第 3 章　需求分析与用例建模

UML 和使用 UML 的软件开发过程是两回事。UML 语言本身只是工具，面向对象的系统分析、设计方法才是面向对象开发的灵魂。系统分析顾名思义就是要发现系统需要做什么而不是怎么做。发现系统要做什么的过程分成两步。首先，明白用户到底需要系统做什么，即捕获用户的需求；其次，为满足用户的需要，系统应该做什么，即完成系统分析。本书将系统分析分为需求分析和面向对象系统分析两个过程。需求分析是软件生命周期中的一个重要阶段，也是开发系统首先要做的第一步工作。客户需求是系统开发的源泉，系统要做什么，都是由需求得来的。尤其是系统必须具有的功能和性能，系统要求的运行环境以及预测系统发展的前景。作为一种通用的建模语言，UML 适用于系统开发过程中从需求规约描述到系统完成后测试的不同阶段。UML 软件开发过程中的需求分析阶段，主要通过建立用例模型来描述系统的业务需求，必要时可以用用例规约或活动图来进一步描述相关用例的功能。本章重点是进行需求分析，指导学生识别系统的执行者，提取和发现业务用例，分析用例之间的关系，并建立系统的业务用例模型，然后用用例规约或活动图描述功能。

本章目的：

- 掌握客户需求分析的要点及需求分析规格说明报告的书写格式。
- 通过绘制用例图及其正文描述来完成客户需求分析的方法。
- 掌握 UML 的用例模型建模方法。
- 掌握活动图的绘制方法，并且能够绘制活动图。

经常听到学生抱怨，说学了 UML 不知该怎么用，或者画了 UML 图却觉得没什么作用。其实，就 UML 本身来说，它只是一种交流工具，它作为一种标准化交流符号，在面向对象分析和设计的过程中作为开发人员间，甚至是开发人员与客户之间传递信息的工具。另外，UML 也可以看做是面向对象思想的一种表现形式。下面通过实例讲解如何进行面向对象的分析与设计，如何更好地应用 UML 工具。面向对象分析与设计的第一步，就是了解用户需求，并将其转换为业务用例图。

3.1　客户需求分析

需求分析阶段是开发过程中第一重要的阶段，如果不能准确地理解客户需要什么，那么就无法构造出正确的系统。如果不了解客户的领域及客户需要解决的问题，那么所有的用例分析都无济于事。客户需求将决定整个项目要承办方具体做些什么，承办方只有明确了客户的需求，才能进行系统设计、编程、测试和维护等工作。在初始需求阶段，需要获得客户的业务模型，然后根据业务模型建立计算机模型。要建立一个符合客户需要的计算机系统，首要条件是完全彻底搞清楚客户的业务，而不是预先假设已有一个计算机系统，再让客户假想计算机系统

帮他们做什么。然而，实际情况是大多数情况下，用户并不清楚自己究竟想要什么。因此，期望仅仅依靠用户获得完整的需求是完全不现实的。所以，需求分析阶段，开发人员必须进行细致的调查研究，以便较好地理解客户的要求。将客户非形式的需求陈述转化为完整的需求定义，再由需求定义转换到相应的需求规格说明。

需求分析阶段的首要工作是要深入了解用户的实际工作领域，通过客户和软件开发人员之间的沟通，了解客户的需求。然后与问题领域专家讨论，分析问题领域的业务范围、业务规则和业务处理过程，明确系统的责任、范围和边界，确定系统的需求，并建立需求模型。在 UML 中需求模型使用用例图来表示。

3.1.1　系统调查

系统调查是系统开发过程中的基础工作，通常分为初步调查和详细调查，是一种十分有效的需求获取方法，也是系统开发不可缺少的过程和手段。需求获取技术包括用户访谈、用户调查、现场观摩用户的工作流程、观察用户的实际操作、考查文档、开需求讨论会、从行业标准及规范中提取需求、采用原型法和其他技术等。这个活动也称为信息收集或者数据收集。调查研究分为两个阶段，一是初步的调查，在可行性分析阶段进行，即先投入少量的人力对系统进行大致的了解，分析其开发的可行性；二是详细的调查，这是在系统分析阶段进行的，即在确定系统可行并立项后，投入大量的人力，展开大规模、全面详细的系统调查。

1. *初步调查*

开发单位接受任务以后，将对现行系统进行初步调查。其目的是对现行业务给出一个概括性描述。着重了解现场存在的主要问题，找到现行系统症结，获取足够的信息以协助制定待建系统的开发方案，决定待建系统是否能够立项。它作为新系统设计的出发点。

初步调查是接受客户提出建立新系统的要求后，系统研制人员与用户管理人员的第一次沟通。调查研究技术有：对现有文档、表格和数据库进行抽样；实地访问；观察工作环境；发调查表；面谈；原型化和联合需求计划等。

初步调查的范围是全方位的，需要对经济、技术、管理和开发环境等方面的内容进行调查。初步调查的重点是了解用户与现行系统的总的情况，现行系统与外部环境的联系，现行系统的现有资源，外界的约束条件等。具体来说了解如下内容：

（1）现行系统的概况。了解现行系统的规模、系统目标、发展历史、组织结构、管理体制、人员分工、技术条件、技术水平等。

（2）系统外部环境。现行系统和外部环境有哪些联系，哪些外部条件制约系统的发展。

（3）现行系统的资源。现行系统有哪些资源，信息系统的状况等。

（4）用户资源和要求。开发新系统用户可以提供的人力、物力和财力等情况，用户的时间要求、功能要求、非功能要求和开发目标等。

（5）现行系统存在的问题。在初步调查中可以设计一些调查表，通过这些调查表可以更好地收集一些信息。了解现行系统存在的主要问题。

详细调查是在系统初步调查的基础上进行的。详细调查与初步调查的对象与方法相同，只不过调查的内容要详尽。详细调查的目的是完整地掌握现行系统的状况，发现系统存在的问题和薄弱环节，为系统分析和建立新系统逻辑模型打下基础。详细调查研究需要做大量细致的实际工作，是一个相当长期的过程。

2. 详细调查

（1）详细调查的原则

在进行详细调查时可遵循以下的原则：

1）系统性原则。从整体出发，全面地对问题进行分析比较。调查从系统的总目标出发，逐步进行分解，逐步求精，逐步细化。即根据调查目的和调查目标，对各项问题进行分类，规定每项问题应调查收集的资料。

2）计划性原则。在调查前要列出详细计划，有针对性地对目标进行调查，做到有的放矢。可以列出调查表、通过拟定调查表，收集有关信息管理系统的基础资料。制定调查的工作进程、工作进度，可以使整个调查活动有节奏地进行，使每一位从事调查工作的人员行动有方向。对调查进度进行监督检查，可以及时发现问题，克服薄弱环节，保证整个调查顺利进行。

3）科学性原则。调查所得数据要进行科学分析，切忌主观臆断，只有坚持科学性原则，才能在错综复杂的环境中避免或减少调查失误，为信息系统的建立提供科学依据。

4）前瞻性原则。业务过程是不断变化的，用户的需求也是变化的。调查结果会随着经济环境、政治气候等诸多因素的变化而变化，在调查中要看到系统调查的可变性，否则就不能了解潜在的需求。

（2）详细调查的内容

开发整个组织的 MIS，应该坚持全面调查和重点调查相结合的方法。详细调查的内容分为全面调查内容和重点调查内容。

1）全面调查的内容。与初步调查一样，要了解现行系统的发展历史、现状、规模、经营状况、业务范围、与外界的联系、确定系统的边界；对系统的组织结构进行调查，了解各个部门的权限、职责、人员分工和关系等；了解系统的资源状况，现有系统的物资、资金、设备、建筑平面布局和其他的资源。如果有计算机配备，要了解计算机的功能、容量和外设等情况；了解系统的约束条件，如系统在资金、人员、设备、处理时间和方式等方面的限制条件和规定；系统目前运行的薄弱环节；系统目前的开发状况、投入的资金、人员等；各部门对现行系统和拟建系统的态度，是否满意以及满意的程度等。

2）重点调查的内容。详细调查的重点是业务流程以及数据的调查。在进行调查时，要弄清楚某项业务做什么？为什么做？由谁来做？在哪里做？何时做以及如何做等，在做的过程中产生了哪些数据。即：

What：做什么？已经做过了什么？遵循什么程序？

Why：为什么需要经过那些流程？能否改变这些流程？

Who：谁来做？谁负责执行系统中的各项程序？为什么要他来做？可否换别人来做？

Where：在哪里做？要执行的业务流程在哪里？为什么？它们可以在哪些地方执行？如果在其他的地方执行是否更有效？

When：什么时间做？程序什么时候执行？为什么在这个时候执行？是否有最佳的执行时间？

How：如何做？流程如何执行？为什么要这样执行？是否可以采用其他的方式将它做得更好？

数据调查的内容主要包括输入信息、输出信息、信息处理过程、存储方式、代码信息和信息需求调查等。

输入信息调查的内容包括输入信息的名称、使用的目的、收集方式、发生周期、信息量、

编码方式、保存期、相关业务、使用文字和其他。

输出信息调查的内容有：输出信息的名称、使用的目的、使用单位、发生份数、发送方式、使用文字、输出时间、输出方式以及其他方面。

信息处理过程调查的内容有：处理内容、处理周期、处理方法、处理时间、处理场所及其他。

存储方式调查的内容有：文件名称、保管单位、保存时间、总信息量、保密要求使用频率、删除周期、追加周期、增加、删除比率。

代码信息调查的内容有：代码名称、分类方式、编码方式、使用目的、起始码、终止码、未使用码、备码率、追加或废弃频率及其他。

信息需求调查的内容有：所需信息名称、需求目的、需求单位、需求者、时间和期限、所需信息的形式、信息表达的要求等。

（3）调查的方法与策略

在调查研究的过程中主要是对数据进行收集。数据收集工作量很大，因此，要掌握有关数据收集的一些知识，既要研究获取信息的渠道、数据的来源，又要研究调查的方法等。

1）获取信息的渠道。获取需求信息的渠道包括用户或客户、公司研发管理部门、公司技术管理部门、项目实施部门、营销管理部门、旧有系统的研发项目组、项目组内。

2）调查数据的来源。数据的来源有：

① 组织正式报告（对于手工系统）；

② 各种卡片、报表；

③ 会议决议；

④ 现行系统的说明性文件（局部计算机化的系统）；

⑤ 各种流程图；

⑥ 计算机文件（或数据库），系统的数据组织结构；

⑦ 组织外的数据来源；

⑧ 上级下达的各种文件和各项任务指标；

⑨ 与本单位密切相关的其他单位的有关信息。

3）调查的方法：

① 询问法。询问法（Questioning）是将所要调查的事项以当面、书面或电话等方式，向被调查者提出询问，以获得所需要的资料，它是市场调查中最常见的一种方法。通常应该事先设计好询问程序及调查表或问卷，以便有步骤地提问。询问法通过面谈、电话、邮寄资料以及留置问卷等方式进行调查。面谈调查是调查人与被调查人面对面地询问有关问题，从而取得第一手资料的一种调查方法。在面谈前要做好准备，设定面谈的对象、目标、问题等，做好面谈记录。这一方法具有回收率高、信息真实性强、搜集资料全面等优点，但所需费用高，调查结果容易受调查人业务水平和态度的影响。电话调查是通过电话向被调查人进行询问以获取相关的信息的方法。这种方式速度快、省时间、费用低，但由于通话时间不宜过长，因而不易收集到深层信息。邮寄调查是通过邮寄的方式来调查相关的信息的方法。这种方式调查区域广泛，调查成本低，真实性强，结果可靠。但回收率低，回收时间长，并且调查者难以控制回答过程。留置问卷调查是面谈调查和邮寄调查的结合，常采用调查表（Questionnaire）。

② 观察法。观察法（Observation）是由调查人员到各种现场进行观察和记录的一种调查方法。被调查者往往是在不知不觉中被观察调查的，总是处于自然状态，因此，收集的信息较

为客观、可靠、生动、详细。但这种方法一般只能观察到事实的发生，观察不到行为的内在因素，所需费用也大。

③ 实验法。实验法（Experimentation）是先在小范围内进行实验，然后再研究是否大规模推广的一种调查方法。如业务的吞吐量、各项工作的时间、费用等。这种方法比较科学，结果准确，但调查成本高，实验时间长。

④ 抽样调查法。从调查对象中选择部分作为样本，加以调查，再从调查结果推断出总体情况的调查方法。抽样（Sample）可以采用系统抽样（System Sample）、分层抽样（Stratified Sample）和随机抽样（Random Sample）等方式进行，样本的主要作用是它可以代表整体。

⑤ 查阅档案资料法。调查人员通过查阅企业的各种文档、表格和数据库，如企业的计划、财务记录与报表、各种档案、工作记录、汇报总结、统计数据、各种录音、录像资料、流程图、设计文档和教师操作手册等来获取所需的基本信息。

⑥ 联合需求计划。联合需求计划（Joint Requirement Planning，JRP）也有的称为联合应用开发（Joint Application Development，JAD），它是一个方法论，它将一个应用程序的设计和开发中的客户或最终用户聚集在一起，通过一连串的合作研讨会获得需求，也叫 JRP 会议。通常通过一个两至五天的集会，让开发者与顾客能够快速有效，而且深入地探讨需求并取得共识。具体结果是产生完整的需求文件。传统调查方法中，开发者通过一系列面谈或查阅资料等得到客户输入信息来调研系统需求，联合需求计划被认为其成倍地加快了开发的速度，并且增大了客户的满足感，因为客户参与了开发的全过程。

计划一个 JRP 会议包括 3 个步骤：

①选择 JRP 会议地点。JRP 会议应该在公司工作地点以外召开。在外面举行 JRP 会议，与会者的精力集中在与 JRP 会议有关的问题和活动上，避免他们在工作地点出现打断和分心。JRP 会议的典型房间布局如图 3.1 所示。

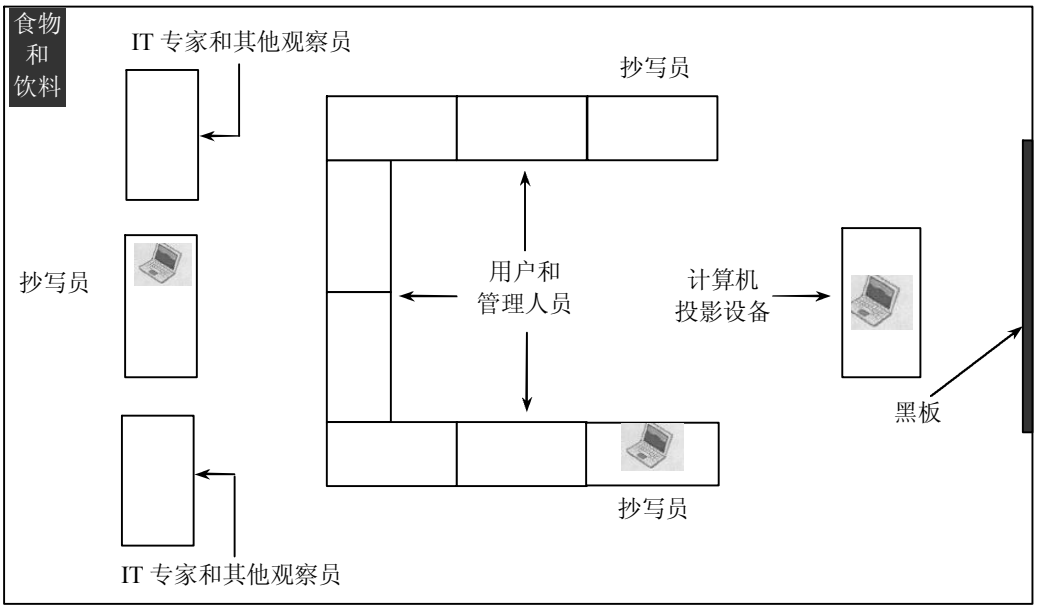

图 3.1　JRP 会议的典型房间布局

②选择 JRP 会议参加者。参加者包括 JRP 主持人、抄写员和用户团体代表。

③准备 JRP 会议议程。JRP 主持人必须准备材料，以简要地向与会者介绍会议的范围与目标。会议议程应该包括开始、主题和结论三个部分。开始部分用于交流会议的预期，沟通基本规则，影响或激发与会者参与；主题部分用于细化要在 JRP 会议中涉及的主题或问题；结论部分用于留出时间总结当天的会议，提醒与会者在会上没有解决的问题的讨论将继续进行下去。JRP 会议的成功很大程度上依赖于计划以及 JRP 会议主持人和抄写员的能力。

