

# 第5章. 确定项目的前景与范围

## 5.1. 引言

在开始一个项目之初，首先要考虑的一个问题是——为什么要启动该项目？也就是说项目的目标是什么？

项目的目标就是系统的业务需求。在很多情况下，涉众可以清晰地表达出系统的业务需求，但这种情况并不多见。在更多的情况下，需要进行一些分析工作，才能得到系统的业务需求，如图5-1所示。

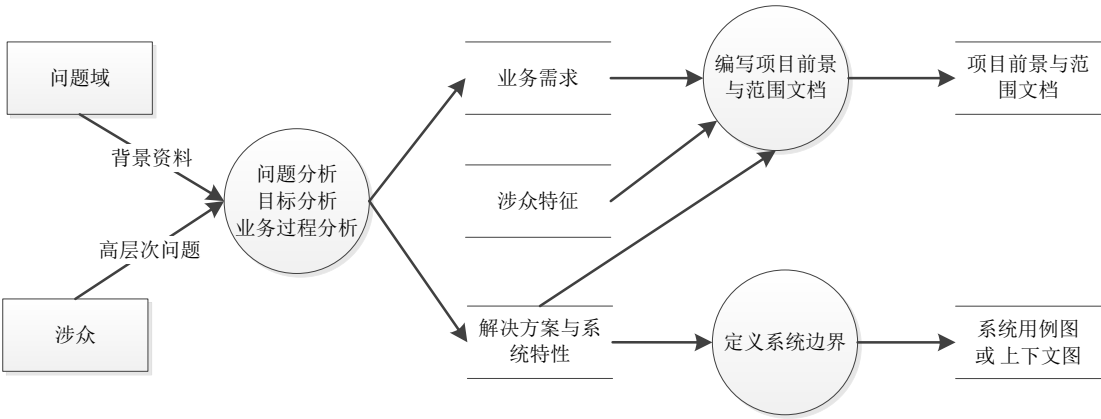


图5-1 确定项目前景与范围过程

为得到业务需求，简单情况下可以进行问题分析，复杂的情况下考虑进行目标分析，必要的时候辅以业务过程分析。

在进行问题分析、目标分析、业务过程分析时，还可以为目标的达成设计相应的高层解决方案，探索解决方案的基本功能特性。系统的高层解决方案及其功能特性可以帮助回答项目启动之初的第二个问题——“项目打算做些什么”。

根据系统的高层解决方案和系统特性，可以定义系统的上下文环境，建立系统的边界，这将是需求后期阶段需求分析活动的起点。

业务需求、高层解决方案及系统特性都应该被记录下来，定义为项目前景与范围文档。前景与范围文档中还会包含部分涉众分析的结果——涉众特征描述。前景（Vision）描述了产品用来干什么以及最终将是个什么样子。范围（Scope）则指出了当前项目是要解决的产品长远规划中的哪一部分。前景声明将所有涉众都统一到一個方向上来。范围声明为项目划定了需求的界线。

## 5.2. 问题分析

涉众在现实世界当中遇到问题时，才会试图引入软件系统，因此他们对问题是感触颇深的。这样，当涉众无法清晰地表达业务需求时，就可以转为从对问题的了解和分析开始，逐步得到业务需求及其解决方案，如图5-2所示。

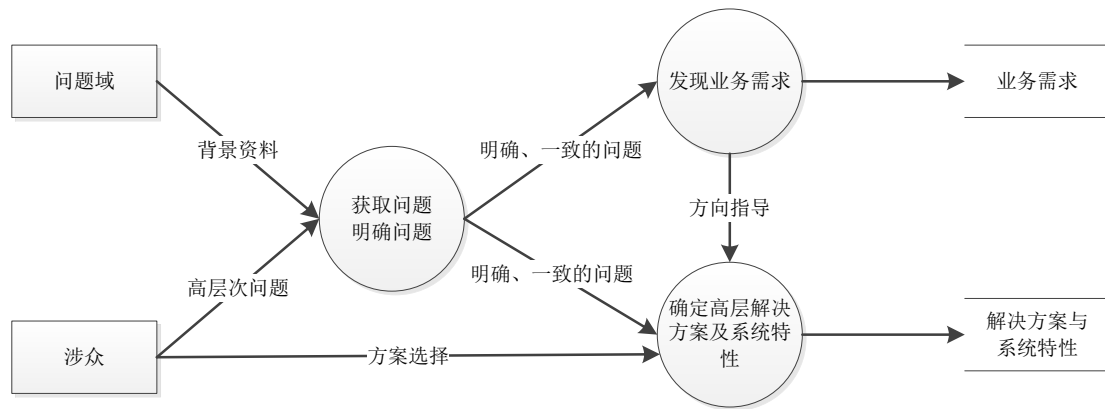


图5-2 问题分析过程

为发现业务需求而需要探讨的问题是指一些高层次的问题，是和组织的战略目标、利益分配、政策规划、业务流程等内容相关的问题。那些和具体业务的细节相关的问题不属于高层次问题。

下面就逐一描述问题分析的各个步骤。

### 5.2.1. 获取问题

问题分析的前提是获取问题，这可以通过收集背景资料或者与涉众沟通来实现。

收集背景资料时要收集业务描述及其统计数据（详细参见第6章硬数据采样部分），关注业务困难与问题。与涉众的沟通主要通过面谈（详细参见第8章面谈部分）。

从收集的资料中，可以分析、发现问题。

例如，从下面一段描述资料中，可以发现问题P1～P4

“×××连锁商店是一家刚刚发展起来的小型连锁商店，其前身是一家独立的小百货门面店。原商店只有销售的收银部分使用软件处理，其他业务都是手工作业，这已经不能适应它的业务发展要求。首先是随着商店规模的扩大，顾客量大幅增长，手工作业销售迟缓，顾客购物排队现象严重，导致流失客源。其次是商店的商品品种增多，无法准确掌握库存，商品积压、缺货和报废的现象上升明显。再次是商店面临的竞争比以前更大，希望在降低成本，吸引顾客，增强竞争力的同时，保持盈利水平。”

P1：手工作业销售迟缓，效率不高。

P2：商店的商品品种太多，无法准确掌握库存。

P3：成本不够低，导致竞争力不强，盈利水平不够。

P4：顾客不够多，销售额不高，盈利水平不够。

对于发现的每一个问题，都要逐一执行下面的“明确问题→发现业务需求→定义问题解决方案及系统特性”，得到每一个问题的业务需求、解决方案（特性、边界及约束）。将所有问题的结果综合起来，就能够得到整个系统的业务需求和解决方案（特性、边界及约束）。

### 5.2.2. 明确问题

要分析涉众的问题，首先要明确问题，将它们变得清晰，变得适宜进行分析。这个过程从问题和相关的背景描述开始。

1. 对问题达成共识

问题一般由单方涉众提出，因此在和所有涉众对其进行讨论之前，先要就问题本身达成一致，形成共识。具体的方法就是用标准化的格式描述问题，并在涉众之间取得认同。一个标准化描述的格式如表5-1、5-2所示。

表5-1问题描述格式

要素	内容
ID	问题标识
提出者	提出问题的涉众
关联者	影响该问题的解决或者受问题解决影响的相关涉众
问题	对问题的描述
影响	描述具体的影响

表5-2 问题描述示例

要素	内容
ID	P2
提出者	总经理
关联者	业务经理、总经理
问题	商店的商品品种太多，无法准确掌握库存
影响	部分商品积压占用库存成本 部分商品经常缺货影响顾客满意度 部分积压商品会超保质期产生报废，增加成本

2. 收集背景资料，判断问题的明确性

达成共识的问题是一致的问题，但一致的问题不一定是明确的问题。问题的明确性要求它们具备以下两点：（1）易于理解；（2）能指明解决的方向。不符合这两点是属于不明确的问题，往往是模糊的问题或者看上去无法使用软件系统进行解决的问题。

判定问题的明确性，需要分析和理解问题域，因此要收集和问题相关的背景资料。常见的背景资料如：

- 组织的自我介绍
- 组织的业务描述
- 组织的章程
- 组织的门户网站资料

在业务比较复杂的情况下，也可能会需要深入的知识，它们往往是对背景知识进行需求分析之后的结果。

例如，对表5-2的问题P2，需要收集的资料有：库存管理的基本活动、积压现象及统计、缺货现象及统计、报废现象及统计。

在理解问题背景的情况下，可以对问题的明确性形成判断。对于明确的问题，可以直接进行“发现业务需求”活动。对于不明确的问题，就需要进行下一步骤“分析不明确问题”。实践中，绝大多数问题都是明

确的问题，少数问题属于不明确的问题。一定要以是否能够理解并解决为判断标准，不要陷入“为了问题而问题”的误区——试图发现每个问题背后的问题。

例如，在常见的图书管理系统开发当中，问题P5就会因为比较模糊而被判断为不明确，而P6中的每一个问题都是比较明确的。再例如，在一个生产企业的销售系统当中，P7的问题看上去是软件系统无法提供帮助的，它就可以被判断为不明确（因为看上去似乎销售系统无解决生产的问题）。

- P5. 图书管理员：图书总是无法上架。
- P6. 图书管理员：图书的内容分类不合适，导致无法分类上架。
  - 图书上架的工作太繁杂，导致来不及上架。
  - 图书的借阅不遵守章程，不能保证及时上架。

P7.生产的废品过多。

3. 分析不明确问题，发现问题背后的问题

对于不明确的问题，尤其是那些初看上去无法解决的问题，不要直接拒绝和丢弃。可以尝试去发现涉众提出不明确问题的原因，理解不明确问题背后深藏的问题。

在大多数情况下，通过和涉众进行接触，解释问题不明确的原因，然后询问相关人员就可以得到问题背后的问题。例如对P5，只要简单询问图书管理员一个问题即可——“什么原因导致了图书无法上架？”。

在复杂的情况下，可以使用一些简单的方法来帮助与涉众的沟通过程。例如对P7，可以建立如图5-3所示的鱼骨图并与涉众一起逐一分析每个原因分支，也可以收集数据资料(如图5-4所示)发现真正的原因。

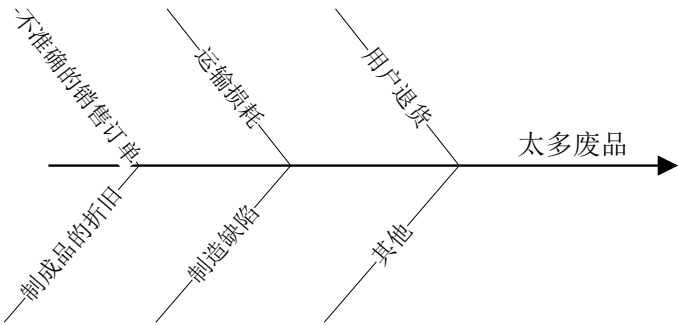


图5-3 P7原因的鱼骨图

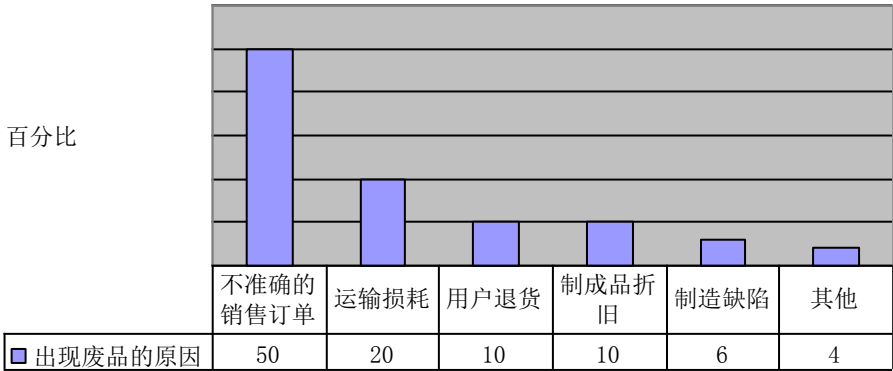


图5-4 P7的原因分析图

如果深入分析后确认问题是无法解决的，就可以拒绝并丢弃了。如果深入分析后发现问题还是可以间接

解决或者部分解决的，就需要重新更准确地定义问题，并转向后面的解决过程。例如，对问题P7，分析图5-4的数据可以发现通过解决“不准确的销售订单”，能够部分解决“生产废品太多”，就可以将P7定义为更准确的P8。

P8. 销售订单不准确，导致产生太多废品。

再次强调：不要为了探究而探究，切忌陷入盲目追逐问题背后问题的无休止过程，在问题可以明确时就应该适可而止。

### 5.2.3. 发现业务需求

每一个明确、一致的问题都意味着涉众存在一些相应的期望目标，即业务需求。因此，确定每一个问题对应目标的过程就是发现业务需求的过程。一般情况下，业务需求就是问题的反面。例如，对问题P1~4和P8，可以发现它们的业务需求分别为BR1~4、BR5。

BR1：在系统使用6个月后，商品积压、缺货和报废的现象要减少50%

BR2：在系统使用3个月后，销售人员工作效率提高50%

BR3：在系统使用6个月后，店铺运营成本要降低15%

范围：人力成本和库存成本

度量：检查平均每个店铺的员工数量和平均每10,000元销售额的库存成本

BR4：在系统使用6个月后，顾客增加10%，销售额度要提高20%

BR5：提供更准确的销售订单，在系统使用后3个月内，减少50%因此而产生的废品。

需要再次重复强调2.3.1节的内容：业务需求可验证的数值指标是通过研究问题域的背景资料得出的。

业务需求也需要得到所有涉众的一致认同。不同涉众对同一个问题的目标要求，或者不同的业务需求之间，可能会互相矛盾，这些矛盾在这个阶段必须得到妥善的解决。在大多数情况下，业务需求的冲突可以通过涉众之间的协商达成一致。实践当中发现，在得到详细的影响及代价分析的情况下，涉众一般很懂得互相妥协。在必要的情况下，决策者或者项目主管也可能被要求解决涉众之间的业务需求冲突。

为了得到一致认同的业务需求，可以扩展表5-1的问题描述至表5-3所示，让所有涉众都就“目标”描述内容达成共识。

表5-3 问题及其业务需求描述的标准化格式

要素	内容
ID	问题标识
提出者	提出问题的涉众
关联者	影响该问题的解决或者受问题解决影响的相关涉众
问题	对问题的描述
影响	描述具体的影响
目标	问题解决的目标，即业务需求

### 5.2.4. 定义问题解决方案及系统特性

仅仅理解问题和发现目标并不能自动地解决问题和达到目标,解决问题还需要需求工程师为每一个问题发挥创造力,建立解决方案。

#### 1. 建立问题解决方案

对每个明确、一致的问题,需求工程师要发现各种可行的候选解决方案,分析不同方案的业务优势和代价,然后通过和涉众的协商进行选定。对问题解决方案的描述可以如表5-4所示。

表5-4 问题的解决方案描述

要素		内容
ID		问题标识
解决方案	方案描述	概要描述解决方案
	业务优势	该解决方案所能带来的业务优势
	代价	该解决方案将花费的代价

建立候选解决方案时既要分析问题域背景,又要发挥需求工程师的创造性,但根本上还要依赖需求工程师的创造性。

例如,可以为P2建立如表5-5所示的问题候选解决方案。

表5-5 P2问题的候选解决方案

要素		内容
ID		P2
解决方案1	方案描述	准确的库存管理,记录入库和出库,提供实时的库存分析数据,可以及时地发现可能的积压、缺货、报废现象。
	业务优势	及时发现积压、缺货与报废,可以尽早处置。
	代价	对积压、缺货的预测可能不准确,会因此而产生代价。
解决方案2	方案描述	制定促销策略,处置可能的积压和报废商品。
	业务优势	通过促销,可以减少积压和报废商品带来的损失。
	代价	促销本身会产生代价。
解决方案3	方案描述	根据过去的销售情况预测未来的销售数据,并据此调整商品购买时机和数量。
	业务优势	可以做到成本最小化。
	代价	如果预测不准确,产生较多的缺货现象会降低顾客满意度。
解决方案4	方案描述	对积压、报废和缺货现象比较频繁的商品进行调整。将总是积压和报废的商品调整出销售目录,为总是缺货的商品引入新的同类商品。
	业务优势	可以比较长远地解决积压、缺货、报废现象。
	代价	无

需求工程师给出的问题只能是候选解决方案,最终的解决方案还要由涉众自己来决定,因为只有他们自己清楚所愿意接受的优势与代价。例如,在表5-5所描述的候选解决方案之中,涉众可能只会选择解决方案1、2与4,因为相对于成本的降低,他们更担心缺货现象造成的顾客满意度降低的后果。

## 2. 确定系统特性和解决方案的边界

在选定解决方案之后,要进一步明确该解决方案需要具备的功能特征,即系统特性(Feature)。然后依据这些功能特征,分析解决方案需要和周围环境形成的交互作用,定义解决方案的边界。

### (1) 系统特性

特性是对一系列内聚的相互联系的需求、领域特征和规格的总称[Classen2008]。通常,一个特性内聚于一个目标与任务,反映了系统与外界一次有价值的完整互动过程。例如,在P2的解决方案1、2、4中,蕴含的系统特性如SF1~SF7所示。

