息。

候选人能够通过外部招聘网站投递简历和关联意愿职位（如前程无忧、智联招聘等，可通过建立假设的简历接口来进行模拟）。

除外部来源的简历外，系统提供简历上传的功能供招聘人员使用；基本逻辑应该能够和企业外部简历投递部分复用。

由于简历来源多样化，很可能出现同一个候选人从多来源投递简历的情况，因此需要能够识别同一候选人的不同简历，避免重复招聘的发生。

2. 找简历——精确定位、快速锁定

面对海量信息，招聘人员常感觉无从下手，因此要求系统能够提供强大的检索功能。

能够对海量信息(简历数在50万左右,需要制造测试数据)进行多维度、多条件、多关键字的查询。

简历搜索能够将候选人基本信息(如姓名、性别、学历、候选人状态等)和简历信息结合进行,如搜索“本科学历,已录用,且简历中包含‘Java’关键字”的简历。

显示简历搜索结果时的页面响应需要在5s内。传统的数据库查询方式对于数十万条简历信息进行关键字模糊查询的效果会非常不理想,需要有切实有效的解决方案。

3. 人才资源池——避免人才浪费、降低成本

软件外包企业应该先要有足够的人才储备。招聘团队需要构建大型的人才资源池。

人才资源池中的候选人可以来自于公司内部处于闲置状态的员工,外包行业由于项目周期原因,在项目内部会有周期性闲置,当其他项目有招聘需求时,招聘团队可以优先考虑人才资源池中的合适候选人,做到最大化地发挥公司现有资源,降低招聘成本。同时避免员工因项目闲置而产生的倦怠情绪,增强企业归属感。

人才资源池中的候选人也可以来自于外部投递进入的简历。

需要设计进入资源池的逻辑,即满足何种要求的候选人能够进入,以及满足何种要求时退出资源池;并考虑自动和手动两种情况。

对于外部投递进入的候选人进入资源池的条件,要求能够和招聘过程中的面试评价体系相结合。

4. 职位发布——自动、统一

多来源的简历需要依靠多来源的职位发布。传统的招聘系统不能做到职位的自动发布更新,需要招聘团队购买多个招聘门户网站账号。这一方面增加了招聘成本,另一方面也不能保证职位发布的统一性,不利于企业形象的构建与提升。

要求系统能够做到一个平台,多种渠道。仅利用招聘系统这一个平台对职位进行发布、更新、删除操作,即可对各种职位去向渠道产生影响。

要求系统能够及时反馈同步情况,以便招聘人员对职位进行下一步处理。

5. 招聘管理——灵活、共享

考虑到人才招聘的重要性和复杂性,需要制订详细的人才招聘计划,对于不同岗位的人才也需要制定不同的招聘流程。比如技术人员的招聘,需要在招聘流程中添加技术面试环节;针对客户需求,决定是否需要添加客户面试环节;根据岗位需求不同,决定是否需要添加背景调查环节等。

