第3章 业务需求分析

CHAPTER 3

软件项目一定是为了实现某些业务目标而实施，而不是因为项目建设方觉得有趣而做的。软件项目通过改变现有系统的业务运作方式，最终帮助客户实现高层的战略目标和长远规划。在学习软件工程方法论时，一定要认识到业务是如何与软件开发过程发生联系的。软件工程方法论通常假设当软件开发项目开始运作前，业务建模已经完成。但实际情况是，很多软件开发组织在项目开始之前并没有对业务进行深入分析。这容易导致一个典型的现象：当一个软件开发新手被问到为什么要开发某些系统功能时，他无法清晰阐述理由。究其根本原因是软件开发人员并没有真正理解系统功能所对应的业务需求。

本章就如何展开业务需求分析进行阐述，为后续的系统需求分析提供工作基础。业务需求分析阶段主要工作内容如图 3-1 所示。

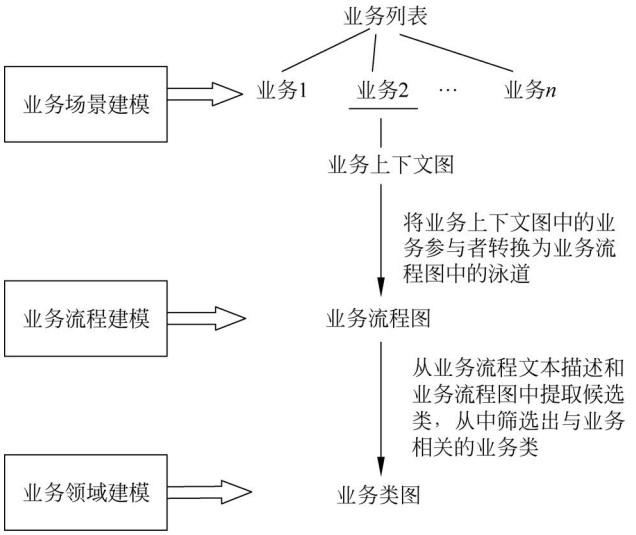


图 3-1　业务需求分析的主要内容

在图 3-1 中，业务需求分析的主要工作包括三个步骤。

第一步：以组织为边界识别出业务执行者和组织交互实现的核心业务列表，借助交互分析技术创建上下文图，完成业务场景建模。

第二步：在业务上下文图的基础上，针对每一个业务进行业务流程分析，借助流程图技术创建业务流程图，完成业务流程建模。

第三步：针对每个业务流程创建局部的领域类图，借助类图技术创建业务类图，然后将所有的局部领域类图进行合并、抽象、提取，形成全局的领域模型，完成业务领域建模。

深入思考 3.1 在软件生命周期中,有一个系统需求分析阶段,为什么本章要把业务需求分析单独拿出来介绍?

企业观点:在企业实践中发现,很多软件开发新手不清楚“需求”在不同阶段的含义,也不清楚业务需求和系统需求的区别。在软件开发实践中,业务需求才是真正体现组织价值的,软件开发项目立项是为了更好地实现业务需求。业务需求分析的目的是为下一步厘清系统需求奠定基础,业务需求分析是进行系统需求分析前的一项重要工作。

详情请参见微课视频 3-1。

3.1 需求

3.1.1 系统与软件

在讨论各种需求的差异与联系之前,先来搞清楚两个容易混淆的概念:系统与软件。系统和软件这两个词在工作实践中经常互换使用,在沟通中会造成一定的混乱。

什么是系统?

在《汉语大词典》中对“系统”的解释是:自成体系的组织,同类事物按一定秩序和内部联系组合成的整体。

在韦氏大辞典中对“系统”的解释是:系统是“有组织的或被组织化的整体;结合着的整体所形成的各种概念和原理的综合;由有规则的相互作用、相互依存的形式组成的诸要素集合”。

总之,系统被定义为由一些相互联系、相互制约的要素构成的复杂整体。尽管系统一词频繁出现在社会生活和学术领域中,但不同的人在不同的场合往往赋予它不同的含义。它可以用来描述计算机系统、电力系统、生物系统或者业务系统等各种不同类型的系统。

广义的系统中包含的构成要素很多。对于工程项目中提到的系统,系统构成要素一般包括硬件、软件、固件、人员、信息、技术和服务等。如果将范围缩小到计算机系统,并且不考虑开展业务活动所需的人员、服务等其他要素,本书简化地认为,构成计算机系统的要素主要包括硬件系统和软件系统,即

$$\text{计算机系统} = \text{硬件系统} + \text{软件系统}$$

例如:要开发一个共享单车管理系统,这个系统就包含硬件系统(单车车体+智能锁)和软件系统(智能锁控制软件+后台计费管理系统)两部分,此时的系统需求就需要从硬件和软件两方面分别进行描述。

如果待开发的计算机系统不包含硬件部分,就称其为一个纯软件系统。例如:人力资源管理系统就是一个纯软件系统。此时,可以认为“系统”等同于“软件”,“软件需求”等同于“系统需求”。

本书围绕软件工程的相关理论展开描述,因此,如无特别说明,本书后续提到的系统都是指软件系统。

3.1.2 需求分类

“需求”这个词语本身的概念很宽泛,可以用在各种场合。在项目开发过程中,存在各种加了前缀的需求,如产品需求、市场需求、业务需求、系统需求等。

什么是“需求”?

首先了解“干系人”的概念,《项目管理知识体系》中将干系人定义为:“积极参与项目或

其利益将受到项目执行的正面或负面影响的个人或机构”。国际业务分析协会(International Institute of Business Analysis，IIBA)在《业务分析知识体系指南》中，对需求的定义采用了电气与电子工程协会(IEEE)多年前的一个定义：需求是干系人解决问题或者实现目标需要的条件或者能力。在这个简短的定义中，并没有提到需求呈现的表现形式或格式，因此，需求分析工程师可以根据项目需要，采用各种能够清晰表达需求的形式来展现，以便进行讨论、分析、文档化和确认。常见的需求呈现形式是：文字、图、表等。在企业软件开发实践中，软件开发人员常常需要与用户沟通和讨论需求，在需求文档中会使用各种类型的图表来表达需求。

下面对软件项目开发实践中常见的需求类型进行简单分类。

1. 按需求层次分类

在国际系统工程协会(International Council on System Engineering，INCOSE)编著的《系统工程手册》中，有一张“需要到需求的转换”图，如图 3-2 所示。在图 3-2 中，左侧是从企业组织层面思考问题，描述需要；右侧展示了需求工程中的三个维度(业务需求、利益攸关者需求、系统需求)，强调了从需要向需求的转换。

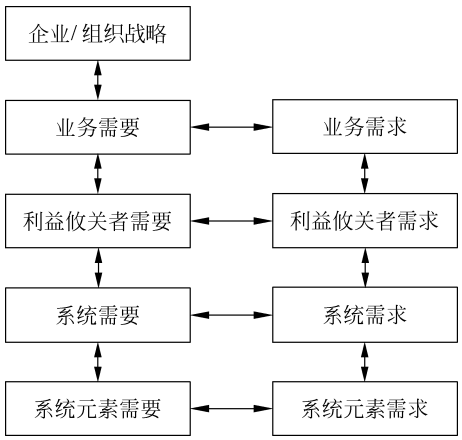


图 3-2　INCOSE《系统工程手册》需要到需求转换图

企业需要系统工程，软件需要软件工程。软件工程将系统工程中的规范化流程管理应用到软件开发过程中。本书结合系统工程的需求形成理论和软件开发过程，在图 3-2 的基础上，将软件项目的需求自上而下、由粗到精进行分层，形成如图 3-3 所示的需求层次。

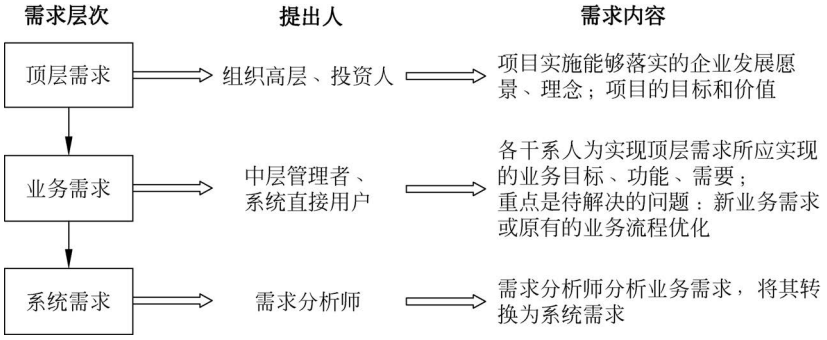


图 3-3　需求层次分类

在图 3-3 中，按照项目开发实践中需求获取的顺序以及需求之间的关联关系，将需求分为 3 个层次：顶层需求、业务需求、系统需求。

1) 顶层需求

顶层需求的提出人通常是项目投资人、项目建设方的决策者、项目实施方的负责人等,他们对项目提出了目标和期望。顶层需求的描述要反映提出人为什么要实施该项目,希望项目达到什么样的目标,目标的阐述通常采用愿景、战略、理念、价值等形式来表达。

“顶层需求”处于图 3-3 的最顶层,由图 3-2 中的“企业/组织战略”转换而来。“顶层需求”为下一层“业务需求”的内容给出了方向和目标,但它并不描述组织具体需要哪些业务以及业务如何处理。这些任务交给业务需求去描述,业务需求是为了落实目标而提出的。

例如:在本书的“智慧社区养老服务系统”案例中,顶层需求包括如下具体内容:

(1) 减少社区养老服务中心运营所需的人力与设备费用;

(2) 改进社区养老服务管理水平;

(3) 为主管社区养老服务的政府部门构建养老服务体系、决策社区养老服务相关政策、指导社区养老工作提供信息化支撑;

(4) 项目实施后,能够大大增加享受相关养老服务的社区居家老年人群体覆盖面;

(5) 居家养老的老年人群体对于社区提供的养老服务体验更好,得到更全面、更方便、更快捷、更高效的老年人服务。

从上述内容可以看到:“智慧社区养老服务系统”顶层需求的描述是从项目建设方——民政部门的角度出发,阐述了系统投入运行后,能够为民政部门的养老服务工作所带来的价值和正面影响,定性地提出对系统的期望和总体目标。

2) 业务需求

业务需求的提出人通常是项目实施方组织中的中层管理者和系统直接用户。中层管理者从落实顶层需求的角度出发,提出业务层面的需求;系统直接用户从实际工作出发,提出希望系统可以解决的具体业务问题或实现的业务目标。

将图 3-2 中的“业务需求”保留,并将“利益攸关者需求”转换为“干系人需求”,合并到业务需求这一层中(干系人和利益攸关者是同一个英文单词 stakeholder 的不同翻译,为保持与原图一致,图 3-2 继续使用了 INCOSE《系统工程手册》中的“利益攸关者”,在本书中统称为“干系人”),形成图 3-3 中的“业务需求”。

“业务需求”处于图 3-3 的中间层。向上看,业务需求的实现能够支撑组织高层所提出的愿景、理想等顶层需求;向下看,业务需求提出了软件系统运行所应支持的业务。

业务需求描述的是客户在具体的业务场景下应完成的任务,因此有时业务需求也被称为“用户需求”。业务需求应使用客户能够理解的语言进行表达。需求分析工程师应在充分理解业务需求的基础上,正确运用业务术语来描述业务需求。业务需求是签订合同和验收系统的基础,未来客户验收的不是系统实现了多少功能模块,而是客户提出的业务需求是否被满足。

3) 系统需求

本书中的系统是指软件系统,系统需求就是指软件需求。软件需求是指用户对系统在功能、行为、性能、设计约束等方面的期望,是系统或系统部件要满足合同、标准、规范或其他正式文档规定所需具有的条件或能力。系统需求是系统的直接用户从实际工作出发向系统提出的需求。

将图 3-2 中的“系统需求”保留,并将“系统元素需求”合并到“系统需求”这一层中,形成图 3-3 中的“系统需求”。

“系统需求”处于图 3-3 的最下层。系统需求提出了软件系统运行应实现的功能,实现了这些功能就可以满足上一层提出的业务需求。需求分析工程师为落实业务需求,从用户的角

度出发，对业务需求进行分析、提取、归纳和整理，最终获取系统层面的需求。在从业务需求到系统需求的转换过程中，来自软件开发组织的需求分析工程师需要得到来自项目实施单位业务人员的支持和帮助，深入全面地开展需求调研、理解业务需求，并从技术的视角分析和解决业务问题，最终从业务需求推导出帮助用户完成业务应满足的系统需求。

2. 按需求性质分类

1）功能需求

功能需求描述了系统应具备的、开发人员必须实现的软件功能。用户利用这些功能完成任务，满足业务需求。

功能需求描述了用户在使用某个功能时会与系统发生一系列的交互。例如："无论何时何地，系统都应该为职员提供请假申请功能"，这是一个常见的功能需求表述示例。在现代软件需求理论中，更加强调从用户的角度出发来描述需求，通常使用"用例""用户故事"等技术来描述功能实现过程中的交互。本书将重点介绍用例技术。

2）非功能需求

除了功能需求之外，对系统提出的其他需求都是非功能需求。非功能需求的存在会影响系统是否能够提供持续、稳定、安全和高效的服务。

非功能需求包括两方面：质量需求和设计约束。质量需求可以进一步分为性能需求、易用性需求、可维护性需求、安全性需求、可靠性需求等。设计约束主要包括技术选择的限制条件(非技术因素决定的技术选型)和运行环境(预期的软、硬件环境)。

(1) 性能需求：从响应时间、吞吐量、资源利用率和精度等方面提出需求。常见表述示例为：页面间跳转时间≤2s；精准搜索反馈结果时间≤1s；模糊搜索反馈结果时间≤3s；当加载数据量大，等待时间超过 3s 时，需要提供加载进度条及预计加载时间；系统需要支持每秒 300 个事务数(300TPS)；支持并发用户数为 500；定位精度误差不超过 50m。

(2) 易用性需求：从易学习性、易操作性、用户错误防御机制和用户界面美观等方面提出需求。常见表述示例为：在系统投入运行后的 1 个月内，90％的用户应该可以独立完成生成财务报销单的任务；用户在阅读了系统使用说明手册后，可以独立完成 95％的功能；对用户输入的内容应进行内容、长度限制等类型检测。