SF1: 处理批量商品出/入库,掌握库存减少/增加

SF2: 记录销售所影响的商品出库,掌握库存减少

SF3: 记录退货所影响的商品入库,掌握库存增加

SF4: 分析店铺商品库存,发现可能的商品积压、缺货和报废现象

SF5: 根据积压、缺货和报废现象调整销售的商品目录

SF6: 制定促销手段,处理积压商品

SF7: 将报废商品自动出库

### (2) 面向对象方法的问题解决方案边界——问题的用例图

系统特性描述了问题解决方案的内容,但还需要有另一种更精确的方式描述问题解决方案的边界,尤其是其与外界的交互,在面向对象方法下使用针对问题解决方案的用例图。

用例及用例图的详细情况请参见第7章。用例图描述了外部角色在与解决方案的交互中完成任务与目标。用例图可以从系统特性中抽取,找到系统特性中所蕴含的外部角色及其交互任务。

例如,从SF1~SF7中,可以发现的角色及其交互任务如下,建立问题用例图如图5-5所示:

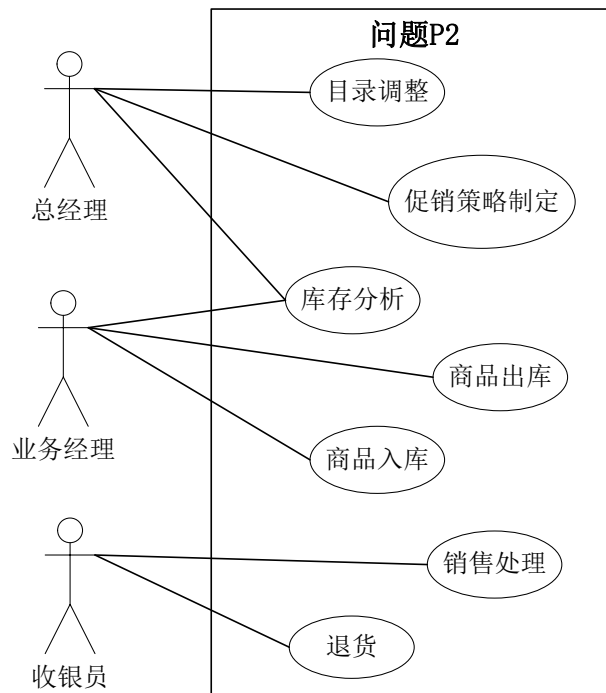


图5-5 问题P2的用例图

SF1：业务经理：商品入库、商品出库。

SF2：收银员：销售处理。

SF3：收银员：退货处理。

SF4：业务经理：库存分析；总经理：库存分析。

SF5：总经理：商品目录调整。

SF6：总经理：制定促销策略。

SF7：业务经理：商品报废。

### （3）结构化方法的问题解决方案边界——问题的上下文图

结构化方法使用上下文图描述解决方案与环境的交互。上下文图的详细情况参见第12章12.2.3节。

上下文图关注解决方案与环境之间的信息流输入/输出，以此界定解决方案的边界。上下文图可以从下面几个方面从系统特性中抽取其所蕴含的数据输入/输出关系：

- 它需要的信息由谁提供？
- 它产生的信息由谁使用？
- 谁控制它的执行？
- 谁会影响它的执行？

例如，从SF1～SF7中，可以发现的数据流如下，建立问题上下文图如图5-6所示：

SF1：商品入库数量：业务经理（源头），系统内部使用；商品出库数量：业务经理（源头），系统内部使用。

SF2：销售商品数量：收银员（源头），系统内部使用。

SF3：退货商品数量：收银员（源头），系统内部使用。

SF4：商品状态报告：系统内部数据源，业务经理与总经理（使用者）。

SF5：商品目录变更信息：总经理（源头），系统内部使用。

SF6：促销策略：总经理（源头），系统内部使用。

SF7：报废商品及其数量：系统内部数据源，业务经理（使用者）。

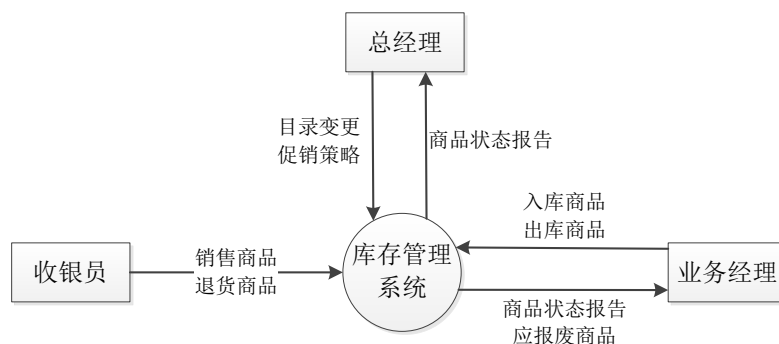


图5-6 问题P2的上下文图

### 3. 确定解决方案的约束

约束是对问题解决方案的进一步限定，会影响到设计师、程序员等后续开发者的工作决策，是一个需要重视又常被忽视的因素。



[Leffingwell1999]建议使用如表5-6所示的检查列表来帮助发现约束。

表5-6 发现约束的检查列表，源自[Leffingwell1999]

约束源	问题
经济的	有哪些财政或者预算上的约束？
	有货物成本和价格上的要求吗？
	有任何法律许可问题吗？
行政的	有产生影响的内部或外部政治问题吗？
	有什么需要部门间协调的问题吗？
技术的	在技术的选择上有什么限制吗？
	是否必须使用既有的平台和技术进行工作？
	对新技术的应用会被禁止吗？
	有可能使用COTS软件包吗？
系统的	要建立在现有系统基础之上吗？
	要维护和现有系统的兼容性吗？
环境的	需要支持哪些操作系统和环境？
	有环境的约束吗？其灵活度怎样？
	符合法律法规吗？
	有安全性需求吗？
	可能会被哪些其他标准限制？
进度及资源的	进度要求如何？
	会被限制在已有资源上吗？
	可以使用外部人力吗？
	可以暂时或永久的扩展资源吗？

例如，对问题P2的解决方案，可以发现约束如表5-7所示。

表5-7问题P2的解决方案的约束

约束源	约束	理由
技术	Constraint-1使用J2EE平台	Web方式符合互联网发展趋势，同时J2EE的开源性质可以降低成本
系统	Constraint-2使用MySQL数据库管理系统	降低成本
行政	Constraint-3不同连锁店之间的积压商品流转由各自的业务经理自行决定	利于不同连锁店之间的工作协调

### 5.3. 目标分析

作为一种实践方法，问题分析将每一个问题都独立对待，这使得它易于操作却只能适用于简单情况，因

为复杂情况下的不同问题之间会存在相互依赖关系。

相比之下，目标分析使用目标建模技术作为基础，能够处理问题、目标、特性、角色、任务等各种因素的相互依赖关系。

### 5.3.1. “目标”概念——面向目标的需求工程方法

“目标”一词就一直在需求工程中被广泛使用。因为需求概念本身，无论是需要被满足的愿望、需要被达到的目标还是需要被实现的前景，都带有目标的意味。但是，这些用法都是非正式的。

面向目标的需求工程（Goal-Oriented Requirements Engineering）方法是一种正式定义“目标”概念，并以此为基础开展需求工程活动的方法。也就是说面向目标需求工程方法中的“目标”概念是被明确定义的，有相应的方法和技术负责它的解释和使用。

面向目标的需求工程方法是指向整个需求工程的[Lamsweerde2004]，对当今需求工程方法和技术的影响比较普遍[Lamsweerde2000a]，但是它的核心作用表现在前期需求阶段[Mylopoulos2006]——项目前景与范围定义活动。

面向目标的需求工程方法正得到越来越广泛的认同和应用。“有相当数量的需求工程方法在组织和归类需求的内容时开始使用目标作为高层抽象。面向对象方法现在也将目标看作是用例一个重要部分，使用它来组织用例。……很多现存的方法学也开始整合对目标的分析与处理技术。目标概念在需求工程方法中的广泛被接受现象说明：目标已经成为了需求工程常用的核心概念”[Kavakli2002]。“目标将会补充传统方法中的实体(Entities)概念和行为(Activities)概念，一起成为需求工程建模与分析的基本对象类别”[Yu1998]。

虽然目标概念在需求工程中的重要地位已经得到了一致的认同，可是人们却一直没能就目标概念的解释和使用达成一致，也就说人们还没有能够就目标概念在需求工程中准确定位达成共识，面向目标的方法仍然缺乏统一性[Yu1998, Kavakli2002, Lapouchnian2005]，尤其是不同方法下的图示有很大差异。因此，本章下面只能是综合面向目标需求工程方法的一些共性，参考KAOS方法图示，介绍它们在定义项目前景与范围活动中的作用——目标分析。需要了解详细内容的读者请参考各种方法（KAOS[Dardenne1993, Lamsweerde1995]、NFR[Mylopoulos1992, Chung2000]、I\*[Yu1997]、GBRAM[Anton1996, Anton1997]）的专门文献。

### 5.3.2. 目标模型

#### 1. 目标

##### （1）目标的定义

对目标概念的定义存在着不同的版本。例如，[Anton1994]将目标（Goal）定义为业务、组织或者系统的高层目的（Objective），它们说明了系统被开发的原因，并用于指导企业内各种层次上的决策。[Lamsweerde2000a]认为目标（Goal）是软件系统应该通过行为者（Agent）的协作而予以实现的目的（Objective），这些行为者来自于被开发的软件系统内部或者系统的应用环境之中。而综合这些不同的定义，简言之就是：目标是系统被开发的目的。

目标模型的描述是需要精确描述和定义的，在对目标的定义当中，能够精确描述目标含义的非形式化的

说明总是需要的[Zave1997]。但非形式化的自然语言描述是无法进行计算机处理的，所以除了非形式化的说明之外，各种面向目标方法还分别给出了半形式化或形式化的描述。这些半形式化和形式化的描述采用了图形符号和文本符号的描述方式，具备了一定的语义和语法规则。其中一个重要的规则是每个目标都拥有一些特征属性，常见的有类型、名称、说明（specification）、优先级、可行性、效用（utility）等。

例如，KAOS方法对火车控制系统中的一个目标描述如图5-7所示。上面是图形描述，下面是精确定义。目标名称为DoorsClosedBetweenStations，属于目标类型SafetyGoal，它要求当火车在两个站台之间运行时火车门保持关闭。关注（Concerns）是目标达成涉及的问题域对象[Dardenne1993]。“o”、“W”是时序逻辑运算操作，常见的时序逻辑操作符如表5-8所示。

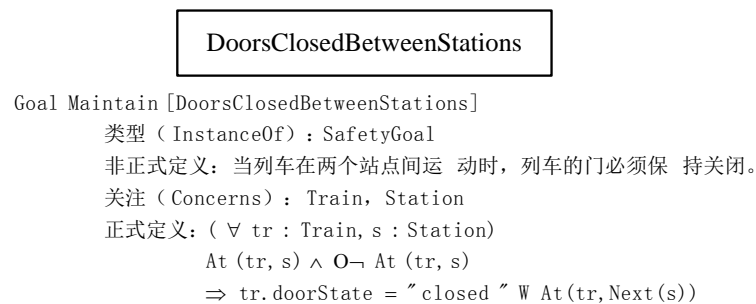


图5-7 KAOS对目标的描述，源自[Bertrand1997]

表5-8 时序逻辑常用操作符

操作符	用法	含义
○	○Q	在下一个状态中Q必须为真
●	●Q	在前一个状态中Q必须为真
◇	◇Q	Q终将在未来的某个时间点上为真
◆	◆Q	过去的某个时间上Q曾经为真
□	□Q	在所有后续路径中，Q必须始终为真
■	■Q	在过去的路径中，Q必须始终为真
U	P U Q	P必须一直为真直到将来的某一点Q为假
W	P W Q	P必须一直为真直到将来的某一点Q为真

目标的描述包括三个方面的信息[Lamsweerde2001]：类型（Type）、属性（Attribute）与链接（Link）。类型将在下面的第（4）部分详细介绍，链接将在下面的“2.关系”部分详细介绍。属性除了包含图5-7所示的必需属性（名称（DoorsClosedBetweenStations）、类型、关注、定义（正式与非正式））之外，还可以有优先级、主体、拥有者等其他可选属性。

## （2）目标的层次

目标可以在不同的抽象层次上进行描述。它可以在战略高层进行描述（达成组织的某种战略），如G1；也可以进行技术上的低层描述（实现某种操作），如G2。目标可以针对不同的内容。它可以是针对系统功能的目标，也可以是针对非功能的目标。目标的不同层次可以组成树状结构。

G1：降低5%的运营成本

Goal Achieve[OperationCostDecrease]

类型：SatisfactionGoal

非正式定义：降低5%的运营成本

关注：OperationCost

正式定义： $\forall c: \text{OperationCost}$

Decrease(c)

$\Rightarrow \Diamond \text{DecreasePercent}(c) \geq 5$

G2：在商品距离报废期还有10天时，业务经理应该得到提示

Goal Achieve[CommodityOutofDateNoticed]

类型：SatisfactionGoal

非正式定义：在商品距离报废期还有10天时，业务经理应该得到提示

关注：Commodity, GeneralManager

正式定义： $\forall c: \text{Commodity}, m: \text{GeneralManager}$

$c.\text{guaranteePeriod}() \leq 10$

$\Rightarrow \Diamond \text{Notify}(m, c)$

### (3) 目标的主体 (Agent)

目标不会自动达成，需要有主体的参与和履职。主体是系统环境中的主动部分，可以是人、硬件，更可以是软件[Dardenne1993]。例如，开发商品销售系统时的目标G1的主体可能有库存管理员、商品销售系统、收银员、信用卡刷卡器、税务系统接口.....它们共同的协作才能将运营成本降低5%。

主体是有主动性的，他们可能需要约束自己的主动性行为，以确保目标的实现。例如收银员看到系统提示的找零数额，就应该实际支付相应的零钱。再例如，总经理发现有商品积压后，就可以对其进行促销。

越是抽象、粗粒度、范围广的目标，参与的主体越多。越是具体、精细、精确的目标，参与的主体越少。例如，目标G2的主体应该只有商品销售系统。如果一个目标的主体只有待开发的软件系统一个，那么该目标就可以等同于需求了。如果一个目标的主体只有系统环境中的一个对象（例如用户），那么该目标就可以等同于假设（Assumption）与依赖（Dependency）了。

要区分目标的主体与拥有者，拥有者通常是涉众，这些涉众期望目标达成但不一定参与目标达成过程。例如目标G1的拥有者是总经理，他并不一定会参与目标达成过程。

### (4) 目标的分类

目标可以被分成不同的类型。最常见的是将其分为功能目标（Functional Goal）和非功能目标（Non-functional Goal）。功能目标是期望系统提供的服务，非功能目标是期望系统满足的质量。

功能目标和非功能目标又可以根据一定的特征进一步的细分。例如，功能目标可以分为满足型目标（Satisfaction Goal）和信息型目标（Information Goal）[Dardenne1993]，满足型目标是对行为者请求的满足，信息型目标是为了保持对行为者的信息告知。非功能目标可以依据质量模型的属性细分为安全目标（Safety Goal）、性能目标（Performance Goal）、可用性目标（Usability Goal）等等。

目标又可以被分为软目标 ( Soft Goal ) 和硬目标 ( Hard Goal ) [Mylopoulos1992]。软目标是指无法清晰判断是否满足的目标,例如关于可维护性的目标。硬目标则是那些可以通过一些技术确认其是否满足的目标,例如关于性能指标的目标。

依据其正式定义的特点, [Dardenne1993]将目标总结为5种基本模式,并以此为基础进行目标的规格与关系处理[Darimont1996]:

- 实现 ( Achieve ) :  $P \Rightarrow \Diamond Q$  //如果将来某一时刻Q为真 ( 被满足 ) , 则目标实现
- 终止 ( Cease ) :  $P \Rightarrow \Diamond \neg Q$  //如果将来某一时刻Q为假 ( 被终止 ) , 则目标实现
- 保持 ( Maintain ) :  $P \Rightarrow \Box Q$  //将来任一时刻Q都为真, 则目标实现
- 避免 ( Avoid ) :  $P \Rightarrow \Box \neg Q$  //将来任一时刻Q都为假, 则目标实现
- 优化 ( Optimize ) : 最大化Maximize (目标功能) 或 最小化Minimize (目标功能)

## 2. 关系

除了核心的目标概念之外,目标模型的另一个核心要素是元素之间的关系,又称为链接。目标模型的链接有二个方面:①目标之间的关系,包括精化( Refinement )关系、阻碍( Obstruction )关系与冲突( Conflict )关系;②目标与其他模型元素之间的链接。这些链接构成了目标模型的结构基础。

### (1) 目标精化

一个高层次目标G可以精化为低层次目标 $\{G_1, G_2, \dots, G_n\}$ :