要求系统能够提供可配置的招聘流程,且招聘进度清晰可见,招聘内容可共享。

招聘人员能够及时分享候选人,通过邮件发送等方式共同探讨候选人情况。

能够形成全面的“评价体系”,让每个招聘人员都能够了解候选人的情况,避免重复工作。

为加强企业与应聘者的沟通,希望在招聘过程中能够加强与候选人的联系,提升企业形象,比如进入面试环节的短信/邮件发送,面试失败时的婉拒邮件等。

6. 报表管理——清晰、便捷

要求系统能够提供常用报表和自定义报表,用于招聘团队绩效考核、候选人情况一览、

招聘状态一览等。

报表可根据权限进行有条件的下载。

可自定义报表，选择需要统计的对象、字段后自动生成报表。

可根据报表形成 Dashboard，给用户最直观的印象。

7. 个人工作管理——简单、安全

系统提供个人工作台，整合当前用户的所有工作情况。

要求在个人工作台中展示当前用户所关注的候选人、招聘各阶段工作等情况一览。

要求提供个人工作交接服务。

8. 权限管理——用户权限、角色管理安全

要求系统具备必要人员控制、权限控制、角色控制，由管理员为职能人员分配权限，让招聘工作更加安全、高效。

3.10.3 用例模型

根据前面的项目说明对系统的用例进行整理，形成系统级用例，如图 3-7 所示。

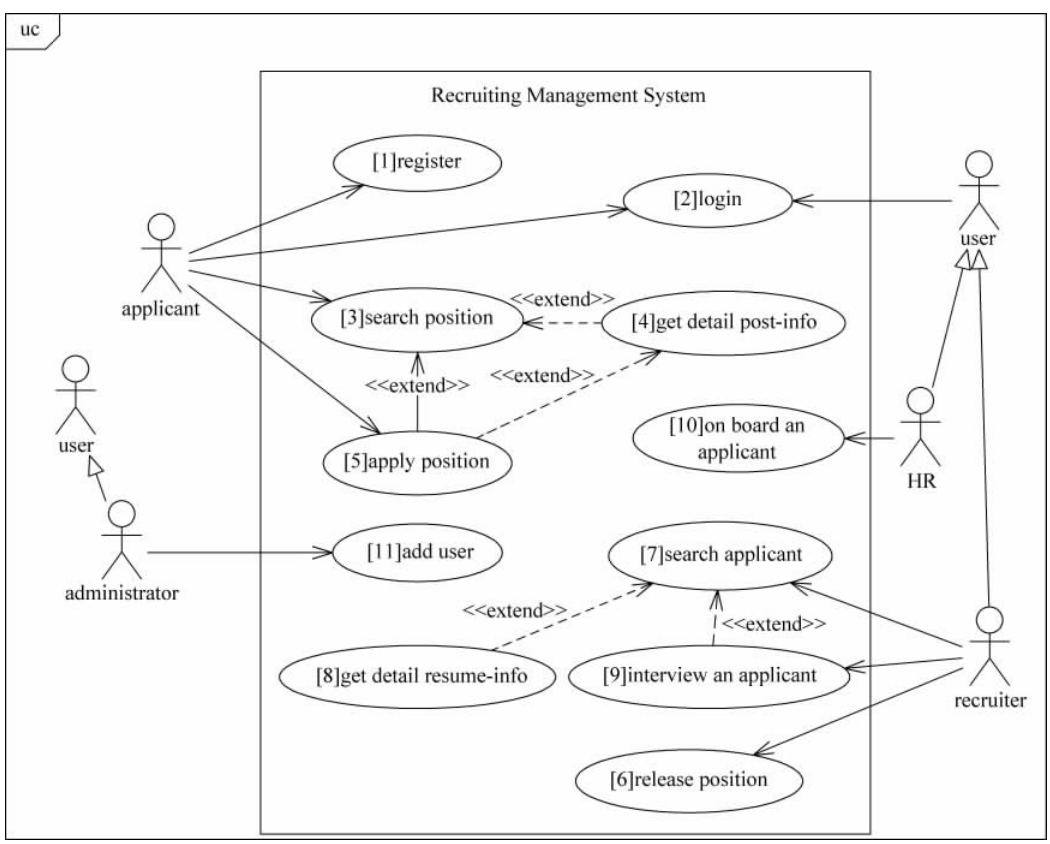

图 3-7 整理后的招聘管理系统用例图

由于篇幅所限，这里只列出部分用例的用例描述。

1. 用例：UC1 注册

参与者：应聘者。

概要描述：当一个应聘者试图访问东软的门户网站，他必须首先注册一个账户，并且填写个人简历。

基本流程：

1. 应聘者单击登录页面页面上的“注册”链接。
2. 系统显示注册页面。
3. 应聘者填写表单。
4. 应聘者提交表单。
5. 系统确认两个密码的一致性。
6. 系统确认用户名注册的唯一性。
7. 系统保存账户信息。
8. 设置应聘者的状态为“未申请状态”。
9. 系统显示登录页面。