(3) 可维护性需求：从维护响应时间、易追踪性等方面提出需求。常见表述示例为：接到普通修改请求后，90％的修改时间不超过 1 个工作日，最长不超过 2 个工作日；评估后为重大需求或设计修改的应在 1 周内完成；应提供日志记录系统；可追踪系统的使用操作情况。

(4) 安全性需求：从保密性、防泄露、权限控制和防攻击等方面提出需求。常见表述示例为：严格控制访问权限，用户在经过身份认证后，只能访问其权限范围内的数据，只能进行其权限范围内的操作；保护数据不被非法/越权访问和篡改，要确保数据的机密性和完整性；提供运行日志管理及安全审计功能，可追踪系统的历史使用情况；能经受来自互联网的一般性恶意攻击，包括病毒攻击、口令猜测攻击和黑客入侵等。

(5) 可靠性：从易恢复性、容错性和成熟性等方面提出需求。常见表述示例为：系统应能正确处理运行过程中出现的各种异常情况，如：人为操作错误、输入非法数据、网络不稳定等，不影响用户的操作及数据完整；要求系统 7×24h 运行，全年持续运行故障停运时间累计不能超过 10h；每 1000h 最多发生 1 次故障等。

(6) 技术选择的限制条件(非技术因素决定的技术选型)：对于软件开发而言，有些技术选型不是由技术团队决定的，而是会受到项目实施单位的影响。常见表述示例为：数据库管理系统必须采用具有自主知识产权的产品等。

(7) 运行环境(预期的软、硬件环境)：运行环境的选择会受到客户实际的软硬件环境的制约。在整理需求时，应将预期的软硬件环境描述出来，以选择最佳技术方案。常见表述示例为：项目实施组织为村一级政府部门，网络带宽不超过 100M，大部分工作计算机内存不超过 2GB 等。

3.1.3 需求工程

软件需求工程包括创建和维护软件需求文档所必需的一切活动，软件需求工程的主要工作内容包括需求开发和需求管理两部分，如图 3-4 所示。

在图 3-4 中，需求开发是主线、是目标；需求管理是支持、是保障，这两方面的工作相辅相成。

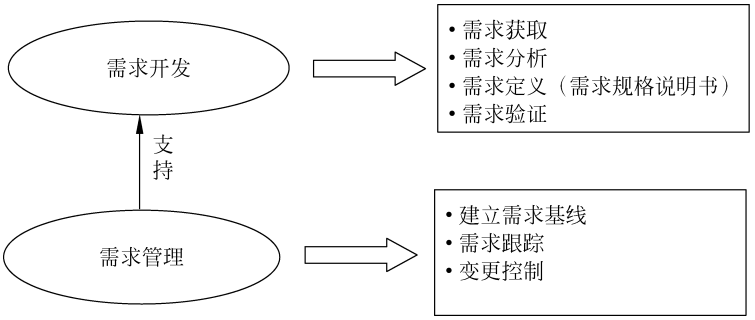


图 3-4 软件需求工程的主要工作内容

1. 需求开发

需求开发是由需求分析工程师与用户接触、交流，并对市场需求进行分析的一系列活动。需求开发通常包括需求获取、需求分析、需求定义(编写需求规格说明书)和需求验证 4 个阶段。

1) 需求获取

需求获取是从人、文档或者环境中获取需求的过程。

需求获取阶段的任务包括：收集要开发系统的背景资料；定义项目前景和范围；选择信息的来源(包括用户、表单、报表、相关的文档和领域专家等)；选择需求获取方法(包括面谈、调查表、观察等方法)；记录需求获取结果(包括文字、图、表、音频、视频等媒介)。

2) 需求分析

需求分析是根据需求获取阶段的成果，确定问题的边界，定义一个需求集合。利用建模和分析技术对需求集合进行整理汇总，建立系统的需求模型，然后根据需求模型将业务需求转化为系统需求。

需求分析阶段的任务包括：对开发系统的背景资料进行分析；确定系统的边界；通过数据流图、E-R 图、用例图、序列图、类图等建模工具进行需求描述，完成需求建模；确定不同系统需求的优先级；对不同用户间的需求冲突进行协商。

3) 需求定义

需求定义是指明确和描述系统开发过程中所需的功能、性能、接口、约束条件以及其他相关要求的过程。通常，需求定义以需求规格说明书文档的形式被确定下来。

4) 需求验证

需求验证阶段的任务是确保需求规格说明书文档能准确地反映用户的意图。通过需求验证发现需求规格说明书中的问题并及时修正。

2. 需求管理

需求管理是在需求开发工作结束后，为保证后续的开发工作是以软件需求规格说明书文档中的内容为标准和目标所做的工作。需求管理通常包括建立需求基线、需求跟踪和变更控制等。

1）建立需求基线

在 RUP 中，需求基线的定义为：逐项列举的在应用程序的某个特定版本中提交的一组需求的集合。特定版本意味着在整个软件的开发过程中将出现多个不同的版本；一组需求集合是需求的一个子集，可以理解为纳入基线的需求是将投入开发的需求，而其他的则是已提出但还没有被接受或没有着手开发的需求。

2）需求跟踪

为了避免在开发过程中发生与需求基线不一致或者偏离的风险，控制开发的质量、成本和时间，人们提出了需求跟踪的方法。需求跟踪能够在需求变化中协调系统的演化，保持各项开发工作对需求的一致性。

3）变更控制

当需求基线确定之后，添加新的需求或修改原有的需求都必须通过需求变更流程来操作，同时记录变更情况、变更日期、变更原因等。需求变更管理的目标是控制变更，而非避免变更。变更控制的目标是减少变更对开发工作的影响。

3.2　业务建模与 UML 概述

3.2.1　业务建模

业务需求分析在可行性研究阶段就已经开始了：通过分析业务问题找出可能的解决方案，对方案进行成本/效益分析等。但在可行性研究阶段，只是初步地、笼统地进行了业务分析，当可行性研究评审通过，项目正式立项后，将展开详细的业务分析工作。

在业务分析过程中，需要来自软件开发组织的需求分析工程师（本书不区分业务需求分析工程师和系统需求分析工程师，统称为需求分析工程师）和来自项目提出方的业务专家两方面人员一起合作完成业务分析阶段的任务。业务专家从业务视角来分析业务流程、探讨如何解决业务问题；需求分析工程师需要学习像业务专家一样思考、理解组织目标，并从技术视角分析问题、提出解决方案，从而支持组织顶层目标的实现。

1. 系统建模及模型

系统建模是建立系统抽象模型的过程，每个模型从不同的视角表示系统。在业务需求分析阶段建立业务模型，是为了帮助人们理解业务流程；在系统需求分析阶段建立系统需求模型，是为了得到明确的系统需求；在系统设计阶段建立设计模型，是为了描述系统如何实现；在系统运行阶段建立运行模型，是为了描述系统的部署和运行结构。

在绘制系统模型时，可以灵活运用图形化建模表示方法，目的是使模型更容易理解。在软件生命周期的每个阶段，图形化模型都可以作为一种讨论问题的方式推动软件开发进程，并且在达成共识后，以文档的形式将模型确定下来，前提是应当正确地使用相应的建模表示法，并且确保模型是对系统的准确描述。

2. 业务模型的概念及作用

在业务分析的过程中，一个重要的任务是把在需求调研阶段以各种形式获取的、各种不同

呈现类型的资料进行抽象和简化,这个抽象的过程称为建立业务模型,简称业务建模。抽象后得到的以文字、图、表等形式来呈现业务需求的结果称为业务模型。业务需求是为了实现顶层需求而提出的,业务建模的目的是从项目提出方的角度明确系统应该提供的价值。

业务模型的作用主要体现在以下三方面。

1) 厘清现行业务

厘清现行业务的现状:参与者、干系人、主要解决的业务问题和业务流程等。业务模型是对现行业务的简化和抽象表述。通过业务模型,人们可以直观地把握和理解现行业务,业务模型对理解复杂的业务领域更有帮助。

2) 业务流程再造

不断改进业务流程是组织提高管理水平、增加竞争力的有效途径。通过对业务模型的分析有助于找到业务流程中需要优化的业务环节,改进现行业务流程,从而实现组织提出的愿景和期望。

3) 建立系统需求分析模型的基础

业务模型是系统需求分析阶段的重要输入,是建立系统需求分析模型的基础。

深入思考 3.2 需求是技术无关的,那么在业务建模时是否可以完全不考虑技术?

企业视角:理论上讲,在业务建模阶段,不但要保证需求的技术无关性,还要保证需求不要深入细节。因为在这个阶段,最重要的事情就是要了解业务的全貌,深入细节会舍本逐末。但在企业实践中,很难避免在业务建模阶段讨论技术。如果客户组织已有一个现行系统,就难以避免讨论新旧系统的兼容或者重新规划等问题。

详情请参见微课视频 3-2。

3. 业务建模类型

描述一个业务需要三个基本要素:业务主体、流程和数据。业务主体是指客户所属组织机构的部门或个人,也可以是其他软硬件系统;流程是指为达到特定的价值目标,由不同的业务主体合作完成的一系列活动;数据是指执行活动所需要的或在执行活动过程中产生的信息。业务需求分析过程将围绕这三个要素,从不同角度对业务需求进行业务建模。

1) 业务场景建模(从业务主体的角度看)

业务场景建模作为一种需求分析技术,用途十分广泛。什么是业务场景?业务场景就是企业和商家需要在某个特定的场景中,适时提供给消费者可能需要的以及关联的产品或服务。

业务场景建模成果:针对场景所要达成的业务目标,基于生态圈、价值链分析和业务部门需求进行需求调研,识别业务场景参与者;结合各参与者需求,分析业务目标和事件,识别业务场景,绘制业务场景模型,形成业务场景列表。

2) 业务流程建模(从流程的角度看)

业务流程建模的目的是帮助人们理解业务执行过程,了解部门职责及部门之间的业务衔接。

业务流程建模成果:针对某一个业务,识别所有业务活动并描述这些业务活动之间的关系,了解这些业务活动执行所需的信息、执行完毕后产生哪些数据,同时标识出这些业务活动所属的职责部门或岗位,绘制业务流程图。

3) 业务领域建模(从数据的角度看)

业务领域建模是在业务流程描述的基础上,针对每一个业务过程,识别出业务实体,确定实体之间的关系后生成业务对象模型。

业务领域建模成果:以面向对象的视角,建立描述现实世界中各种事物之间关系的业务

类图(或称领域类图)。业务类图的建立将为在系统需求分析阶段构建分析类图打下基础。

4. 业务建模语言

目前,业务建模领域还没有出现大家普遍接受的业务建模语言,下面介绍两种影响比较大的业务建模语言。

(1) 业务流程建模符号:业务流程建模符号(Business Process Modeling Notation,BPMN)是由非营利组织——业务过程管理组织(Business Process Management Initiative,BPMI)开发的一套标准。BPMI 于 2004 年 5 月对外发布了 BPMN 1.0 规范。后来 BPMI 并入到对象管理组织(Object Management Group,OMG),OMG 于 2011 年推出 BPMN 2.0 标准,对 BPMN 进行了重新定义。BPMN 的主要目标是提供一些被所有业务用户容易理解的符号,从创建流程轮廓的业务分析到这些流程的实现,直到最终用户的管理监控。

(2) UML 业务建模:统一建模语言(Unified Modeling Language,UML)是一种定义良好、易于表达、功能强大且普遍适用的可视化建模语言。1997 年,UML 被 OMG 采纳成为工业标准,UML 的出现有力推动了建模技术在软件业的应用和推广。UML 提供了一套标准的制图技术和模型符号,它不仅包含软件建模的功能,还包含了业务建模、流程建模等多种建模功能。目前 80%的建模工具是在 UML 基础上加以变化或扩展形成的。

5. 业务建模工具

在业务建模过程中,选择的工具应支持上述建模语言或建模符号。主流的业务建模工具包括 Microsoft Visio、Enterprise Architect(EA)、Rational Rose 等。

本书案例从业务需求分析开始,采用 UML 作为建模语言,选用 Microsoft Visio 作为建模工具,介绍软件生命周期各阶段的建模成果。

3.2.2　UML 概述

从 20 世纪 70 年代开始,不断地有面向对象的建模语言面世,但新的问题也随之而来——没有统一的标准。面向对象的分析与设计方法(Object Oriented Analysis Design,OOAD)在 20 世纪 80 年代末至 90 年代中期得到了充分的发展,UML 就是这个发展过程中的产物。

1. UML 发展历史

- 1991 年,由 James Rumbaugh 等 5 人提出了面向对象的建模技术(Object Modeling Technology,OMT)方法,采用面向对象的概念,引入各种独立于语言的表示符号,所定义的概念和符号可用于软件开发的分析、设计和实现的全过程。该方法是一种面向对象的开发方法,开发工作的基础是对真实世界的对象建模,然后围绕这些对象使用分析模型来进行独立于语言的设计,面向对象的分析建模促进了对需求的理解,有利于开发更清晰、更容易维护的软件系统。
- 1994 年,Jacobson 提出了面向对象的软件工程方法 OOSE,它是一种在 OMT 的基础上用于对功能模型进行补充指导的系统方法。其最大特点是面向用例,并在用例的描述中引入了外部角色的概念。
- 1994 年 10 月,Grady Booch 和 James Rumbaugh 首先将 Booch 93 和 OMT-2 统一,并于 1995 年 10 月发布了第一个公开版本 UML 0.8。
- 1996 年 6 月和 10 月,在 Booch、Rumbaugh 和 Jacobson 三人的共同努力下,发布了两个新的版本:UML 0.9 和 UML 0.91,并将 UM 重新命名为 UML。
- 1996 年,DEC、HP、IBM、Microsoft、Oracle、Rational Software 等公司和组织成立了 UML 成员协会,职责是完善、加强和促进 UML 的定义工作。

- 1997 年 1 月,UML 1.0 版本发布。
- 1998 年,UML 1.2 版本发布。
- 1999 年,UML 1.3 版本发布。
- 2003 年,UML 1.5 版本发布。
- 2005 年 7 月,UML 2.0 版本正式发布,这是对 UML 的一次重大修订。
- 2011 年,UML 2.4 版本发布。
- 2012 年,UML 成为国际标准化组织(International Organization for Standardization,ISO)规定的国际标准规范。
- 2015 年,UML 2.5 版本发布。