- 如果一系列子目标 $\{G_1, G_2, \dots, G_n\}$ 的完成有助于目标G的完成,那么G与 $\{G_1, G_2, \dots, G_n\}$ 之间就是AND 精化关系。此时任意两子目标 $G_i$ 与 $G_j$ 之间是互补的。
  - 如果更进一步,子目标 $\{G_1, G_2, \dots, G_n\}$ 的完成能够直接保证G的完成 $\{G_1, G_2, \dots, G_n\} = G$ ,那么G与 $\{G_1, G_2, \dots, G_n\}$ 之间就是完备 ( Complete ) AND 精化关系。
- 如果任一子目标 $G_i$ 都是G的替代方案,那么G与 $\{G_1, G_2, \dots, G_n\}$ 之间就是OR 精化关系。此时,任意两子目标 $G_i$ 与 $G_j$ 之间是互相替代的。

一个目标模型的精化关系示例如图5-8所示,它描述了火车管理系统的目标模型片段。在图5-8中,火车管理系统主要有三个高层的软目标 服务更多的旅客 ( ServeMorePassengers ), 尽可能降低成本 ( Costs , 类型Min ) 和安排运输 ( SafeTransport ) 。

对ServeMorePassengers的工作可以同时 ( AND精化 ) 从增加新班次 ( NewTracksAdded ) 和缩短班次间隔 ( TrainsMoreCloselySpaced ) 两个方面来实现。缩短班次间隔则可以通过 ( OR精化 ) 减少站点间运行时间 ( TimeBetweenStations , 类型Min ) 来实现。

降低成本的实现可以考虑降低新投资 ( DvlptCosts , 类型Min ) 或者 ( OR精化 ) 降低运营成本 ( OperationCosts , 类型Min ) 。

在实现安全运输的措施当中,有三个是必须同时 ( AND精化 ) 达到的:一、要保持安全的车距 ( WCS-DistBetweenTrains , 类型Maintain ) ;二、列车的速度要保持在轨道能够承受的范围内 ( TrackSegmentSpeedLimit , 类型Maintain ) ;三、列车不要进入已经关闭的站台 ( TrainEnteringClosedGate , 类型Avoid ) 。

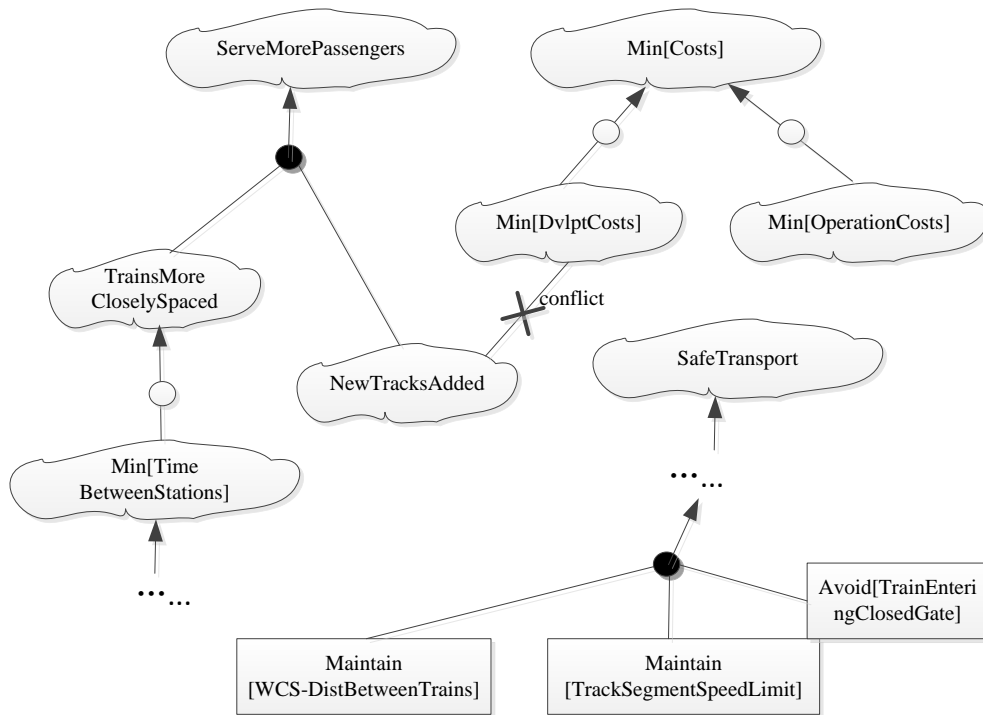


图5-8 目标模型精化关系示意图，修改自[Lamsweerde2001]

## (2) 目标阻碍

精化关系只考虑了能够使得高层目标顺利完成的理想子目标，但是实际情况中，很多具体细节情况会使得高层目标无法完成，这就是阻碍关系要描述的内容[Lamsweerde1998a]。

如果子目标 $O$ 的达成会使得高层目标 $G$ 失败 $O \models \neg G$ ，那么 $O$ 与 $G$ 的关系就是阻碍关系。

阻碍关系如图5-9所示，其中的AccelerationSentInTimeToTrain目标与AccelerationCommandNotSentInTimeToTrain目标之间就是阻碍关系。另外三个子目标NotSent、SendLate、SentToWrongTrain则是对AccelerationCommandNotSentInTimeToTrain的OR精化。

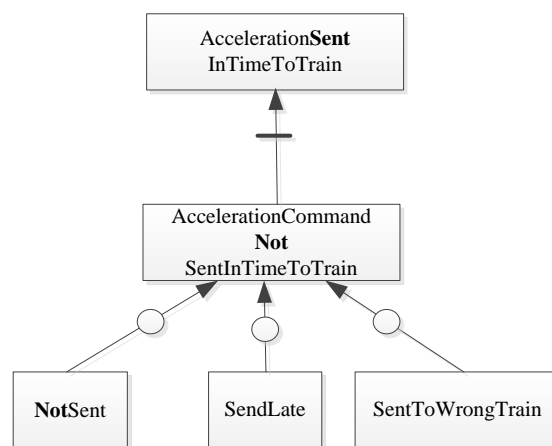


图5-9 目标模型阻碍关系示意图

对阻碍目标的考虑能让建立的软件解决方案更加坚固、完备[Alrajeh2012]，更好地应对异常事件与场景。但这要求描述目标 $G$ 及其实现时，不仅要靠考虑有助于目标 $G$ 实现的AND精化、OR精化等 $\{G1, G2, \dots, Gn\}$ 理想情况，还要专门考虑对阻碍目标 $O \models \neg G$ 的AND精化、OR精化的情况 $\{O1, O2, \dots, Om\}$ 。

### (3) 目标之间的支持 (Support)、冲突 (Conflict) 关系

精化关系与阻碍关系都是从单个目标的实现出发进行考虑的,前者是能够促使目标实现的分解与细化手段,后者是使得目标失败的阻碍情景。

支持、冲突关系是对多个目标之间关系的考虑。Support链接表示一个目标对其他目标的支持作用。也就是说一个目标的实现对其他目标的实现有促进作用。Conflict链接表达的含义正好相反,它是说一个目标的实现对其他目标的实现有阻碍作用。

支持与冲突关系可以是多元的,例如目标 $G_1, \dots, G_n$ 中,只要有 $G_i$ 成功,则必然会有 $G_j, j \neq i$ 难以成功,但更常见的是二元的。例如在图5-8中, NewTracksAdded与DvlptCosts两个目标就是二元的冲突关系。

在二元的支持、冲突关系中,也可以使用如下标识标记关系:

- ++ ( Make ) : 一个目标的成功可以直接保证另一个目标的成功;
- + ( Help ) : 一个目标的成功可以让另一个目标更容易成功;
- - ( Hurt ) : 一个目标的成功会使得另一个目标的成功更加困难;
- -- ( Break ) : 一个目标的成功会直接导致另一个目标的失败。

在KAOS方法中,会将支持关系处理为OR精化关系。

### (4) 目标与其他需求模型元素的链接

目标可以与其他模型元素建立关系,这些模型元素有主体 ( Agent )、场景 ( Scenario )、操作 ( Operation )、任务 ( Task )、资源 ( Resource )、UML元素等等。

如图5-10所示, Assignment链接用来连接目标和主体,表示为实现目标而需要参与主体。OR Assignment表示目标的实现可以交由多个主体中的一个来完成。AND Assignment表示目标的实现必须由多个主体一起共同完成。

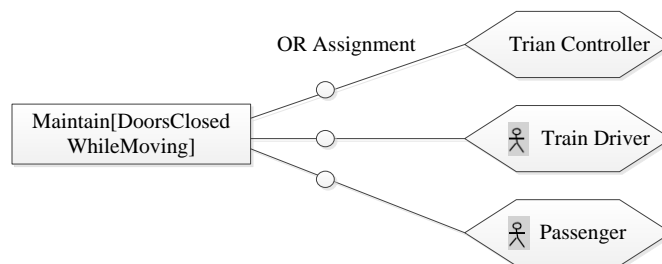


图5-10 目标与主体之间的关系示例

如图5-11所示, AND Operationalization链接和OR Operationalization链接用来连接目标和操作,描述了目标实现过程中需要执行的操作,  $G \models \{Spec(Op1), \dots, Spec(Opn)\}$ 。

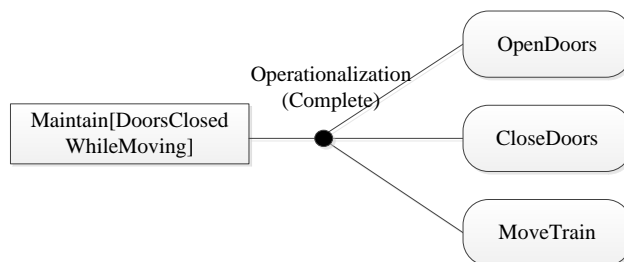


图5-11 目标与操作之间的关系示例

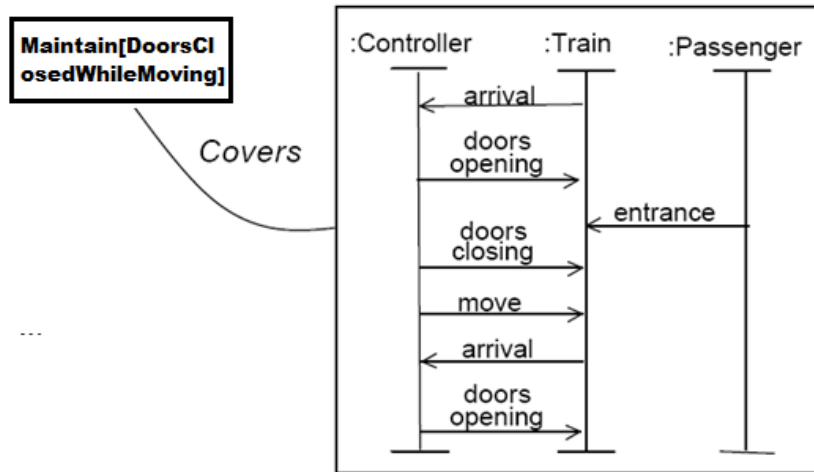


图5-12目标与场景之间的关系示例

如图5-12所示,Cover链接用于连接目标和场景,表示场景涉及所有行为都是目标所包含的操作的子集。Cover链接的使用一方面可以帮助需求获取行为的开展(针对分解后的子目标进行需求获取),另一方面也可以帮助进行获取需求的组织(将需求内容组织为场景(这一点将在下一章介绍),再将不同场景归类汇总到目标)。

如图5-13所示,Concerns链接用来描述目标与应用领域对象(数据资源)之间的关系。所有被目标定义为Concerns属性的应用领域对象,都应该与目标存在Concerns链接关系。

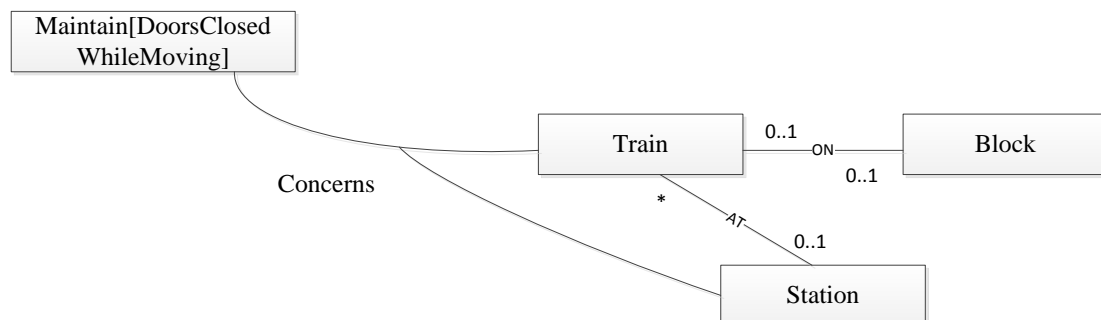


图5-13 目标与应用领域对象(数据资源)之间的关系

### 5.3.3. 目标分析过程

面向目标的需求工程方法是可以应用于需求工程各个阶段的,这一点通过目标与其他需求模型元素之间的关系能得到充分体现。只是本书只重点讲述目标模型在需求工程前期阶段的关键作用——目标分析,所以本书后面不会涉及目标模型与其他需求模型元素关系的建立细节,感兴趣的读者可以参考专门著作,在后续需求开发中可以自行尝试将用例/场景、UML模型等其他需求模型元素整合入目标模型。

目标分析是在需求工程前期阶段发生的,在非技术层次(目标的主体通常是涉众,尤其是不能只有待开发系统一个)上,建立目标模型,定义项目前景与范围的活动,如图5-14所示。



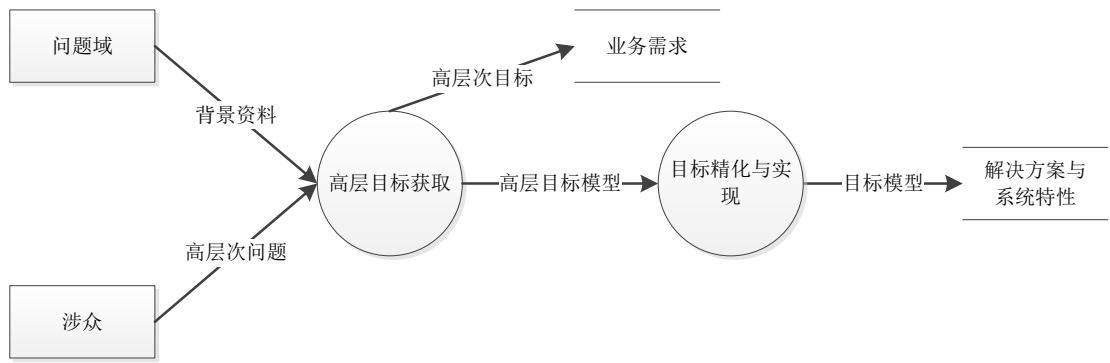


图5-14 目标分析过程

1. 高层目标的获取

对目标的识别并不是一件容易的工作。有些时候，涉众可以明确地提起一个目标，或者收集的材料会清晰地反映一个目标，但更多的时候，目标是需要努力去发现和获取的。

对系统的现状、背景和问题的分析往往能够发现高层次目标[Oshiro2003,Regev2005]，在这一点上与分析过程基本一致，只是目标分析需要将识别出来的目标、系统特性等组织为相互联系的目标模型，而不是默认为不同问题、特性是相关独立的。

例如，在面对5.2.1所述的连锁商店销售系统项目时，也需要分析其问题P1~P4，并建立最高层目标BR1~BR4（5.2.3节），这些目标可以被抽取定义为业务需求。

只是，目标分析并不简单地使用文本描述BR1~BR4，而是将它们组织为目标模型，如图5-15所示，它不仅更直观地表现了所有目标，更能够体现目标之间的关系。

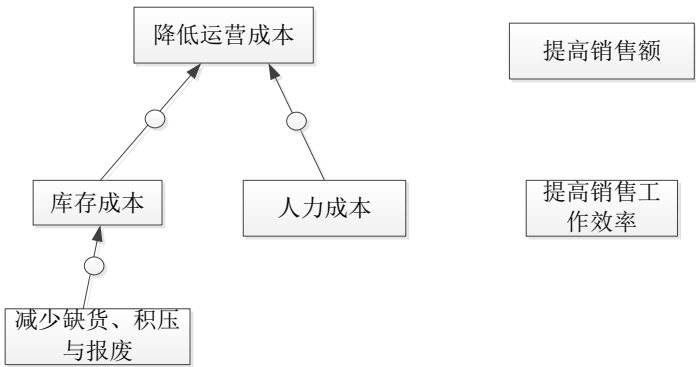


图5-15 连锁商店销售系统的高层目标模型

2. 目标精化

得到高层目标模型之后，还需要对高层目标进行精化，并根据精化结果完善目标模型。

对一个高层目标的精化需要获取高层目标的相关信息，并进行分析，发现其子目标。总结不同方法的目标精化过程，可以将目标精化工作分为下列几个方面：

（1） 获取对高层目标的描述

通过收集背景资料，运用面谈、原型等需求获取方法，可以收集对高层目标的描述信息。

常见的描述信息包括：

- 企业规章、政策；
- 任务描述；
- 业务过程；
- 场景描述（参见第7章）。

[Regev2005]建议尤其要注重对信息持有情况的获取，例如存储、加工、输出数据信息，因为这很常见，意味着相应Maintain目标的存在。例如在收集图5-15所示高层目标“减少缺货、积压与报废”的描述信息时，可以发现需要持有库存数据，包括销售数据、退货数据、批量入库数据和批量出库数据，那么就可以建立1个目标（库存数据）和4个AND精化子目标（销售数据、退货数据、批量入库数据和批量出库数据），如图5-16所示。