分支流程：

(5a) 如果两个密码不一致，系统将给出“密码不一致”的提示，并显示注册页面。

(6a) 如果账户已经存在，系统将在注册页面上给出提示“该用户名已经存在”。

2. 用例：UC7 筛选简历

参与者：招聘人员。

概要描述：招聘人员选择职位，输入关键字，系统提供满足条件的应聘者列表。

基本流程：

1. 招聘人员进入招聘人员主页面。
2. 系统在招聘人员主页面上显示当前招聘人员发布的职位信息列表。
3. 招聘人员在招聘人员主页面上选择一个职位。
4. 招聘人员在招聘人员主页面输入查找资格关键字。
5. 系统将显示符合条件的申请了该职位的应聘者列表。
6. 招聘人员单击“详细信息”链接。
7. 系统执行“查看简历详细信息”用例。

分支流程：

6.a 招聘人员单击面试链接，系统执行“面试应聘者”用例。

3. 用例：UC6 发布职位

参与者：招聘人员。

概要描述：招聘人员在门户网站上发布职位信息。

基本流程：

1. 招聘人员单击招聘人员主页面上的“发布职位”链接。

2. 系统显示为招聘人员录入职位信息的发布职位页面。
3. 招聘人员输入职位信息并提交请求。
4. 系统保存职位信息。
5. 系统保存发布的日期。
6. 系统保存招聘人员与职位之间的关系。
7. 系统重新显示发布职位页面，并提示“职位发布成功!”。
8. 招聘人员在发布职位页面单击“返回”链接。
9. 系统显示招聘人员主界面。

分支流程：无。

3.11 知识拓展

3.11.1 需求分类的补充

前面介绍了需求分为功能需求和非功能需求，事实上，通过需求获取的不同渠道，也可以从下面几个方面进行分类。

1. 领域需求

领域需求的来源不是系统的用户，而是系统应用的领域，反映了该领域的特点。它们主要反映了应用领域的基本问题，如果这些需求得不到满足，系统的正常运转就不可能。领域需求可能是功能需求，也可能是非功能需求，其确定所需的领域知识。它经常采用一种应用领域中的专门语言来描述。

2. 业务需求

反映组织机构或客户对软件高层次的目标要求，这项需求是用户高层领导机构决定的，它确定了系统的目标规模和范围。

3. 用户需求

用户使用该软件要完成的任务。

4. 系统需求

容易被忽视的要求通常是为了保证整个系统能够正常运行的辅助功能，用户一般不会意识到。

事实上，不同类型系统的需求之间的差别并不像定义中的那么明显。若用户需求是关于机密性的，则表现为非功能需求，但在实际开发时，可能导致其他功能性需求，如系统中关于用户授权的需求。