2. UML 的特点

UML 称为统一建模语言,它的命名本身就体现了它的 3 个特点。

1) 统一(Unified)

统一表示 UML 是一种通用的标准,它被 OMG 认可,为软件工业界的一种标准。UML 表述的内容能被各类人员所理解,包括客户、领域专家、分析师、设计师、程序员、测试工程师及培训人员等。UML 为不同类型的人员进行充分沟通提供了一个统一的平台。

2) 建模(Modeling)

建模表示 UML 不是一种编程语言,而是从不同角度对系统进行描述。它可以用于业务建模、流程建模、数据库建模等多种应用领域。模型本质上是为参与系统开发的各类人员提供一种各方都能理解的语言,使他们能够基于此来理解业务、需求以及系统的体系架构。

3) 语言(Language)

语言表明 UML 是一种描述规范,而不是一种方法论或开发框架。UML 使用一套按照特定规则和模式组成的符号系统,包括图形符号、自然语言和形式语言相结合的方法来描述目标,为不同工作领域的人们沟通时提供标准化的建模语言,以便他们开发和交换有意义的模型。

3. UML 与 RUP

UML 是建模语言,能够用它的表示和规则为系统进行面向对象的建模,但它仅仅是一种系统建模语言,并没有告诉建模人员应该如何使用它。为了使用 UML,就需要软件过程来指导。

UML 独立于特定的编程语言和开发过程,可以将它运用于许多软件过程。由于 UML 是伴随面向对象技术产生的,在第 1 章中提到的 RUP 正是目前影响较大的、面向对象的软件开发过程。因此,可以说 RUP 是一种特别适应于 UML 的生命周期方法。RUP 提出了一整套以 UML 为基础的开发准则,它为软件开发人员有效地使用 UML 提供了指导。

4. UML 的图

本书不过多介绍 UML 中的元素和规则,主要结合软件过程介绍 UML 各种图的用法。UML 2.0 标准一共定义了 14 种图,分为结构图和行为图两大类,如图 3-5 所示。

1) 类图

类图描述了系统中对象的类型以及它们之间存在的各种静态关系,是系统静态对象结构的图形描述。

2) 对象图

对象图描述了一组对象及它们之间的关系。对象图是类图的一个实例,是系统在某个时间点的详细状态的快照。和类图一样,对象图给出系统的静态设计视图或静态进程视图。

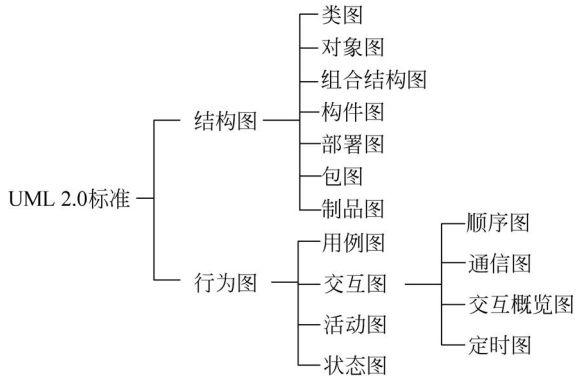


图 3-5　UML 2.0 的 14 种图

3）组合结构图

组合结构图描述了一个组合结构的内部结构以及它们之间的关系。组合结构图用于画出组合结构的内部内容。它用来表示系统中逻辑上的“组合结构”。

4）构件图

构件图展现了一组构件之间的组织和依赖。它与类图相关，通常把构件映射为一个或多个类、接口、协作。

5）部署图

部署图展现了运行处理结点以及其中的构件配置。部署图给出了体系结构的静态实施图。它与构件图相关，通常一个结点包含一个或者多个构件。

6）包图

包图描述了模型的组织。包可以把所建立的各种模型组织起来，形成各种功能或用途的模块，并可以控制包中元素的可见性，以及描述包之间的依赖关系。

7）制品图

制品图用于面向对象系统的物理建模。制品图展示了一组制品之间的组织及其依赖关系。利用制品图可以对存在于结点上的物理事物进行建模，物理事物是指可执行程序、库、表、文件和文档等。制品图实质上是针对系统制品的类图。制品图通常与部署图一起使用。

8）用例图

用例图是指由参与者、用例、边界以及它们之间的关系构成的用于描述系统功能的视图。它用来描述边界内部的系统与边界外部的用户（或其他系统）之间的交互视图，强调的是系统能够做什么，而不是系统如何工作。

9）顺序图

顺序图也可称为序列图，是一种交互图。顺序图展示了在用例的特定场景中，对象如何与其他对象交互。通过描述对象之间发送消息的时间顺序展示对象如何进行动态协作。

10）通信图

通信图也是一种交互图，它强调收发消息的对象或参与者的组织关系。顺序图和通信图表达了类似的基本概念，但它们所强调的概念不同。顺序图强调的是时序，通信图强调的是对象之间的合作关系而不是时间顺序。在 UML 1.X 版本中，通信图被称为协作图。

11）交互概览图

交互概览图是 UML 2.0 新增的交互图之一，它主要描述交互（特别是关注控制流），但是抽掉了消息和生命线，使用活动图的表示法，可以看作是活动图和顺序图的混合物。

12）定时图

定时图也是一种交互图，它强调消息跨越不同对象或参与者的实际时间，而不仅仅是关心消息的相对顺序。

13）活动图

活动图描述了具体用例的实现流程，说明活动之间的依赖关系。活动图对系统的功能建模和业务流程建模特别重要。

14）状态图

状态图是对一个单独对象的行为建模，它用来表示指定对象在整个生命周期响应不同事件的不同状态，强调事件导致的对象状态的改变。状态图对于接口、类或协作的行为建模尤为重要。

3.3 业务场景建模

业务场景建模通常是从用户的角度来理解业务，一个业务场景可以通过以下 3 个要素来进行表述：

- 谁：识别参与者，用人或者系统描述。
- 在什么环境下：识别上下文，用时间、空间和状态描述。
- 干什么和遇到什么问题：识别完成的事情，用活动序列描述。

常见的场景建模技术包括上下文图、用例图、用户故事等。

1. 上下文图

上下文图是一个分析模型，通常在项目的早期使用，以确定调查的范围。特点：以系统为中心，被环境中的所有外部实体和交互系统包围，没有系统内部结构的细节。

2. 用例图

用例图是指由参与者、用例、边界以及它们之间的关系构成的用于描述系统功能的视图。特点：简洁、直观、站在用户的角度来描述系统和外部参与者之间的关系。

3. 用户故事

用户故事在软件开发过程中被作为描述需求的一种表达形式。为了规范用户故事的表达，便于沟通，用户故事通常的表达格式为：作为一个〈用户角色〉，我想要〈完成活动〉，以便于〈实现价值〉。

本书在业务需求分析阶段选择上下文图作为业务场景建模工具，在系统需求分析阶段选择用例图作为系统功能建模工具。

3.3.1 上下文图

根据范围界定的不同，上下文图分为业务上下文图和系统上下文图。

在业务需求分析阶段，业务上下文图以组织为边界，以业务为黑盒子，标识出参与业务的组织内部参与者和外部参与者及其发起的业务事件；在系统需求分析阶段，系统上下文图以系统为边界和黑盒子，标识出与系统进行交互的参与者及其发起的业务事件。

1. 业务上下文图

以茉莉餐厅的核心业务“就餐”业务为例，其业务上下文图如图 3-6 所示。

在图 3-6 中，外部的大矩形盒子代表“组织边界”，在本例中为茉莉餐厅。内部的小矩形盒子代表“业务边界”，在本例中为就餐业务。业务上下文图的建模步骤如下：

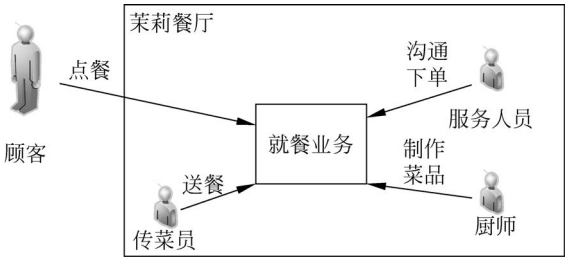


图 3-6　业务上下文图示例

(1) 标识组织外部参与者。顾客是“茉莉餐厅”的外部参与者。

(2) 明确组织外部参与者发起的业务事件。由顾客发起“点餐”业务事件。

(3) 寻找业务相关的内部参与者。餐厅为了处理“就餐业务”,需要餐厅内部的服务人员、厨师和传菜员协同配合,各自执行负责的业务活动后,才能完成就餐业务。

在图 3-6 中,完成“就餐”业务需要四个参与者:顾客、服务人员、厨师和传菜员,业务上下文图揭示了组织内部和外部参与业务交互的参与者。需要特别说明的是,在下一步使用泳道图绘制业务流程图时,业务上下文图中的各个参与者将对应泳道图中的各个泳道。

深入思考 3.3　在图 3-6 中,厨师在就餐业务过程中发起的“制作菜品”事件是明显的人工环节,它和待开发的餐厅软件系统似乎无关,将这个参与者列出来是否多余?

参考答案:在业务上下文图中列出所有的业务事件是业务需求分析的一个重要线索,业务事件是响应外部用户请求、实现组织价值的核心途径,因此也是组织进行业务流程再造的重要切入点。在业务需求调研的访谈过程中,向每个参与者提问的内容可能是:“你现在面临的问题是什么?你在什么业务中遇到了这个问题?如果这个问题解决了,将会为组织带来什么好处?”以厨师为例,厨师的业务事件是“制作菜品”,在调研访谈时,厨师可能会提到:“现在顾客下的单子虽然是打印出来按顺序放置的,但有时候不小心单子的顺序乱了,会导致早来的顾客等待时间比晚来的顾客还长,经常会被投诉,这是我遇到的最大问题”,那么针对这个问题,在业务流程再造时,就可以考虑为后厨安装一套“后厨餐饮管理系统”,将整个下单和出菜流程信息电子化。

因此,在绘制业务上下文图时,不要省略那些目前看来是人工完成的环节和业务事件,随着技术的进步和组织管理思维的转变,每个业务环节都有可能被优化。

详情请参见微课视频 3-3。

2. 系统上下文图

系统上下文图定义了所开发的系统和系统外部实体(如使用人员、硬件设备和其他软件系统)之间的边界和接口。系统上下文图的目标是通过明确系统相关的外部因素和事件,促进更完整地识别系统需求和约束。

例如,经过分析评判,认为茉莉餐厅目前存在的问题是:点餐环节花费的人力较多,顾客体验不够良好;后厨的出菜顺序和菜品数量管理混乱。拟通过运行智能餐厨系统来改善茉莉餐厅现状。“智能餐厨系统”上下文图示例如图 3-7 所示。

3.3.2　案例的业务场景建模

本节案例围绕社区养老服务的实际工作,从各干系人的角度出发,确定各自的业务目标,按照业务上下文图建模的基本步骤完成业务场景建模。

1. 识别组织边界

组织边界一般就是业务运行所属的组织。本案例的核心业务是养老服务,所属组织为养

老服务的直接管理组织——社区。所以,“社区”就是本案例的业务组织边界。

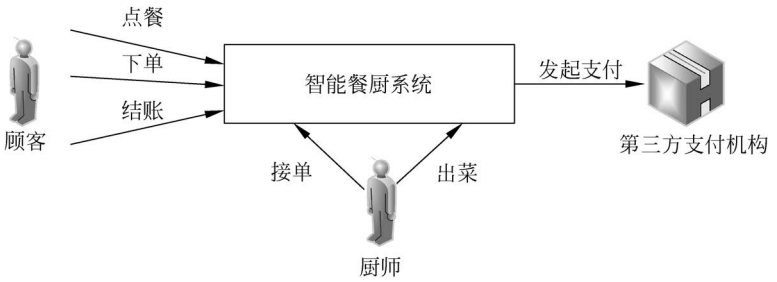


图 3-7 “智能餐厨系统”上下文图示例

2. 寻找外部参与者参与的核心业务事件,确定业务列表

由于一个组织的价值是通过组织作用于外部用户产生的,所以首先寻找组织的主要外部用户,由他来激发组织内部的响应,也就是首先确认外部参与者。在本案例中,外部参与者为老年人/老年人亲属、养老服务提供商、接单人员、民政局养老负责人。然后确定每一个外部参与者与业务交互的主要目的,并从中抽取业务。业务抽取过程如下所述。

(1) 老年人/老年人亲属希望社区提供相关的养老服务,如果服务出现问题,可以找社区进行投诉;养老服务提供商希望社区能公平、公正地对老年服务订单进行派单,同时自身也能向接单人员公平、公正地派单;接单人员希望能准确、方便地完成接单工作,能获得准确的老年人相关信息并完成客户的订单支付确认工作。据此抽象得到“养老服务订单处理”和“养老服务投诉处理”业务。

(2) 民政局养老负责人希望社区能定期提供其养老服务的工作开展情况,以对社区工作进行评估;民政局养老负责人会定期为满足条件的老年人发放养老券,用以支付某些养老服务。据此抽象得到“养老服务评估”和“养老券发放”业务。

根据上述分析,获取社区在养老服务方面的业务列表是:养老服务订单处理、养老服务投诉处理、养老服务评估、养老券发放。

3. 绘制业务上下文图

下面以“养老服务订单处理”业务为例,绘制其业务上下文图,如图 3-8 所示。

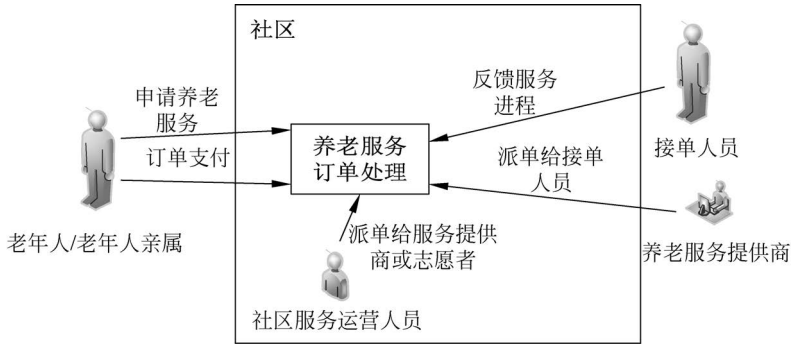


图 3-8 “养老服务订单处理”业务上下文图

在图 3-8 中,完成“养老服务订单处理”业务需要四个参与者。其中,三个外部参与者为老年人/老年人亲属、养老服务提供商、接单人员;一个内部参与者为社区服务运营人员。老年人/老年人亲属发起的业务事件为“申请养老服务”和“订单支付”;接单人员发起的业务事件为“反馈服务进程”;养老服务提供商发起的业务事件为“派单给接单人员”;社区服务运营人