#### （2）从高层目标描述中发现AND精化关系

分析高层目标的描述信息，可以发现AND精化关系。

常见的AND精化关系场景有：

- 同一个目标有不同场景：每个Gi代表一个典型场景，任意Gi与Gj代表不同的场景；
- 完成目标有连续过程：每个Gi代表G完成过程中的一个状态，Gi、Gi+1代表两个连续的状态；
- 完成目标需要有多多个方面紧密配合：Gi与Gj紧密联系或互相支持；
- 目标有不同质量环境及表现：每个Gi代表不同质量要求下的G的完成。

例如，在分析图5-15所示高层目标的描述信息时，可以发现

- 销售工作包括销售处理和退货处理两个场景，那么就可以为目标“提高工作效率”建立2个AND精化子目标，如图5-16所示。

- “减少积压缺货、积压与报废”目标的实现需要先后执行四个步骤：持有库存数据→分析数据尽早发现可能的缺货、积压与报废→预先处置可能的缺货、积压与报废。于是就可以为“减少积压缺货、积压与报废”目标建立4个AND精化子目标（积压与另外2种情况的处置策略不同），如图5-16所示。

- “减少人力成本”的实现需要减少人员与提高人员工作效率两个方面的紧密配合，所以可以为目标“减少人力成本”建立2个AND精化子目标，如图5-16所示。

#### （3）从高层目标描述中发现“候选办法”发现OR精化关系

从高层目标描述中如果发现目标的实现可以有多种可以相互替代的“候选办法”，那么就可以建立相应的OR精化关系。

例如，目标“提高销售额”可以从“让更多的人来买”和“卖的更多”两个方面分别想办法，只有它们中任意一个能满足，就能保证“提高销售额”目标的满足，所以可以建立如图5-16所示的OR精化关系。

#### （4）考虑阻碍目标实现的情况（avoid目标）

除了要考理想情况下的目标实现方法之外，还要考虑阻碍目标的问题。

[Lamsweerde2000b, Alrajeh2012]建议从下列方面考虑阻碍目标：

- 考虑使得子目标失败的阻碍，添加针对子目标的阻碍目标。
- 考虑主体行为导致目标失败的阻碍，添加针对高层目标的阻碍目标。
- 考虑预防失败的情况，即为高层目标添加Avoid目标，避免阻碍的出现。

- 发现太理想化的目标，重新处理目标，缩小目标范围或为太理想化的目标添加约束。
- 发现因为应用领域不一致导致的目标阻碍，将应用领域转为一致消除阻碍现象。

例如，目标“利润下降”和“顾客流失”就是为预防失败而添加的Avoid目标，如图5-16所示

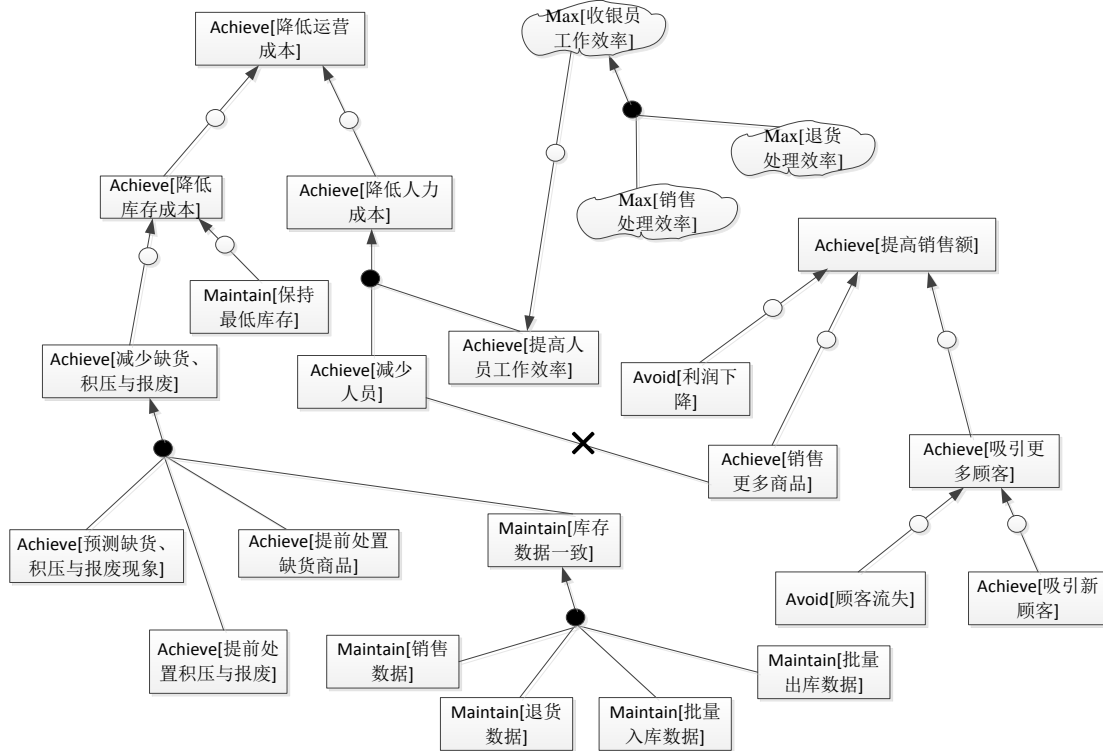


图5-16 连锁商店销售系统的完整目标模型

#### （5） 发现目标冲突关系

在各独立目标之间，要注意发现冲突关系。尤其要关注那些[Lamsweerde1998b]：①不同视点(View)之间的冲突；②正式定义之间相矛盾的冲突。例如“减少人员”与“销售更多商品”两个目标就是不同视点之间的冲突，如图5-16所示。

#### （6） 对高层目标问“How”，对低层目标问“Why”，完善层次结构

前面所述各个建立目标模型的事项中，都要坚持“对高层目标问‘怎么实现’，对低层目标问‘为什么需要’”的原则，完善层次结构。

每次考虑“怎么实现”一个目标时，要充分分析其参与的主体，尤其是涉众，了解他们各自对目标的观点和贡献往往能够帮助发现那些描述信息不足的目标的精化线索[Webster2005]。

在考虑“怎么实现”时，还要注意区分“目标”与“任务”的不同：①目标是一个条件（硬目标）或条件倾向（软目标），一个目标的完成有很多种方式，可以用不同的任务来实现；②任务是一个活动，是实际情况的反映不是条件倾向，任务完成是指活动结束而不是某个条件得到满足；③任务的执行有成功的、好的，也有不成功的、坏的，只有成功的、好的任务执行才能促使目标的满足。

需要专门强调的是，目标精化并不是一个自上至下分解的过程，而是一个不断获取、发现和分析的过程[Webster2005]。

### 3. 目标实现

这个阶段的主要任务是收集与目标相关的需求信息，讨论可能的候选解决方案，确定最终的解决方案。

目标实现的最主要工作有两个：①将最底层目标分配给主体；②设计实现最底层目标的操作（任务）。每个工作与问题分析活动的解决方案设计一样，由需求工程师给出候选方案，涉众做最后的选择决策。

如图5-17是连锁商店销售系统的目标模型片段的主体分配实现，其中“提前处置缺货商品”目标不需要软件系统的参与，完全由总经理在应用领域中自行解决。图5-18是相应的操作（任务）实现。

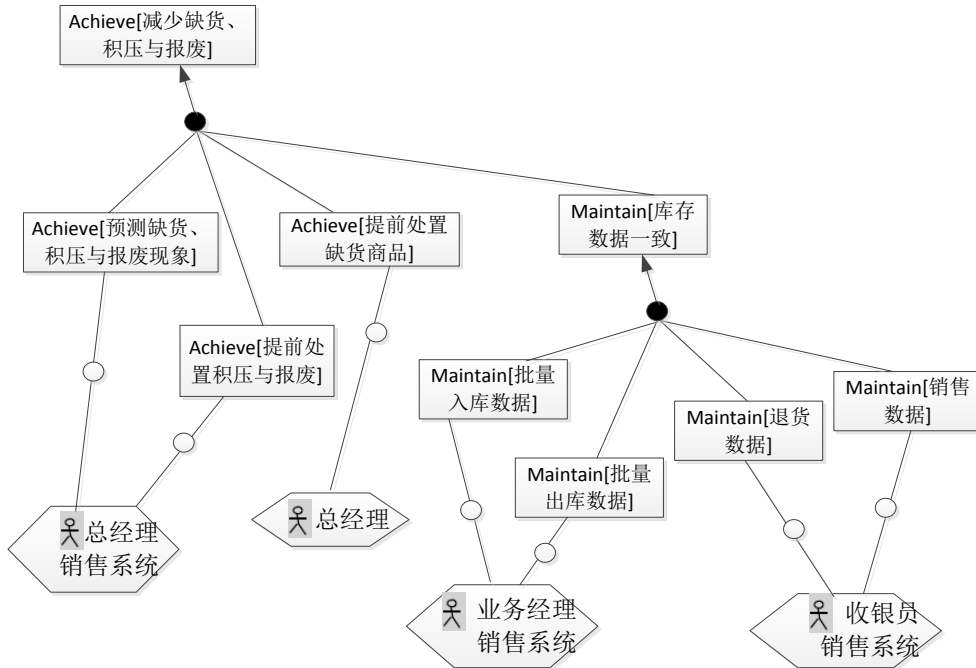


图5-17 连锁商店销售系统的目标模型片段的主体分配实现示例

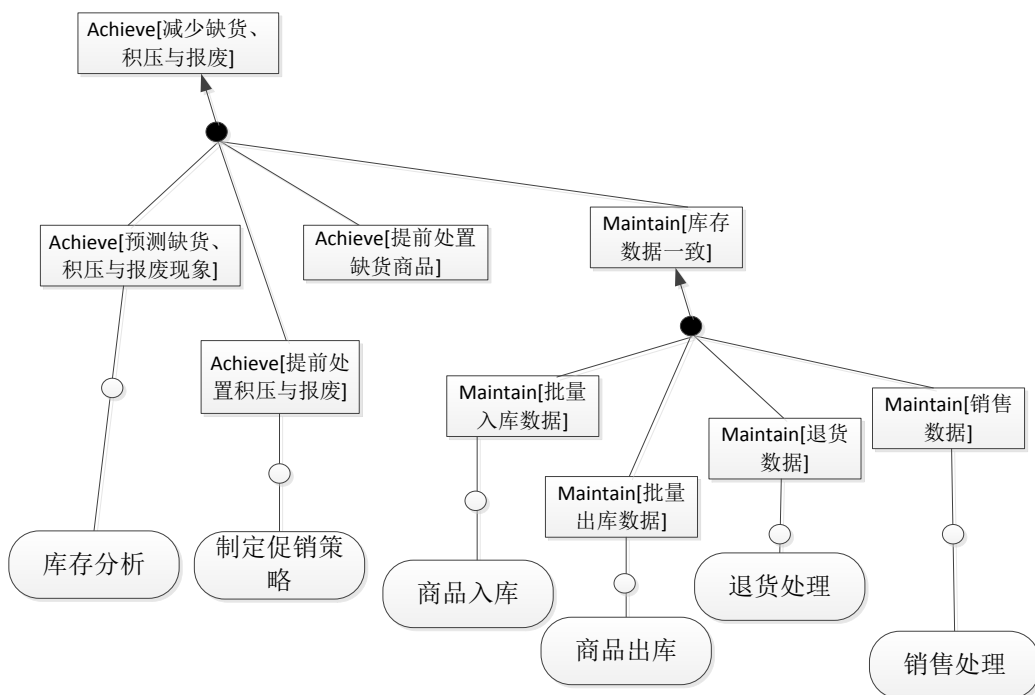


图5-18 连锁商店销售系统的目标模型片段的操作实现示例

在进行目标实现时,如果一个目标的责任都被分配给了涉众,待开发的软件系统没有介入,例如图5-17的“提前处置缺货商品”目标,那么就意味着该目标没有被纳入项目范围,但必须将其要记为项目假设与依赖,因为它虽不在项目范围却能够影响最终目标是否能成功满足。

目标模型本身更深入、准确地说明了项目的前景,主体分配中涉及的需要与待开发系统互动的主体限定了系统的边界,操作实现中出现的操作/任务说明了项目的功能特性,它们共同构成了项目的范围。

## 5.4. 非功能需求分析

### 5.4.1. 为什么需要非功能需求分析

功能需求与非功能需求 (NFRs, Non-Functional-Requirements) 都是需求的重要部分,但需求工程技术发展主要关注在功能需求的开发上,忽略了非功能需求,尤其是质量需求[Chung2000]。

但实践中因为非功能需求而导致系统故障甚至项目失败的案例已经很多了,大型复杂系统的开发中非功能需求已经超过功能需求成为决定项目成败的关键因素,这也是软件体系结构设计以非功能需求设计决策为工作重点的原因[Jansen2005, Clements2006]。

非功能需求的技术处理有两个方面[Mylopoulos1992]:面向产品的 (Product Oriented),主要目的是保证最终的产品能够符合非功能需求;面向过程的 (Process Oriented),在整个软件开发过程中始终关注非功能需求的处理,包括其获取、分析、设计、实现及验证。

近二十年中,面向过程的非功能需求处理技术得到了很大的发展,其中一个重要发现是:在需求工程的早期阶段应该获取和分析非功能需求,而在需求工程的后期阶段则关注于这些需求的完备性、一致性和自动化验证[Yu1997]。

### 5.4.2. 非功能需求分析的困难是什么?

传统需求工程技术更多地关注功能需求而不是非功能需求是因为:①传统软件规模没有现在的软件系统这么复杂,对非功能需求进行处理的要求没有现在这么突出;②非功能需求分析比功能需求分析要困难得多。

首先是获取非功能需求比较困难,因为涉众不会认识到并主动提起非功能需求。

其次是非功能需求分析困难,这主要体现在几个方面[Cysneiros2004a]:

- 非功能需求不集中,在系统中散布。功能需求是比较集中的,通常可以映射为单个或多个任务,也可以被实现为系统的单个单位(例如模块)。但非功能需求却是分散的,一个非功能需求可能会被分布在很多甚至大部分系统任务中,例如易用性、性能、法律约束等。不集中就意味着难以界定它们的边界,也就难以描述和处理。

- 非功能需求不独立,依赖于功能需求。功能需求是可以相对独立的,没有非功能需求的存在,功能需求仍然是完整的。但没有功能需求,非功能需求就没有了存在的必要,因为非功能需求是对功能需求的约束,不具有独立存在的价值。这意味着处理非功能需求时很难脱离功能需求进行独立考虑,难度自然较大。

- 作为非功能需求主体的质量需求比较复杂,需要被分解为特征-子特征的层次结构(参见第2章的质量模型)进行处理,这使得非功能需求的处理技术要也比较复杂,能够处理层次结构。

● 非功能需求不独立，相互冲突、依赖。不同的功能需求是相对独立的，它们之间的联系是受限的。但不同非功能需求之间却可能存在着紧密的联系，例如性能与安全就是一对冲突的非功能需求——安全性越高性能越低，反之亦然。所以处理非功能需求时要同时折中、权衡的因素远多于单个非功能需求的内容，难度自然也高得多。

总的来说，非功能需求分析的根本困难是不独立性，而传统上获取与分析非功能需求时，总默认为每一个非功能需求分析是独立的。非功能需求分析需要有一种能够将独立NFR及其对外依赖关系综合考虑的技术。