上述的调查结果如表 3.1 所示。

表 3.1　调查结果对照表

调查类型	收集资料
通过与用户交谈而收集到的信息	面谈记录、有效的调查问卷、观察记录、会议记录
现有的文档和文件	商业使命和战略声明、商业表格和报告及计算机显示样例、操作手册、作业描述、培训手册、现有系统的流程图和文档、咨询报告
基于计算机的信息	联合设计会议的结果、现有系统的 CASE 存储库内容和报告、系统原型的现实报告

此外，在调查时还可以采用快速应用开发（RAD）方法，以便获得更多的用户需求信息。

4）调查的策略。在进行调查研究时，通常不要直接进行面谈，而是先收集可以通过其他方法收集的信息。调查研究的策略是：了解现有文档、表格、报告和文件；如果合适，观察工作中的系统；根据已经收集到的信息，设计并分发调查表，澄清你没有完全理解的问题；进行面谈来验证和澄清最困难的问题；对于没有理解的功能需求或者需要被验证的需求，可以构造获取原型；使用合适的调查研究技术验证事实。这个策略并不是不可改变的，可根据实际情况选择调查策略。但基本思想是在面谈之前收集尽可能多的事实。

3.1.2　系统需求陈述

面向对象分析的第一步，就是获取用户的需求。需求调查的目的是通过各种途径获取用户的需求信息，由于在实际工作中，大部分客户无法完整地讲述其需求，因此，需求获取是一件看似简单，做起来却很难的事情。在需求获取过程中，主要需要弄清楚三个问题，即：明确需要获取的信息（What）、明确所获取信息的来源和渠道（Where）和怎样获取需求（How）。

1. 需求的获取

需求是客户在项目立项时就有的一个远景。客户需求将决定在整个项目中需要承办方具体做些什么，即承办方的任务。承办方在明确了需求后，就可以开始后期的设计、开发、测试、部署等工作。需求获取在软件工程中非常重要，因为后续的设计、开发等都基于软件需求。如果软件需求获取不正确或在需求开发过程中很多功能没有挖掘出来，那么在后期选择弥补时，将会造成项目延期以及成本大幅度增加的严重后果。软件需求分为三个层次。

业务需求：反映了组织机构或客户对系统高层次的目标要求。

客户需求：描述了用户使用产品所能完成的任务。

功能需求：说明了软件的功能，用户使用这些功能以完成任务。

要了解系统的需求，首先要获取客户的需求。在实际的工作中大部分的客户都无法完整地表述其需求。需求获取看起来是简单的事情，但是做起来却是非常难的工作。在进行需求获取时只要完成三方面的工作，具体如下：

（1）明确需要获取的信息（What）。需求分析师应在需求获取前明确需要获取的信息，以确保在实施需求获取时有的放矢。通常需要获取的信息有三大类：与问题域相关的背景信息（如业务资料、组织结构图和业务处理流程等）；与要解决的问题直接相关的信息；用户对系统的特别期望与施加的任何约束信息。

（2）明确所获取信息的来源和渠道（Where）。需求分析师还应确定获取需求信息的来源与渠道，以提高需求分析师在需求获取阶段的工作效率，使得所收集的信息更加有价值、更加全面。需求信息的来源有：客户的需求；竞争对手产品的优势和不足；国家政策、业务规则及相关行业标准；实施产品设计所需满足的需求；执行测试验证工作所需满足的需求；系统安装和维护所需满足的需求等。获取需求的渠道有：客户或用户；公司的研发管理部门；公司的技术管理部门；项目的实施部门；营销管理部门；旧系统的研发项目组和项目组内部等。

（3）如何获取需求（How）。项目经理应选择至少一种需求获取技术获取相关的需求，作为需求分析的依据。需求获取技术主要有：客户访谈；客户调查；现场观摩用户的工作流程；行业标准或规范中提取；文档考证；需求讨论会或原型法等。

2. 需求陈述的撰写

为获得正确的业务模型，要建立需求（场景）陈述。通常，需求陈述的内容包括：问题范围、功能需求、性能需求、出错处理需求、接口需求、约束、应用环境、假设条件及将来可能提出的要求等。总之，需求陈述应该阐明“做什么”而不是“怎样做”。它应该描述用户的需求而不是提出解决问题的方法。应该指出哪些是系统必要的性质，哪些是任选的性质。应该避免对设计策略施加过多的约束，也不要描述系统的内部结构，因为这样做将限制实现的灵活性。对系统性能及系统与外界环境交互协议进行描述。此外，对采用的软件工程标准、模块构造准则、将来可能做的扩充以及可维护性要求等方面进行描述。

书写需求陈述时，要尽力做到语法正确，而且应该慎重选用名词、动词、形容词和同义词。不少分析人员书写的需求陈述，都把实际需求和设计决策混为一谈。系统分析员必须把需求与实现策略区分开，后者是一类伪需求，分析员至少应该认识到它们不是问题域的本质性质。需求陈述可简可繁。对人们熟悉的传统问题的陈述，可能相当详细，相反，对陌生领域项目的需求，开始时可能写不出具体细节。因此，需要反复地与客户进行沟通。

绝大多数需求陈述都是有二义性的、不完整的、甚至不一致的。某些需求有明显错误，还有一些需求虽然表述得很准确，但它们对系统行为存在不良影响或者实现起来造价太高。另外一些需求初看起来很合理，但却并没有真正反映用户的需要。应该看到，需求陈述仅仅是理解用户需求的出发点，它并不是一成不变的文档。不能指望没有经过全面、深入分析的需求陈述是完整、准确、有效的。随后进行的面向对象分析的目的，就是全面深入地理解问题域和用户的真实需求，建立起问题域的精确模型。

系统分析员必须与用户及领域专家密切配合协同工作，共同提炼和整理用户需求。在这个过程中，很可能需要快速建立起原型系统，以便与用户更有效地交流。

下面阐述自动取款机 ATM 系统的需求陈述。

某银行拟开发一个自动取款机系统，它是一个由自动取款机、中央计算机、分行计算机及柜员终端组成的网络系统。ATM 和中央计算机由总行投资购买。总行拥有多台 ATM，分别设在全市各主要街道上。分行负责提供分行计算机和柜员终端。柜员终端设在分行营业厅及分行下属的各个储蓄所内。该系统的软件开发成本由各个分行分摊。

银行柜员使用柜员终端处理储户提交的储蓄事务。储户可以用现金或支票向自己拥有的

某个账户内存款或开新账户。储户也可以从自己的账户中取款。通常，一个储户可能拥有多个账户。柜员负责把储户提交的存款或取款事务输进柜员终端，接收储户交来的现金或支票，或付给储户现金。柜员终端与相应的分行计算机通信，分行计算机具体处理针对某个账户的事务并且维护账户。

拥有银行账户的储户有权申请领取现金兑换卡。使用现金兑换卡可以通过 ATM 访问自己的账户。目前仅限于用现金兑换卡在 ATM 上提取现金（即取款），或查询有关自己账户的信息（例如，某个指定账户上的余额）。将来可能还要求使用 ATM 办理转账、存款等事务。

所谓现金兑换卡就是一张特制的磁卡，上面有分行代码和卡号。分行代码唯一标识总行下属的一个分行，卡号确定了这张卡可以访问哪些账户。通常，一张卡可以访问储户的若干个账户，但是不一定能访问这个储户的全部账户。每张现金兑换卡仅属于一个储户所有，但是，同一张卡可能有多个副本，因此，必须考虑同时在若干台 ATM 上使用同样的现金兑换卡的可能性。也就是说，系统应该能够处理并发的访问。

当用户把现金兑换卡插入 ATM 之后，ATM 就与用户交互，以获取有关这次事务的信息，并与中央计算机交换关于事务的信息。首先，ATM 要求用户输入密码，接下来 ATM 把从这张卡上读到的信息以及用户输入的密码传给中央计算机，请求中央计算机核对这些信息并处理这次事务。中央计算机根据卡上的分行代码确定这次事务与分行的对应关系，并且委托相应的分行计算机验证用户密码。如果用户输入的密码是正确的，ATM 就要求用户选择事务类型（取款、查询等）。当用户选择取款时，ATM 请求用户输入取款额。最后，ATM 从现金出口吐出现金，并且打印出账单交给用户。

3.1.3　系统需求分析

从广义上来看需求分析包括需求的获取、分析、规格说明、变更、验证、管理等一系列需求工程。从狭义上来看需求分析是指需求的分析、定义过程。需求分析就是分析软件用户的需求是什么。在软件工程中，需求分析指的是在建立一个新的或改变一个现存的系统时描写新系统的目的、范围、定义和功能所要做的所有的工作。需求分析是软件工程中的一个关键过程。在这个过程中，系统分析员和软件工程师确定顾客的需要。只有在确定了这些需要后才能够分析和寻求新系统的解决方法。由于每个人对新系统的功能和特征都有自己的观点和期望，所以，调查研究活动通常会产生互相矛盾的需求。需求分析的目的就是发现和解决需求中的这些问题并达成一致意见。调查研究和需求分析活动联系非常密切，并且经常交织在一起。如果在调查研究的过程中获取的需求被查出存在问题，就需要分析员对这些问题进行专门的分析。

1. 需求分析的任务

需求分析阶段的工作包括定义需求、分析与综合、编制需求规格说明书和评审等。

（1）定义需求。定义需求就是从系统角度来理解软件，确定对所开发系统的综合要求，并提出这些需求的实现条件，以及需求应该达到的标准。这些需求包括：功能性需求和非功能性需求。功能性需求描述一个系统必须提供的活动和服务（做什么）；非功能性需求描述一个满意的系统的其他特征、特点和约束条件等，包括系统的性能需求（要达到什么指标）、环境需求（如机型，操作系统等）、可靠性需求（不发生故障的概率）、安全保密需求、用户界面需求、资源使用需求（软件运行所需的内存、CPU 等）、软件成本消耗与开发进度需求，以及预先估计以后系统可能达到的目标等。

（2）分析与综合。逐步细化所有的软件功能，找出系统各元素间的联系、接口特性和设

计上的限制，分析他们是否满足需求，剔除不合理部分，增加需要部分。最后，综合成系统的解决方案，给出要开发的系统的模型（做什么的模型）。

（3）编制需求规格说明书。描述需求的文档称为软件需求规格说明书，是需求分析阶段的成果，作为提交给下一阶段的文档资料。

（4）评审。对功能的正确性、完整性和清晰性，以及其他需求给予评价。评审通过才可进行下一阶段的工作，否则重新进行需求分析。

2. 需求分析的过程

为了更清晰地表示出需求分析的主要过程，用一个图来表示，如图 3.2 所示。

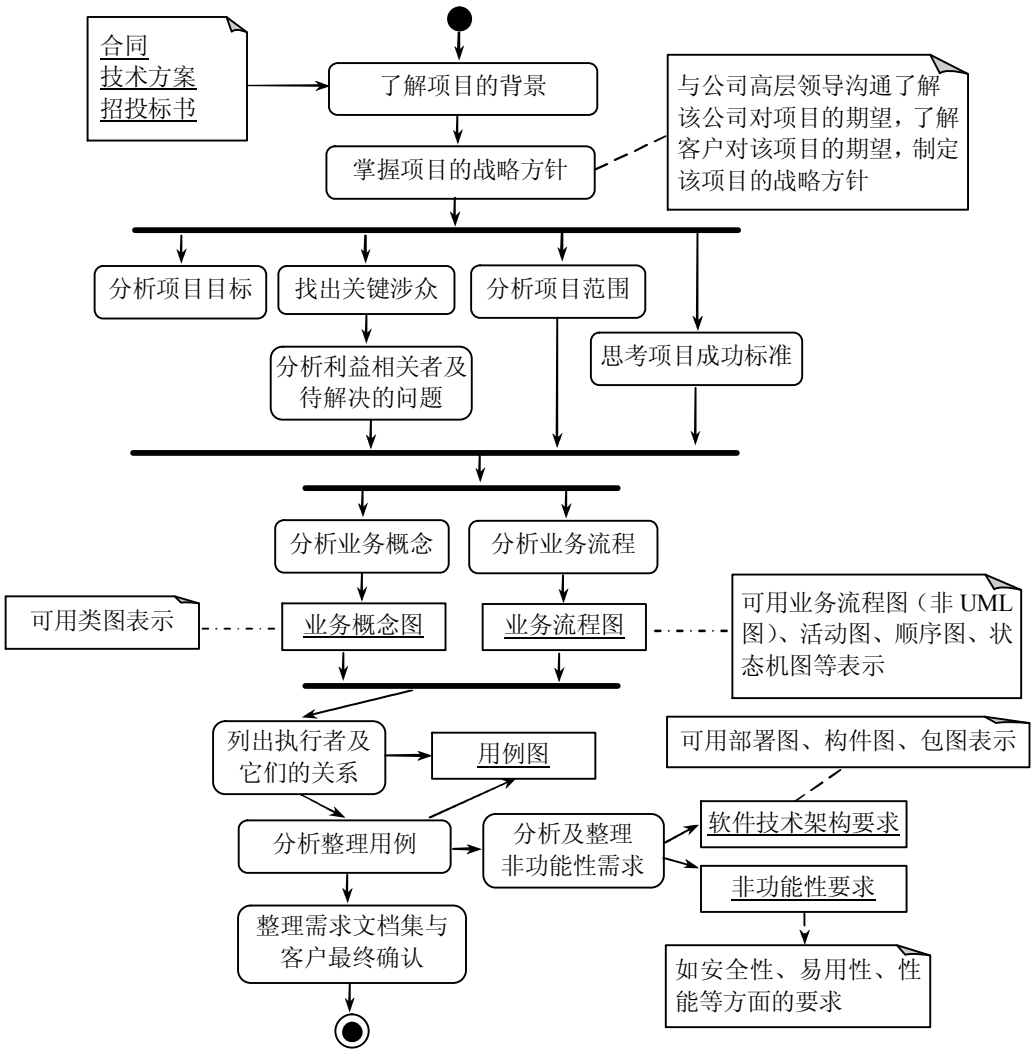

图 3.2　需求分析的过程

3. 需求分析的特点

需求分析是一项重要的工作，也是最困难的工作。该阶段工作有以下特点：

（1）用户与开发人员很难进行交流。在软件生存周期中，其他四个阶段都是面向软件技术问题，只有本阶段是面向用户的。需求分析是对用户的业务活动进行分析，明确在用户的业

务环境中软件系统应该“做什么”。但是在开始时，开发人员和用户双方都不能准确地提出系统要“做什么？”。因为软件开发人员不是用户问题领域的专家，不熟悉用户的业务活动和业务环境，又不可能在短期内搞清楚；而用户不熟悉计算机应用的有关问题。由于双方互相不了解对方的工作，又缺乏共同语言，所以在交流时存在着隔阂。

（2）用户的需求是动态变化的。对于一个大型而复杂的软件系统，用户很难精确完整地提出它的功能和性能要求。一开始只能提出一个大概、模糊的功能，只有经过长时间的反复认识才逐步明确。有时进入到设计、编程阶段才能明确，更有甚者，到开发后期还在提新的要求。这无疑给软件开发带来困难。

（3）系统变更的代价呈非线性增长。需求分析是软件开发的基础。假定在该阶段发现一个错误，解决它需要用一小时的时间，到设计、编程、测试和维护阶段解决，则要花 2.5、5、25、100 倍的时间。