员发起的业务事件为“派单给服务提供商或志愿者”。

3.4　业务流程建模

业务场景建模通过业务上下文图确定了业务的内外部参与者及其发起的业务事件，那么对于业务流程如何运转，就需要将业务展开进行业务流程分析。对于流程简单的业务，可以通过文本进行描述；对于流程较复杂的业务，一般借助流程图来完成业务流程分析。

3.4.1　流程图模型

适用于描述流程的模型有很多，下面重点介绍三种常见的流程图：系统流程图、跨职能流程图、UML 活动图。

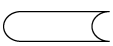
1. 系统流程图

系统流程图是概括描绘系统物理模型的传统工具，它的基本思想是用图形符号以黑盒子形式描绘系统里面的每个具体部件（程序、文件、数据库、表格、人工过程等），并表达数据在系统各个部件之间流动的情况。系统流程图的绘制通常采用自上而下、自左向右的顺序原则。

系统流程图中的某些符号与程序流程图中的符号形式相同，但是系统流程图与程序流程图是不同的。程序流程图表达了对信息进行加工处理的控制过程，重点在控制流；系统流程图表达了信息在系统各部件之间的流动情况，重点在数据流。

系统流程图的常用基本符号及含义如表 3-1 所示。

表 3-1　系统流程图常用基本符号

符　号	名　称	说　明
	处理	表示能引起数据改变的处理功能
	输入输出	表示输入或输出
	判断	表示判断或开关类型功能
	联机存储	表示可以通过输入输出设备访问的数据库中的数据存储
	文档	表示打印输出数据，也可表示打印端输入数据
	数据流	表示数据流动方向
	数据库	表示存储数据的数据库
	显示	显示终端或部件，用于输入输出
	人工输入	人工输入的脱机处理
	人工操作	人工完成的处理
	既定处理	表示已经命名好的处理
	通信链路	通过远程通信线路或链路传送数据

下面以“点外卖”业务流程为例，绘制系统流程图，如图 3-9 所示。

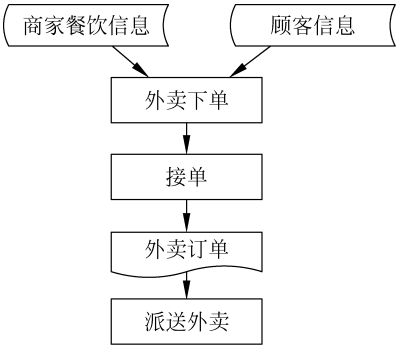


图 3-9 “点外卖”业务系统流程图示例

在图 3-9 中,系统首先获取商家餐饮信息和顾客信息,在两方面信息都具备的情况下可以完成外卖下单,然后商家接单生成外卖订单,最后将外卖派送到顾客手中。

系统流程图有两个缺点:

(1) 系统流程图虽然重点在于表达数据在系统各部件之间的流动,但是在绘制时,数据和加工是混合在一起的,数据流和控制流没有明确的区分。

(2) 系统流程图中的处理只有动作,无法表现执行动作的业务实体,使得业务流程无法清楚表述负责该业务的职能部门。

2. 跨职能流程图

跨职能流程图可以解决系统流程图表达上的缺陷。跨职能流程图是商业建模领域的标准工具。一个组织的业务流转通常需要各个职能部门的通力协作,很多业务流程不仅涉及组织内部的多个部门,甚至需要组织外部的机构配合完成。在这样的情形下,使用跨职能流程图(也称泳道图)定义各泳道的职责,能够使流程更加清晰。每条泳道代表一个人、部门或系统,每个泳道中的活动代表所属泳道的职责。因此,在业务需求分析过程中,业务流程图选用跨职能流程图是一个较好的选择。

下面以工厂生产的采购流程为例,绘制采购业务的跨职能流程图,如图 3-10 所示。由此案例初步认识跨职能流程图中的各种基本元素。

工厂生产通常的采购基本流程为:采购人员根据需求部门所提出的采购需求制定采购计划,采购计划需要满足公司生产计划所需。然后邀请一些潜在的供应商开始招标,供应商参加招标会竞标后确定供应商。选定供应商后,准备合同的签订(确定付款条件、货运方式、售后服务等),合同需要需求部门、法务部门、财务部门审查确认后才能够正式签订,生成采购订单。之后供应商发货,工厂收到货物后由质检部门验收,验收合格后进行财务结算付款。

在图 3-10 中,跨职能流程图特有的框架元素如下所述:

1) 泳道

共有 6 个部门参与完成采购业务流程。6 个部门分别对应 6 个泳道,包括采购部、需求部门、法务部门、财务部门、质检部门和供应商。每个部门负责的业务在各自的泳道中描述,职责清晰。

2) 任务阶段

采购业务分为两个阶段:签订合同前和签订合同后。将不同阶段要执行的业务活动划分清楚。

跨职能流程图的常用基本符号及含义如表 3-2 所示。

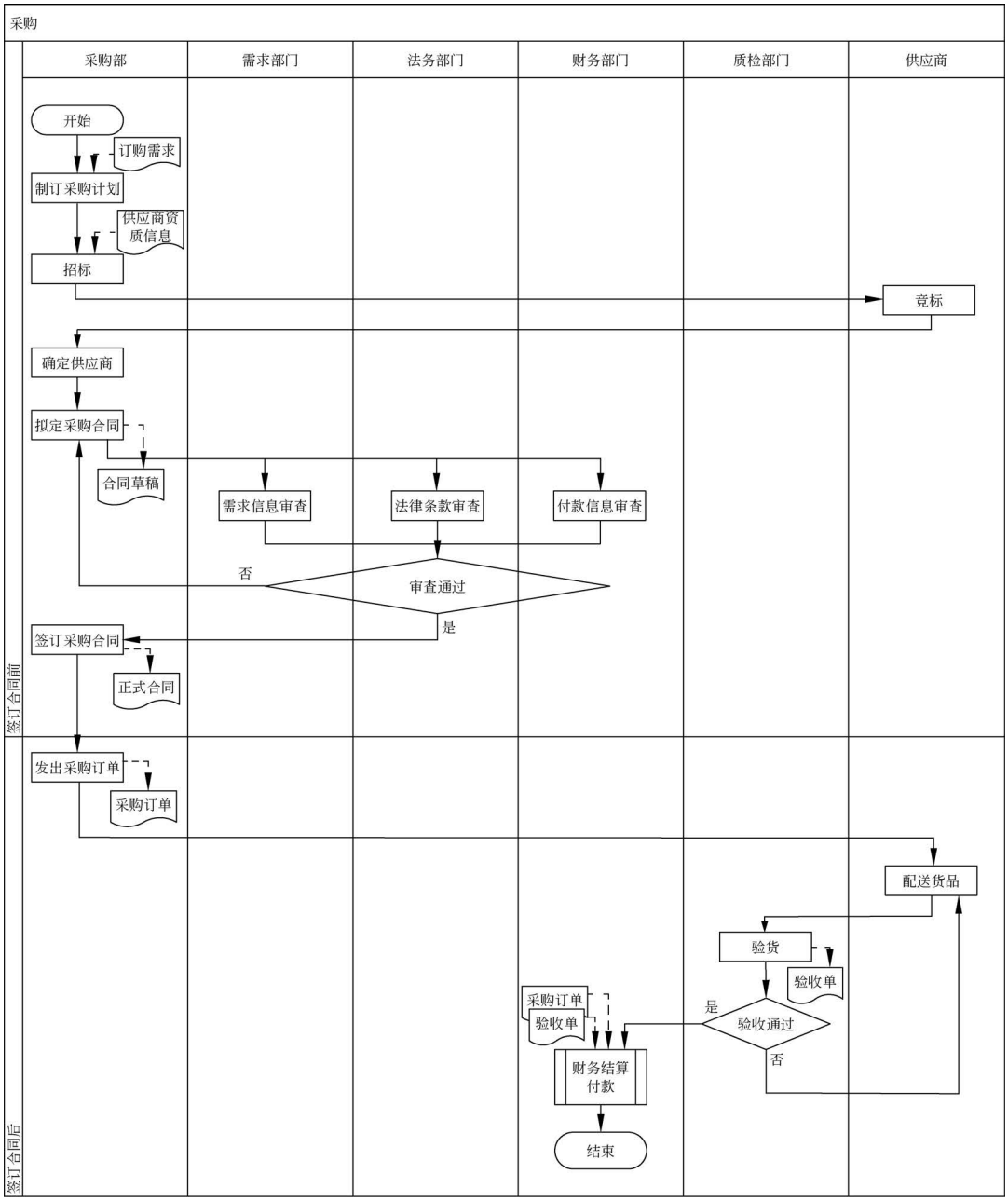


图 3-10　采购流程的跨职能流程图

表 3-2　跨职能流程图常用基本符号

符　　号	名　　称	说　　明
	开始/结束	只能有一个开始结点，可以有多个结束结点
	判断	表示判断或开关类型功能
	处理	表示能引起数据等改变的单个操作
	文档	表示支持流程运行所需的文件、数据
	处理流	表示处理流程的流动方向

续表

符　号	名　称	说　明
	数据流	表示在处理执行时涉及的数据 箭头指向处理：代表处理所需的数据 箭头从处理向外指出：代表处理所产生的数据
跨职能流程特有的元素		
	泳池	泳池是泳道图的一个外部框架，泳道、流程都包含于泳池里 泳池是随着泳道的拖放自动形成的，无需绘制
	泳道	泳池里可以创建多个泳道 根据泳道摆放的方向不同，分为垂直泳道图和水平泳道图 泳道垂直于画布摆放，称为垂直泳道图 泳道水平于画布摆放，称为水平泳道图
	流程阶段	通过任务阶段来区分，明确各个阶段需要处理的任务环节。如：售前—下单—售后 阶段划分是随着分割线的拖放自动形成的，无需绘制 如果没有放置分割线，则流程图不区分阶段，只有一个阶段
	分割线	表示阶段的划分 水平分割线用于垂直泳道图的阶段划分 垂直分割线用于水平泳道图的阶段划分

图 3-10 中的实线线条代表采购业务所需的各种活动之间的跳转流程，虚线线条代表执行某个业务时所需的数据或产生的数据。不同的标识将活动跳转的控制流和活动执行时的数据流做出了明确的区分。例如：在执行“制订采购计划”活动时需要输入“订购需求”信息，在执行“签订采购合同”后会输出“正式合同”文档。

注意：图 3-10 中的“财务结算付款”使用了系统流程图中的“子流程”符号，代表“采购业务流程”走到财务结算步骤时，在此不展开说明，详情参见“财务结算付款”业务流程。

3. UML 活动图

UML 活动图是 UML 规范中定义的一种图，是 UML 对系统的动态行为建模的常用工具，主要描述活动的顺序，可以用来对业务过程和工作流程建模，与传统的流程图十分相似。活动图具有更多与程序开发相关的语义信息，对于软件开发更具有指导性。

活动图的常用基本符号及含义如表 3-3 所示。

表 3-3　活动图常用基本符号

符　号	名　称	说　明
	开始	只能有一个开始结点
	结束	可以有多个结束结点
	活动	表示一个动作、操作
	控制流	表示活动跳转流程
	分支	可以有一个进入控制流和多个离开控制流 每个离开控制流都有一个判断条件，条件之间必须互斥，满足该条件时就执行它指向的活动
	分岔	并行控制流。分岔有一个进入控制流和多个离开控制流，离开控制流是并发处理

续表

符　号	名　称	说　明
	汇合	并行控制流。汇合有多个进入控制流和一个离开控制流，进入控制流是并发处理
	泳道	泳道将活动图中的活动状态分组，每一组表示一个特定的类、人或部门，它们负责完成组内的活动 每个活动都明确属于一个泳道，不可以跨越泳道

下面以细化后的“点外卖”业务流程为例说明活动图的绘制。

顾客通过外卖平台点外卖的业务流程通常是：顾客浏览餐饮信息，选择好餐食下单，之后选择支付方式准备付款，若在规定时间内付款成功，则商家会接到订单，否则订单会被取消。商家接到订单后，在准备餐食的同时，有外卖小哥接到送外卖的派单并赶往商家，当外卖小哥到达和商家完成餐食打包这两个条件都满足时，即可进入派送途中，直到餐食被送至顾客手中，此次外卖交易结束。基于上述点外卖的业务流程绘制活动图，如图 3-11 所示。

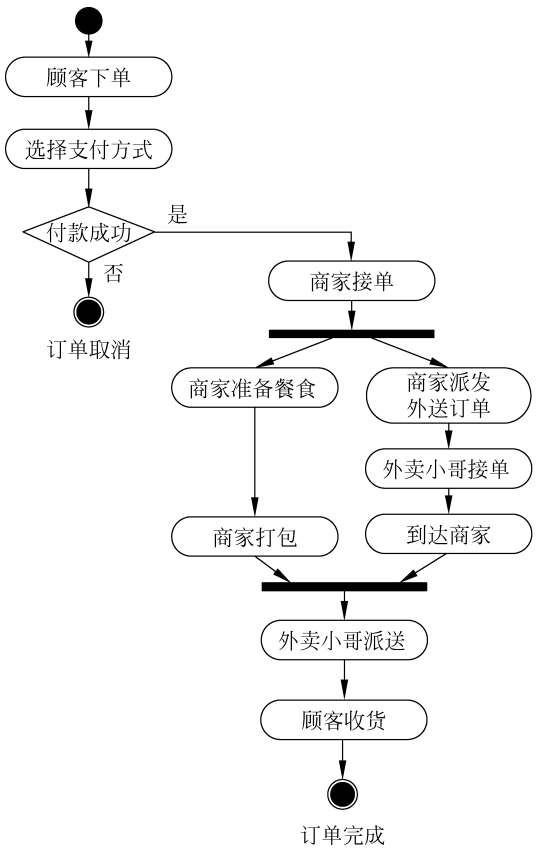


图 3-11　点外卖流程的活动图

在实际的控制流中，除了顺序、分支和循环结构外，还有一种是并发事件流，在 UML 活动图中，使用分岔与汇合来表示事件的并发处理。在图 3-11 中体现了分岔与汇合的用法。在本例中，当商家接单后，系统将并行处理两方面事务：一是餐饮商家根据订单信息开始餐食制作、打包；二是商家派发外送订单，外卖小哥接单后，赶往餐饮商家准备送货。这两类事件是并发处理，当这两个并发处理都完成时，控制流汇合，跳转到“派送”活动中。