### 5.4.3. 使用面向目标的方法分析非功能需求

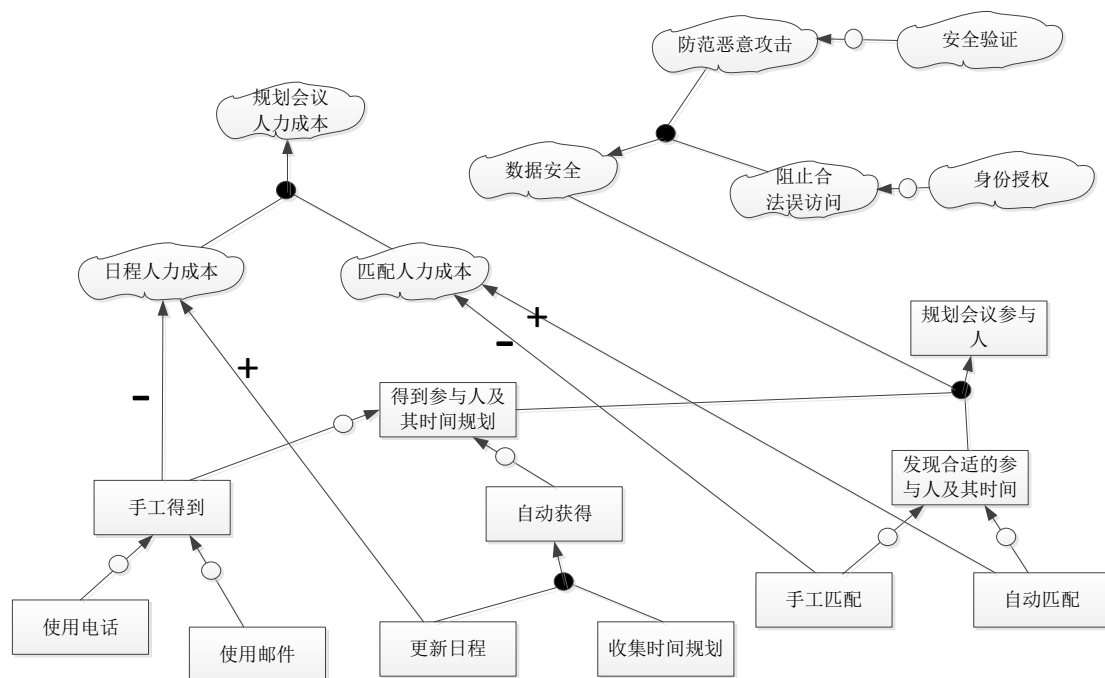
#### 1. 概述

Mylopoulos、Chung等人的工作 [Mylopoulos1992, Chung1993, Chung1995]使人们认识到目标模型及面向目标的方法是进行非功能需求分析的有效手段，并建立了NFR方法[Chung2000, Cysneiros2004b]。

如图5-19所示，NFR方法将非功能需求建模为目标：

● 在早期信息较为抽象和不充分时建模为软目标，后期添加目标满足标准修正为硬目标，这样符合需求开发的基本过程。

- 让非功能目标支持、阻碍或精化功能目标，描述非功能需求对功能需求的依赖关系。
- 让单个非功能目标依赖多个不同的功能目标，描述非功能需求的散布特性。
- 使用支持、阻碍（冲突）精化描述不同非功能目标之间的折中与冲突关系。
- 依赖质量模型，高层非功能目标精化为子非功能目标，描述非功能需求的层次结构关系。



很明显，目标模型适合于非功能需求分析。但需求工程中的非功能需求处理并不仅仅是分析，还需要获取、文档化、验证等其他工作。

所以，NFR方法是一个以目标模型为基础同时包含获取、与其他需求模型的整合及分析、规格化、验证等其他需求开发活动的工作框架。

限于篇幅，本书只着重讲述获取非功能需求及建立目标模型的简单过程，更详细的内容请参考专门著作 [Chung2000, Cysneiros2004b]。

获取非功能需求及建立目标模型的简单过程主要有三个步骤，分别如下：

## 2. 依赖功能需求识别、获取非功能需求目标

非功能需求难以获取是因为涉众意识不到非功能需求的存在及其重要性，所以无法明确地表达出来，但这并不意味着涉众完全没有提及非功能需求。事实上，在涉众表达功能需求时，会附带很多“关注”因素，这些因素往往就是高层次的非功能目标。

例如，在涉众描述一个功能时，

- 如果强调该功能一定要“好用”，就意味着存在易用性目标（Usability Goal）。
- 如果强调某个任务过程要小心谨慎，就意味着存在可靠性目标（Reliability Goal）。
- 如果强调用户比较多、数据比较多或计算比较复杂，就意味着可能有性能目标（Performance Goal）。
- 如果强调要容易调整、修改、增加新功能，就意味着存在可维护性目标（Maintainability Goal）。
- 如果强调数据的敏感性（比如A用户不能看到B用户的数据），就意味着存在安全目标（Safety Goal）。
- 如果强调功能关联着技术环境（比如强调使用浏览器...，使用其他软件系统...），就意味着可能存在可移植性目标（Portability Goal）。
- .....

所以NFR方法依赖于对功能需求的描述来获取非功能需求目标，它希望首先按照目标分析的过程获取、分析并描述功能需求目标，然后分析功能需求目标描述中的“关注”因素，发现依赖于功能需求目标的非功能需求目标。

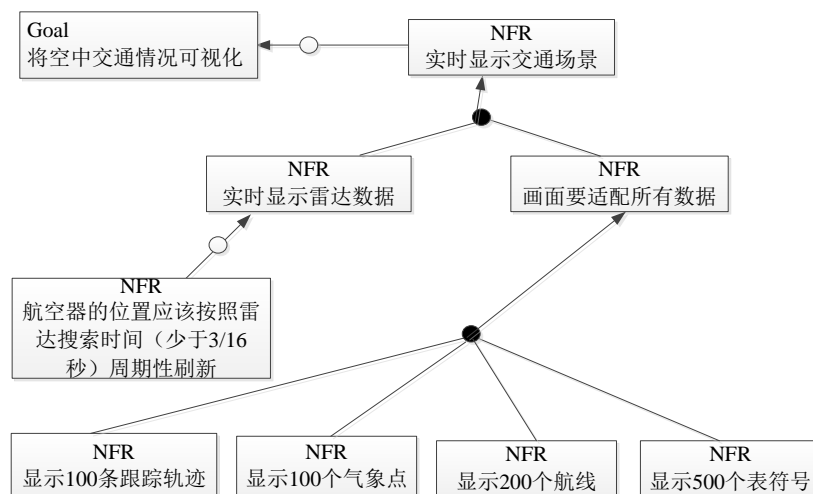


图5-20 NFR识别、获取非功能需求目标示例

如图5-20所示，功能目标“将控制交通情况可视化”中的文字“可视化”可能意味着HCI有质量要求，

仔细描述后可以发现，可视化意味两件事情：①将众多复杂数据清晰地显示在一个屏幕上；②实时刷新动态数据。于是就可以发现相应的非功能需求目标，分析后可以建立如图5-20所示的目标模型。

### 3. 根据非功能需求层次结构，精化非功能需求目标

有些时候获取到的非功能需求目标是比较简单的，例如图5-20所示的“实时显示雷达数据”。但更多的时候获取到的非功能需求是比较复杂、抽象的，典型情况是只获取到了质量模型的特征级质量需求。这时可以依据质量模型或其他非功能需求层次结构[Chung2000]描述，将特征级的质量需求目标精化为子特征级的质量需求子目标[Doerr2005]。例如图5-19中，将安全目标“数据安全”精化为“防范恶意攻击”、“阻止合法误访问”两个更准确的子目标。

### 4. 量化底层非功能需求目标的验收标准

在高层NFR目标模型中，因为复杂、抽象等原因，主要使用软目标描述非功能需求目标。但最终的规格化及验证需要可以验收的非功能需求目标。所以，到了NFR分析后期，要为NFR目标模型中的底层非功能需求目标确定量化验收标准，如图5-20所示。

## 5.5. 业务过程分析

问题分析方法将每一个问题、目标、特性等都看作是相互独立的，所以只能完成简单系统的前景与范围定义任务。目标分析能够表达问题、目标、特性之间的依赖关系，所以能够完成较为复杂系统的前景与范围定义任务。但是仍有些系统的目标、特性（尤其是涉众任务）之间存在着极其紧密的关系，远超过目标模型链接关系的表述能力，典型情况是系统中存在着复杂的业务过程，这时就需要使用业务过程模型，进行业务过程分析。

业务过程的复杂性是随着1980s后期及1990s早期业务过程再造BPR ( Business Process Reengineering ) 的广泛应用而被人们认识到的，人们开始寻找技术手段来描述企业的业务过程。当时发现DFD有流程的描述能力，但又有很多局限性。为此，人们依据DFD“流”处理的思想，建立了专门用来描述组织业务流程的业务过程模型BPM ( Business Process Model )，并发展至今，需要了解其详细情况的读者请参见[OMGBPMN]。

BPM的思想被面向对象的主流技术UML吸收和采纳，建立了活动图 ( Activity Diagram )。活动图是描述业务过程和对象行为的模型，它以“流”（控制流和数据流）处理为侧重点描述过程与行为，以“令牌 ( Token )”平衡为手段保证过程与行为中的复杂并发协同现象。

### 5.5.1. 活动图

#### 1. 常用节点与流

活动图中比较常用的节点如图5-21所示：

- 动作 ( Action ) 节点：每个动作节点都是一个可执行的任务或行为。动作的执行会产生一次过程处理或数据转换。在业务过程分析中，每个动作都是业务过程中的一个重要步骤，通常是涉众的一个任务和多个任务的集成。

- 控制 ( Control ) 节点：控制节点用来协同活动图中其他节点之间的流转。



- 初始 ( Initial ) 节点：活动开始流转的地方。
- 活动结束 ( Final ) 节点：活动停止流转地方。
- 决策 ( Decision ) 节点：有一个流入流和多个流出流，决策节点在多个流出流中决策选择一个。
- 合并 ( Merge ) 节点：有多个流入流和一个流出流，用于将多个可能的流入流转换为一个的流出流。
- 分叉 ( Fork ) 节点：将一个流入流分割为多个并发的流出流。
- 汇合 ( Join ) 节点：汇合多个并发的流入流转换为一个流出流。
- 对象 ( Object ) 节点：是指一个特定对象的实例，通常在活动图的特定时间点出现和使用。
  - 对象节点可以是动作节点的输入和输出，此时对象节点被称为动作的输入Pin或输出Pin。

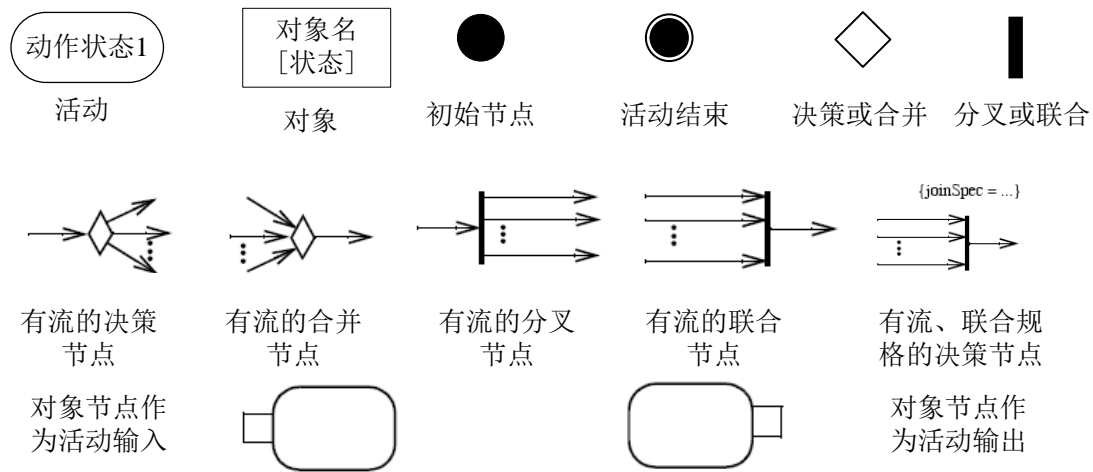


图5-21 活动图常用节点

活动图中的流有两种，如图5-22所示：

- 控制流 ( Control Flow )：用于在前一个活动节点完成后启动下一个活动节点的活动边 ( Activity Edge )。
- 对象流 ( Object Flow )：一个能够传递数据和对象的活动边。

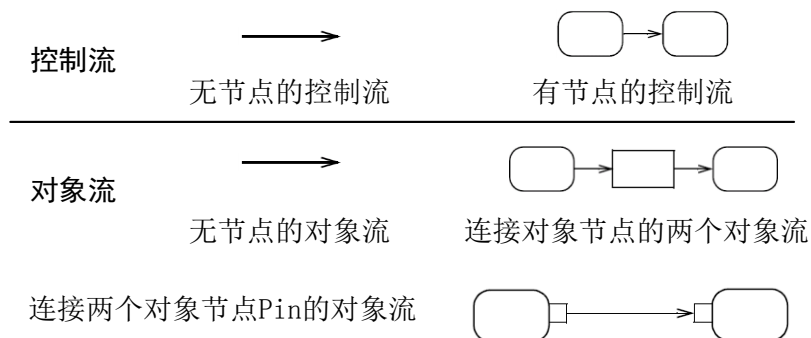


图5-22 活动图的流

一个使用常用节点与流表达的活动图如图5-23所示。

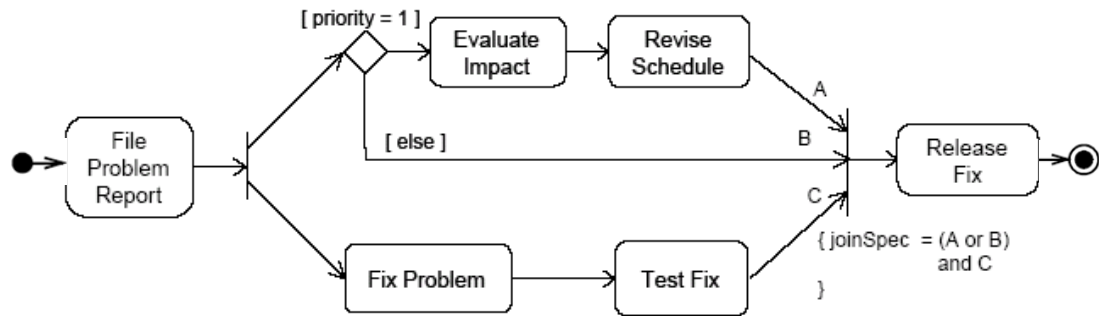


图5-23 使用常用节点与流的活动图示例

## 2. 令牌平衡

活动图的一个关键是令牌平衡，它用于验证活动图流转是否正确。如果一个活动图的流转出现了令牌缺失（有些节点得不到令牌）、令牌丢失（有些令牌没有被传递到结束就无法继续传递了）或令牌冗余（有多余的令牌没有被处理），那么就意味着活动图的业务流转是有问题的。

典型的令牌流转是：

- 一个控制流传递一个令牌。
- 一个对象流传递一个令牌。
- 决策节点从流入流接收到一个令牌后传递给多个流出流中的一个。
- 合并节点从多个流入流中只要接收到一个令牌就传递给流出流。
- 分叉节点从流入流接收一个令牌后，复制出多份令牌，给每个流出流都传递一个令牌
- 汇合节点从每个流入流都接收一个令牌，或者流入流的令牌接收情况符合汇合格，就向流出流传递一个令牌。

一个令牌不平衡的活动图如图5-24所示。在Receive Order动作节点之后的决策节点会丢失令牌，当业务不符合“order accepted”条件时，令牌就没有继续传递的路径了，出现丢失。在Fill Order动作节点之后会出现令牌缺少，因为Fill Order只接收了一个令牌，自然无法同时传递给Ship Order和Send Invoice两个动作节点，因为这需要至少两个令牌。在Close Order动作节点的流入流部分又会产生令牌多余，Close Order只需要一个令牌，但是却能接收到两个令牌，自然会有一个多余的。一个正确的活动图应该如图5-25所示。