以上软件需求的分类方法视不同类型的软件可能稍有差异。图 3-8 是软件需求各组成部分之间的一种常见关系。

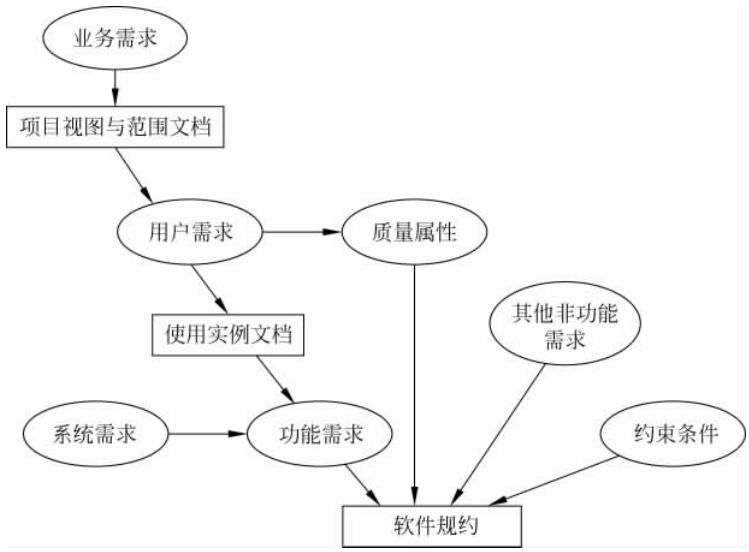

图 3-8 软件需求各组成部分之间的关系

3.11.2 需求开发过程

需求工程包括需求开发和需求管理两个方面。需求管理是一种系统化方法，可用于获取、组织和记录系统需求并使客户和项目团队在系统变更需求上达成并保持一致。需求开发是一个包括创建和维持系统需求文档所必需的一切活动的过程。它包含 4 个通用的高层需求工程活动：系统可行性研究、需求导出和分析、需求描述和文档编写、需求验证。图 3-9 说明了这些活动之间的关系，也说明了在需求开发过程的每个阶段将产生哪些文档。

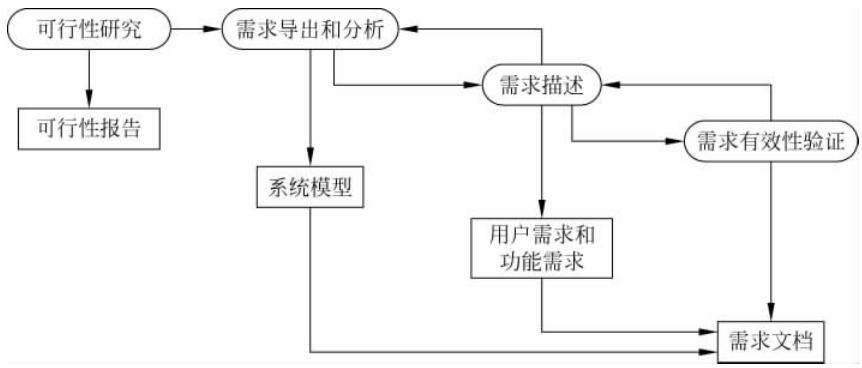

图 3-9 需求开发过程模型

(1) 可行性研究。它指明现有的软件、硬件技术能否实现用户对新系统的要求，从业务角度来决定系统开发是否划算以及在预算范围内能否开发出来。可行性研究是比较便宜和省时的。结果就是要得出结论：该系统是否值得进行更细致的分析。

(2) 需求导出和分析。这是一个通过对现有系统分析、与潜在用户和购买者讨论、进行任务分析等导出系统需求的过程，也可能需要开发一个或多个不同的系统模型和原型。这些都会帮助分析了解所要描述的系统。

(3) 需求描述。需求描述就是把在分析活动中收集的信息以文档的形式确定下来。在这个文档中有两类需求：用户需求是从客户和最终用户角度对系统需求的抽象描述；功能需求是对系统要提供的功能的详尽描述。

(4) 需求有效性验证。这个活动检查需求实现、一致和完备。在这个过程中，不难发现需求文档中的错误，然后必须加以改正。

当然，需求过程中的各项活动并不是严格按顺序进行的。在定义和描述期间，需求分析继续进行，这不排除在整个需求工程过程中不断有新的需求出现。因此，分析、定义和描述是交替进行的。

在初始的可行性研究之后，下一个需求工程过程就是需求导出和分析。在这个活动中，软件开发技术人员要和客户及系统最终用户一起调查应用领域，即系统应该提供什么服务、系统应该具有什么样的性能以及硬件约束等。

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讨、确定描述这些需求的多种解决方案。参与需求获取的人员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。

所有对系统需求有直接或间接影响力的人统称为项目相关人员。项目相关人员包括使用系统的最终用户和机构中其他与系统有关的人员、正在开发或维护其他相关系统的工程人员、业务经理、领域专家等。以下原因增加了系统需求导出和分析的难度。