图 3-11 所示的活动图为基本活动图，与系统流程图类似，基本活动图与系统流程图一样，同样无法体现每个业务活动是由哪个岗位或角色执行的。因此，在基本活动图的基础上，可以通过添加泳道来明确各个活动由谁负责，变形后的活动图称为“带泳道的活动图”。

对图 3-11 的基本活动图添加泳道后，形成带泳道的活动图，如图 3-12 所示。

点外卖流程中活动所涉及的角色包括：顾客、商家、外卖小哥，对应图 3-12 中的三个泳道。在图 3-12 中，每个活动、分支只能属于一个泳道，而分岔、汇合可以跨泳道。活动图中的箭头体现了活动控制流，泳道体现了负责泳道内活动的角色。

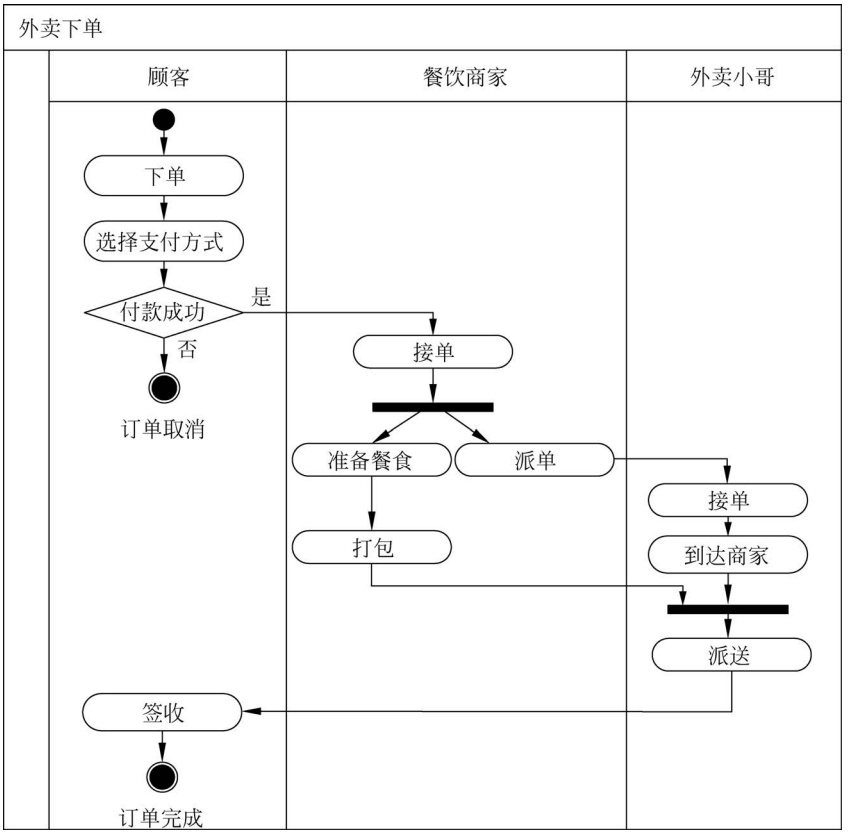


图 3-12　点外卖流程——带泳道的活动图

3.4.2　业务流程图

业务流程是指一组为客户创造价值并相互关联的活动。在业务需求分析过程中，业务流程图只是手段，不是目的。绘制业务流程图的目的是：在业务流程图建模完成后，对模型进行深入分析，帮助组织寻找业务流程中能够提升业务价值的环节加以改进。

本书选用“带泳道的活动图”作为绘制业务流程图的建模工具。

以图 3-6 茉莉餐厅上下文图中的“就餐”业务为例，根据“从上下文图中识别出的各个参与者正是下一步分析业务流程中泳道图对应的各个泳道”原则，获得“就餐”业务流程图的泳道为顾客、服务员、传菜员、厨师。

绘制现行的茉莉餐厅顾客就餐业务流程图，如图 3-13 所示。

通过业务流程图能够揭示出现有业务流程中的一些瓶颈环节，从而获得业务流程改进的启示和可能。改进后的茉莉餐厅顾客就餐业务流程图如图 3-14 所示。

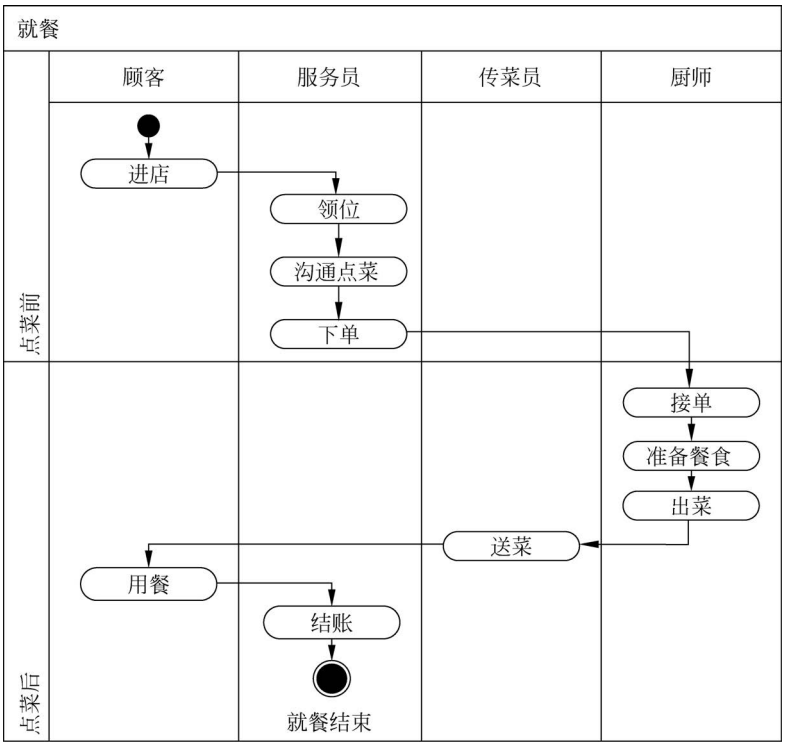


图 3-13　现行的茉莉餐厅顾客就餐业务流程图

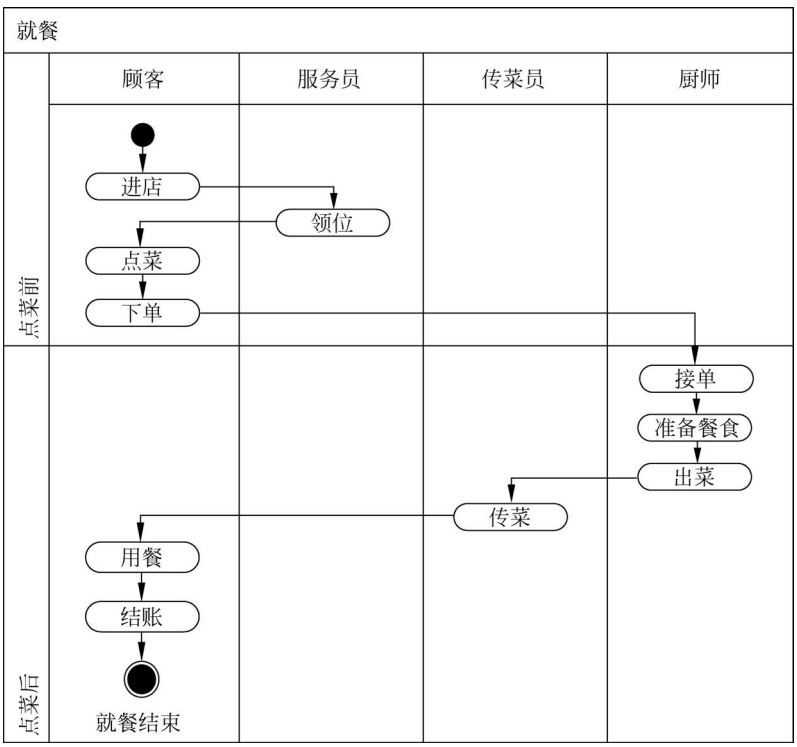


图 3-14　引入“自助点餐系统”后的就餐业务流程图

在图 3-13 的就餐流程中,服务员所在泳道的活动包括:配合顾客点菜下单;在就餐过程中配合顾客加单;当顾客询问菜品目前是否已开始烹饪时,到后厨确认以便答复顾客的问询。整体看就餐业务流程中服务员的工作量比较大,餐厅在此环节需要投入的人力较多。图 3-13 中原本由服务员负责的“点餐”和“付款”的业务活动,在图 3-14 中改进为交由顾客去完成,目前流行的小程序或公众号技术为点餐环节的业务流程改进提供了技术上的可能性。图 3-14 中,顾客也可以通过微信小程序自行查询目前菜品的进展,能够大大减少服务员的工作量,进而降低餐厅的人力成本。

当然,在改进业务流程的同时,也会面临其他相关问题。例如,顾客使用小程序或公众号自助结账,就有可能出现“逃单”的问题,如何避免或减少此类现象发生是餐厅需要考虑的;引入“自助点餐系统”的系统购置费用和人力成本的节约费用相比,需要从经济效益上进行分析,判断是否能够真正节约企业开支,等等。所以,当业务流程图建模完成后,对业务流程提出改进方案时,需要对流程改变带来的影响做预估,才能更好地减少变更带来的不利影响,最大化体现业务流程改进为企业带来的价值和意义。

3.4.3 案例的业务流程建模

本节以“养老服务订单处理”业务为例,进行业务流程建模。

1. 社区现行“养老服务订单处理”业务流程

目前,社区有一个“养老服务中心”为老年人提供各种服务,服务种类主要包括三种服务。

(1) 饮食:养老服务中心开设了怡老餐厅,如果老年人有饮食方面的需求,需要提前电话预约,然后在规定时间内自行出门到餐厅吃饭。

(2) 娱乐:养老服务中心开设了棋牌室,如果老年人有娱乐方面的需求,无需预约,可以自行到棋牌室参与各项活动。

(3) 日常体检:养老服务中心开设了医务室,无需预约,能够进行体重、身高、血压、心率、血糖等方面的日常检查。

社区目前的养老服务种类比较单一,而且对于行动不便、只能居家的老年人基本没有提供实质性的帮助。由于能够自理的老年人到餐厅就餐的人数较少,医疗健康方面的服务不专业使得医务室的开设也形同虚设,社区养老服务中心面临无法维持日常开销的局面,社区没有真正发挥出基层管理部门在养老服务方面中的作用,也无法真正落实国家倡导的居家养老政策。

2. 改进后的“养老服务订单处理”业务流程

本着“专业的事情交给专业的人去做”的业务理念,社区养老服务中心不再从事具体的养老服务事务,而是从养老服务实施中心变身为养老服务调度中心,为老年人群体和养老服务提供者之间搭建一个沟通平台。借助“智慧社区养老服务系统”,以提供上门服务为主,可以为老年人群体提供更多种类的养老服务;充分兼顾行动不便的居家老人,全面覆盖老年人群体。

“智慧社区养老服务系统”投入运行后,老年人从社区获取养老服务的业务流程变更,如图 3-15 所示。图 3-8“养老服务订单处理”业务上下文图中的四个参与者:老年人/老年人亲属、养老服务提供商、接单人员、社区服务运营人员被转换为图 3-15 业务流程图中的四个泳道。

(1) 当老年人有养老服务需求的时候,有 3 种途径可以完成养老服务下单。

① 若老年人不愿意记电话号码,可以通过购买定制的智能手机来解决。定制手机的智能终端上有红绿两个按钮,绿色按钮代表日常的服务请求,红色按钮代表紧急求助,两个按钮都可以通过系统拨号连接到“智慧社区养老服务系统”,系统能够显示拨号智能终端所绑定的老