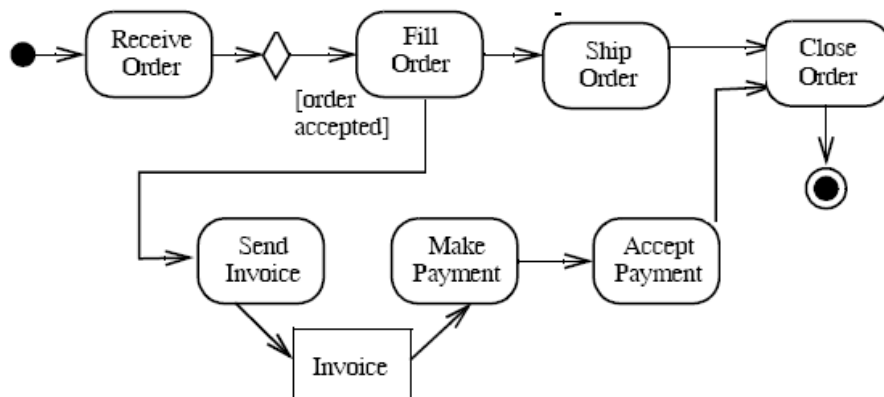


图5-24 令牌不平衡的活动图示例

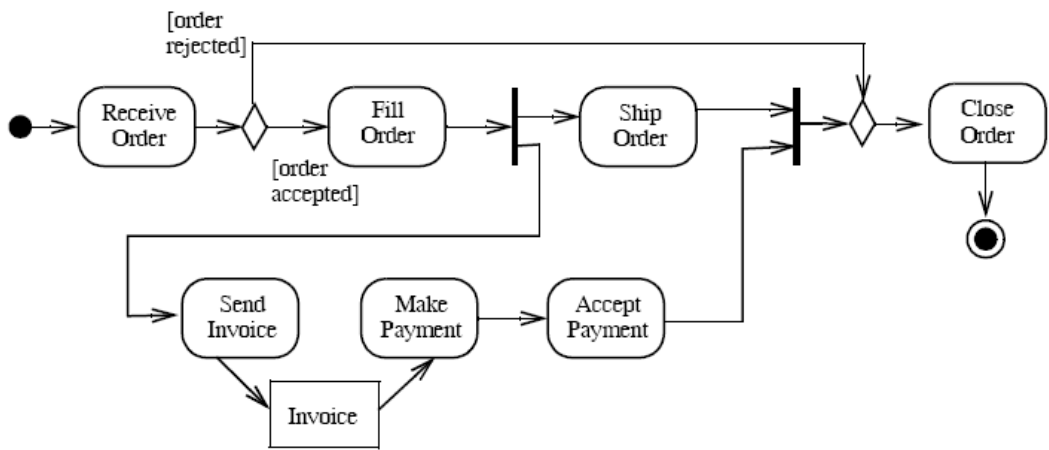


图5-25 令牌平衡的活动图示例

为了既能保持令牌平衡，又能描述复杂的实际业务，活动图引入了如图5-26所示几个复杂元素：

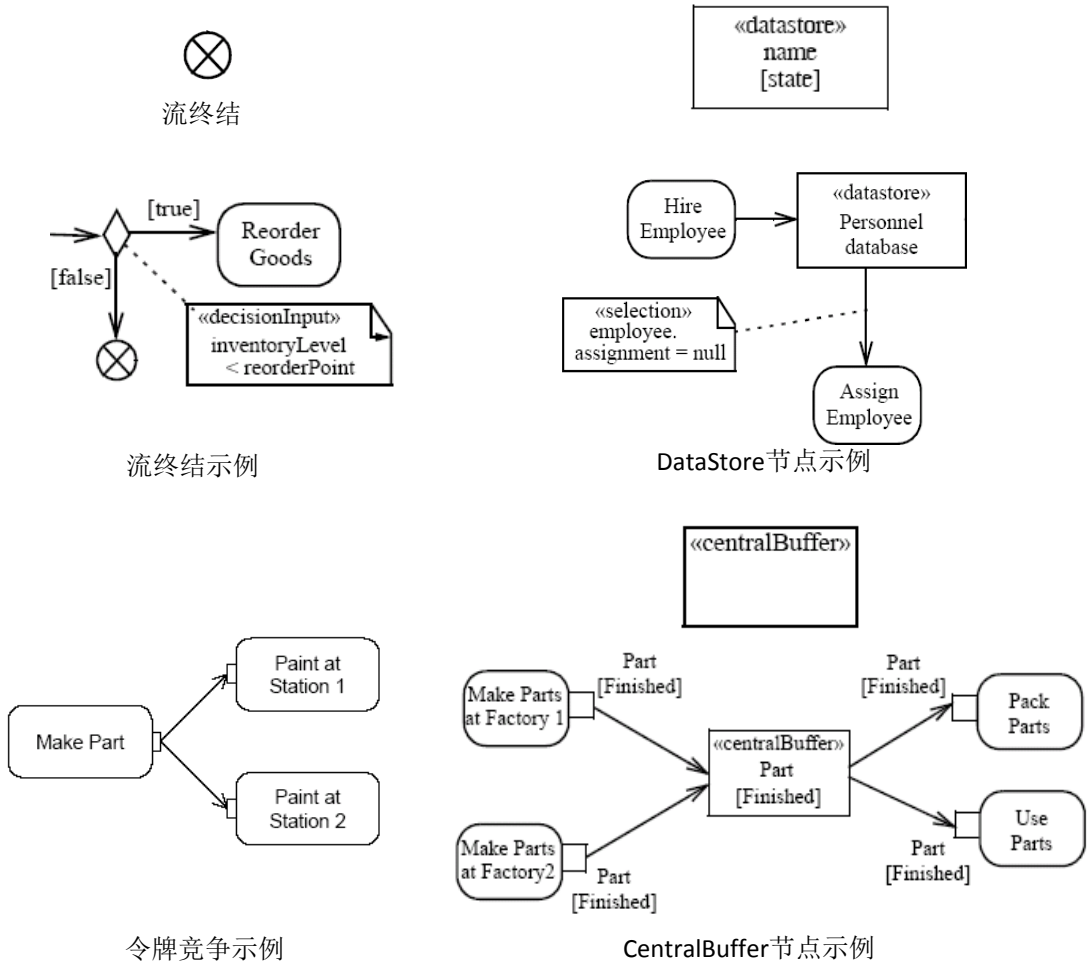


图5-26 活动图的复杂元素

● 流终结（Flow Final）：理论上，一个令牌只能从初始节点开始向后传递，一直传递到活动结束节点，代表业务过程成功结束，否则就是令牌不平衡，业务过程失败。但在实际业务中，的确存在业务过程失败的情况。例如，在学校的人才培养过程中，入学是初始节点，毕业是活动结束节点，理论上一个入学的学生应该成功毕业，但实际情况中也会有中途退学的现象。为了解决业务过程失败的实际情况，活动图引入

了流终结，传递到流终结的并不是一个成功结束的令牌，但却不会继续传递。流终结不等同于令牌丢失，因为令牌丢失意味着业务过程的流转设计不周密，被丢失的令牌没有被处理，就好比一个学生没有得到任何处理就不见了而且也没有人能够发现。但业务过程中的流终结是有意设计的，流终结也是一个行为，意味着学校意识到了学生不能毕业的失败现象，并会安排专人处理。流终结可以保证业务过程失败时也能令牌平衡

- **CentralBuffer节点**：是一个对象节点，管理多个对象输入和多个对象输出。CentralBuffer节点用于描述实际业务过程中的缓冲区或仓库，会将输入的令牌暂存排队，并最终随一个对象输出向后传递。

CentralBuffer不影响令牌平衡。

- **DataStore节点**：是一种特殊的CentralBuffer节点，它暂存的是可以一次生成无数次使用的资源（例如数据库数据），所以一个令牌传递到DataStore后就像是消失了一样不再继续传递，同时DataStore又可以在需要时无限制地生成新的令牌向后传递。DataStore对令牌平衡的计算会有影响，需要重视。

- **令牌竞争**：在实际业务过程中，有些业务流转的目的地是随机的，例如如果有一个课程同时开设了两个班，那么学生就可以随机地选择一个班上课，但不会同时在两个班上课，活动图就将这种情况描述为竞争。令牌竞争不影响令牌平衡，多个随机目的地中只有一个能够得到令牌。

### 3. 信号与事件机制

为了描述异步协同的业务过程，活动图引入了信号与事件机制，如图5-27所示：

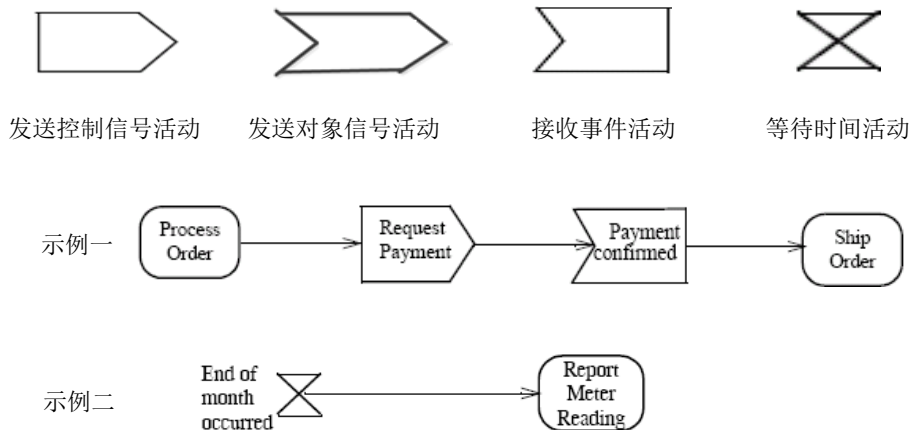


图5-27 活动图的信号与事件机制

- **发送信号动作（Send Signal Action）**：动作从输入中创建一个信号实例，并传送给目的地。
  - 发送的信号可以仅是控制信号，也可以是带有数据的对象信号。
  - 发送信号是异步协同，所以发送信号时并不向目的地传递令牌，发送信号动作的令牌仍然保留在本地，继续向后续节点传递。
- **接收事件动作（Accept Event Action）**：动作等待一个符合特定条件的事件的发生。
  - 有两种类型的接收事件动作：1等待接收发送信号动作发送的信号；2等待时间事件的发生，时间事件可以是绝对时间事件（例如1949年10月1号），也可以是相对时间事件（例如10分钟后）。
  - 时间事件会产生新的令牌，但信号接收事件不会产生令牌。也就是说，如果一个信号接收事件动作没有收到令牌时，即使有信号发送也无法接收。一个有令牌的信号接收事件接收了一

个信号时，令牌并不会增多，还是一个，传递给后续节点。

#### 4. 异常与中断机制

为了描述实际业务过程中的异常情况，活动图引入了异常与中断机制，如图5-28所示：

- 异常处理（Exception Handler）：如果保护节点出现了特定类型的异常，就自动执行异常处理动作。如果异常没有被捕获，就会被传递给上一层被封装的保护节点。如果异常一直被传递到最顶层都没有被捕获，系统就可能会出现未预料的行为。

- 可中断活动区（Interruptible Activity Region）：是一个支持中断其中令牌的活动组。只要区域内存在令牌，该区域就可以接收中断事件。只要接收到中断事件，区域内的所有令牌都会终止，并沿着中断边向外传递一个令牌。