3.2　需求建模

过去人们使用数据模型、过程模型、原型系统，以及需求规格说明之类的工具来描述需求，但是这些工具对于没有受过软件开发实践教育的用户来说很难理解。因此，系统开发应该以用户为中心进行。用例是一种描述系统需求的方法，使用用例的方法来描述系统需求的过程就是用例建模。UML 中建模过程一般都是从用例图开始的。用例模型包括用例图和用例描述，用例描述可以用用例规格说明，也可以用活动图来表示。将每个业务用例都绘制出相应的活动图，再将其中的“活动”整合，就得出所有备选系统用例。

用例模型有两类：业务用例模型和系统用例模型。从字面的意义来看，确实很难分清两者究竟在做些什么工作。因此应该弄清楚两者之间的区别。RUP 定义为“业务用例是从一个外部的，增加值的角度来描述一个业务过程。为了给这个业务的涉众创造价值，业务用例是超越组织边界的业务过程，很可能包括合作伙伴和供应商”。这个定义标识了一些重要点，如一个业务用例描述的是业务过程——而不是软件系统过程。一个业务用例为涉众创造价值，这些涉众要么是业务参与者要么是业务工作者。一个业务用例可以超越组织的边界。系统用例的设计范围就是这个计算机系统设计的范围，它是一个系统参与者，与计算机系统一起实现一个目标。系统用例就是参与者如何与计算机技术相联系，而不是业务过程。业务用例模型与系统用例模型之间主要有三大不同之处：设计范围、白盒测试与黑盒测试，以及业务操作。业务用例着重于业务操作，它们表示实现业务目标的业务中的具体工作流。业务过程可能涉及手工和自动过程，并且在一段长期的时间内进行。系统用例着重于要设计的软件系统，参与者如何与软件系统进行交互？在系统用例说明中书写的事件流应该足够详细，从而用作编写系统测试脚本的出发点。

系统用例模型应该划分子系统以对应不同的功能。系统用例几乎总是以黑盒形式编写的，它们描述了软件系统之外的参与者如何与被设计的系统进行交互，系统用例详细阐明了系统需求。系统用例模型的目的是从涉众的角度说明需求，而不是设计如何满足需求。

这二者最大的不同点在于：业务用例模型仅关注于企业部门的业务，而系统用例模型则关注于系统本身实现后的互动。业务用例模型和系统用例模型有共同的图素，但是在意义上是完全不同的。

3.2.1 用例建模

需求分析主要是定义业务用例模型。业务用例模型的目的在于描述企业的内部组织结构；描述企业各部门的业务；关注于角色和系统的交互界面。用例建模被认为是描述信息系统功能需求的最佳实践，用例模型描述系统外部的参与者所理解的系统功能。用例建模是使用用例的方法来描述系统功能需求的过程，建模的本质是通过抽象获得被建模对象的关键要素，然后基于特定的目的和视角利用图形把模型元素展示出来。用例图是 UML 图中较为重要和常用的一种，常常用于软件开发的分析阶段，也能用于软件的系统测试阶段。

1. 用例图

用例图是描述系统与其他外部系统以及用户之间交互的图形，即用例图描述了谁将使用系统，用户希望以什么方式与系统交互。用例图确定系统中所包含的参与者、用例和两者之间的对应关系，它描述的是关于系统功能的一个概述，描述软件应具备哪些功能模块以及这些模块之间的调用关系。用例图包含了用例和参与者，用例之间用关联来连接以求把系统的整个结构和功能反映给非技术人员（通常是软件的用户）。它的建立是系统开发者和用户反复讨论的结果，描述了开发者和用户对需求规格达成的共识。用例模型关心的是系统做什么，而不是如何实现。因此，它首先描述待开发系统的功能需求；其次，它把系统看成黑盒子，从外部参与者的角度来理解系统；第三，它驱动了需求分析之后各阶段的开发工作，不仅在开发过程中保证了系统所有功能的实现，而且被用于验证和确认所开发的系统，从而影响到开发工作的各个阶段和 UML 的各个模型。用例建模的目的是将需求规约变为可视化模型，并得到用户确认；给出一个清晰、一致的关于系统做什么的描述，确定系统的功能要求；提供从功能需求到系统分析、设计、实现各阶段的度量标准；为最终系统测试提供基准，据此验证系统是否达到功能要求；为项目目标进度管理和风险管理提供依据。用例模型是需求分析阶段的主要成果。

用例图展示了用例之间以及用例参与者之间是怎样相互联系的。用例图用于对系统、子系统或类的行为进行可视化，使用户能够理解如何使用这些元素，并使开发者能够实现这些元素，它将系统功能划分成对参与者（即系统的理想用户）有用的需求，而交互功能部分被称作用例。用例图的主要目的是帮助开发团队以一种可视化的方式来理解系统的功能需求，包括基于基本流程的“角色”之间的关系，以及系统内用例之间的关系。

用例模型的建立过程通常是：先画用例图，然后对用例图编写用例说明。在编写用例说明的时候，对一些复杂的、认为有必要的地方，绘制活动图、状态图或顺序图，方便阅读者理解。用例图关注的是三部分内容：用例、参与者和关系。

（1）用例。用例是系统执行的一组动作序列，并为执行者产生一个可供观察的结果，这个结果对系统的一个或多个参与者是有价值的。用例描述一个系统做什么，而不是怎么做。用例有系统用例和业务用例，用例习惯上也叫系统用例，是一种软件需求定义的方法或形式。业务用例实际上是一种业务流程，它从业务组织的外部，即业务参与者的角度定义业务组织提供的服务。当然业务用例还包括一些内部流程，它可能不是由业务参与者启动的，如采购流程等。因此，业务用例只是使用了用例的思想和形式而已，研究的主题是完全不同的。系统用例研究软件系统，借助用例定义软件系统需求，而业务用例研究一个目标组织，借助业务用例定义目标组织应该具有哪些业务流程，以及这些流程应该是什么样子的。用例用动名词短语的业务术语命名，用例名可以是简单的名称，如 Place Order，也可以在用例名前加表示它所属的包的名称，称为受限名，如 Sensors∷Calibrate location。用例名可以是一个文字串，其中包括任意数

目的字母、数字和大多数标点符号（冒号除外，它用来将类目与它所属的包的名称分隔开来）。用例用一个水平的椭圆表示，用例名显示在椭圆的上面、下面或内部，如图 3.3 所示。

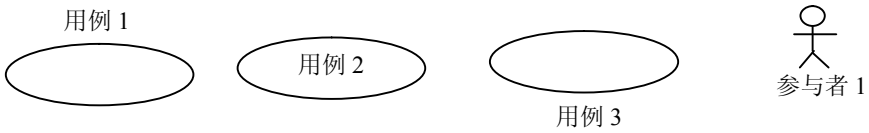

图 3.3　用例与参与者符号

（2）参与者。参与者表示用例的使用者在与这些用例进行交互时所扮演的角色的一个紧密的集合，也叫角色，通俗地讲，就是和系统打交道的人、系统、设备等。角色是一个群体概念，代表的是一类能使用某个功能的人或事，角色不是指某个个体。在实际问题中，一个人可以有多种角色，一个角色可以有多个人。“系统交互”指的是角色向系统发送消息，从系统中接受消息，或是在系统中交换信息。在画用例图的过程中，参与者往往是第一个被确定的，因为系统或者用例在开始时是模糊的，但是参与系统的角色是最容易明晰的。有了参与者之后，根据参与者与系统的交互，以及参与者要求的功能，可以进一步确定系统和用例。参与者也叫执行者，是用例的使用者。参与者不是指人或事物本身，而是指系统以外的，在使用系统或与系统交互中所扮演的角色，如小明是图书馆的管理员，他参与图书馆管理系统的交互，这时他既可以作为管理员这个角色参与管理，也可以作为借书者向图书馆借书，在这里小明扮演了两个角色，是两个不同的参与者。参与者在画图中用简笔人物画来表示，人物下面附上参与者的名称，如图 3.3 所示。参与者主要有四类：主要业务参与者、主要系统参与者、外部服务参与者和外部接收参与者。

（3）关系。关系可分为三类：第一类是参与者和用例之间的关联关系，表明参与者主要使用系统的哪些用例；第二类为用例和用例之间的关系，主要分为包含关系、扩展关系和泛化关系；第三类为参与者和参与者之间的泛化关系。关系及其表示法如表 3.2 所示。

表 3.2　关系及其表示

关系	功能	表示法
关联	参与者与其参与执行的用例之间的通信途径	———
扩展	在基础用例上插入基础用例不能说明的扩展部分	<< extend >> - - - - - - - - - - →
泛化	用例之间的一般和特殊关系，其中特殊用例继承了一般用例的特性并增加了新的特性；参与者之间的泛化关系	———▷
包含	在基础用例上插入附加的行为，并且具有明确的描述	<<included>> - - - - - - - - - - →

角色和角色之间如果存在一般和部分的关系，可以用泛化来表示，如图 3.4 所示。

用例和参与者之间是关联关系，一般角色为用例的启动者，用单向关联，否则为双向关联，如图 3.5 所示。

用例之间的关系有三种：

① 如果一个用例包含另一个用例的行为，则这两个用例之间存在包含关系，或者说一个用例使用了另一个用例的行为，被调用用例可以是事先定义好的，也可以是在各种环境下使用

的公共行为，调用是无条件的。包含关系用一个虚箭头表示，从调用用例指向被调用用例，如图 3.6（a）所示。包含关系表示一种从属关系，即子用例是主用例中相对独立的、必须调用的一部分功能。在用例分析中，应当将多个用例都共有的、相对独立的功能提取出来形成一个子用例，为日后代码复用提供有力保障。

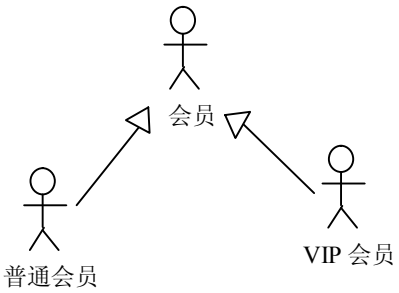

图 3.4　角色之间的关系及表示

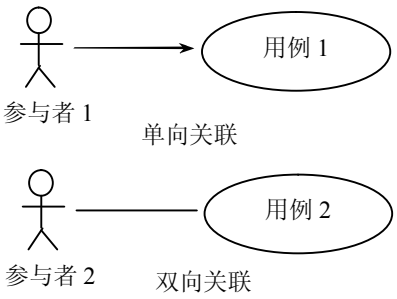

图 3.5　销售管理系统

② 如果对一个用例添加某些额外的行为，包含此额外行为的用例和原来的用例之间就构成了扩展关系，扩展用例本身提供了一种离散的行为，可以把自己添加到执行用例的环境中去，增强了执行用例的行为，等于在不改变执行用例的情况下，给系统添加新的行为。扩展用例的执行是有条件的，这个条件叫扩展点，可以用注释在图中表示，也可以在用例描述中加以说明。扩展关系用虚箭头表示，箭尾在扩展用例上，箭头在执行用例上，如图 3.6（b）所示。扩展关系表示一个功能是对另一个功能的扩展，即被扩展功能不一定调用扩展功能，但扩展功能是对被扩展功能的加强与延伸。

③ 如果一个用例是另一个用例的特例，这两个用例之间就是泛化关系，父用例不具备完整的事件流，不具备执行能力，而子用例实现其高级行为，如图 3.6（c）所示。这种关系表示在语义上比较困难，UML2.0 对此又没有明确的描述定义，因此在实际中这种关系使用非常少，甚至很多书连提都不提这种关系。用例之间的关系实例如图 3.6（d）所示。

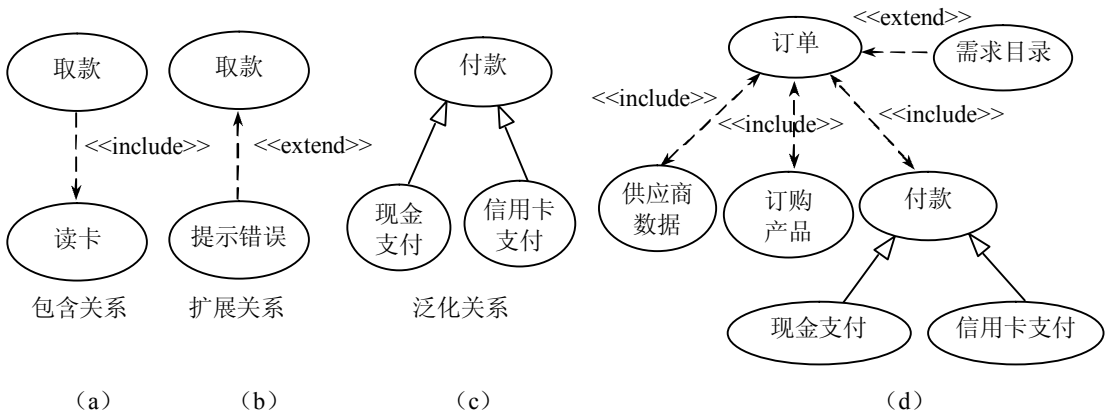

图 3.6　用例的关系表示

2. 用例描述

用例描述是业务事件以及用户如何同系统交互以完成任务的文字描述。用例图可以直观

地展现需求中的所有用例、参与者、系统边界，以及它们之间的关系，但这还不足以表达需求分析所要求表达的内容。用例图必须辅之以用例说明，才能完整清楚地表达。针对每一个用例都应该有一个用例规约文档与之相对应，该文档描述用例的细节内容。

对于用例描述的内容，一般没有硬性规定的格式，但一些必须的或者重要的内容还是应该写进用例描述里面的。用例描述一般包括：简要描述（说明）、前置（前提）条件、基本事件流、其他事件流、异常事件流、后置（事后）条件等。

3. 用例建模的步骤

在用例建模的过程中，建议的步骤是先找出参与者，再根据参与者确定每个参与者相关的用例，最后再细化每一个用例的用例规约。具体如下：

（1）确定将要设计的系统范围和它的边界。

（2）确定系统外的参与者。

（3）从参与者（用户）和系统对话的角度继续寻找以下两方面的特征：寻找参与者怎样使用系统；系统向参与者提供什么样的功能。把离用户最近（接口）的用例作为顶层用例。

（4）对复杂的用例做进一步分解，并确定底层用例以及用例间的关系。

（5）对每一用例做进一步细化。

（6）寻找每一个用例发生的前提条件和发生后对系统产生的结果。

（7）寻找每一个用例在正常条件下的执行过程。

（8）寻找每一个用例在非正常条件下的执行过程。

（9）用 UML 建模工具画出分层的用例模型图。

（10）审核用例模型。

（11）编写用例模型图的补充说明文档。

用例分析，从一开始就对原始需求进行了功能划分，将相关功能划分到一起，将共有功能提取出来，标志出功能间的依赖与非依赖关系（包含表示的是一种依赖关系，而扩展表示的是一种非依赖关系）。这是一种 OOA，它为日后的 OOD 提供了强有力的支持，而这些都是需求规格说明书所不具有，或者不完全具有的功能。

4. 业务用例建模

当进行一个项目的需求分析时，首先要听用户谈他们的需求，或者看用户提交的业务需求文档。用户一定会提出一个又一个的功能或要求，它们中的每一个要求就成为了最初的用例。分析这些用例，关注它们的每一个参与者，以及它们相互之间的关系，这就形成了最初的用例模型。在采用用例分析的方式与客户沟通需求的时候，应当着重关注的是参与者及其目标，即每个功能的参与者是谁，完成这个功能的目标是什么，以及如何完成这个目标。这时的用例说明采用概述的方式，即只进行主成功场景（基本流程）的描述。此后，继续细化用例，各个用例的替代场景（分支流程）逐渐被整理出来，用例再一步步细化。另外，开发人员对一些需求的认识一开始可能存在着偏差，因此，需要不断更正用例描述。同时，一些新的功能可能被用户提出来，形成一些新的用例。如此反复数轮之后，项目需求的整体框架才逐渐清晰。然后应该讨论系统边界，对用例模型进行再一次调整。