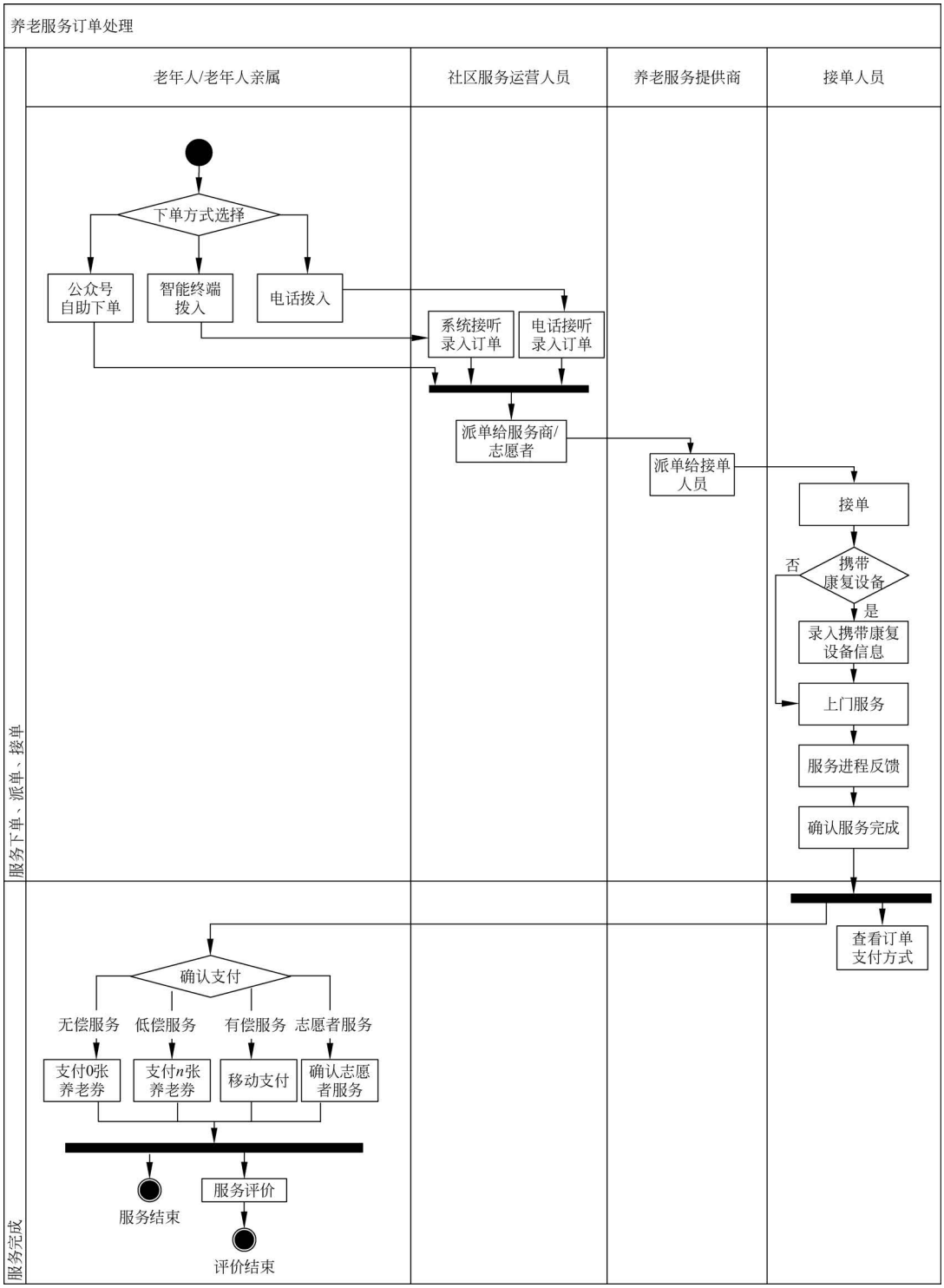


图 3-15　“智慧社区养老服务系统”的“养老服务订单处理”业务流程图

年人年龄、住址等相关信息，方便社区服务运营人员快速处理老年人需求。

② 若老年人不愿意购买定制的智能终端，可以通过普通的手机打电话到社区养老服务中心，由社区服务运营人员将老年人的个人信息及服务需求录入到系统中，同样也可以满足老年人需求。

③ 为老年人亲属提供自助下单功能。针对有些不能自理的老年人,或者老年人亲属想为老年人在线申请一些养老服务,可以通过“智慧社区养老服务系统”开通的小程序或公众号,自行查阅服务种类及详情,选择合适的服务下单。

(2) 社区服务运营人员根据服务性质派单给养老服务提供商或者志愿者。

老年人如果只是比较烦闷,想找个人说说话或者陪同去商场购买东西等,这类需求可以派单给志愿者完成;如果需要理发、送餐、上门抽血等专业服务,则需派单给相应的养老服务提供商。养老服务提供商接单后在企业内部派单,接单人员接单后上门服务,如果需要使用康复设备,还应将设备使用信息录入系统。养老服务完成后,接单人员应及时确认。

(3) 老年人或老年人亲属应在服务完成后及时支付订单。

服务订单的支付方式分为 4 种:第 1 种是无偿服务,需支付 0 张养老券(即使如此也需要进行订单支付,以确认服务流程结束);第 2 种是低偿服务,需支付与服务类型相对应的养老券数量;第 3 种是有偿服务,需通过微信、支付宝等方式支付订单;第 4 种是志愿者服务,只需要确认志愿者服务完成即可。这 4 种支付方式其实分为三大类:第一类是养老券支付。养老券由政府部门根据社区管辖内的老年人年龄和身体状况,按照一定的养老券发放规则每月发放,无偿服务和低偿服务实际上是由政府以养老券的形式为养老服务买单,为老年人群体提供一定次数的养老服务,提高老年人群体生活满意度;第二类是自费支付。这种情况适用于只支持自费的服务或者可以支付养老券的服务但养老券已消费完毕,可将服务折算为相应的自费金额;第三类是志愿者服务。这种情况可以为志愿者累计义务服务的小时数,累积到一定数量后为志愿者发放志愿者服务证书予以表彰。

3.5 业务领域建模

业务领域建模的目标是了解这个业务领域中有哪些业务实体,它们之间存在什么样的关系。在业务领域建模的过程中,一般采用“先局部,后全局”的方法,先针对每个业务流程创建局部的领域类图,然后再将所有的局部领域类图进行合并、抽象和提取,形成全局的领域模型。业务领域建模的产物主要是类图。

3.5.1 类图

1. 类与对象

在面向对象的思想中,将现实世界中的万物都看作是对象,这符合人类的思考习惯。“对象”是具体的事物,每种事物都有自己的属性和行为。“类”是对现实生活中某一类具有共同属性和行为的事物的抽象。类与对象是面向对象思想中最基本的概念。

首先以餐厅点餐的场景来理解面向对象思想,了解类和对象的概念。

人的大脑对事物的理解有个归类的过程,会把每个具体物体归类在某个类别之下。面向对象思想也是对各种事物进行归类。思考每个顾客点餐的场景都有什么相似性,如何做归类?直观看到的场景是:顾客翻阅菜品信息,然后选择菜品下单。根据人的认知能力可知,在每张餐台就餐的顾客对象是一类事物,可以抽取为一个类,就是“顾客”类,张先生和李女士就是具体的顾客对象。同理,被翻阅的每一道菜品,可以抽取为“菜品”类,红烧鱼和土豆丝就是具体的菜品对象。点餐场景中还有一类具有抽象性的事物就是顾客下单的内容,可以将其抽取为“订单”类。

类的对象需要执行特定的行为才能描述一个特定的场景。从上述点餐场景中抽取的类有

哪些行为呢？顾客把“查询菜品”这件事交由“菜品”类负责（或称把查询菜品消息传递给菜品），“菜品”类就执行“查询”行为，同时接收执行“查询菜品”行为所需的相关信息——“菜品名称”；顾客把“下单”这件事交由“订单”类负责（或称把下单消息传递给订单），“订单”类就执行“下单”行为，同时接收执行“下单”行为所需的相关信息——“菜品”。点餐场景中类之间的交互关系如图 3-16 所示。

图 3-16 呈现了面向对象方法中对象之间的消息传递机制：在面向对象的世界中，对象间的相互联系是通过传递“消息”来完成的。对象之间通过“消息传递”驱动对象执行一系列的操作，从而完成某一任务。换句话说，行为的启动是通过将“消息”传递给对此行为负责的对象来完成的，同时还将附加执行该行为所需的信息（参数），收到消息的对象则会执行相应的“方法”（行为）来实现传达消息者的要求。

总之，在面向对象的思想中，用类和对象描述现实世界中的事物，用消息和方法模拟现实世界中对象之间交互所构成的场景。

2. 类的表示法

在类的 UML 图中，使用一个长方形描述一个类的主要构成。将长方形垂直地分为 3 层：第 1 层是类名，第 2 层是属性，第 3 层是操作，在属性和操作之前的符号代表可见性。当类图重点表示类之间的关系时，下面两层可以省略，以突出重点研究内容。类的表示法示例如图 3-17 所示。

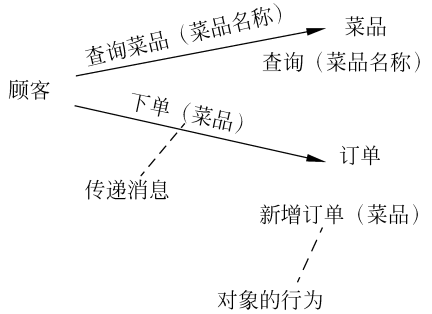


图 3-16　点餐场景中类的交互关系

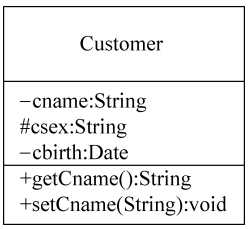


图 3-17　类的表示法示例

1）类名

给类命名时，应遵守下列规则：

- 如果类名使用英文单词，那么类名的首字母应大写。在图 3-17 中，顾客类被命名为“Customer”。类名也可以使用中文进行命名，尤其在业务需求建模阶段，可以帮助客户方理解，便于沟通。
- 类名应见名知义，最好使用业务领域中的术语，不要随意造词。类名应具有明显的可识别度，命名为“A、B”之类的名称就不合适，因为无法从“A、B”这样的名称中初步判断类的含义。类名应含义明确，在餐厅就餐场景中，“顾客”比“用户”就描述得更精准。
- 当类名由几个英文单词复合而成时，应遵循“驼峰规则”：类名每个单词的首字母使用大写。例如，餐厅经理类可命名为 DinnerManager。

2）属性

- 属性用于描述类的本质特征。在面向对象编程中，属性被称为类的成员变量。在图 3-17 中，使用 cname（顾客姓名）、csex（顾客性别）和 cbirth（顾客生日）三个属性来描述每个具体的顾客对象。需要注意的是，只有软件系统需要使用的那些属性才应抽取出来。例如，如果顾客的职业和软件系统运行无关，则无需列出该属性。

- 属性的命名规范也应遵循“驼峰规则”：属性的首字母使用小写，其余单词的首字母使用大写。例如，下单日期属性可命名为 orderDate。
- 属性层描述属性的语法格式是：“可见性 属性名：数据类型”。数据类型既可以是基本数据类型(整数、实数、布尔型等)，也可以是复杂数据类型(类、接口、枚举、数组等)。

3) 操作

- 操作用于刻画类的功能。在图 3-17 中，getCname()是一个用于获取顾客姓名的操作，在面向对象编程中也称为成员方法。
- 方法的命名规范也应遵循“驼峰规则”：方法的首字母使用小写，其余单词的首字母使用大写。例如，修改顾客姓名方法可命名为 setCname(String)。
- 方法名后面有一对小括号，括号内是对象接收传递消息时所附加的信息，称为参数。参数可以没有，也可以是一个或多个。当参数有多个时，参数之间用逗号分隔。参数的命名同属性的命名规则。例如，获取两个整数的最大值的方法可命名为 getMax(int,int)。
- 操作层描述方法的语法格式是：“可见性 方法名(参数列表)：返回值类型”。参数列表中只需要标明每个参数的类型即可，不需要写参数名称。返回值类型表示调用方法后，被返回数据的类型。如果没有返回值，则用 void 表示调用方法后无返回数据。

4) 可见性

属性和操作名称前面的符号表示属性或操作的可见性，UML 类图中表示可见性的符号有四种：

＋：表示 public(公用的)，对所有类可见；

＃：表示 protected(受保护的)，对该类的子孙可见；

－：表示 private(私有的)，只对该类本身可见；

～：表示 friendly(友好的)，或称包访问权限，只对同一包下声明的其他类可见(什么符号都不使用也可以表示友好权限)。

3. 类之间的关系

日常生活中需要解决的问题经常是复杂的、高度抽象的，从问题中抽象出的类之间往往也是有关系的。类之间的关系通常可以分为以下几种：泛化关系、实现关系、依赖关系、关联关系、聚合关系和组合关系。在 UML 类图中，类之间的关系概览如表 3-4 所示。

表 3-4 类之间的关系

序号	关系名	关 系 描 述	图 符
1	泛化关系	描述了一般事物与该事物中的特殊种类之间的关系	空心三角实线箭头
2	实现关系	描述了一个类实现了一个接口的关系	空心三角虚线箭头
3	依赖关系	表示一个类依赖于另一个类的定义	虚线箭头
4	关联关系	表示两个类之间存在某种语义上的联系，它是所有关系中最通用的	实线

续表

序号	关系名	关 系 描 述	图　　符
5	聚合关系	表示“整体与部分”之间的弱关联关系	空心菱形箭头
6	组合关系	表示“整体与部分”之间的强关联关系	实心菱形箭头

1）泛化关系

泛化关系的本质是继承关系。如果一个类(称为子类)继承另外一个类(称为父类)，此时可以说，子类从父类继承而来，父类则是子类的泛化。在 UML 类图中，表示继承关系的空心三角实线箭头由子类指向父类。

图 3-18 是一个继承关系示例，很多情形下父类是抽象类，在 UML 类图中用斜体表示。在图 3-18 中，Animal(动物)是抽象父类，Dog(狗)和 Cat(猫)是子类。Dog 和 Cat 继承了 Animal，也可以说 Dog 和 Cat 可以被泛化为 Animal。

2）实现关系

实现关系描述了一个类实现接口的功能。实现是类与接口之间最常见的关系。接口是一组规则的集合，它规定了实现接口的类必须拥有的一组规则。接口由类去实现，以便使用接口中的方法。一个类可以实现多个接口。在 UML 类图中，接口用<< interface >>标识，表示实现关系的空心三角虚线箭头由实现类指向接口。

图 3-19 是一个实现关系示例。Car(汽车)是一个实现类，它实现了两个接口 MusicPlayer(音乐播放器)和 Usb(USB)。同一个接口 Usb 可被不同的实现类 Car 和 PC 实现。由示例可知：多个无关的类能够实现同一接口；一个类能够实现多个无关的接口。

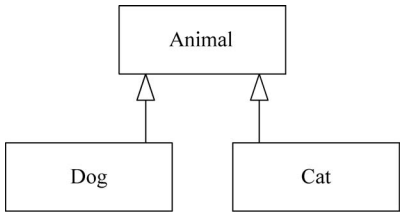


图 3-18　继承关系示例

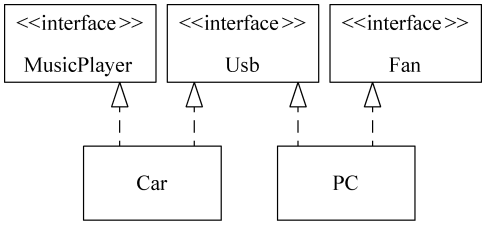


图 3-19　实现关系示例

3）依赖关系

依赖关系是类之间最弱的关系，是指一个类依赖于另一个类的定义。这种依赖关系具有偶然性、临时性，是非常弱的关系。在 UML 类图中，表示依赖关系的虚线箭头由依赖类指向被依赖类。