项目相关人员通常并不真正知道他们希望计算机系统做什么。让他们清晰地表达出需要系统做什么是件困难的事情，他们或许会提出不切实际的需求。项目相关人员用他们自己的语言表达需求，这些语言会包含很多他们所从事的工作中的专业术语和专业知识。需求工程师没有客户的领域中的知识和经验，而他们又必须了解这些需求。不同的项目相关人员有不同的需求，他们可能以不同的方式表达这些需求。需求工程师必须发现所有潜在的需求资源，而且能发现这些需求的相容之处和冲突之处。政治上的因素可能影响系统的需求。管理者可能提出特别需求，因为这些允许他们在机构中增加他们的影响力。经济和业务环境决定了分析是动态的，它在分析过程期间会发生变更。因此，个别需求的重要程度可能改变。新的需求可能从新的项目相关人员那里得到。

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。需求开发活动通常包括如下 14 个步骤。

(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他非功能需求。

(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) 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象还未理解的需求。

(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。

需求导出和分析过程的通用过程模型如图 3-10 所示。

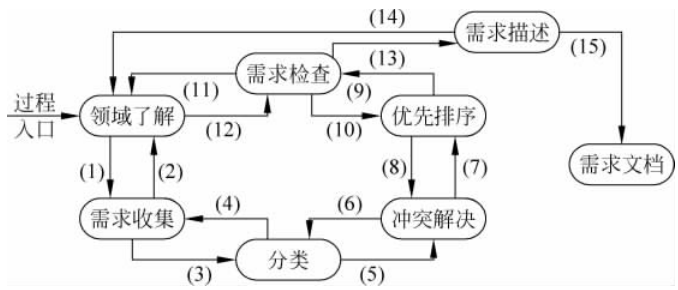

图 3-10 需求导出和分析过程

过程活动包括以下内容。

(1) 领域了解。分析人员一定要了解应用领域。举例来说,为一家超级市场做系统开发,则分析人员一定要了解超市场的运作方式。

(2) 需求收集。这是一个与项目相关人员沟通以发现他们需求的过程。很显然,在这个活动期间能对领域有进一步的了解。

(3) 分类。收集的需求是无序的,需要对其重新组织和整理,将其分成相关的几个组。

(4) 冲突解决。在有多个项目相关人员参与的地方,需求将不可避免地会发生冲突。这个活动就是发现而且解决这些冲突。

(5) 优先排序。在任何一组需求中,一些需求总是会比其他的更重要。这个阶段包括和项目相关人员交互以发现最重要的需求。

(6) 需求检查。检查需求是否完全、是否一致以及是否与项目相关人员对系统的期待相符合。

从图 3-10 可以看出,需求导出和分析是一个重复过程,从一个活动到另一个活动会有持续不断的反馈。过程循环从领域了解开始,以需求检查结束。分析人员在每个回合中都能进一步加深对需求的理解。

小结

俗话说,万事开头难。作为软件生命周期最重要的阶段之一,需求分析最根本的任务是确定用户到底需要一个什么样的软件系统。具体地说,是应该确定系统必须具有的功能和性质、系统要求的运行环境,通过分析得出系统详细的需求模型。

需求分析的结果是软件开发的基础,必须仔细验证它的正确性,开发人员必须同用户取得完全的一致,需求分析的文档也应被用户认可。但是这并不意味着随着项目的进展,需求

不会再发生变化。因此，为了更准确、更具体地确定用户的需求，往往通过原型法来加强同用户的沟通。此外，作为需求分析的结果，应该制订明确的软件需求规约，并有必要邀请多方人员对所描述功能的正确性、完整性和清晰性共同进行评审。

在构建需求模型的过程中，用例的观点和思维过程带给需求开发的变化比起是否画出正式的用例图显得更为重要。用例的获取、整理和细化是整个 RUP 过程中的一项重要的任务。RUP 过程是用例驱动的，用例的选取、细化将会对整个系统的后续工作产生重要的影响。在本章中给出了产生用例的步骤和方法。它以系统特性为基础，抽取对系统产生重要影响的事件，并将其描述在事件清单当中，细化事件清单中的事件形成事件表，在分析事件表的基础上抽取出用例，并对用例进行相应描述。