业务用例定义标识了一些重要点，比如：

一个业务用例描述的是业务过程——而不是软件系统过程。

一个业务用例为涉众创造价值。这些涉众要么是业务参与者，要么是业务工作者。

一个业务用例可以超越组织的边界。有些架构师对于这一点有非常严谨的态度。许多业

务用例确实超越了组织的边界，但是有些业务用例仅仅关注于一个组织。

Podeswa 在《UML for the IT Business Analyst》中对业务用例的定义：“业务用例：业务过程是描述这个业务的具体工作流的，是一次涉众与实现业务目标的业务之间的交互，它可能包含手工和自动化的过程，也可能发生在一个长期的时间段中。”

这个定义表明了通过实现业务目标创造价值的观点。它通过把一个业务过程描述成一个可能包含手工和自动化过程的具体工作流来详述 RUP 的定义。这个定义还指出，工作流可能发生在一个长期时间段中，所有的这些都十分得重要。

业务用例建模需要注意的问题有：

（1）业务用例是仅从系统业务角度关注的用例，而不是具体系统的用例。它描述的是“应该实现什么业务”，而不是“系统应该提供什么操作”。例如，在实际系统中，“登录”肯定要作为一个用例，但是这是软件系统中的操作，而用户所关注的业务是不包含“登录”的。

（2）业务用例仅包含客户“感兴趣”的内容。

（3）业务用例所有的用例名应该让客户能看懂，如果某个用例的名字客户看不懂是什么意思，它也许就不适合作为业务用例。

（4）用例模型的重点是用例说明而不是用例图。建立用例模型的时候，绘制用例图可能只花费几十分钟，而编写用例说明却要花费数小时甚至数天。用例图只是给人最直观的展示，而用例说明才是对业务需求最详尽的说明。

3.2.2 确定系统边界和范围

1. 确定系统的边界

任何一个系统都有一个边界的问题。边界问题就是确定系统和相邻系统的交接部分。定义系统边界就是定义系统的范围，即哪些元素属于本系统，哪些元素属于相邻系统，明确系统目标范围。对一般的物质性系统，其边界通常比较容易通过物理的方法确定，以物理的形式表达，如小区边界可以以围墙或街道划分，企业边界可以用围墙，也可以用业务范围划分等。但对信息系统的边界，学术界一直没有一种权威的定义和表示方法。其难度主要在于信息系统是一个融于物质系统的特殊系统，其本身既包含有一定的物质成分，又包含有一些非物质成分。同时，所有这些成分几乎又都融于其相邻的系统中，难以单独分割。

信息系统一般都是某种物质系统内的一个子系统，但其又完全融入大系统之中，几乎不可能进行有形的分割，就如人体的肉体系统与血液、神经系统的关系，要完全把血液与神经系统从肉体系统中分割出来是不可能的，只能进行示意，要通过有形的方式来准确表示信息系统及其边界几乎完全不可能。信息系统具有比一般系统更复杂的边界关系，既有物理上的相关，又有信息上的相关。

信息系统边界的定义对于分析与设计信息系统十分重要。有了明确的边界就知道了信息系统分析与设计的范围，可以更好地分析与设计信息系统的内部流程、信息处理方式和信息组织方式；同时，也可以更好地确定设计的信息系统与外部信息系统的信息联系。特别是在企业信息系统多个子系统分析与设计的过程中，子系统边界的确定对于整个信息系统的信息流程优化等方面具有举足轻重的意义。

系统边界是一个软件系统需要处理的整个问题空间的范围。一个软件系统不可能处理所有问题，必须给它定义问题空间的范围。哪些是这个软件可以处理的，哪些则是这个软件不能处理的，也就是项目管理中所说的项目范围。系统的范围是一种形象的描述，实际

上是不存在的，只能通过系统边界来区分系统与系统环境。与需求规格说明书相比，用例分析通过图形的方式，更加明确地定义出了项目范围，以及与其他系统之间的关系，使其更加清楚明了。

信息系统的边界可以从两个方面进行定义，一是通过界定其构成元素来区分它和环境的关系；二是通过界定其边界关系来区分它与环境的联系。这样，可以归纳出信息系统的边界定义为：信息系统的边界就是信息系统内部构成元素与外部有联系实体之间的信息关系的描述与分割，并不需要在它们之间划一条物理边界，而只需要弄清它们之间信息输入与输出的分割。确定边界意味着找出系统中有什么，系统外有什么。

在定义需求时，必须定义要开发的计算机系统的边界，即确定哪些是系统需求，哪些是和系统相关的操作过程的需求，哪些是在系统范围之外的需求。需求提供者常常不太了解系统应该包含哪些内容，因此，他们可能会提出不恰当的需求。需要通过系统边界定义初步剔除那些明显在系统范围之外的需求，以免这些需求干扰后续的分析过程。检查每项原始需求，将它们区分为系统需求、过程需求和应该拒绝的需求。主要考虑如下问题：

（1）某项需求是否是基于不完整的或者不可靠的信息做出的？

（2）某项需求的实现是否需要在系统已定义的数据库之外的信息？

（3）某项需求是否和系统的核心功能相关？

（4）某项需求是否牵涉到系统之外的功能或者设备的性能？

对于问题（1）和问题（2）可以判断是否为过程需求，如果是过程需求，则要求系统的操作者提供这些信息，否则需要复审系统应该处理的数据。

对于问题（3）和问题（4）可以判断是否是系统边界以外的需求。如果是，则它可能是不必要的，也可能是无法实现的需求。

对于操作过程相关的需求和系统边界之外的需求，必须准备一些技术的和经济的论据，说明这些需求被拒绝的理由。这些论据应该是基于这个组织已定义的业务目标或者系统可行性研究的结果。

信息系统边界的定义就是分割信息系统与外部环境的信息输入与输出。即在连接外部环境实体的信息中哪些信息是由信息系统内部产生而输出到外部环境的，哪些信息是由外部环境实体产生而输入到信息系统内部的。信息系统边界的表示方法不止一种，但系统用例图是表示边界的最好的方法。

如果在确定执行者和用例的过程中发现新的需求应该怎么办呢？来看看这一需求是否在你的系统之中。当你发现新的需求时应该考虑的一些问题如下：

这些需求对系统是否是必需的？

这些需求是系统在逻辑上能完成的吗？

新的需求将如何影响目前的风险分析？

这些需求是否可以被系统当前的执行者处理？换句话说，是否有别人对这些需求负责？

这些需求是客户希望系统去做的吗？

这些需求会使产品在市场上变得与众不同吗？

为系统定义一个清晰的边界或许很难。如果希望系统在规定时间和预算范围内完成，就必须具有清晰定义的系统边界。预先花时间进行这项工作是很有价值的。否则，在你的项目生命期内会一直受此困扰。

系统边界是用来表示正在建模系统的边界。边界内表示系统的组成部分，边界外表示系

统外部。在 UML 用例图中，用一个方框来表示系统的边界。所有系统用例都放在框内，所有动作者都位于框外。动作者和用例之间用直线相连。方框内的每一件事物都是系统的一部分，方框外的每一件事物都是系统的外部。参与者画在边界的外面，用例画在边界里面。因为系统边界的作用有时候不是很明显，所以在画图时可省略。

2. 确定系统的范围

在确定好系统的执行者和用例后，系统的边界就确定了。然后就需要确定项目的范围，清晰地定义项目的范围，将不需要做的事情放到一边，同时为需要做的事情划分优先级。

系统范围是指整个系统中安全基线所涉及的对象和考虑的范围，系统范围确定是一个划定系统安全边界的过程，该边界界定了安全基线的管辖范围，即将系统划分为内部和外部两部分，系统内部即为安全基线的系统范围，系统外部即为系统环境，而系统安全边界正是这两部分的接口。如果你不打算在当前项目中构建整个系统，则需要清晰地定义系统将要包括的成分。使用需求优先级的方法确定系统必须包含的事情，并确定没有必要包含的事情。定义系统边界和细化用例时，可能发现系统的更多需求。在继续进行的过程中已经确定了这些需求在你的系统边界之内，但是它们是否在项目的范围之内呢？

3.2.3 确定参与者

通常将系统外部与系统内部交互的事物统称为参与者，也有的称为执行者或者角色。参与者是在系统之外与系统交互的人或事物，每一个执行者都对应一种特定的角色，每一个系统之外的实体对应多种执行者，就好比一个人在系统中会有多种角色一样，又或者几个人可以用一个执行者来表示，因为他们对于系统来讲属于同一个角色。参与者是在系统之外，在系统之内的不是参与者，参与者与系统有明显的系统边界。参与者给大家最直观的认识就是操作系统的那些人，但更准确的说应当是操作系统的那些角色。参与者代表了同系统交互的用户实现角色，而不是代表一个人或者工作职位。事实上，参与者不必是一个人，它可以是一个组织、另一个信息系统、一个外部设备（如传感器）或者是一个时间概念。

如何寻找参与者呢？可以参考如下的信息源：标示系统范围和边界的上下文图；现有的系统文档和系统手册；项目会议和研讨会的记录；现有的需求文档、项目章程或工作陈述。

参与者总是在你的系统之外，他们从来都不是系统的一部分。为了帮助找到执行者，可以在人、其他的软件、硬件设备、数据存储或者网络目录中进行寻找。参与者是在系统之外，透过系统边界与系统进行有意义交互的任何事物。通俗地讲，参与者就是所要定义系统的使用者。寻找参与者可以从以下问题入手：

（1）系统开发完成之后，有哪些人会使用这个系统？

（2）系统需要从哪些人或其他系统中获得数据？

（3）系统会为哪些人或其他系统提供数据？

（4）系统需要与哪些其他系统交互？

（5）系统是由谁来维护和管理并保持其正常运行？

（6）系统需要应付（处理）哪些硬设备？

（7）谁（或什么）对系统运行产生的结果感兴趣？

（8）有没有自动发生的事件？

在这里需要注意的是：系统参与者一定是与系统有直接联系的事物，这里的事物包括人和其他系统，如图 3.7 所示。

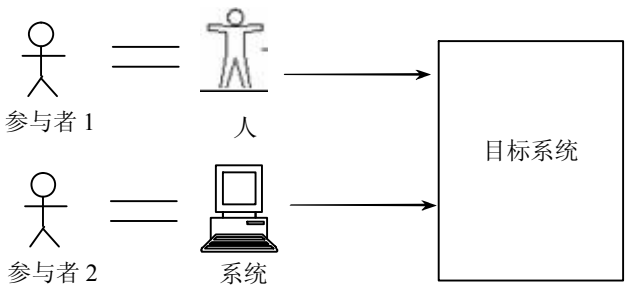

图 3.7　参与者符号

由于参与者事实上就是类，因此，参与者之间也有继承关系（泛化）。参与者之间的泛化关系表示一个一般性的参与者与另一个特殊性的参与者之间的关系，如图 3.4 所示。子参与者继承了父参与者的行为和含义，还可以增加自己的特殊行为和含义，子参与者可以出现在父参与者能出现的任何位置上。因此，参与者之间的关系只有一种——继承关系，表示功能职责上的继承。例如，通常软件系统中都有“普通操作者”，代表的是进入系统的所有用户都应当具有的功能。而软件系统中又有很多“特殊职能者”，他们除了具有普通操作者的功能外，还有自己独特的功能。在这里，“特殊职能者”是对“普通操作者”来说，在系统外部与系统直接交互的人或事物（如另一个计算机系统或一些可运行的进程）。需要注意的是：

（1）参与者是角色，而不是具体的人，它代表了参与者在与系统打交道的过程中所扮演的角色。所以在系统的实际运作中，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于一个参与者，从而代表同一参与者的不同实例。

（2）参与者作为外部用户（而不是内部）与系统发生交互作用，是它的主要特征。

（3）在后面的顺序图等中出现的“参与者”，与此概念相同，但具体指代的含义，视具体情况而定。

3.2.4　确定需求用例

用例图中首先要明确的概念就是用例。用例是系统的一个功能单元，描述了参与者与系统发生的一次交互行为。用例，就是一件事情，要完成这件事情，需要一系列活动。做一件事情可以有很多不同的方法和步骤，也可能会有各种各样的意外情况，因此，这件事情由很多不同情况的集合构成，在 UML 中称为场景。用例必然是以动宾形式出现，且相对独立，但并不意味着用例不与其他用例交互就能独立完成参与者的目的。确定用例主要是看各参与者需要系统提供什么样的服务，或者说参与者是如何使用系统的。如银行的 ATM 自动提款机系统，用户提款就是一个用例。用例是一个用来描述参与者如何使用系统来实现其目标的一组场景的集合。用例强调的是一组场景，这组场景不多但相互之间存在功能上的共性，就像一个大功能模块下的多个子模块。这组场景中的每一个，又分别形成一个个子用例。子用例再细分，又可以再形成各自的子用例。用例分析就是这样由粗到细地逐步细分，从而形成一系列的用例图。用例图分析到多细，应当由业务需求的情况决定。分得过粗，不足以说清楚业务的相关细节，或者是一张用例图信息过多，影响人们的理解；分得过细，不仅会增加工作量，还会丢失许多用例间的相互关系，得不偿失。总之，较为复杂的部分细一些，简单的部分粗一些，保证每个用例图都能保持强烈的相关性，以指导日后的功能划分。

寻找用例的方法：

（1）从原始需求中寻找所包含的功能。

（2）未来的系统，需要和哪些系统发生信息交互？

（3）哪些人将操作未来的软件系统？

（4）系统有哪些受限的条件？

（5）未来的软件界面怎样组织？

（6）系统的响应有哪些去向？

一个用例应有明确有效的目标，一个真实的目标应完备地表达主角的期望，一个有效的目标应当在系统边界之内，由主角发动，具有明确后果。用例的执行结果对参与者来说可观测和有意义，如后台进程监控在需求阶段不应该作为用例出现，而应该作为系统需求在补充规约中定义。

在实际分析中一种普遍的错误认识是认为用例就是功能。从使用者观点出发描述软件是非常适合的。使用者观点告诉需求收集人员，它希望这个系统是什么样，他将怎样使用这个系统，希望获得什么结果，那么软件按照使用者要求提供实现，就不会偏离使用者的预期。使用者的观点实际上就是用例的观点，一个用例就是一个参与者如何使用系统，获得什么结果的一个集合。通过分析用例，得出结构性和功能性的内容，最终实现用例，也就实现了用例的观点。功能是脱离使用者愿望存在的。我们常常说工具具有某个功能，描述的是工具，而不是站在使用者角度描述使用者的愿望。功能用来描述某某东西能做什么，与使用者的愿望无关，描述的是事物固有的性质。用例描述的是使用者的愿望和对系统的使用要求，是一个系统性的工作，这个系统性的工作非常明确，形成一个系统性的目标。一个用例是使用者对目标系统的一个愿望，是一个完整的事件。为了完成这些事件，需要经过很多步骤，但是如果这些步骤不能完整地反映参与者的目标就不能作为用例。业务用例是用例版型中的一种，专门用于需求阶段的业务建模。针对客户业务的模型，也就是现在客户的业务是怎么来建立的。严格来说，业务建模与计算机模型无关，只是业务领域的一个模型，通过业务模型可以得到业务范围，帮助需求人员理解客户业务，并在业务层面上和客户达成共识。例如：图书馆借书系统，计算机可以自动提示读者逾期没有归还图书，但是在业务建模的时候不应该将计算机包括进来。之所以不能把计算机引入，是因为业务范围不等于系统范围，不是所有的业务都能用计算机来实现，不在计算机中实现的业务就不进入系统范围，就不能作为一个需求。

3.2.5 用例模型的关系

用例描述的是系统外部可见的行为，是系统为某一个或几个参与者提供的一段完整的服务。从原则上来讲，用例之间都是并列的，它们之间并不存在着包含从属关系。但是从保证用例模型的可维护性和一致性角度来看，可以在用例之间抽象出包含（include）、扩展（extend）和泛化（generalization）这几种关系。这几种关系都是从现有的用例中抽取出部分信息，然后通过不同的方法来重用这部分信息，以减少模型维护的工作量。