图 3-20 是一个依赖关系示例。OldMan(老年人)类的 use(使用)方法只有传入一个 SmartDevice(智能设备)device 对象，调用 device 对象的 dial(拨号)方法才能发挥作用，因此可以说 OldMan 类依赖于 SmartDevice 类。

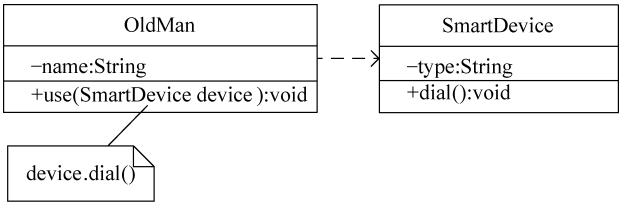


图 3-20　依赖关系示例

从示例中看到：若 A 类依赖 B 类，则 B 类将作为参数或返回值类型在 A 类某个方法中使用，所以依赖关系也可以称为 A 类“use”B 类。表示依赖关系还有两种情形，B 类作为 A 类方法中的局部变量；B 类中的静态方法在 A 类方法中被直接调用，本书不再赘述。

4）关联关系

关联关系指类与类之间的连接，关联关系是比依赖关系更强的一种关系，两个类之间可以长期合作。关联关系又可进一步分为单向关联、双向关联。两个类之间是一个层次的，不存在部分跟整体之间的关系。

(1) 单向关联。

图 3-21 是一个单向关联关系示例。OldMan 类把 SmartDevice 类作为 OldMan 类的一个成员变量，则称 OldMan 类关联 SmartDevice 类。从语义上理解，可以说老年人拥有自己的智能设备，因此关联关系也可以称为 A 类“has”B 类。

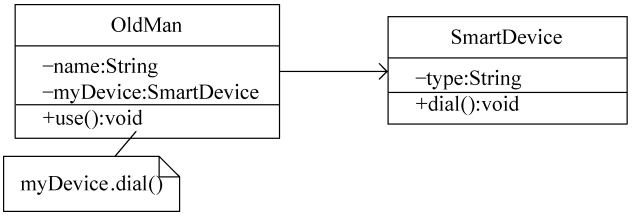


图 3-21　单向关联关系示例

在 UML 类图中，表示单向关联关系的两个类使用带箭头的实线来连接，若 A 类关联 B 类，那么 B 类通常是作为 A 类的成员变量出现，箭头的方向由 A 类指向 B 类。

(2) 双向关联。

图 3-22 是一个双向关联关系示例。OldMan 类既可以把 SmartDevice 类作为 OldMan 类的一个成员变量，表示老年人拥有自己的智能设备；SmartDevice 类也可以把 OldMan 类作为 SmartDevice 类的一个成员变量，表示智能设备拥有自己的主人，它表达了一个双向关联。

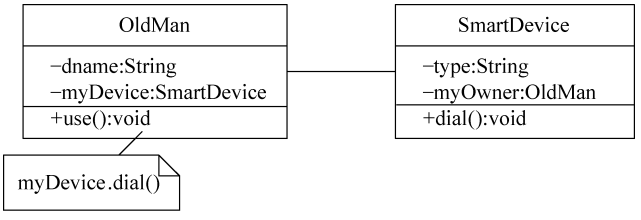


图 3-22　双向关联关系示例

在 UML 类图中，表示双向关联关系的两个类用一个不带箭头的实线来连接，双向关联的双方各自持有以对方类作为数据类型的成员变量。

(3) 自关联。

图 3-23 是一个自关联关系示例。Person（人）类的 children（子女）属性的数据类型是 Person 类自身，即人的子女也是人，它表达了一个自关联。在 UML 类图中，用一个带箭头的实线由类自身指向自身来表达自关联关系，类持有以自身类作为数据类型的成员变量。

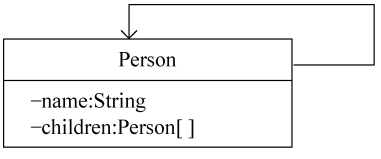


图 3-23　自关联关系示例

(4) 关联关系的多重性。

关联关系的多重性表示两个关联对象在数量上的对应关系，其表示格式为“$n..m$”，n 为

连接对象的最小数目，m 为连接对象的最大数目，当数目不确定时用“*”表示，表示多重性的符号应标注在关联关系连线的两端。

关联关系按照多重性划分，可分为一对一、一对多和多对多三种关系。若 A 类和 B 类之间有关联关系，多重性表示方法及含义如表 3-5 所示。

表 3-5　多重性分类、表示方法及含义

多重性分类	A 类多重性	B 类多重性	含　义
一对一	1	0..1	表示 1 个 A 类对象与 0 个或 1 个 B 类对象关联
	1	1	表示 1 个 A 类对象与 1 个 B 类对象关联
一对多	1	*	表示 1 个 A 类对象与 * 个 B 类对象关联，数量上限和下限均不确定
	1	0..*	表示 1 个 A 类对象与 0 个或多个 B 类对象关联，数量上限不确定
	1	1..*	表示 1 个 A 类对象与 1 个或多个 B 类对象关联，数量上限不确定
多对多	*	*　0..*　1..*	表示多个 A 类对象与多个 B 类对象关联

注意：多重性的表示格式为“$n..m$”，因此多重性的上下限不一定是 0 或 1，例如：多重性可以表示为：“4..*”（下限为 4，上限不确定）或“*..100”（下限不确定，上限为 100）或直接是一个数字 n（精确的 n 个对象）。

在类图中添加了关联关系多重性的示例如图 3-24 所示。Driver（司机）类与 Car（汽车）类之间关联关系多重性为 1 对 0..*，从语义上理解即为 1 个司机可以有 0 台汽车或多台汽车。Car 类与 Wheel（车轮）类之间关联关系多重性为 1 对 4，从语义上理解即为 1 辆汽车必须有 4 个轮子。所以，在类图中添加了多重性表示后，类图的含义会更加清晰。

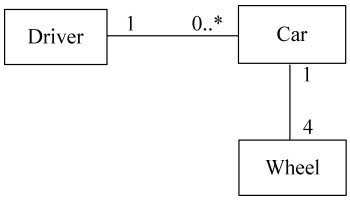


图 3-24　关联关系多重性示例

5）聚合关系

聚合关系是关联关系的一种特例，它体现的是整体与部分的关系，强调“整体”包含“部分”，但是“部分”可以脱离“整体”而存在，是一种“弱拥有”的关系。在 UML 类图中，表示聚合关系的两个类使用空心菱形箭头连接，部分类通常是作为整体类的成员变量出现，空心菱形箭头指向整体类。

图 3-25 是一个聚合关系示例。Wheel（轮胎）是 Car（汽车）的组成部分，Car 类与 Wheel 类之间的关系就是整体和部分的关系。在现实世界，汽车包含轮胎，但是轮胎可以脱离汽车而存在，因此两者之间是聚合关系。在类图建模中，Wheel 类（部分类）作为 Car 类（整体类）的成员变量形式出现。

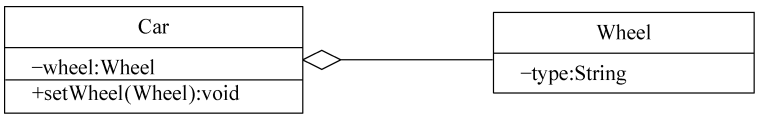


图 3-25　聚合关系示例

需要注意的是：虽然聚合关系和关联关系相似，也表达的是“has”的关系，但是关联关系

中的两个类是同一层次的,而聚合关系中的两个类是整体和部分之间的关系。

6) 组合关系

组合关系也是关联关系的一种特例,它体现的也是一种包含关系,但这种关系比聚合关系更强,也称为强关联。组合关系与聚合关系最大的不同点在于:“部分”不能脱离“整体”而独立存在。在 UML 类图中,表示组合关系的两个类使用实心菱形箭头来连接,部分类通常是作为整体类的成员变量出现的,实心菱形箭头由部分类指向整体类。

图 3-26 是一个组合关系示例。Hand(手)和 Foot(脚)是 Person(人)的组成部分,Hand 类、Foot 类与 Person 类之间的关系就是整体和部分的关系。在现实世界,人有手有脚,并且手脚不能脱离人而独立存在,因此两者之间是组合关系。在类图建模中,Hand、Foot 类(部分类)作为 Person 类(整体类)的成员变量出现。

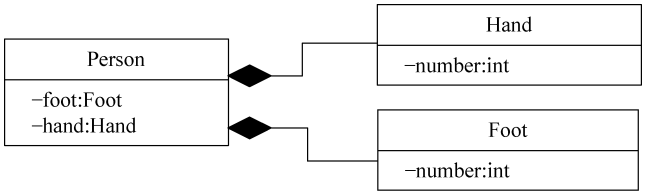


图 3-26 组合关系示例

总结一下:泛化关系和实现关系体现的是一种类与类或者类与接口之间的纵向关系;其他四种关系体现的是类与类之间的横向关系,这四种关系所表现的关联程度的强弱性从强到弱依次为:组合关系>聚合关系>关联关系>依赖关系。

类图的基本语法并不复杂,可能最多学习两三天就可以掌握,然而要真正做到活用类图则需要不断地实践。

3.5.2 业务类图

业务类图是用来描述领域对象模型的一种手段,它以面向对象的视角,描述现实世界中各种事物之间的关系。“业务类”在其他书籍中还有很多叫法,如业务实体、概念类、领域类、实体类等,实质上表示的都是在业务流程中涉及的各种业务主体、信息实体、数据实体等业务对象。

在进行业务类图建模时,最重要的任务是从问题域中抽取类,并定义类之间的关系。在实际工作中,一般先把复杂的问题分解为多个问题域,针对不同的问题域构建业务类图片段,然后再合并、提炼为全局的业务类图。

本书将一个业务作为一个问题域,针对每一个业务进行业务类图建模。业务类图建模的基本步骤如下。

1. 标识类

1) 寻找候选类

寻找候选类的一个常用方法:从业务流程文字描述和业务流程图中,提取一些重要的业务参与者、概念、业务术语等名词或名词短语,把它们作为候选类。

2) 确定业务类

并不是每一个候选类都是合适的业务类。有些名词在业务流程中出现只是为了阐述相关背景,对于系统的运行并没有起什么作用;有些名词实际上是某些业务实体的属性。这些名词都不应该单独抽取为业务类。将候选类列表经过筛选之后,得到的就是业务对象模型所需的业务类。

2. 确定类之间的关系

确定了业务类之后，接下来就是明确类之间的关系。在业务对象模型中，最常见的就是关联关系。绘制好类之间的关系后，还需要进一步完成多重性分析，在类图上进行数量关系的标识。

3. 明确类的属性

经过上面两个步骤后，业务类图的初步框架已经形成。这一步就要把业务类的各个属性加入类图。用类图进行业务建模时，一般不需要将操作标识出来。业务类图的建模过程涉及的信息庞杂，类图的生成也需要经过不断的修改、迭代，逐步完善。

3.5.3　案例的业务类图建模

本节以“养老服务订单处理”业务为例，将“养老服务订单处理”业务作为一个问题域，根据业务流程文字描述及业务流程图，按照业务类图建模的基本步骤逐步构建业务类图模型。

1. 标识类

1）寻找候选类

首先，仔细阅读“养老服务订单处理”业务流程文字描述及业务流程图，将重要的业务参与者、概念、业务术语等名词或名词短语提取出来，把它们作为候选类，得到如下的候选类列表：

老年人、养老服务、社区、智慧社区养老服务系统、智能终端、日常的服务请求、紧急求助、社区服务运营人员、老年人年龄、老年人住址、社区养老服务中心、服务需求、老年人亲属、不能自理的老年人、公众号、服务种类及详情、服务性质、养老服务提供商、志愿者、接单人员、设备、设备使用信息、订单支付方式、无偿服务、养老券、养老券发放规则、低偿服务、有偿服务、电子结账方式、志愿者服务、志愿者服务证书。

2）确定业务类

在选择候选类的时候，是无差别地保留业务流程中的相关名词，在筛选时则需要将不合适的名词排除或将表达同一个实体的名词进行合并。

- “老年人”无疑是一个重要的类，而对于“老年人年龄”“老年人住址”“老年人亲属”“不能自理的老年人”这些名词，可以从中提取出“出生年月”“住址”“亲属”“自理状况”等名词，作为描述“老年人”实体的属性。
- “社区”“社区养老服务中心”这些名词是否应抽象为业务类，要取决于“智慧社区养老服务系统”是只限于在某一个特定的社区运行实施还是要推广到多个社区使用。如果该系统只在一个社区使用，当然无需把“社区”“社区养老服务中心”抽取出来，因为只是一个业务背景的发生地而已。但若从系统未来的扩展性上考虑，最好把“社区”抽取为业务类，以便未来区分不同的“社区”实体。
- “智慧社区养老服务系统”“公众号”指的是要开发的系统本身及前端交互形式，需要排除掉它们。
- “养老服务”指社区能够为老年人提供的服务，是一个很重要的业务类。“日常的服务请求”“紧急求助”是按服务紧急程度划分的两种养老服务种类；“服务种类及详情”是描述服务的相关信息；“服务性质”实际指的是“服务提供者性质”，据此进行划分，可分为养老服务提供商和志愿者；“订单的支付方式”也是服务的一个属性，“无偿服务”“低偿服务”“有偿服务”“志愿者服务”都是这个属性的取值集合中的一个值。根据订单支付方式不同，每一种支付方式还对应着“养老券张数”或“自费金额”，应作为养老

服务的属性。根据上述分析,提取业务类“养老服务”,将“服务紧急程度”“服务提供者性质”“服务种类”“服务种类详情”“订单的支付方式”“养老券张数”“自费金额”等作为“养老服务”业务类的属性。