本章的重点在于描述用例的构建过程，而对于如何描述质量更好、更有效的用例不做深入探讨，如果想了解更多的内容，请参阅相关的参考资料。

强化练习

一、选择题（单选题）

1. 软件需求阶段的工作可以划分为以下 4 个方面：对问题的识别、分析与综合、制定需求规格说明和（　　）。

A. 总结　　B. 阶段性报告

C. 需求分析评审　　D. 以上答案都不正确

2. 各种需求分析方法都有它们共同适用的（　　）。

A. 说明方法　　B. 描述方法　　C. 准则　　D. 基本原则

3. 软件需求规格说明书的内容不应该包括对（　　）的描述。

A. 主要功能　　B. 算法的详细过程

C. 用户界面和运行环境　　D. 软件的性能

4. 需求分析产生的文档是（　　）。

A. 项目开发计划　　B. 可行性分析报告

C. 需求规格说明书　　D. 软件设计说明书

5. 需求分析中，分析人员要从用户那里解决的最重要的问题是（　　）。

A. 要让软件做什么　　B. 要给该软件提供什么信息

C. 要求软件工作效率如何　　D. 要让该软件具有何种结构

6. 需求规格说明书的作用不应包括（　　）。

A. 软件设计的依据

B. 用户与开发人员对软件要做什么的共同理解

C. 软件验收的依据

D. 软件可行性研究的依据

7. 需求分析的最终结果是产生（　　）。

A. 项目开发计划　　B. 可行性分析报告

C. 需求规格说明书　　D. 设计说明书

8. 下面不属于用例图作用的是(　　)。

A. 展现软件的功能　　B. 展现软件使用者和软件功能的关系

C. 展现软件的特性　　D. 展现软件功能相互之间的关系

9. 下面对参与者说法不正确的是(　　)。

A. 是系统的一个实体　　B. 也叫活动者

C. 在系统外部　　D. 与系统发生交互

10. 下面(　　)不属于参与者类型。

A. 人　　B. 设备　　C. 外部系统　　D. 交互对象

二、简答题

1. 需求阶段主要解决的问题是什么？该过程中需要经过哪些主要活动？每项活动的主要任务和目标是什么？

2. 请简单描述一下你对用例的理解。

3. 请根据本章中部分用例的摘要式描述，给出“安排上岗”用例的非正式形式。

4. 请根据课程设计中设定的项目，给出用例的摘要式描述。

5. 请根据第 4 题中整理出来的摘要式用例描述，选择其中的核心用例给出其非正式形式的用例描述。

6. 下面提供的是存在问题的用例描述，请将存在问题的地方修改过来。

用例：买东西

范围：采购应用系统

主执行者：顾客

(1) 顾客使用 ID 和密码进入系统。

(2) 系统验证顾客身份。

(3) 顾客提供姓名。

(4) 顾客提供地址。

(5) 顾客提供电话号码。

(6) 顾客选取商品。

(7) 顾客确定购买商品数量。

(8) 系统验证顾客是否为老顾客。

(9) 系统打开到库存系统的连接。

(10) 系统通过库存系统请求当前库存量。

(11) 库存系统返回当前库存量。

(12) 系统验证购买商品的数量是否足够。

……

7. 宾馆客房业务管理提供客房预订、预订变更、客房入住、退房结账、旅客信息查询几个方面的功能。订房人可以通过电话、短信、网络或面对面等方式预订客房。允许预订人根据自己情况的变化更改预订信息。旅客入住客房前需要出示证件并登记，并要预交一定的押金。旅客提交押金后，柜台工作人员将在计算机上登记旅客信息，分配房间，并打印旅客入住单，旅客持入住单到指定客房入住。旅客离开宾馆前需要退房结账。旅客或宾馆管理人员可以随时查询旅客或客房的入住信息。请建立该问题的用例模型。