1. UML 用例图的包含关系

包含关系（include）是把几个用例的公共行为分离成一个单独的用例，使这几个用例与该单独的用例之间所建立的关系。被抽取出来的单独的用例叫做被包含用例（Inclusion），而抽取出公共用例的几个用例称为基本用例（Base）。包含用例是被封装的，代表在各种不同基本用例中复用的行为。当某用例的事件流过于复杂时，为了简化用例的描述，可以把某一段事件

流抽象成为一个被包含用例；相反，用例划分太细时，也可以抽象出一个基本用例，来包含这些细颗粒的用例。这种情况类似于在过程设计语言中，将程序的某一段算法封装成一个子过程，然后再从主程序中调用这一子过程。在绘制用例关系时，包含关系应绘制成从主用例指向子用例的虚线箭头，并标注为“include”，表示主用例包含子用例。如在银行办业务，取钱、转账和修改密码，都需要核对账号和密码，那么这个行为可以抽取出来，形成一个包含用例，这个包含用例带有复用的意义。如果缺少包含用例，取钱、转账等业务用例将不完整，核对账号也不能脱离取钱、转账等业务用例而单独存在，如图 3.8 所示。包含用例表示的是“必需”，而不是“可选”，这意味如果没有包含用例，基本用例就不完整，同时如果没有基本用例，包含用例就不能单独存在。

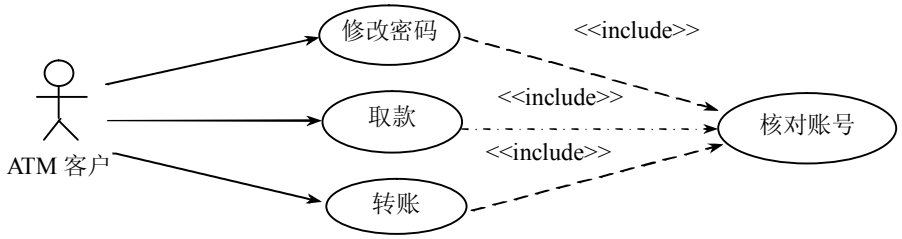

图 3.8　用例的包含关系

2. UML 用例图的扩展关系

扩展关系（extend）是将基本用例中一段相对独立并且可选的动作，用扩展用例加以封装，再让它从基本用例中声明的扩展点上进行扩展，从而使基本用例行为更简练、目标更集中。扩展用例为基本用例添加新的行为，扩展用例可以访问基本用例的属性，因此，它能根据基本用例中扩展点的当前状态来判断是否执行自己。对于包含关系而言，子用例中的业务过程是一定要插入到基础用例中去的，并且插入点只有一个。而扩展关系可以根据一定的条件来决定是否将扩展用例的业务过程插入基础用例业务过程，并且插入点可以有多个。但是扩展用例对基本用例不可见。对于一个扩展用例，可以在基本用例上有几个扩展点。扩展用例带有抽象性质，表示用例场景中的某个“支流”，由特定的扩展点触发而被启动。与包含关系不同，扩展表示“可选”，而不是“必需”，这意味即使没有扩展用例，基本用例也是完整的，如果没有基本用例，扩展用例不能单独存在。如果有多个扩展用例，同一时间用例实例也只会使用其中一个例子。扩展关系应绘制成从扩展用例指向被扩展用例的虚线箭头，并标注为“extend”，表示扩展用例是对被扩展用例的扩展。虚线箭头在 UML 中代表的是一种依赖关系。如基本通话这个用例上可以有呼叫等待、呼叫转移等扩展的功能用例。如果对方通话正忙着，可以用呼叫等待，如果对方不方便接电话，也可以用呼叫转移。因此，可以采用扩展关系来描述，如图 3.9 所示。

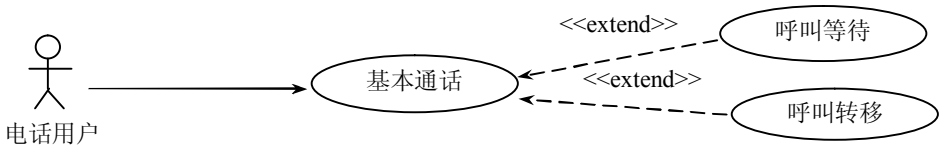

图 3.9　用例的扩展关系

虽然包含和扩展两个概念在指定共同功能（include）和简化复杂用例流程时非常有用，但是这些概念常常被误用。两者的区别如表 3.3 所示。

表 3.3　包含和扩展关系的主要区别

	包含的用例	扩展的用例
这个用例是可选的吗	否	是
没有这个用例基本用例还完整吗	否	是
这个用例的执行是有条件的吗	否	是
这个用例改变了基本用例的行为吗	否	是

3. UML 用例图的泛化关系

泛化关系（generalization）表示子用例和父用例的相似；子用例将继承父用例的所有结构、行为和关系。子用例可以使用父用例的一段行为，也可以重载它。父用例通常是抽象的。在实际应用中很少使用泛化关系，子用例中的特殊行为都可以作为父用例中的备选流存在，代表一般与特殊的关系，它的意思和面向对象程序设计中的继承的概念是类似的。子用例从父用例继承了行为和属性，还可以添加行为和属性来改变已继承的行为。如订票是一个很泛化的用例，具体的用例可以是电话订票、网上订票等，如图 3.10 所示。

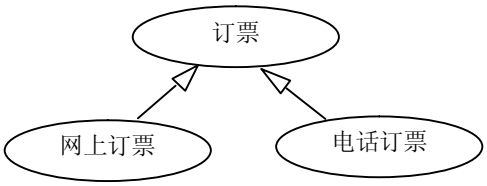

图 3.10　用例的泛化关系

一般来说，可以用“is a”和“has a”来判断使用哪种关系。泛化关系和扩展关系表示的是用例间的“is a”关系，包含关系表示的是用例间的“has a”关系。扩展关系和泛化关系相比，多了扩展点的概念，也就是说，一个扩展用例只能在基本用例的扩展点上进行扩展。在扩展关系中，基本用例一定是一个真实存在的用例，一个基本用例执行时，可以执行、也可以不执行扩展用例。在包含关系中，基本用例可能是、也可能不是一个真实存在的用例，一定会执行包含用例部分。如果需要重复处理两个或多个用例时，可以考虑使用包含关系。

3.2.6　构造业务用例模型图

建立业务模型，查找业务用例都必须使用业务主角，而不是普通参与者。业务主角是客户实际业务里的参与者，没有计算机，就没有抽象的计算机角色。参与者位于系统边界外面，如果把边界内和外的参与业务的人都作为参与者建模，会混乱。如何区分是不是参与者，可通过如下来判断：看他是否主动向系统发出工作；是否有完整的业务目标；系统是否为他服务。参与者主要有四类：主要业务参与者、主要系统参与者、外部服务参与者和外部接收参与者。如超市收银系统，收银员是该系统的参与者，而超市客户则不是。

参与者、用例以及用例间的关系已经明确，这时需要用一种比较直观的方式来表达这些信息。用例图是显示一组用例、参与者以及它们之间关系的图，一个用例模型由若干个用例图

来描述。对于一个系统来说，不同的人进行用例分析后得到的用例数目多少不一。用例是有一定粒度的，开发者往往难于确定用例。如果用例粒度很大，那么得到的用例数就会很少，如果用例粒度很小，那么得到的用例数就会很多。一个基本用例可以分解出许多更小的关键精化用例，这些小的精化用例展示了基本用例的核心业务。与泛化关系不同，精化关系表示由基本对象可以分解更明确、精细的子对象，这些子对象并没有增加、减少、改变基本对象的行为和属性，仅仅是更加细致和明确化了。从业务模型中分析出实现业务目标的那些核心行为和实体，从而描述从一个关键的业务结构能得到一个易于理解的业务框架，这些关键概念就是对业务用例的精化。精化是进行分层。

一个用例图描述用例模型的一个侧面，几个用例图可以完整地描述一个系统或子系统。如根据进销存管理业务得到系统的整体业务用例模型，如图 3.11 所示。

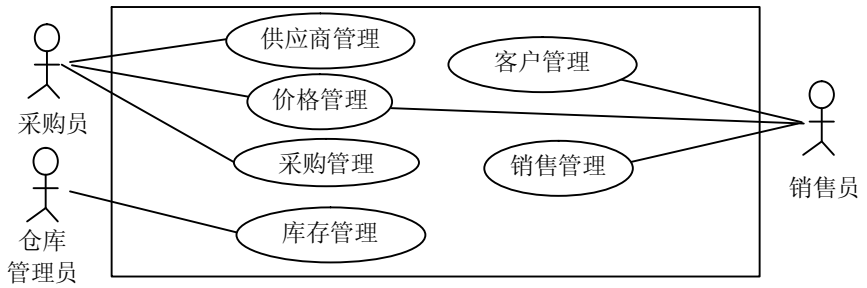

图 3.11　进销存管理系统业务用例图

每个用例又可以进一步细化，如库存管理子系统主要完成入库、出库、库存盘点、库存查询、打印超预警线货物清单、打印年终库存损耗报表、编制库存报表等业务活动，如图 3.12 所示。

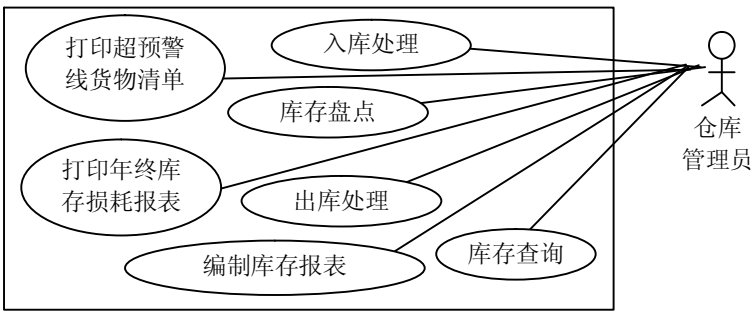

图 3.12　库存管理子系统业务用例图

在业务建模阶段，一个用例描述一项完整的业务流程，如取钱、报装电话、借书等，而填写申请单、查找书目就不是用例。在概念建模阶段，用例能描述一个完整的事件流。可以理解为一个用例描述一项完整业务中的一个步骤，需要归纳和抽象出业务用例中的关键概念模型并为之建模。如宽带业务中有申请报装和申请迁移地址用例，在分析时，可归纳和分解为提供申请资料、受理业务、现场安装等多个业务流程。在系统建模阶段，用例视角是针对计算机的，用例的粒度以一个用例能够描述操作与计算机的一次完整交互为宜，如填写申请单、审核申请单、派发任务，也可以理解为一个操作界面或者一个页面流。在进行顶级用例建模时，需要寻

找角色和角色之间的关系，以及角色和顶级用例之间的关系。

3.2.7 用例规格说明

一些初学者认为，用例模型就是画张用例图，其实这是错误的。用例模型包括用例图和用例规格说明，有时还要辅以一些简单的活动图、状态图和顺序图等。用例图采用图形的方式，形象生动地展现了用例、参与者和系统边界之间的关系。用例图只是简单地用图描述了一下系统，在用例图中用例只是一个简单的“动宾”词语，无法知道其实际过程，如订票，不知道什么时候订，采取什么方式订。因此，对于每个用例，我们还需要有详细的说明，这样就可以让别人对这个系统有一个更加详细的了解。

用例说明是对用例、参与者和系统边界进行的详细描述。在描述过程中，还可以对一些关键的流程，以及这些流程中关键类的状态变化，使用活动图、状态图或顺序图进行图形化的展现。因此，详细描述用例的方式有用例规约描述、活动图、状态图、顺序图和通信图等。在需求分析中尚不涉及状态问题，所以没有必要用状态图、顺序图或通信图表示，但可以使用活动图对用例进行分析，因为活动图对分支、判断、循环等流程概念的表达比其他图清晰，适合描述业务过程。

1. 用例规约描述

一个完整的用例包括参与者、前置条件、场景和后置条件等，如图 3.13 所示。用例建模图画完以后，就要进行描述文档的编写，文档主要用来弥补图形表示的不足和增强系统的完整性。对于用例描述的内容，一般没有硬性规定的格式，但一些必须或者重要的内容还是要写进用例描述里面的。通常包括的内容有：用例名称、参与者、前置条件、后置条件、基本事件流、备用事件流、异常事件流等。

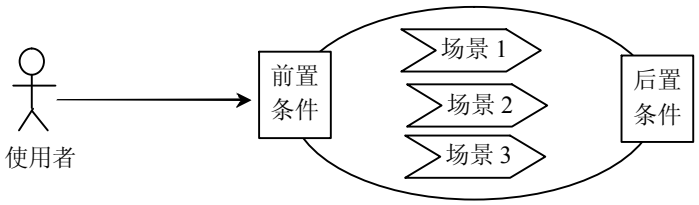

图 3.13 完整的用例

在参考了其他资料的基础上总结出用例描述格式，如表 3.4 所示。

表 3.4 用例描述格式

描述项	说明
用例名称	表明用户的意图或用例的用途，如“查询客户资料”
用例编号[可选]	唯一标识符，在文档其他地方可以通过标识符来引用该用例
用例描述	对用例的角色、目的的简要描述
参与者	与此用例相关的参与者
优先级[可选]	一个有序的排列，1 代表优先级最高
状态[可选]	通常为以下几种之一：进行中、等待审批、通过审查或未通过审查
前置条件	一个条件列表，这些条件必须在访问该用例前被满足

续表

描述项	说明
后置条件	一个条件列表，这些条件必须在完成该用例后被满足
基本操作流程	描述该用例的基本流程，指每个流程都“正常”运作时所发生的事情，没有任何备选流和异常流，而只有最有可能发生的事件流
备选操作流	表示这个行为或流程是可选的或备选的，并不是总要执行它们
异常事件流	表示发生了某些非正常的事情所要执行的流程
被泛化的用例[可选]	此用例所泛化的用例列表
被包含的用例[可选]	此用例所包含的用例列表
被扩展的用例[可选]	此用例扩展点用例列表

用例模型描述必须达到以下要求：

1）语言的互通。用例模型采用的语言必须达到，既能让业务人员看懂，以便给予业务确认，又能让技术人员看懂，以便进行日后的设计开发。因此，用例模型的描述必须是对业务需求最平实的表述，站在业务人员的角度说事儿，不能参杂过多的技术语言。同时，又要站在技术的角度进行分析，详细清楚地表述各个功能的操作流程，并不能使用过于专业的业务术语，或者对必要的业务术语进行解释，让技术人员看懂。

2）清晰准确。用例模型描述必须清晰准确地表达每一个业务需求，在建立用例模型描述的时候，必须明确每一个术语、每一段表述，不能存在二义性。

3）最全面和详尽的业务需求。用例模型描述必须从各个角度，全面地反映客户对系统的期望。用例模型描述对业务需求把握得越全面、越详尽，项目出现偏差和风险的几率就越小。这就要求进行分析时，各个角度都要分析到。

要做到以上几点其实并不容易，所幸的是，用例规格说明规范为我们提供了一个良好的范例。用例说明在用例模型中的作用，甚至超过了用例图。寥寥几页的用例图，需要数十页甚至上百页的用例说明来描述。用例说明需要按照一定的格式，对所有用例图中的所有用例进行描述，如果有必要，还要对参与者、系统边界和部分术语进行描述。

用例描述虽然看起来简单，但事实上它是捕获用户需求的关键一步，虽然很多人能给出用例描述，但描述中往往存在很多错误或不恰当的地方，在描述用例时容易犯的错误有：

（1）只描述参与者的行为，没有描述系统的行为，如表 3.5 所示。

表 3.5　用例描述 1

用例名称	取款
用例描述	客户取款
事件流	
基本事件流	1．储户插入 ATM 卡，并键入密码； 2．储户按“取款”按钮，并键入取款数目； 3．储户取走现金、ATM 卡以及收据； 4．储户离开