- “服务需求”指一次具体的服务要求信息,可以给它更名为业务类“服务订单”,以便于沟通和理解,将支付订单的“电子结账方式”作为其属性。
- “智能终端”是一个相对比较独立的业务类,描述智能终端的型号、厂家、编号等信息,每一个投入运行的智能终端都应和一个老年人信息进行绑定。
- “社区服务运营人员”“养老服务提供商”“志愿者”“接单人员”都是业务流程图中重要的业务参与者,对应的不是一个人而是一种角色,实际对应了多个业务参与者对象,所以应该抽取为业务类。
- “志愿者服务”记录的是每一个志愿者的服务种类、服务小时数、服务情况。如果每次从全部的养老服务记录中去筛选某个志愿者的服务记录,效率会较低,因此,“志愿者服务”比较特别,它是“订单支付方式”属性中的一个取值,同时它也应另外重命名为“志愿者服务记录”后独立为一个业务类。相应地,如果需要记录“志愿者服务证书”的发证时间、发证机关等信息,它就应该独立成为一个业务类。

经过上述分析,确定案例的业务类列表为:老年人、养老券发放规则、社区、养老服务、服务订单、智能终端、社区服务运营人员、养老服务提供商、志愿者、接单人员、志愿者服务记录、志愿者服务证书。

2. 确定类之间的关系

确定了业务类之后,接下来的任务就是厘清类之间的关系,绘制“养老服务订单处理”业务的业务类图如图 3-27 所示。

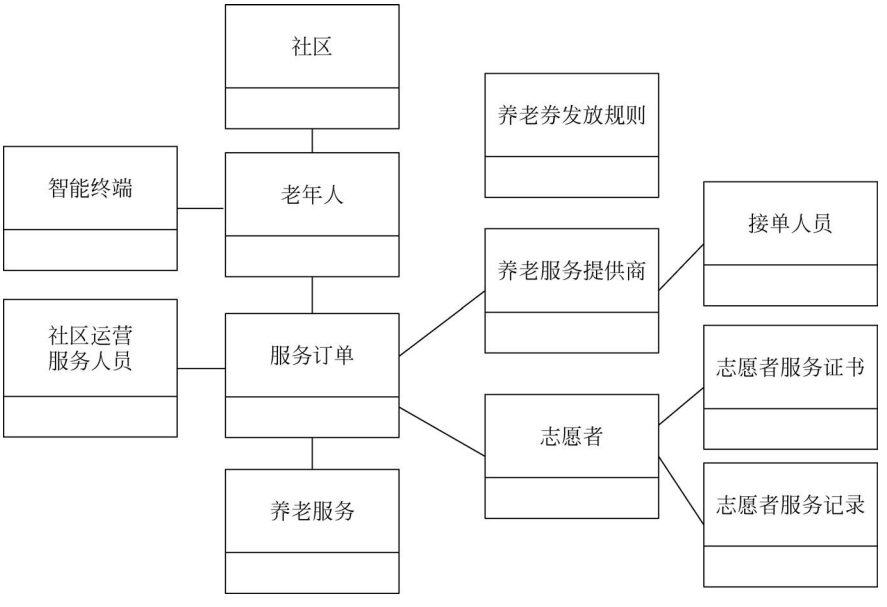


图 3-27 “养老服务订单处理”业务类图

业务类之间通常是关联关系居多,通常应在图 3-27 的基础上,进一步明确标识业务类之间的关联多重性,以更准确地反映业务规则和现实语义,添加多重性后的业务类图如图 3-28 所示。

在图 3-28 中,通过添加多重性传递了更多业务类之间的关联细节,例如:

- 一个老年人身份信息只能绑定一个智能终端,一个智能终端也只能绑定一个老人。

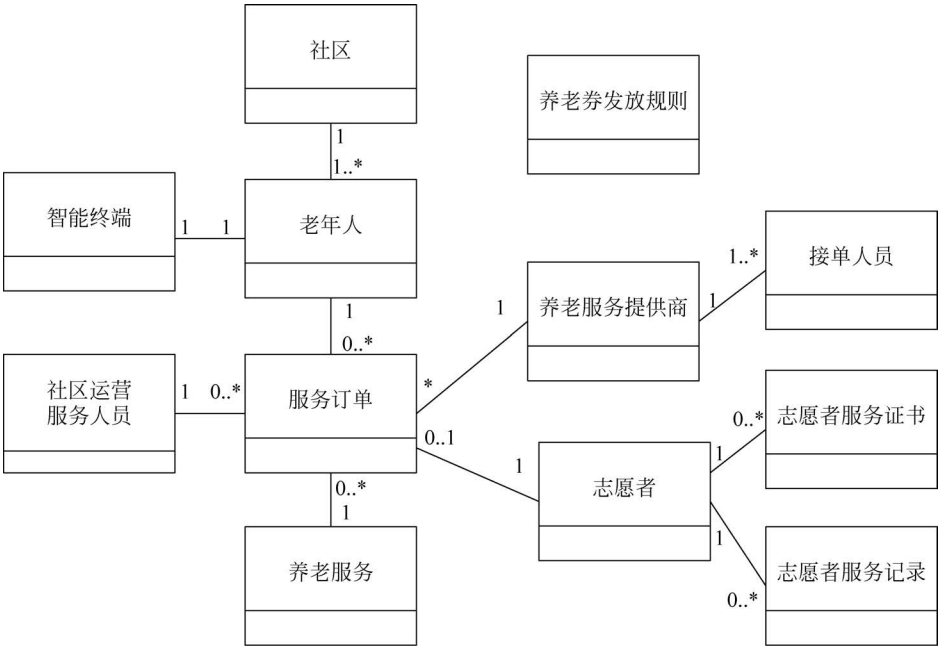


图 3-28　添加多重性后的“养老服务订单处理”业务类图

- 一个社区能包含多个老年人，而一个老年人只能属于一个社区。
- 一个服务订单只能由一个社区服务运营人员来进行派单，只能派给一个养老服务提供商或志愿者，一个订单只能包含一项养老服务。
- 养老服务提供商旗下至少有一名接单人员。
- 志愿者只有完成服务后，才会有相应的志愿者服务记录，并在满足一定小时数后获得志愿者服务证书。

深入思考 3.4　图 3-28 中，“志愿者”与“服务订单”之间的关联关系为什么不是一对多的关系？

参考答案：在《UML 和模式应用》书中提到：多重性的值是指在特定时刻（而不是在某个时间跨度内），一个类有多个实例与对方类的一个实例发生联系。业务类图中的多重性与数据库设计的 E-R 图中实体与实体间的联系有区别，在 E-R 图中更偏重的是数据间的关系，一个志愿者可以对应多条服务订单历史记录，是一对多的关系；而业务类图中的多重性标识更侧重于业务规则的体现：在同一时刻，一个志愿者要么没有进行服务，联系多重性为“0”；要么只能服务一个订单，联系多重性为“1”。

详情请参见微课视频 3-4。

在图 3-28 中，还可以看到一个孤零零的类“养老券发放规则”，它和其他业务类之间都没有联系。这是什么原因造成的呢？

图 3-28 中的业务类图是从“养老服务订单处理”业务流程描述中提取的业务类关联之后，得到的业务对象模型片段，它只是整个业务类图的一部分。而“养老券发放规则”业务类虽然出现在了“养老服务订单处理”业务流程描述中，但它主要用于“养老券发放”业务，形成的业务类图模型简图如图 3-29 所示（业务流程详情不再赘述）。

按照上述建模过程，针对每一个业务，创建局部的业务领域类图片段。最后，案例的全局业务类图将由图 3-28、图 3-29 和更多的业务类图片段合并后，加以抽象、提炼得到。

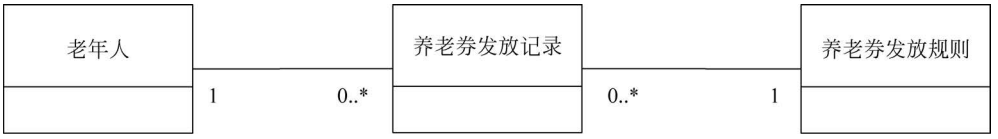


图 3-29 “养老券发放”业务类图

3. 明确类的属性

在找到反映问题域本质现象的业务类,并描述了它们之间的关系之后,结合需求分析为每一个类添加属性。本节案例通过在图 3-28 的基础上添加属性后,得到更加细化的业务类图,如图 3-30 所示。

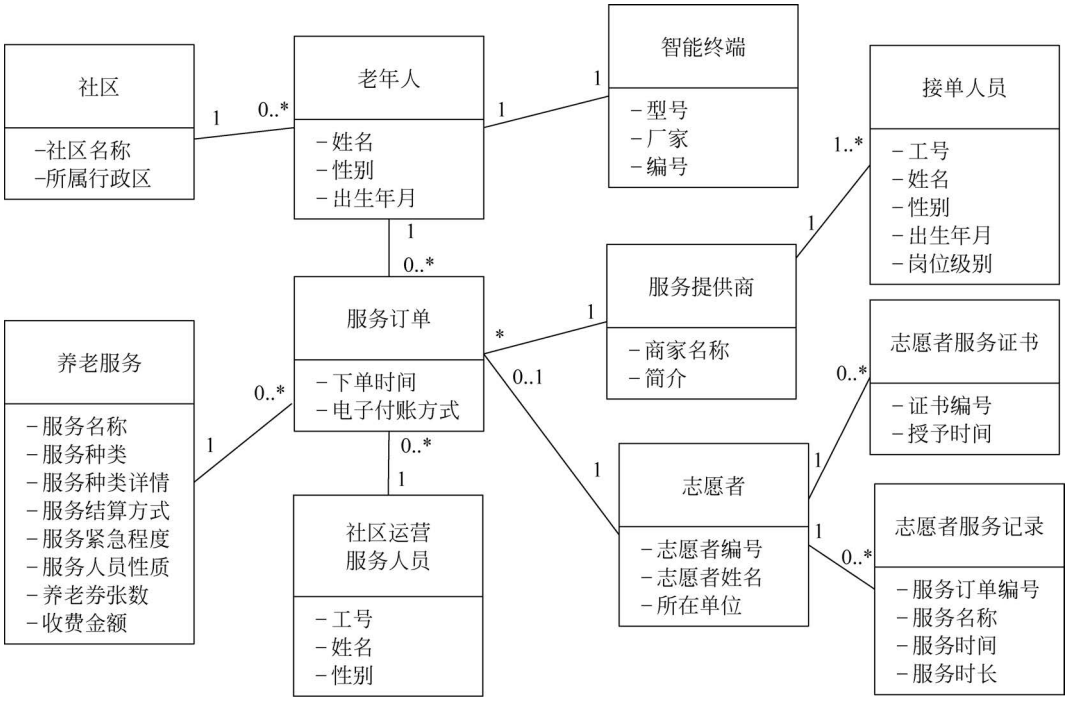


图 3-30 添加属性后的“养老服务订单处理”业务类图

习题

一、选择题

1. 下列不属于非功能需求的是(　　)。

A. 页面间跳转时间≤2s　　B. 支持并发用户数为 50

C. 提供运行日志管理功能　　D. 能够下单购物

2. 下列不属于需求开发阶段的是(　　)。

A. 需求可行性分析　B. 需求获取　　C. 需求分析　　D. 需求定义

3. 下列哪一项不是描述一个业务所需的三个基本要素?(　　)

A. 业务实体　　B. 模型　　C. 流程　　D. 数据

4. UML 是一种(　　)?

A. 编程语言　　B. 建模工具　　C. 建模语言　　D. 软件开发过程

5. 下列哪一项不属于 UML 常用的图?(　　)

A. 部署图　　B. 流程图　　C. 包图　　D. 顺序图

6. 业务上下文图以什么为边界？（　　）

A. 组织　　B. 系统　　C. 业务　　D. 事件

7. 泳道图中，每条泳道不能代表下列哪个选项？（　　）

A. 人　　B. 系统　　C. 活动　　D. 部门

8. 在类的 UML 图中，下列哪一项不能用于描述一个类的主要构成？（　　）

A. 类名　　B. 对象名　　C. 属性　　D. 操作

9. 下列哪一项用于描述类的本质特征？（　　）

A. 功能　　B. 对象　　C. 属性　　D. 操作

10. 下列四种关系中表现关联程度最强的是哪一项？（　　）

A. 组合　　B. 聚合　　C. 关联　　D. 依赖

二、判断题

1. 业务需求更多的是从技术的角度整理出的需求。（　　）

2. 软件需求是指用户对系统在功能、行为、性能、设计约束等方面的期望。（　　）

3. 需求开发为需求管理提供支持和保障。（　　）

4. 业务模型是系统需求分析阶段的重要输入。（　　）

5. 业务流程图是业务流程建模成果。（　　）

6. 在面向对象的思想中，用消息和方法来模拟现实世界中对象之间的交互。（　　）

7. 上下文图常被用于描述系统和外部参与者之间的关系。（　　）

8. 系统流程图表达了信息在系统各部件之间的流动情况，重点在控制流。（　　）

9. 在 UML 活动图中，使用分岔与汇合来表示事件的并发处理。（　　）

10. 类是对对象的抽象。（　　）

三、综合题

1. “进书”业务的主要业务流程如下：书商将采购单和新书送至采购员处进行送检；采购员进行验收，如果新书不合格就退回给书商重新送检，合格就送至编目员；编目员按照国家标准进行分类编号，填写包含书籍基本信息的入库单；库管员验收入库单和新书，如果合格就入库，并更新入库台账；如果不合格就退回。

请用 UML 活动图画出上述“进书”业务流程图。

2. 某慕课平台的在线作业批改系统的核心业务是“批改作业”，业务流程如下：学生通过在线方式提交作业；教师从系统中下载学生提交的作业，对每个题目进行批改、给出分数和评价，之后教师将批改后的作业上传；教务人员抽取批改后的作业样本进行抽检，给出抽检意见形成抽检报告。

（1）请用跨职能流程图画出上述“批改作业”业务流程图。

（2）画出该业务的业务类图。

3. 类图练习

（1）Windows 操作系统中通过文件夹与文件进行资源管理，文件夹中可能又包含文件夹，请用类图表达出文件夹与文件的关系。

（2）农业专家要对苹果、橘子、梨、香蕉等水果做水分、蛋白质、脂肪、膳食纤维、钙、铁、胡萝卜素、维生素等多方面的营养成分分析，请用类图对上述场景进行建模。