#### 5. 分区

活动图的节点与边可以依据特定的目的归类为活动组（Activity Group），一个节点或边可以被同时归类为多个活动组。

活动分区（Activity Partition）是一种最常用的活动组形式，它通常按照组织单位或涉及角色将节点、边划分到不同分区。

分区可以有层次嵌套结构，也开始是多维的，其图示可以使用泳道（Swimlane）也可以直接标记在节点上，如图5-29、5-30、5-31所示。

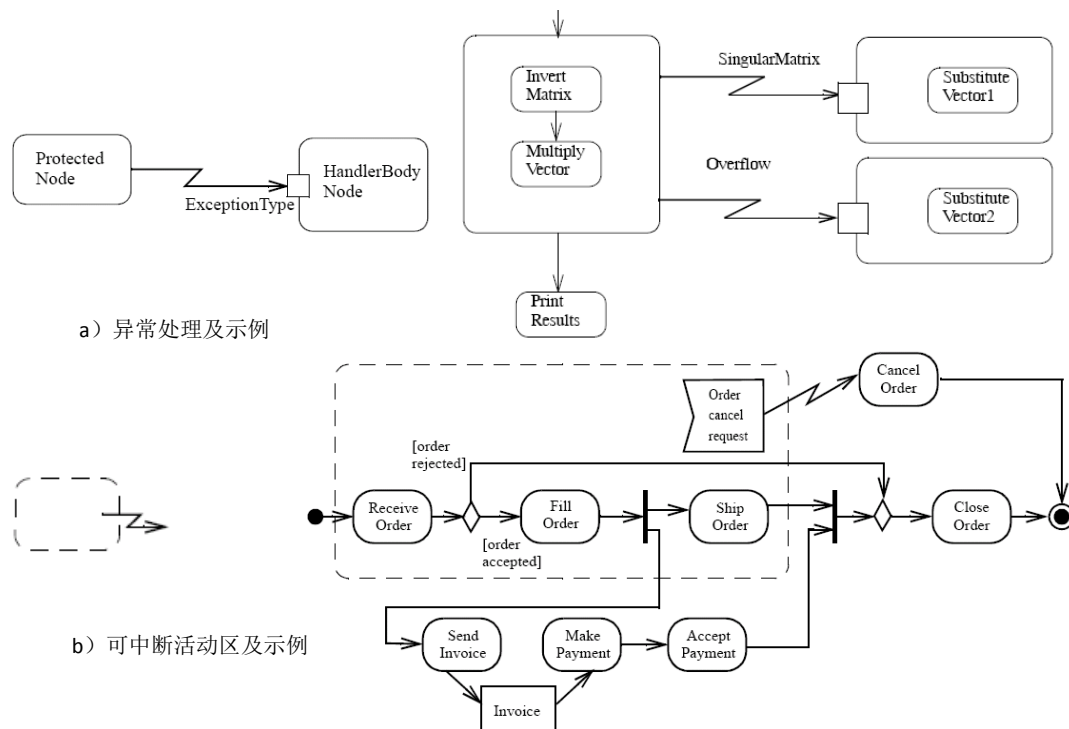


图5-28活动图的异常与中断机制

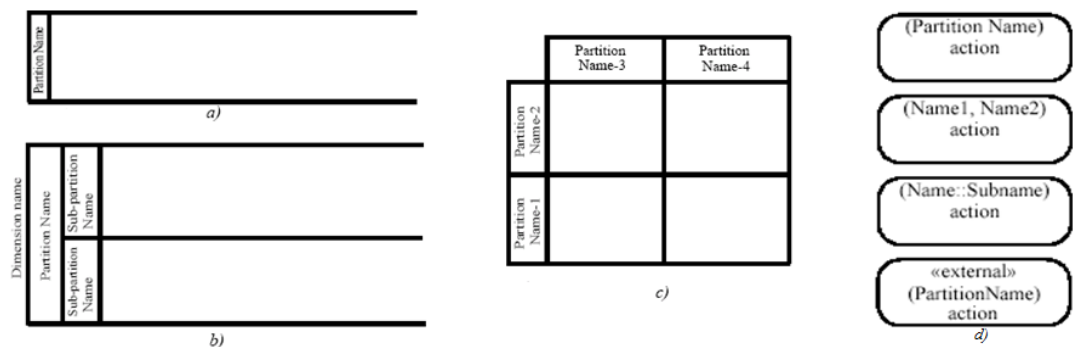


图5-29 分区图示

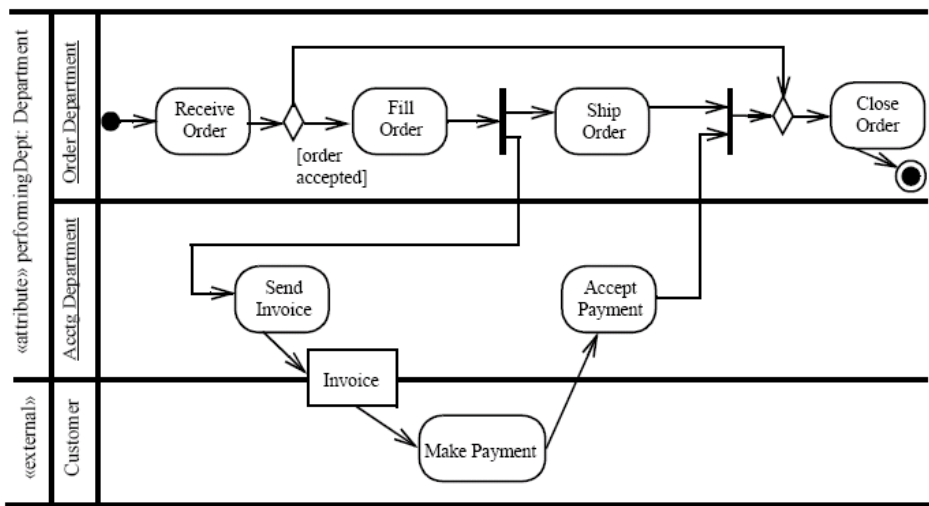


图5-30 使用泳道的活动图分区示例

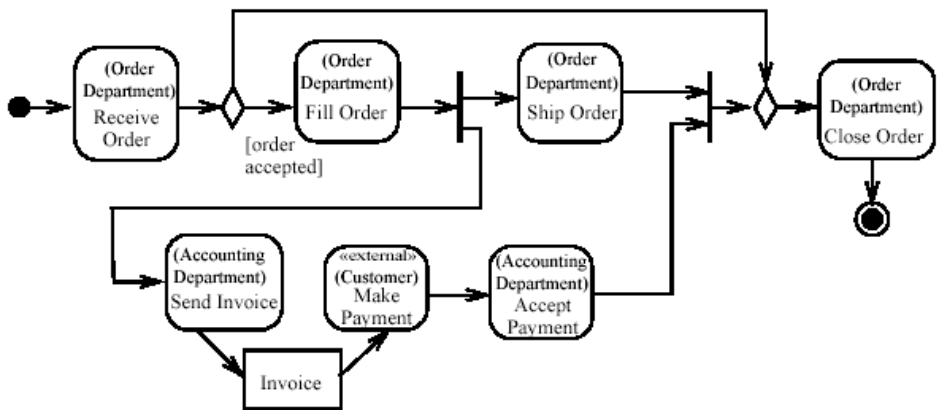


图5-31 使用标记的活动图分区示例

6. 层次结构与图简化元素

为了防止单个活动图过于复杂，难以描述和阅读，活动图使用“活动”（Activity）元素来建立层次性的活动图结构，将单个复杂业务过程分解为多个更简洁的活动图。

活动使用控制和数据流模型，描述包含动作元素的下级行为的协同执行。简单地说，活动就是一个可以带有输入/输出参数的子活动图，如图5-32所示。

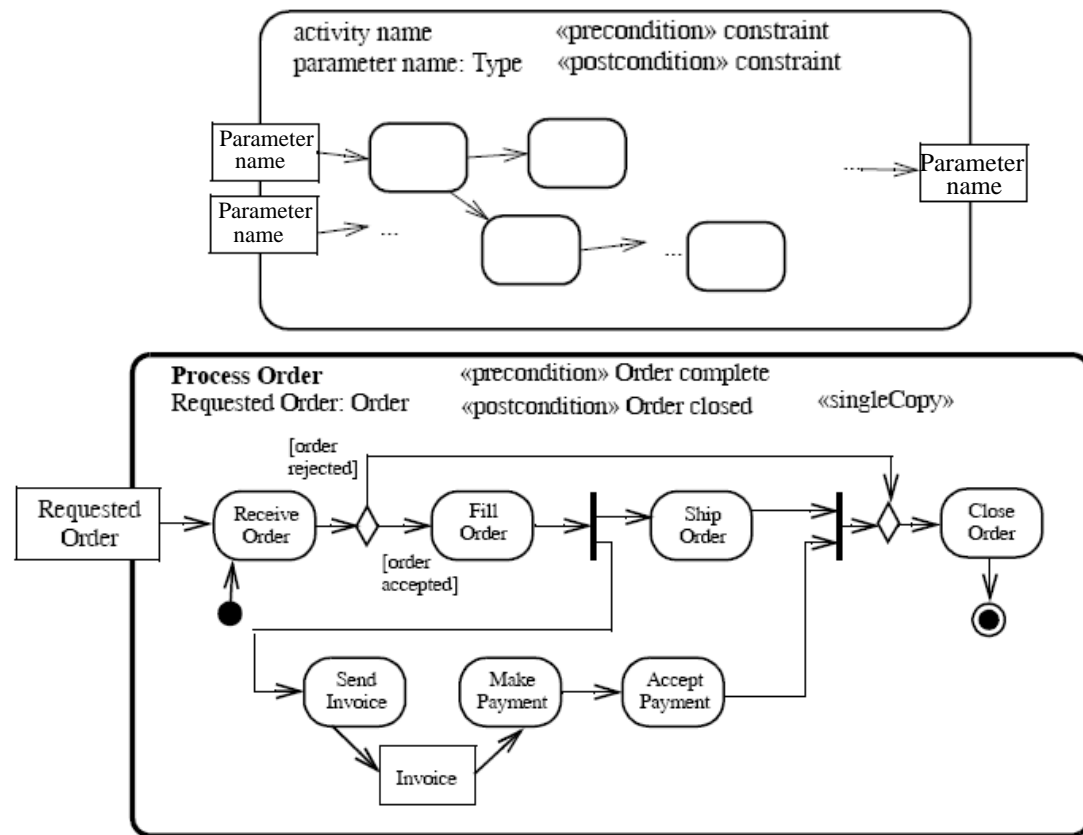
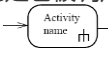


图5-32 活动示意

如图5-32所示，活动可以有输入/输出参数，也可以没有，可以标记前置/后置条件，也可以不标记，但必须有一个活动名称，这也是它被调用时的标识。

在活动图中，可以使用  调用一个活动，如图5-33所示，调用了活动Provide Required Part。被调用的活动Provide Required Part的定义则如图5-34所示。图5-33中活动Provide Required Part的流入流对应着图5-34中的初始节点。图5-34中的活动结束节点对应着图5-33中活动Provide Required Part的流出流。

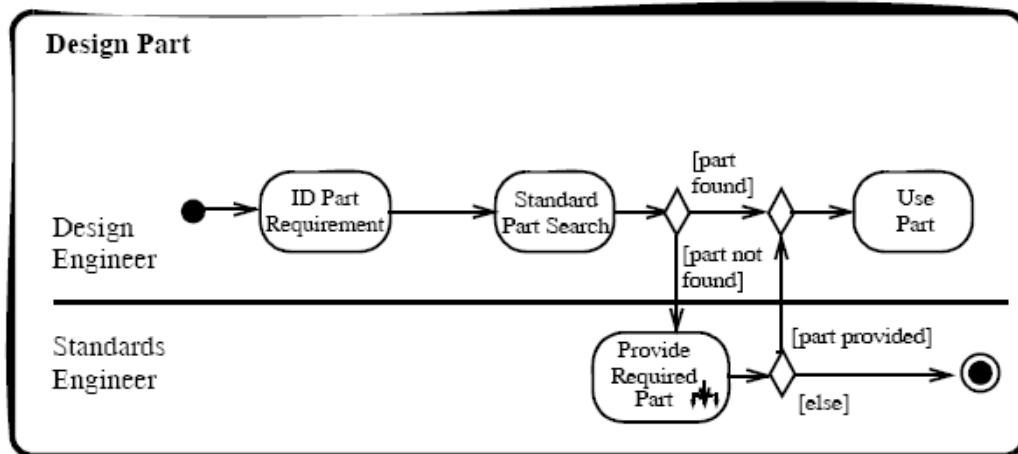


图5-33调用活动示例——调用部分

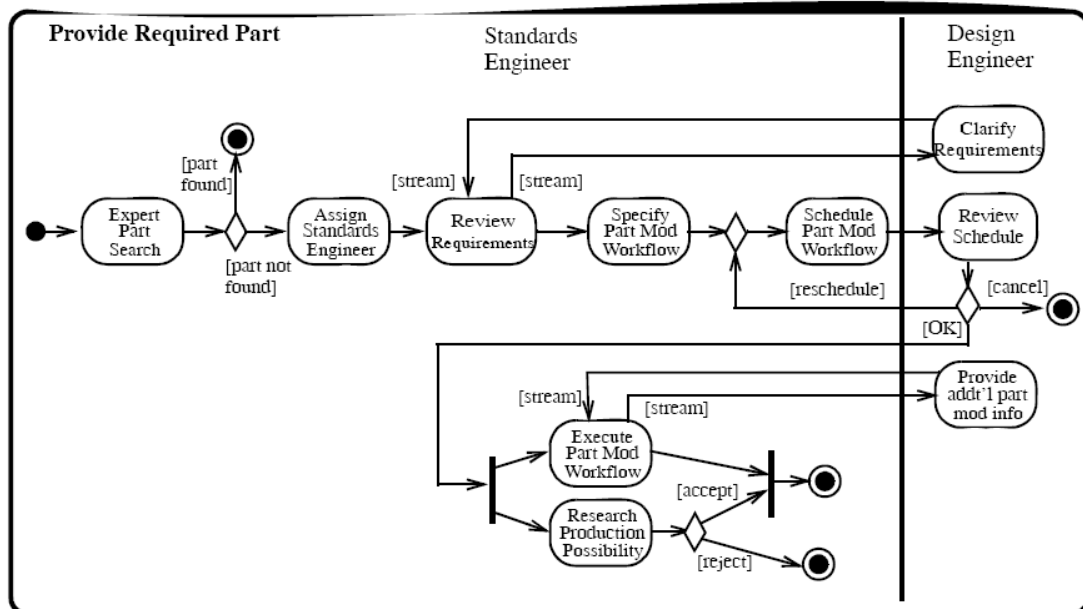


图5-34调用活动示例——被调用部分

为了不让复杂活动图中表示控制流和数据流的箭头交叉混乱,活动图使用了连接件(Connector)元素,如图5-35所示。连接件是成对出现的,就像传送门一样,将从一个连接件端流入的流传送到另一个连接件端流出。

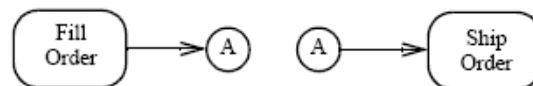


图5-35 连接件示例

## 7. 更多的复杂活动图元素

前面所述的各种活动图元素只是活动图的部分元素,是业务过程描述时较为常用的元素,更多的活动图元素请参见[OMGUMLSuperstructure]。

### 5.5.2. 使用活动图进行业务过程分析



在获取到业务过程描述之后，可以按照如下步骤使用活动图进行业务过程分析：

（1）确定活动图的上下文环境。活动图是对业务过程的描述，建立活动图首先要确定业务过程的处理界限。

（2）分析业务过程中的主要处理步骤的行为。复杂的业务过程总是会按照一定的步骤顺序执行，要发现并列举这些主要的处理步骤。分析清楚这些步骤之间的先后衔接顺序。

（3）分析业务过程中的主要数据流。业务过程往往会利用数据的传递作为工作衔接的重要手段，要发现这些具有重要意义的被传递数据，并分析这些数据在业务过程中不同步骤下的状态差异。

（4）识别参与者。在复杂的业务过程当中，识别主要的流程参与者。这些参与者将分享业务过程中的各项职责与行为，构成活动图中的不同泳道。在业务过程比较简单时，可以不进行参与者识别的工作。

（5）进行职责分配，将业务过程的处理步骤划分到不同的泳道，并将处理步骤和数据流的传递组织起来，建立活动图。

（6）添加活动图的详细信息，完善活动图描述，尤其要注意下列工作：

- 分析不同动作之间的协同是同步还是异步，同步使用控制流和数据流，异步使用信号与事件。
- 分析是否存在业务过程失败场景，添加流终结节点。
- 分析是否存在较为复杂的行为，为其建立活动。
- 分析业务过程中是否有异常，补充异常处理。
- 始终要检查令牌平衡，修正不平衡的节点。

例如，依据如图5-36的描述，可以按照下列步骤建立如图5-37所示活动图：

（1）整个业务过程以卖家发布商品开始，到交易完成结束。卖家与买家注册的问题不属于业务过程范围，可以不予处理。卖家和买家中途取消交易的申请不在业务过程范围内，可以另行描述交易取消申请过程。派送状态查询也可以描述为其他业务过程，不纳入交易业务过程范围内。

（2）可以发现下列几个步骤行为：（卖家）发布商品、（买家）选择商品、（买家）预付款、（中介商）预付款监管、（卖家）通知准备发货、（卖家）选择物流公司、（物流公司）派送物品、（物流公司）更新状态、（卖/买家）查询状态、买家（验收）、中介商（付款）、（审核人员）交易取消。

（3）需要传递的数据流有：商品信息，卖家发布供买家查看；预付款，买家交付中介商；商品派送状态，物流公司更新供卖家、买家查询；货款，中介商交付卖家；货物，卖家交付物流公司，再交付给买家。

（4）参与者信息上两步已有描述。

（5）按照（2）（3）的描述内容建立泳道框架，分配动作，初步连接控制流和数据流。

- 某电子商务厂商准备开发一个网上旧货交易系统，主要功能是完成二手物品交易。其预期的基本交易流程是：
- 1. 卖家注册，发布二手商品；
  - 2. 买家注册，查询待交易的二手商品，选中想要购买的物品；
  - 3. 如果买家选中某二手商品，就进行预付款；
    - 1. 预付款暂时由系统（中介商）监管；
  - 4. 如果卖家选中某二手商品，系统通知卖家准备发货；
  - 5. 接到准备发货通知后，卖家选择物流公司；
  - 6. 如果买家已经完成预付款，物流公司就开始派送物品；
  - 7. 派送过程中，物流公司要持续更新派送状态（已送达地点、时间与联系人）；
  - 8. 在派送过程中，买家和卖家都可以实时查询派送状态。
  - 9. 买家接受到商品后，进行验证和确认，如果验证通过，系统会将预付款最终转账给卖家银行帐号，成功完成交易；
  - 10. 在交易开始至交易结束的整个过程中，买家和卖家都可以申请取消交易。
    - 1. 进入交易取消流程（另有描述），由专门的审核人员负责。

图5-36 业务过程描述示例

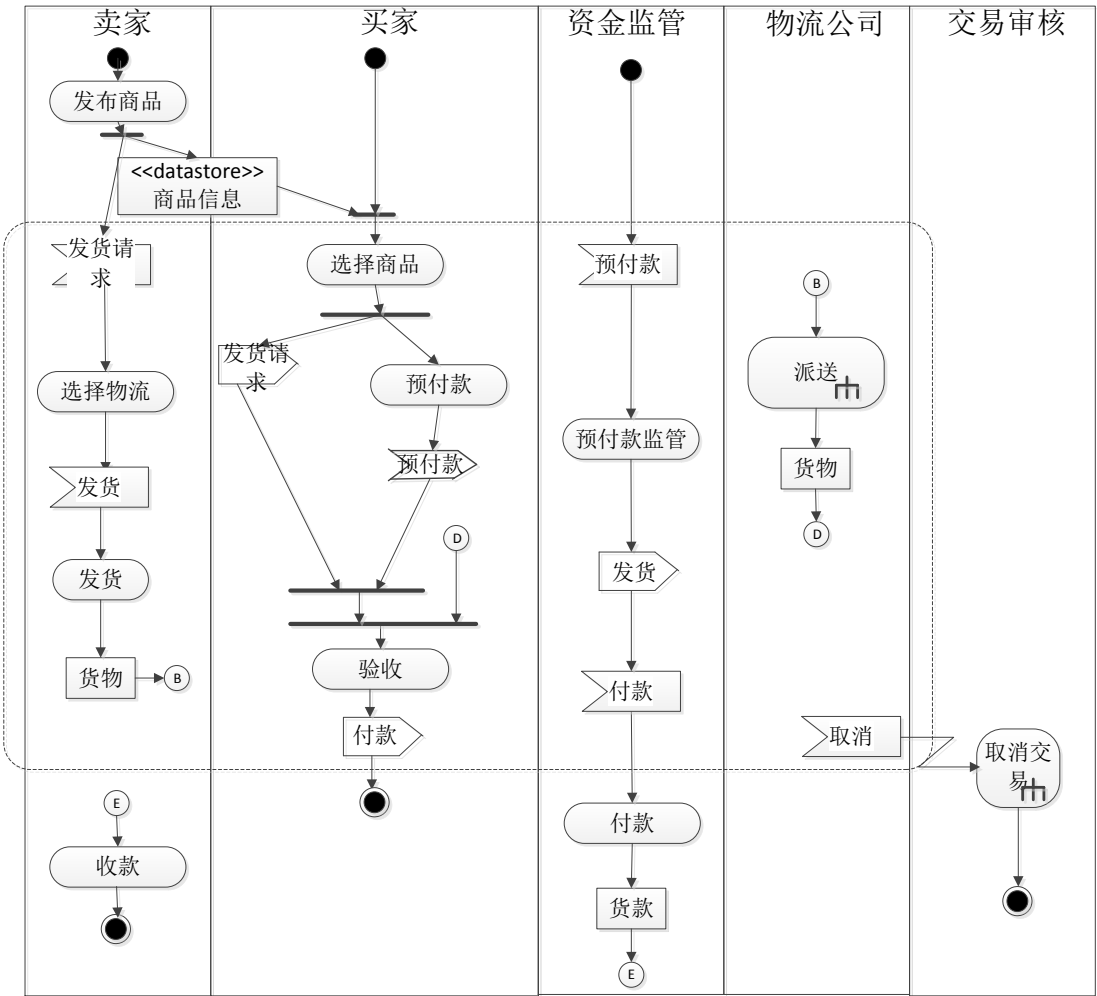


图5-37 业务过程分析示例

(6) 仔细完善第五步的活动图：

- 货物和实际付款两个数据流转比较关键，必须使用同步协同，其他的流转都可以使用异步协同——通过共享系统数据实现异步通信。
- 在描述中未发现业务过程失败场景，事实上这是不可能的（例如买家验收货物不通过情景），所以实践中还需要进一步就此展开获取。
- 描述中表明取消交易是复杂行为，可以建立活动。其实派送也涉及一个复杂的物流过程，也需要建立为活动。在实践中，它们也都需要继续获取细节。
- 交易取消的行为就是异常。

## 5.6. 定义系统边界

系统边界系统与环境互动的界限，定义系统边界可以明确系统需要满足的与外界的交互行为，从而从宏观上界定了系统的功能概要。

系统边界是需求工程后期阶段需求分析活动起始模型，后期的需求分析可以看作是逐一细化系统边界中的对外交互行为的活动。

结构化方法中使用上下文图作为系统边界定义模型，面向对象方法中使用系统用例图作为系统边界定义模型。二者的区别是：上下文图更注重系统与环境的输入输出数据流交互；系统用例图更注重系统与环境的功能性（目的性、任务性）交互。

### 1. 问题分析与系统边界定义

如果前景与范围定义活动主要使用问题分析方法，那么就可以得到每个问题的解决方案边界，将它们合并（叠加、汇总、补充）起来就是整个系统的边界定义。

### 2. 目标分析与系统边界定义

如果前景与范围定义活动主要使用目标分析方法，那么可以从目标模型中抽取系统的边界定义：

- 分配主体包括“将要构建的系统”和系统环境（涉众、硬件、其他系统等）的底层目标往往意味着存在系统与环境的互动，它们就是系统边界定义要考虑的目标，可以称之为边界目标。