（2）只描述系统的行为，没有描述参与者的行为，如表 3.6 所示。

表 3.6　用例描述 2

用例名称	取款
用例描述	客户取款
事件流	
基本事件流	1．ATM 系统获得 ATM 卡和密码； 2．设置事务类型为“取款”； 3．ATM 系统获取要提取的现金数目； 4．验证账户上是否有足够储蓄金额； 5．输出现金、数据和 ATM 卡； 6．系统复位

以上两个错误原因是没有理解用例描述的作用，即描述参与者与系统的交互过程。既然是交互过程，那就要有来有往，“用户做什么”然后“系统做什么”这两样应该成对出现在描述中。

（3）描述过于冗长，如表 3.7 所示。

表 3.7　用例描述 3

用例名称	客户维护
用例描述	对客户资料进行维护
事件流	… …
备选事件流	修改客户资料 1．在客户资料维护的主界面，用户选择查询某客户资料； 2．系统显示符合条件的客户资料，内容包括：客户编号、来源、类型、省份、公司名称，同时在每行的开始位置放置“修改”按钮，以便让用户单击进行资料修改； 3．用户选择某客户记录左边的“修改”按钮，修改某客户资料； 4．系统打开修改客户资料界面； 5．……

2．使用活动图描述用例

前面用文本描述了用例的流程（用例规格说明）。文本描述的好处是容易被创建和修改，它们可以在任何地方被创建，可以很容易地由业务人员和开发人员使用和分享。但是，用文字描述的复杂用例、业务过程和算法可能难以理解。复杂流程采用可视化描述就会好很多。所以，完成了业务用例图后，可以为每一个业务用例绘制一幅活动图。活动图提供了活动流程的可视化描述，可以是在系统、业务、工作流或其他过程中使用。活动图关注被执行的活动以及谁（或什么）负责执行这些活动。活动图还有个很重要的使命就是从业务用例分析出系统用例。一个系统用例应该是实际使用系统的用户所进行的一个操作，如“图书管理”这个业务用例，可以分解出多个系统操作。这里要特别注意这些操作，其中很多“活动”都很可能是一个系统用例（当然，并不是每个都是）。如系统中至少包含以下备选系统用例：登录、注销登录、查看图书列表、修改图书、删除图书。这样，将每个业务用例都绘制出相应的活动图，再将其中的“活动”整合，就得出所有备选系统用例。

在面向对象的分析设计方法中，用例模型主要用于表述系统的功能性需求，系统设计主要由对象模型来记录描述。另外，用例定义了系统功能的使用环境与上下文，每一个用例描述

的是一个完整的系统服务。用例方法比传统的 SRS（软件需求说明）更易于被用户所理解，它可以作为开发人员和用户之间针对系统需求进行沟通的一个有效手段。

3.3　活动图

针对每一个业务用例，整理出其业务流程。业务流程可以用两种方式表达：顺序图或活动图。两种方式各有利弊。顺序图可以很好地表达各个业务对象之间的关系，有助于准确定位用户对系统的功能需求从而获得系统业务用例。但对于流程分支关系表达的不够清楚。活动图可以有效地将业务流程的各种分支准确地表达出来，但在表达对象关系方面不如顺序图。本章主要介绍活动图，顺序图请参看第 5 章。

活动图是状态图的一个变体，用来描述执行算法的工作流程中涉及的活动。业务用例工作流程说明了业务为向所服务的业务主角提供其所需的价值而必须完成的工作。业务用例由一系列活动组成，它们共同为业务主角生成某些工件。工作流程通常包括一个基本工作流程和一个或多个备选工作流程。工作流程的结构使用活动图来进行说明。活动状态代表了一个活动、一个工作流步骤或一个操作的执行。活动图描述了一组顺序的或并发的活动。活动图很像流程图，它显示出工作步骤，判定点和分支。可用于表达一个对象的操作和一个业务过程。

活动图是 UML 用于对系统的动态行为建模的另一种常用工具，它描述活动的顺序，展现从一个活动到另一个活动的控制流。活动图在本质上是一种流程图。活动图着重表现从一个活动到另一个活动的控制流，是内部处理驱动的流程。

3.3.1　活动图的符号

UML 活动图中包含的图形符号有：活动状态（Activity）、动作状态（Actions）、动作状态约束（Action Constraints）、动作流（Control Flow）、开始结点（Initial Node）、终止结点（Final Node）、对象（Object）、数据存储对象（DataStore）、对象流（Object Flows）、分支与合并（Decision and Merge Nodes）、分叉与汇合（Fork and Join Nodes）、异常处理（Exception Handler）、活动中断区域（Interruptible Activity Region）和泳道（Partition），如图 3.14 所示。

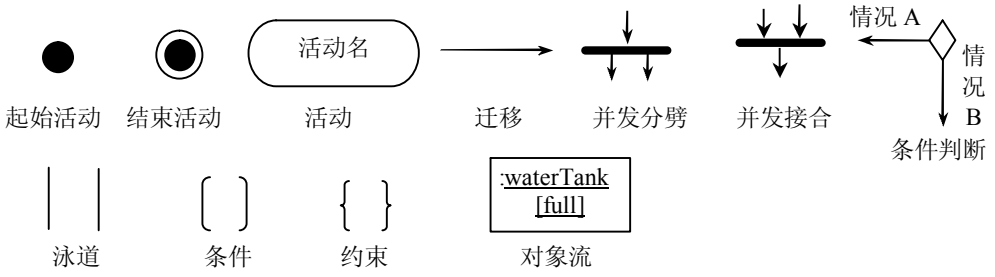

图 3.14　活动图的基本符号

3.3.2　活动图的基本概念

1. 动作状态

动作状态是指原子的、不可中断的动作，并在此动作完成后通过完成转换转向另一个状态。它与活动状态有些相似，但是它们是原子活动并且当它们处于活动状态时不允许发生转换。

动作状态有如下特点：

（1）动作状态是原子的，它是构造 UML 活动图的最小单位。

（2）动作状态是不可中断的。

（3）动作状态是瞬时的行为。

（4）动作状态可以有入转换，入转换既可以是动作流，也可以是对象流。动作状态至少有一条出转换，这条转换以内部的完成为起点，与外部事件无关。

（5）动作状态与状态图中的状态不同，它不能有入口动作和出口动作，更不能有内部转移。

（6）在一张 UML 活动图中，动作状态允许多处出现。

UML 中动作状态用平滑的圆角矩形表示。

2. 活动状态

活动图包含活动状态。活动状态表示过程中命令的执行或工作流程中活动的进行。与等待某一个事件发生的一般等待状态不同，活动状态等待计算处理工作的完成。当活动完成后，执行流程转入到活动图中的下一个活动状态。当一个活动的前导活动完成时，活动图中的完成转换被激发。活动状态通常没有明确表示出引起活动转换的事件，当转换出现闭包循环时，活动状态会异常终止。活动状态用于表达状态机中的非原子的运行，并能够进一步被分解，它们的活动可以由其他的活动图表示。而且，活动状态不是原子的，也就是说它们可以被中断。可以把活动状态看成是一个组合，它的控制流由其他的活动状态和动作状态组成，如工资报表申报审批可以看做是一个活动状态，然后可分解成报表生成、报表提交、报表审批、报表发布四个动作状态。活动图的特点如下：

（1）活动状态可以分解成其他子活动或者动作状态。

（2）活动状态的内部活动可以用另一个 UML 活动图来表示。

（3）和动作状态不同，活动状态可以有入口动作和出口动作，也可以有内部转移。

（4）动作状态是活动状态的一个特例，如果某个活动状态只包括一个动作，那么它就是一个动作状态。

UML 中活动状态和动作状态的图标相同，但是活动状态可以在图标中给出入口动作和出口动作等信息。

3. 分叉控制

活动图可以包含并发线程的分叉控制。对象在运行时可能会存在两个或多个并发运行的控制流，为了对并发的控制流建模，UML 中引入了分叉与汇合的概念。分叉用于将动作流分为两个或多个并发运行的分支，而汇合则用于同步这些并发分支，以达到共同完成一项事务的目的。活动图不仅能够表达顺序流程控制还能够表达并发流程控制，如果排除了这一点，活动图很像一个传统的流程图。

4. 泳道

泳道将UML活动图中的活动划分为若干组，并把每一组指定给负责这组活动的业务组织，即对象。在 UML 活动图中，泳道区分了负责活动的对象，它明确地表示了哪些活动是由哪些对象进行的。这种分配可以通过将活动组织成用线分开的不同区域来表示。在包含泳道的 UML 活动图中，每个活动只能明确地属于一个泳道。

泳道是用垂直实线绘出，由于它们的外观的缘故，这些区域被称作泳道。在泳道的上方可以给出泳道的名字或对象的名字，该对象负责泳道内的全部活动。泳道没有顺序，不同泳道

中的活动既可以顺序进行也可以并发进行，动作流和对象流允许穿越分隔线。每条泳道代表整个工作流程的某个部分的职责，该职责由组织的某个部门来执行。泳道最终可以由组织单元或者业务对象模型中的一组类来实施。

泳道之间的排序并不会影响语义。每个活动状态都指派了一条泳道，而转移则可能跨越数条泳道。

5. 对象流

对象流是动作状态或者活动状态与对象之间的依赖关系，表示动作使用对象或动作对对象的影响。用 UML 活动图描述某个对象时，可以把涉及到的对象放置在 UML 活动图中并用一个依赖将其连接到进行创建、修改和撤销的动作状态或者活动状态上，对象的这种使用方法就构成了对象流。

对象流中的对象有以下特点：

（1）一个对象可以由多个动作操作。

（2）一个动作输出的对象可以作为另一个动作输入的对象。

（3）在 UML 活动图中，同一个对象可以多次出现，它的每一次出现表明该对象正处于对象生存期的不同时间点。

对象流用带有箭头的虚线表示。如果箭头是从动作状态出发指向对象，则表示动作对对象施加了一定的影响。施加的影响包括创建、修改和撤销等。如果箭头从对象指向动作状态，则表示该动作使用对象流所指向的对象。对象流状态表示活动中输入或输出的对象。如果活动有多个输出值或后继控制流，那么箭头背向分叉符号。同样，多输入箭头指向结合符号，对象结点和可中断的区域等用得少。

6. 活动的分解

一个活动可以分为若干个动作或子活动，这些动作和子活动本身又可以组成一个 UML 活动图。不含内嵌活动或动作的活动称之为简单活动，嵌套了若干活动或动作的活动称为组合活动。组合活动有自己的名字和相应的子 UML 活动图。

3.3.3　活动图的构建

要创建一个 UML 活动图，需要反复执行下列步骤。

（1）标识需要活动图的用例。一个系统用例模型包含多幅用例图，每幅图又包含多个用例，一般情况下，不需要对每个用例绘制活动图，只有当实现该用例的步骤繁杂或者有特殊需要的时候才会画它。因此，要首先确定建模的内容，即要对哪个用例建立活动图。在“进销存管理系统”中，“销售合同”用例和“核对付款单”用例需要画出活动图。

（2）建模每一个用例的主路径。在创建用例的活动图时，需要先确定该用例一条明确的执行工作流程，建立活动图的主路径，然后以该路径为主线进行补充、扩展和完善。在“销售合同”用例中，销售合同从签订到履约的整个过程组成了该活动图的基本执行轨迹。

（3）建模每一个用例的从路径。首先根据主路径分析其他可能出现的工作流的情况，这些可能是活动图中还没有建模的其他活动，可能是处理错误，或者是执行其他的活动。然后对不同进程中并发执行的活动进行处理。“销售合同”中的“核对货物清单”和“核对付款单”就是 2 个从路径。

（4）添加泳道来标识活动的事务分区。泳道是将活动用线条分成一些纵向的矩形，这些矩形称为泳道。每个矩形属于一个特定的对象或部门（子系统）责任区。使用泳道可以把活动

按照功能或所属对象的不同进行组织。属于同一个对象的活动都放在同一个泳道内，对象或子系统的名字放在泳道的顶部。“销售合同”涉及到“仓库管理”、“合同管理”和“财务管理”三类功能，所以划分为不同的泳道，便于对“销售合同”用例进行详细的分析。销售管理中的“销售合同”用例的活动图如图 3.15 所示。

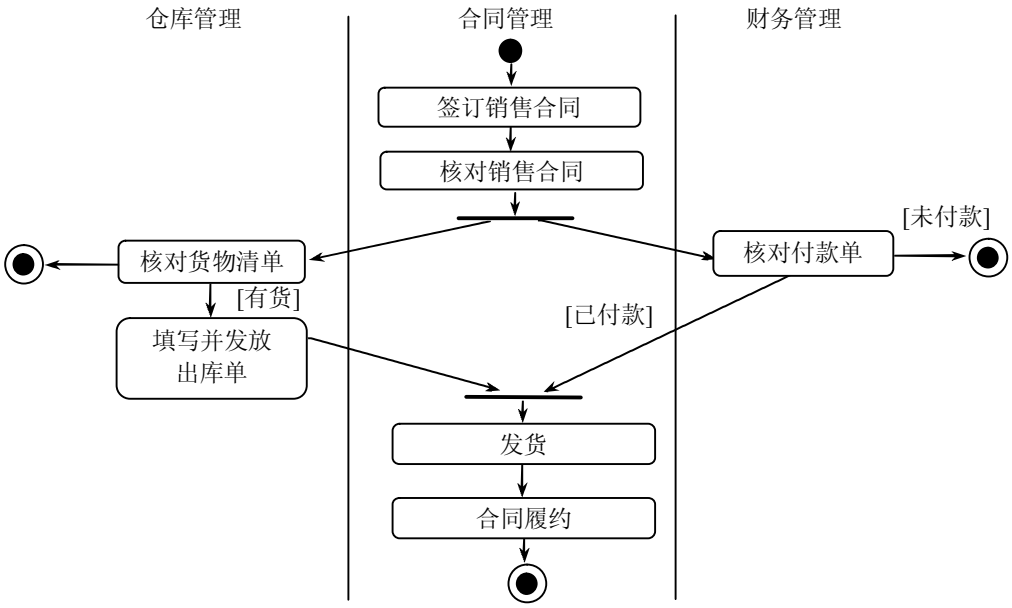

图 3.15　销售合同从签订到履约的活动图

（5）改进高层的活动。对于一个复杂的系统，需要将描述系统不同部分的活动图按照结构层次关系进行排列。在一个活动图中，其中的一些活动可以分解为若干个子活动或动作，这些子活动或动作可以组成一个新的活动图。采用结构层次的表示方法，可以在高层只描述几个组合活动，其中每个组合活动的内部行为在展开的低一层活动图中进行描述，便于突出主要问题。如图 3.15 描述的销售合同从签订到履约的活动中，“核对付款单”就是一个组合活动。该组合活动可以进一步地分解为如图 3.16 所示的活动图。

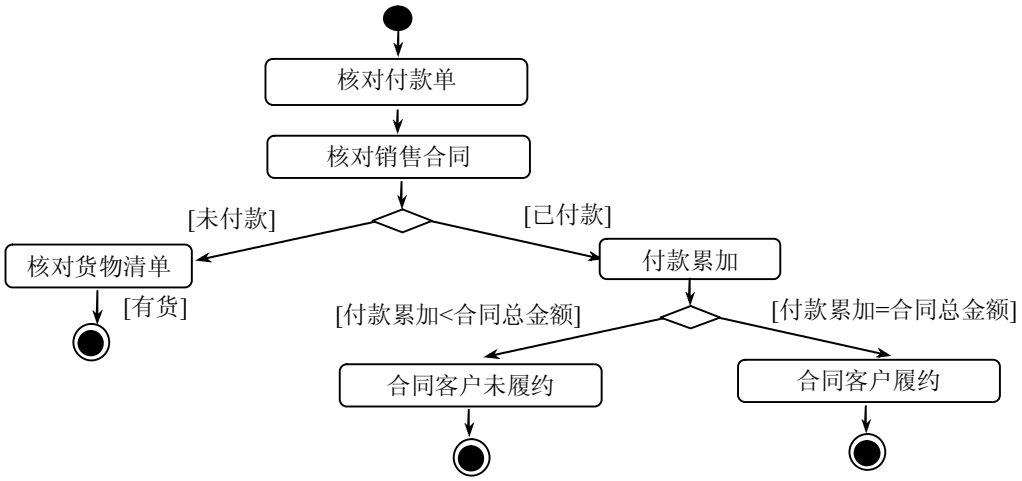

图 3.16　“核对付款单”子活动图

（6）进一步对细节进行完善。对前面的活动图进行补充和完善。

活动图的主要应用有：

（1）描述用例的行为。活动图对用例描述尤其有用，它可建模用例的工作流，显示用例内部和用例之间的路径；它也可以向读者说明需要满足什么条件用例才会有效，以及用例完成后系统保留的条件或者状态。

（2）理解工作流程。活动图对理解业务处理过程十分有用。可以画出描述业务工作流的活动图与领域专家进行交流，明确业务处理操作是如何进行的，将会有怎样的变化。可以显示用例内部和用例之间的路径，说明用例的实例如何执行动作以及如何改变对象状态。

（3）描述复杂过程的算法。在这种情况下使用的活动图和传统的程序流程图的功能是差不多的。活动图不过是 UML 版的程序流程图，常规的顺序、分支过程在活动图中都能得到充分的表现。每一个活动图只能有一个开始状态，但是可以有无数个结束状态。很多人将 UML 的活动图理解为大家经常使用的业务流程图，特别是在活动图中也可以使用泳道。但要注意，UML 的活动图和一般的流程图是有区别的。

接下来就是要充分地和客户沟通，找出业务流程的错误与不足之处。这一步对建立正确的系统模型至关重要。此处产生的错误会延续到系统验收、部署阶段，会极大地增加开发成本。因此必须和用户反复沟通，获得用户的认可，并尽量找出所有的分支情况。

UML 活动图与流程图的区别：

（1）流程图着重描述处理过程，它的主要控制结构是顺序、分支和循环，各个处理过程之间有严格的顺序和时间关系。而 UML 活动图描述的是对象活动的顺序关系所遵循的规则，它着重表现的是系统的行为，而非系统的处理过程。

（2）UML 活动图能够表示并发活动的情形，而流程图不能。

（3）UML 活动图是面向对象的，而流程图是面向过程的。所以两者的最大区别在于，活动图是以 OOAD 为指导的，以对象为分析基础，配合状态机图使用，为的是对用例的执行流程进行描述，而不是以现实中的业务流程为视角。

3.4　需求分析规格说明

软件需求说明（Software Requirements Specification，SRS）的编制是为了使用户和软件开发者双方对该软件的初始规定有一个共同的理解，使之成为整个开发工作的基础。根据国标 GB/T9385-2008 计算机软件需求规格说明规范的要求，软件需求说明包括以下的内容：

1 引言

SRS 的引言部分应当提供整个 SRS 的概述，包括以下各条：目的；范围；定义、简称和缩略语；引用文件；综述。

1.1 目的

描述 SRS 的目的；说明 SRS 的预期读者。

1.2 范围

通过名称识别要生产/开发的软件产品；必要时说明软件产品将要或不做什么；描述规定的软件的应用，包括相关的收益、目标和目的；如果上层规格说明（如系统规格说明）存在，与上层规格说明类似的陈述保持一致。

1.3 定义、简称和缩略语

本条提供对正确解释 SRS 所要求的所有术语、简写和缩略语的定义，这些信息可以通过引用 SRS 中的一个或多个附录、或者引用其他文件的方式来提供。

1.4 引用文件

提供 SRS 引用的所有文件的完整清单；标识出每个文件的名称、报告编号（适用时）、日期、出版组织；标明可以获得引用其他文档的方式。

1.5 综述

描述 SRS 的其余章条包含的内容；说明 SRS 是如何组织的。

2 总体描述

描述影响产品及其需求的一般因素，而不叙述具体的需求。相反，它提供需求的背景并使它们更易理解，在 SRS 的第 3 章将详细定义这些需求。

产品描述；产品功能；用户特点；约束；假设和依赖关系和需求分配等。

2.1 产品描述

把产品置于其他有关产品的全景之下。如果产品是独立的和完全自我包含的，这里宜如实给予陈述。正如平常出现的那样，如果 SRS 定义的产品是较大系统的组成部分，则本章宜将软件的功能性与较大系统的需求相联系，而且宜识别软件和系统之间的接口。

使用框图展示较大系统的主要部分、相互联系以及外部接口是有帮助的。

更高层规格说明本条也宜描述在各种不同的约束下软件如何运行，如这些约束包括：系统接口；用户界面；硬件接口；软件接口；通信接口；内存；运行；现场适应性需求。

2.2 产品功能

给出软件将执行主要功能的概要。例如，某个会计程序的 SRS 可在此部分关注客户账户维护、客户财务报表及发票准备，而不涉及这些功能要求的大量细节。

有时，本条需要的功能概要可直接从分配具体功能到软件产品的更高层规格说明（如果存在）中摘录。为了清晰，应当注意：功能宜以这样的方式组织，以使顾客或第一阅读该文件的任何读者对功能列表容易理解；可以使用文本或图示的方法，显示不同的功能及其之间的关系。这样的图示不比现实产品的设计，但简要显示变量之间的逻辑关系。

2.3 用户特点

给出软件产品预期用户的一般特征，包括教育程度、经验、专业技术情况。它不宜指出具体的需求，但宜给出 SRS 第 3 章中为何规定某些具体需求的原因。

2.4 约束

给出将会限制开发人员选择的其他事项的一般描述。包括：法规政策；硬件局限（如信号时间要求）；与其他应用接口；并行操作；审核功能；控制功能；高级语言需求；信号握手协议（如 XON-XOFF、ACK-NACK）；可靠性需求；使用的关键性；安全和保密安全考虑。

2.5 假设和依赖关系

列出影响 SRS 规定需求的每个因素。这些因素不是软件设计的限制条件，但是，它们的任何变更可能影响 SRS 中的需求。例如，某个假设可能是软件产品制定的硬件具有某个特定操作系统，如果事实上该操作系统不能使用，那么 SRS 将做相应的修改。

2.6 需求分配

识别可能推迟到系统将来版本的需求。

3 具体需求

本章宜包括足够详细的所有软件需求，使设计人员能够设计系统以满足这些需求，并且

使测试人员能够测试该系统满足这些需求。贯穿本章，对于用户、运行人员或其他外部系统，每个规定的需求应当是外部可理解的。这些需求至少应当包括，每个系统输入（激励）、每个系统输出（响应），以及系统通过响应某个输入或支持某个输出所执行的所有功能。由于这通常是 SRS 篇幅最大和最主要部分，以下原则适用：规定的具体需求宜符合好的 SRS 的特征；具体需求宜引用较早的相关文件；所有的需求宜是唯一可标识的；宜注意需求组织，使其具有最大的可读性。

在考察组织需求的具体方式之前，了解 3.1 到 3.6 组成需求的各个不同项是有益的。

3.1 外部接口需求

本条宜是软件系统所有输入和输出的详细描述。它宜是对前面的接口描述的补充，不宜重复前面已有的信息。

宜包括以下内容和格式：项的名称；目的描述；输入源和输出目的地；有效范围、准确度和/或容限；测量单位；定时；与其他输出/输入的关系；屏幕显示/组织；窗口格式/组织；数据格式；命令格式；结束消息。

3.1.1 用户界面

指出用户使用软件产品时的界面需求。若用户通过显示终端操作，则需指定如下需求：对屏幕格式的要求；报表或菜单的页面显示格式及内容；用户命令的格式。列出输出错误信息的格式。尽可能采用开发工具构造界面，使需求定义和设计、编码相衔接，应遵从《界面设计规范》。

3.1.2 硬件接口

软件产品与系统硬设备之间每一接口的逻辑特点；硬件接口支持的设备；软件与硬件接口之间以及硬件接口与支持设备之间的约定。

3.1.3 软件接口

描述该软件产品与其他有关软件接口关系，并指出这些软件的名字、助记符及版本号。

3.1.4 通信接口

说明各种通信接口及协议。

3.2 类/对象

3.2.1 类/对象 1

3.2.1.1 属性（直接的或继承的）

3.2.1.1.1 属性 1

……

3.2.1.1.n 属性 n

3.2.1.2 功能（服务、方法、直接的或继承的）

功能需求宜定义软件在接受和处理输入以及处理和产生输出中必须发生的基本动作。一般情况下使用“系统应…”的方式来陈述。这些包括：对输入有效性的核查；操作的准确顺序；异常情况，包括：溢出、通信设施、错误处理和恢复；参数影响；输入和输出的关系，包括：输入/输出顺序、从输入到输出转换的公式。

尽管将功能需求划分为子功能或子过程可能是适当的，但这并不意味着软件设计同样以这样的方式划分。

3.2.1.2.1 功能需求 1.1

……

3.2.1.2.m 功能需求 1.m

3.2.1.3 消息（接受的或发送的通信）

3.2.2 类/对象 2

…….

3.2.p 类/对象 p

3.3 性能需求

正常情况下和峰值工作条件下，在一定时间内要处理的数据总量；响应时间；输出结果精度。

本条宜规定软件或人与软件互作用的整体静态的和动态的数量化需求。静态数量化需求可能包括：支持的终端数量；支持同时运行的用户数量；要处理的信息量和类型。

有时，静态数量化需求包含在命名为“能力”的独立部分。

动态数量化需求可能包括，如在正常和高峰工作负载条件下，在某时段内处理的事务处理数、任务数和数据量。所有这些需求宜以可测量的方式规定。如应在小于 1s 内处理 95%的交易量，而不是操作方不需等待事务处理结束。

注：适用于某个具体功能的数量化限制，通常作为该功能处理描述部分予以规定。

3.4 设计约束

宜规定可能由其他标准、硬件局限等引发的设计约束。

3.5 软件系统属性

有一些软件属性可以作为需求。规定所需求的软件属性是重要的，这样才能客观地验证属性的实现情况。

3.6 其他需求

3.5 需求分析用例建模案例

在这里以大家熟悉的“图书馆管理信息系统”为例介绍需求分析，并用用例图、用例描述和活动图表示。

3.5.1 需求陈述

由于“图书馆管理信息系统”很复杂，在这里仅以高校“图书管理”为例进行介绍。在对图书管理系统进行详细调查与分析后，写出其需求陈述的文档。

“图书管理信息系统”的用户需求陈述如下：

在图书流通管理系统中，管理员要为每个读者建立借阅账户，并给读者发放不同类别的借阅卡（借阅卡可提供卡号、读者姓名、类别、单位、职称等）。持有借阅卡的读者可以借阅、归还、预约和续借图书，不同类别的读者可借阅图书的范围、数量和期限不同。读者来图书馆借书，可能先查询馆中的图书信息，查询可以按书名、作者、图书编号、关键字进行。如果查到则记下书号，交给流通组工作人员，等待办理借书手续。如果该书已经被全部借出，可做预定登记，等待有书时被通知。如果图书馆没有该书的记录，可进行缺书登记。

办理借书手续时要先出示借阅证，借阅图书时，先输入读者的借阅卡号，系统验证借阅卡的有效性和读者是否可继续借阅图书。如果借阅卡无效，让读者到办公室进行补办；如果借书数量超出规定，则不继续借阅；如果有过期图书，则需要交纳罚款；如果可以借书，则显示

读者的基本信息（包括照片）供管理员人工核对。借书时系统登记图书证编号、图书编号、借出时间和应还书时间。如果是预约图书，还要修改预约记录。

当读者还书时，流通组工作人员根据图书证编号找到读者的借书信息，察看是否超期。如果已经超期，则进行超期处罚；如果图书有破损、丢失，则进行破损及丢失等处罚。登记还书信息，做还书处理，同时查看是否有预定登记，如果有则发出到书通知。

图书采购人员采购图书时，要注意合理采购。如果有缺书登记，则随时进行采购。采购到货后，编目人员进行验收、编目、上架、录入图书信息、发到书通知。图书馆管理人员定期对图书进行盘库，如果图书丢失或旧书淘汰，则将该书从书库中清除，即图书注销。

以上是图书管理系统的基本需求。经过与图书馆工作人员的反复交流，他们提出了下列建议：

1）当借阅的图书到期时，希望能够提前用短信或电子邮件方式提示读者。

2）读者希望能够实现网上查询和预定图书。

3）应用系统的各种参数设置最好是灵活的，由系统管理人员根据需要设定，如借阅期的上限，还书提示的时间，预定图书的保持时间等参数。

3.5.2　需求分析

为了理解系统所要解决的业务问题，以便掌握用户需求，可以采用用例图进行需求建模。用例图通过列出用例和角色，显示用例和角色的关系，从而给出了目标系统功能。用例建模如下：

（1）确定参与者。通过对系统需求陈述的分析，可以确定系统有如下执行者：读者，流通组工作人员，办公室工作人员，采购员，编目人员。读者可查询图书信息和个人借阅信息，还可以在符合续借的条件下自己办理预约及续借图书；流通组工作人员完成读者借书、还书、预约和续借图书及罚款等管理工作；办公室工作人员为读者办理借书证有关事宜；采购人员对图书进行订购；编目人员进行图书的编目。

（2）确定用例。在确定参与者之后，结合图书管理的领域知识，进一步分析系统的需求，识别出的用例有：借书、还书、缺书登记、预约、取消预约、查询、通知、读者管理、图书管理、处罚、采购、编目、图书催还等。

（3）系统用例的审查与细化。要对系统用例逐一进行审核。用例的审核必须有用户参与，可以通过用户访谈和讨论会的形式对系统用例进行审核。确定每一个用例实现了用户的哪些需求。是不是用户的每一个需求都有相应的用例来实现。基本路径和扩展路径是否完备，是否存在冗余。

（4）确定用例之间的关系。确定参与者和用例之后，进一步确定用例之间的关系。业务用例图如图 3.17 所示。

在用例图中的系统用例仅仅是一个名字，还需要对每个用例进行详细描述。在建立完用例图后，需要为图书流通管理系统业务用例图中某些用例编写用例文档。用例文档中应包括如下内容：名称；描述；前置条件；后置条件；活动的基本过程和扩展活动等。在用例文档中还可添加一些可选内容，如参与者、状态、扩展点、被包含的用例、变更历史等。如借阅的用例描述如下：

用例名称：借书

用例描述：当读者前来图书馆借书时，流通组工作人员启动该用例，该用例检查读者的

有效性及借阅记录，检查图书是否在库，实现读者借书活动。

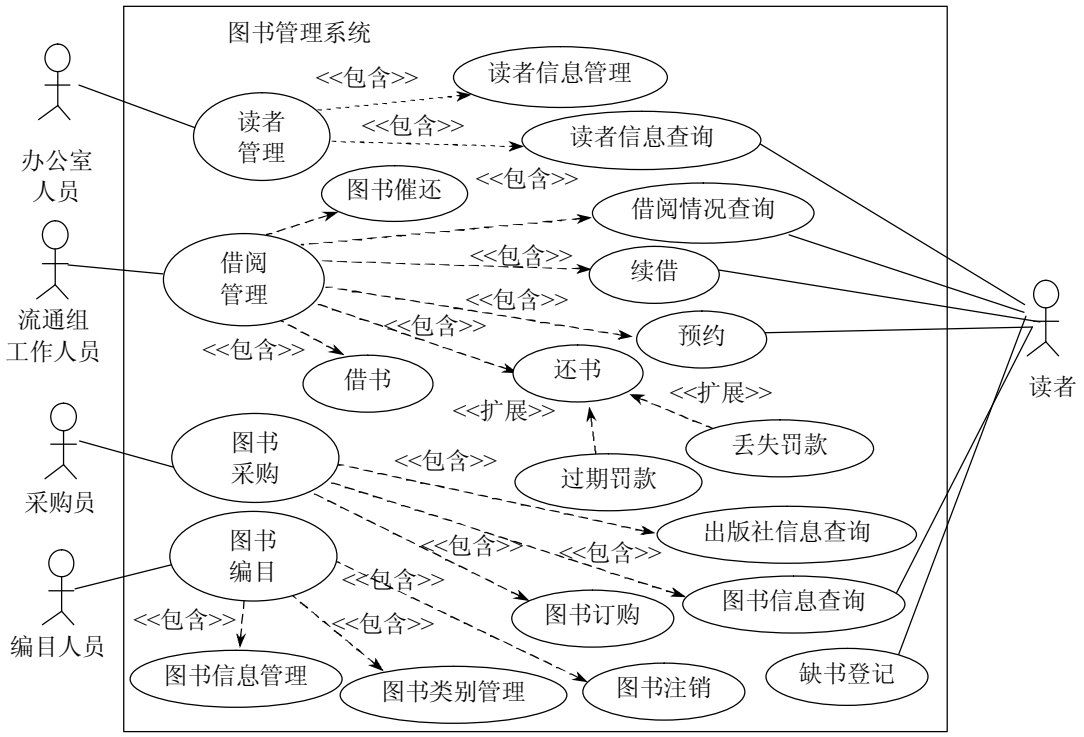

图 3.17　图书管理业务用例图

参与者：管理员。

前置条件：

流通组工作人员要先执行登录用例，才能启动借书用例。

读者号存在

图书号存在

图书在库

后置条件：

记录了借阅信息

图书库存减少

创建借还书记录

活动的基本过程：

①输入读者账号；

②输入图书编号；

③添加一条借书记录；

④“图书信息表”中“现有库存量”–1；

⑤“读者信息表”中“已借书数量”+1；

⑥提示执行情况；

⑦修改预约记录；

⑧清空读者、图书编号等输入数据；

重复第②～⑧步，直到读者选择结束借阅；

⑨选择“退出”；

⑩返回上一级界面；

扩展路径：

读者号不存在，显示读者无效

读者号存在，借书数量已经超限，显示数量超限

读者号存在，图书号无效，显示图书不存在

图书号，读者有效，图书不在库存，转预定处理。

每个用例都可以建立一个活动图。读者借书的活动图如图 3.18 所示。

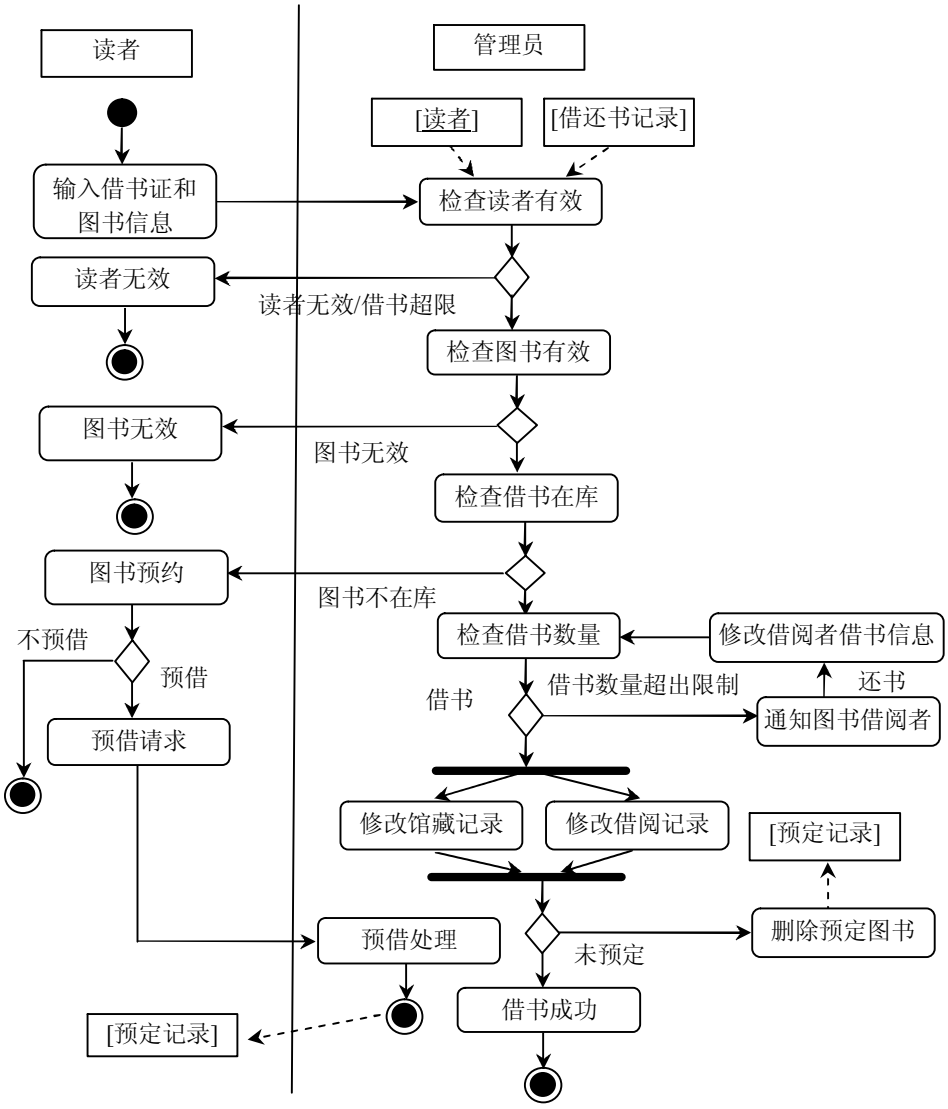

图 3.18　借书活动图

3.5.3 系统开发方案

1. 新系统目标

图书馆管理信息系统是为了适应图书馆综合管理的需求，改变传统管理模式，加速图书馆管理的自动化、标准化和科学化，而建立的一个整体性的图书馆操作系统。它可以为图书馆管理决策部门提供可靠的信息依据，为提高图书馆的社会效益服务。具体如下：

（1）在图书馆采访、编目、流通、阅览等业务部门全部实现自动化管理，书目数据实现标准化。

（2）充分发挥图书馆馆藏的作用，提高藏书利用率。

（3）读者可通过公共查询系统进行馆藏查询、个人数据查询，实现远程查询。

（4）自行办理图书预约、续借手续，自动进行各种统计和计算，提供辅助决策支持，以缩短决策周期。

了解图书馆服务的相关信息，加强与读者沟通，还可根据不同授权，检索、利用图书馆光盘镜像服务器提供的中文镜像数据、光盘数据等。

2. 新系统方案

（1）系统规划及初步开发方案。根据系统的开发目标，以及现行系统存在的主要问题，建议新系统采用微机网络系统，能与校园网及 Internet 连接，便于与供书商交流。能够实现业务管理自动化；输入、输出标准化；文献存储高密度化；情报利用大众化。新系统具有图书的采编、流通、读者服务和系统维护等功能，如图 3.19 所示。

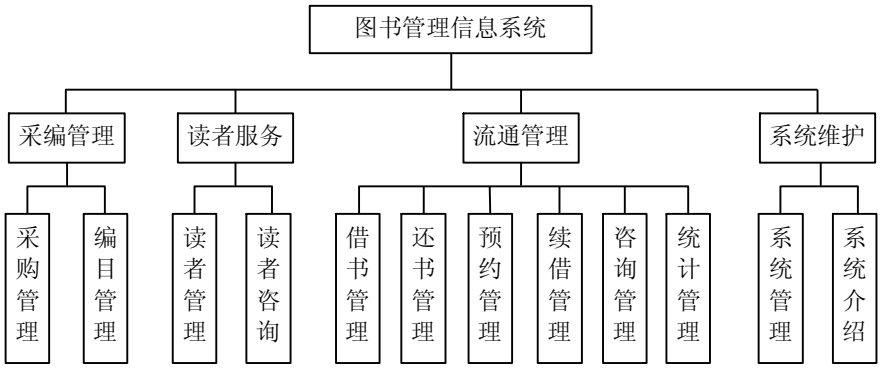

图 3.19　图书管理系统的功能

（2）系统的实施方案。根据新系统的开发方案，确定整个项目的阶段性目标，列出分段实施进度计划与计划安排等情况（略）。

（3）投资方案（略）。

（4）人员培训及补充方案（略）。

图书馆管理信息系统虽然对现行的管理体制有影响，但不强烈，重点是加强基础设施建设，以适应自动化管理。专业人员的变动不大，除了增加一部分计算机专业人员外，经过培训，现有的人员将逐步适应自动化管理的要求，学会图书馆管理信息系统的使用。

3.5.4　系统可行性分析

1. 技术可行性分析

目前已经成功地建立了许多复杂的管理信息系统，而图书馆管理信息系统是比较简单的，系统开发人员具备开发的能力，有成熟的 C/S 和 B/S 技术。因此从技术上来说，完全可以建成一个适用的图书馆管理信息系统。

2. 经济效益分析

图书馆管理信息系统所产生的经济效益，与众多的因素有关，不宜采用传统的一次性投资效益估计，因为开发系统的投资用在管理领域，但经济效益体现在科研、教学等诸多方面。它可以使管理体制合理化和管理信息标准化，可以使文献更好地被利用，可以改进管理的手段。新系统的统计分析功能更强大，可以更好地为文献采购提供依据，使得采购的文献使用性更强；可以及时了解书库中图书在库情况及读者借阅情况，可以进行预约和催还，更好地提高文献的利用率。管理信息系统所带来的效益是很难定量估计的，但新系统使用后可以减少工作人员，因此，从经济上说是可行的。

3. 运行管理方面

现有的图书馆管理人员只要进行培训完全可以胜任工作，对于缺少的计算机管理人员可以通过招聘解决。现有的运行环境只要稍加改进就可以保证新系统运行，从运行管理方面看也是可行的。

4. 结论

由于管理信息系统的开发在国内外是一个技术上成熟的系统，并且有可行的开发方案和切实的工程技术保证，有学校领导的大力支持以及人员和资金的保证，因此开发图书馆管理信息系统是完全可行的。

小　结

在实际的系统开发过程中，用户的需求往往是很难捕捉的，而且经常变动。甚至连用户自己也常常无法准确描述自己的需求，他们的需求往往在看到软件产品后才逐步地清晰起来。因此在需求分析阶段应该采用好的需求分析方法和技术，以保证得到高质量的用户需求。用例是系统、子系统或类和外部的参与者交互的动作序列的说明，包括可选的动作序列和会出现异常的动作序列。用例命名往往采用动宾结构或主谓结构，但最常见的是动宾结构。系统需求一般分功能性需求和非功能性需求，用例只涉及功能性需求。用例之间可以有泛化关系、包含关系和扩展关系等。参与者是指系统以外的、需要使用系统或与系统交互的东西，包括人、事物和系统等，参与者之间可以有泛化关系。用例的描述是用例的主要部分。用例的描述格式没有一个统一标准，不同的开发机构可以采用自认为适合的格式。用例分析结果的好坏与分析人员的个人经验和领域知识有很大关系。用例模型包括用例图和用例文档（用例描述）。

用例是一种用来探索需求的技术，而需求和设计之间的区别在于需求解决的是系统“做什么”的问题；而设计则是针对需求中提出的问题，解决系统该“怎么做”的问题。需求调研的过程是发现和界定问题的过程，设计的过程是寻找解决方案的过程。需求分析的思维方式是总结和抽象，系统设计的思维方式是分解和细化。用例规约包括执行者、前置条件、交互步骤、主事件流、备选事件流和后置条件等。活动图描述了业务用例中，用户可能会进行的操作序列。

活动图描述的是对象活动的顺序关系所遵循的规则，它着重表现的是系统的行为，而非系统的处理过程。活动图是面向对象的，能够表示并发活动的情形。活动图有个很重要的使命：从业务用例分析出系统用例。

复习思考题

一、选择题

1．用例之间的关系主要有（　　）。

A．聚合　　B．继承　　C．扩展　　D．包含

2．基于用例图的需求捕获的第一步就是确定系统的参与者，在寻找系统参与者时，可以根据以下（　　）等问题来确定。

A．系统同环境如何进行交互　　B．由谁安装系统

C．系统为哪些对象提供信息、服务　　D．系统的使用者是谁

3．如果用例 B 是用例 A 的某项子功能，并且建模者确切地知道在 A 所对应的动作序列中何时将调用 B，则称（　　）。

A．用例 A 扩展用例 B　　B．用例 A 继承用例 B

C．用例 A 包括用例 B　　D．用例 A 实现用例 B

4．如果用例 A 与用例 B 相似，但 A 的动作序列是通过改写 B 的部分或者扩展 B 的动作而获得的，则称（　　）。

A．用例 A 实现用例 B　　B．用例 A 继承用例 B

C．用例 A 扩展用例 B　　D．用例 A 包括用例 B

5．如果用例 A 与用例 B 相似，但 A 的功能较 B 多，A 的动作序列是通过在 B 的动作序列中的某些执行点上插入附加的动作序列而构成的，则称（　　）。

A．用例 A 扩展用例 B　　B．用例 A 包含用例 B

C．用例 A 继承用例 B　　D．用例 A 实现用例 B

6．在 UML 中，（　　）表示使用软件系统的功能，与软件系统交换信息的外部实体。

A．执行者　　B．类　　C．用例　　D．用例图

7．用例之间的关系主要有（　　）。

A．包含　　B．继承　　C．扩展　　D．聚合

8．在 UML 活动图中，（　　）表示一个操作完成后对其后续操作的触发。

A．信息流　　B．控制流　　C．初始活动　　D．活动

9．对于活动图，以下说法正确的有（　　）。

A．活动图适用于精确地描述单个用例中的处理流程，也可用来描述多个用例联合起来形成的处理流程，表达相对复杂的业务操作或软件处理过程，有时甚至可以针对类中某个复杂的操作用活动图给出实现细节

B．活动图中包含控制流和信息流，控制流表示一个操作完成后对其后续操作的触发，信息流则刻画操作期间的信息交换

C．活动图的基本建模机制包括结点、边及泳道

D．活动图描述实体为完成某项功能而执行的操作序列，其中的某些操作或者操作的

子序列可以并发和同步

10.（　）是构成用例图的基本元素。

A．参与者　　B．泳道　　C．系统边界　　D．用例

11．下面不是用例间主要关系的是（　）。

A．扩展　　B．包含　　C．依赖　　D．泛化

12．下列对系统边界的描述不正确的是（　）。

A．系统边界是指系统与系统之间的界限

B．用例图中的系统边界是用来表示正在建模的系统的边界

C．边界内表示系统的组成部分，边界外表示系统外部

D．我们可以使用 Rose 绘制用例中的系统边界

二、填空题

1．在 UML 提供的图中，_____用于描述系统与外部系统及用户之间的交互。

2．用例图两个最核心的元素是_____与_____。

3．用例的组成要素是_____、_____和_____。

4．用例中的主要关系有_____、_____和_____。

5．用例图中以实线方框表示系统的范围和边界，在系统边界内描述的是_____，在边界外描述的是_____。

三、名词解释

需求分析　活动图　动作状态　活动　泳道

四、综合题

1．简述扩展、包含和泛化三种 UML 依赖关系的异同。

2．什么是用例？什么是用例图？两者之间有什么不同？

3．简述用例模型的组成元素以及建模步骤。

4．用例图在系统中有什么作用？

5．试述如何识别用例？

6．用例之间的三种关系各使用在什么场合？

7．请问在设计系统时，绘制的用例图是多一些好，还是少一些好，为什么？

8．请简述为何在系统设计时要使用用例图。它对我们有什么帮助？

9．用例之间的关系各使用在什么场合？

10．请画出 ATM 取款机取款的用例图，为取款进行用例规约描述。

11．请画出图书馆采购管理的用例图，为查询进行用例规约描述。

12．支持跨行业务；插入卡片；输入密码；选择服务；取钱；存钱；挂失卡片；缴纳费用；警示骗子；三次错误吞没卡片。请判断一下哪些是有效用例。