- 分析边界目标所覆盖的场景和操纵的操作（任务），可以得到系统用例，并据此建立面向对象的系统边界定义——系统用例图。

- 分析边界目标所关注的数据对象，可以得到系统与环境的输入/输出流，并据此建立结构化的系统边界定义——上下文图。

### 3. 业务过程分析与系统边界定义

如果前景与范围定义活动使用了业务过程分析方法，那么分析的业务过程可以帮助完善系统边界的定义：

- 活动图的每一个动作都可能是（未必一定是）一个用例，可以据此完善面向对象方法的系统用例图。
- 活动图的每一个对象流都可能是（未必一定是）一个系统与环境的输入/输出数据流可以据此完善结构化方法的上下文图。

### 5.7. 前景与范围文档\*

业务需求、高层次解决方案和系统特性都应该被定义到项目前景与范围文档之中，以为后续的开发工作打好基础。有些组织使用项目合约或抽象的业务用例文档来实现类似目的。

前景与范围文档主要由需求工程师来完成，但文档的负责人一般是项目的投资负责人、执行主管或其他类似角色。

一个可以参考的前景与范围文档模版如模版5-1所示。

1 业务需求

1.1 应用背景

1.2 业务机遇

1.3 业务目标

1.4 业务风险

2 项目前景

2.1 前景概述

2.2 主要特性

2.3 假设与依赖

3 项目范围

3.1 第一版范围

3.2 后续版本范围

3.3 限制与排除

4 项目环境

4.1 操作环境

4.2 涉众

4.3 项目属性

词汇表

参考资料

附录

模版5-1 项目前景与范围文档模版

下面将分别说明模版5-1的各项内容。

#### 5.7.1. 业务需求

主要目的是清晰的解释系统的业务需求。业务需求描述了新系统将带给投资人、购买者和用户的主要利益，说明了项目的最终目标。对于不同类型的产品，如信息系统、商业软件包和实时控制系统，业务需求的着重点也会不同。

##### 一、应用背景

概述系统开发的应用背景，描述原有的应用状况，说明新系统开发的动机。必要的情况下，还需要说明

\* 本部分的主要内容源自[Wiegers2003]

应用的历史延续过程。

例如，一个自助餐厅在线订餐系统的应用背景描述如下：

目前，Process Impact公司的大多数员工平均每天要花费60分钟去自助餐厅选择、购买和用午餐，其中大约有20分钟要花在公司和自助餐厅之间的往返、选择午餐和以现金或信用卡方式结账上。当员工到自助餐厅之外去用午餐时，他们平均有90分钟时间不在岗。有些员工提前给自助食堂打电话预订午餐，请自助餐厅准备好他们选择的午餐。但是，员工并不总是能够如愿以偿，因为自助餐厅有些食物已卖完。而与此同时，自助餐厅又在浪费大量的食物，因为有些食物没有卖掉而只好倒掉。早餐和午餐同样面临着这样的问题，只是到餐厅用餐的员工人数比午餐要少的多。

## 二、业务机遇

如果开发的是商业产品，这部分描述的是存在的市场机遇以及产品要参与竞争的市场。如果是企业信息系统，则应描述要解决的业务问题或需要改进的业务流程，以及系统的应用环境。这部分内容还应对已有的产品和可能的解决方案进行比较评估，指出新产品的优点。说明有哪些问题因为没有该产品而在当前无法解决。还要说明该产品怎样符合市场潮流、技术发展趋势或企业的战略方向。另外，还应有一段简短的说明描述如果需要为客户提供一个完整的解决方案，还需要哪些其他的技术、过程和资源。

自助餐厅例子的业务机遇描述如下：

许多员工都通过自助餐厅的一个在线订餐系统提出订餐请求，要求在指定的日期和时间内将所订的午餐送到公司的指定地点。通过这样一个系统，使用这一服务的员工可以节约相当可观的时间，而且订到自己喜欢食物的机会也增大了。这既提高了他们的工作生活质量，也提高了他们的生产率。自助餐厅提前了解到客户需要哪些食物，就可以减少浪费，并提高员工的工作效率。要求送货上门的订餐员工将来还可以从本地的其他饭店来订餐，这就大大扩大了员工对食物的选择范围，并通过与其他饭店的大量购餐协议而有可能节约费用。Process Impact公司也可以只在自助餐厅订午餐，而在其他饭店订早餐、晚餐、特定事件的用餐和周末会餐。

## 三、业务目标与成功标准

用量化和可衡量的方式概述产品提供了哪些重要的业务利益。如果其他文档（如业务用例文档）中已包含了这些信息，此处指明参考文档即可，不必重复其内容。这一部分还应明确涉众如何定义和判断项目的成功。说明哪些因素对项目获得成功的影响最大，无论这些因素是否处于组织的控制范围内。还要定义可衡量的标准，用于评估各项业务目标是否已实现。

业务目标的例子如下所示：

BO-1：在第一版应用之后的6个月内，减少食物的浪费。

度量标准（Scale）：每周被自助餐厅工作人员扔掉的食物价值。

计量方法（Meter）：检查自助餐厅库存系统的日志。

理想标准：减少50%

一般标准：减少30%

最低标准：减少20%

BO-2：在第一版应用之后的12个月内，减少15%自助餐厅的运作成本。

BO-3：在第一版应用之后的3个月内，每个员工每天的有效工作时间平均增加20分钟。

成功标准的例子如下所示：

SC-1：在第一版应用之后的6个月内，目前在自助餐厅用午餐的员工中，有75%的人使用在线订餐系统。

SC-2：在第一版应用之后的3个月内，对自助餐厅满意度的季度调查评价要提高0.5，而在第一版应用之后的12个月内，这种满意度要提高1.0。

#### 四、业务风险

概述与产品开发相关的主要风险。风险类别包括市场竞争、时间安排、用户认可、实现技术以及可能对业务造成的负面影响。要评估每一项风险可能造成的损失、发生的几率以及对它的控制能力。找出所有可能降低风险的必要措施。如果在业务用例分析或类似文档中已经给出了这些信息，此处只需指明出处而不必重复该信息。

业务风险的示例如下：

RI-1：使用该系统的员工太少，减少了对系统开发和变更自助餐厅经营过程的投资回报。

可能性0.3，影响为9。

RI-2：其他本地饭店可能并不认同减价是员工使用这一系统的正当理由，这会减低员工对该系统的满意度，并可能会减少他们对这一系统的使用。

可能性为0.4，影响为3。

### 5.7.2. 项目前景

这一部分建立系统的战略前景，该系统将实现业务目标。前景为产品生命周期中所有的决策提供了背景。详细的功能需求或项目计划信息不应包括在这一部分内。

#### 一、前景概述

用一个简洁的声明概括系统的长期目标和意图。声明应当反映能够满足不同涉众需求的平衡的观点。前景声明可以理想化，但应当以当前或预期的市场现状、企业结构、团体战略和资源限制为依据。

前景概述的示例如下：

对那些希望通过公司自助餐厅或其他本地饭店在线订餐的员工来说，“自助餐厅订餐系统”是一个基于Internet的应用程序，它可以接受个人订餐或团体订餐，结算用餐费用，并触发将预计餐送到Process Impact公司内的指定位置。与当前的电话订餐和人工订餐不同，使用“自助餐厅订餐系统”的雇员并不需要到食堂内去用餐，这既可以节约他们的时间，又可以增加他们对食物的选择范围。

系统上下文图如图5-38所示：

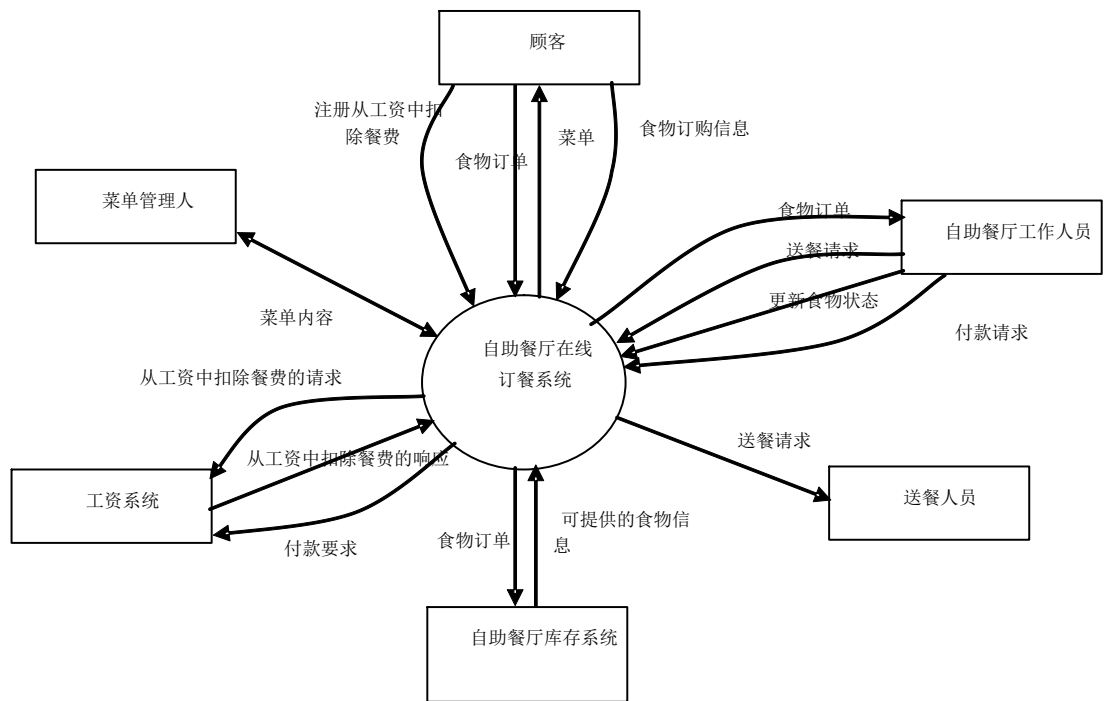


图5-38自助餐厅订餐系统上下文图

二、主要特性

为新产品的每一项主要特性或用户功能进行固定的、唯一的命名或编号，突出其超越原有产品或竞争产品的特性。给每项特性一个唯一的标号，这样可以追溯其去向——用户需求、功能需求和其他系统元素。

系统特性的示例如下：

- FE-1：根据自助餐厅提供的菜单来订餐。
- FE-2：根据其他本地饭店的送货菜单来订餐。
- FE-3：请求送餐。
- FE-4：创建、浏览、修改、删除用餐预订。
- FE-5：通过公司的内联网访问系统，或者授权员工通过外部Internet访问系统。

.....

三、假设与依赖

记录构思项目和编写前景与范围文档过程中涉众所提出的每一项假设。由于一方所做的假设往往不为其他各方所知，因此通过将所有的假设记录下来并进行检查，各方就能对项目潜在的基本假设达成一致。这样便能够避免可能的混乱以及这种混乱会在将来造成的影响。

这一部分还记录项目对不在自身控制范围内的外部因素的主要依赖关系。这类外部因素包括悬而未决的行业标准或政府法规、其他项目、第三方厂商及开发伙伴等。

假设与依赖的示例如下：

- AS-1：自助餐厅内有可以访问公司内网的计算机和打印机。
- AS-2：自助餐厅有送货人员和送货车辆，最多比请求的送货时间晚15分钟。
- DE-1：如果某饭店有自己的联机订餐系统，那么“自助餐厅订餐系统”必须能与这一系统进行双向通

信。

### 5.7.3. 项目范围

项目的范围定义了解决方案的概念和范围，同时也要表明系统不能提供哪些功能，它可以帮助涉众建立现实的期望。

#### 一、第一版范围

概述计划在产品的第一个版本中实现的主要特性。描述产品的质量特性，产品依靠这些特性为不同类别的用户提供预期利益。

如果目标是集中开发力量和维持合理的项目进度，就不要企图在1.0版本中包含所有可能的需求。那样会导致项目范围在不知不觉中增大，使得进度延误。应该把注意力集中在那些能够在最短时间内，以最适宜的成本，为大多数用户提供最大价值的特性上。

#### 二、后续版本范围

如果要采取阶段性的开发方式，需要决定推迟实现哪些特性，并为后续的版本做出时间安排。后续版本能够实现更多的需求和特性，并可完善最初的功能。随着产品的不断成熟，系统的性能、可靠性和其他质量特征也将得到改进。

第一版和后续版本的范围定义示例如下：

特性	版本1	版本2	版本3
FE-1	用午餐菜单定标准餐；费用支付方式是从工资中扣除	除午餐外，也可以订早餐和晚餐；费用的支付方式可以是信用卡	
FE-2	不实现	不实现	完全实现
FE-3	送餐地点仅限公司内部	送餐地点也可是公司外	
FE-4	如果有时间就实现	完全实现	
FE-5	完全实现		

#### 三、限制与排除

管理范围蔓延的方法之一是，定义项目包含的需求与不包含的需求之间的界线。此处应列出涉众可能希望得到，但不在产品或其某个特定版本计划之内的功能和特性。

限制与排除的示例如下：

LI-1：自助餐厅的有些食物不适宜送货，因此“自助餐厅订餐系统”的顾客使用的送货菜单是食堂整个菜单的一个子集。

LI-2：“自助餐厅订餐系统”只能用于Process Impact公司总部内的自助餐厅。

### 5.7.4. 项目环境

#### 一、操作环境

描述系统将用于什么样的环境，定义关键的可用性、可靠性、性能等质量属性要求。这些信息对系统的结构定义有着重要的影响。



和操作环境相关的问题包括：

- 用户是地理分散的还是集中的？
- 不同的用户会在什么时间访问系统？
- 数据在何处生成，用于何处？
- 访问数据时的最大响应时间是否已知？
- 用户能否容忍服务中断？
- 是否需要提供访问安全控制和数据保护？

## 二、涉众

描述项目涉众的相关信息，重点介绍不同类型的客户、目标市场和目标市场中的用户类别，说明他们和系统密切相关的一些特征。

详细的涉众描述见第6章。

## 三、项目属性

要想更有效的进行决策，涉众就必须就项目的相关属性及其优先级达成一致。这些属性包括：特性（功能、范围）、质量、成本、进度和人员。

对任何一个特定的项目而言，上述每个属性都有三种影响因素：

- 驱动因素（Driver）：重要的成功目标；
- 约束因素（Constraint）：项目必须在一定的限制下开展工作；
- 可调整因素（Degree of Freedom）：可以根据其他方面进行平衡和调整的因素。

项目经理的目标是：在约束施加的限制内，合理安排可调整因素，获得最大的驱动因素。

在项目属性之间互相不可调和时，属性间的优先级顺序指导项目管理者采取正确的行动。例如，对于急需面市的系统，其进度是第一优先级，这样在项目无法按照预定计划前进时，就可能会推迟特定功能的实现，或者增加人员和投资。再例如，对于人员（或费用）受限的系统，人员（费用）是第一优先级，在项目出现偏离时就可能延迟系统的完成期限，或者推迟部分功能的实现。除特殊情况之外，质量都是一个不应该被牺牲的项目属性。

项目属性的示例如下：

属性	执行者	约束因素	可调整因素
进度			计划××日期完成第一版，××日期完成第二版；在不包括责任人评审的情况下，最多可超过期限3个星期
特性		1.0版本中要求实现的特性必须完全可操作	
质量		必须通过95%的用户验收测试；必须通过全部的安全性测试；所有的安全事务都	

属性	执行者	约束因素	可调整因素
		必须遵守公司的安全标准	
人员	团队规模包括一名兼职的项目经理，两名开发人员，和一名兼职的测试人员；如果有必要，还可以再增加兼职的开发人员		
费用			在不包括责任人评审的情况下，财政预算最多可超支15%

## 引用文献

- [Alrajeh2012] Alrajeh, D., Kramer, J., Lamsweerde, A., Russo, A., Uchitel, S., Generating Obstacle Conditions for Requirements Completeness Proc, ICSE'2012: 34th International Conference on Software Engineering, Zurich, June 2012.
- [Anton1994] Anton, A., McCracken, W., Potts, C., Goal Decomposition and Scenario Analysis in Business Process Reengineering. Proc. 6th Conference On Advanced Information Systems Engineering (CAiSE'94), Utrecht, Holland, June 1994.
- [Anton1996] Anton, A., Goal-Based Requirements Analysis. Proc. Second IEEE International Conference on Requirements Engineering (ICRE'96), Colorado Springs, USA, April 1996.
- [Anton1997] Anton, A., Goal Identification and Refinement in the Specification of Software-Based Information Systems. Ph.D. Thesis, Georgia Institute of Technology, Atlanta, GA, USA, June 1997.
- [Bertrand1997] Bertrand P., Darimont R., Delor E., Massonet P., Lamsweerde A., *GRAIL/KAOS: an environment for goal driven requirements engineering*, Proceedings de ICSE-97: 19th International Conference on Software Engineering, Boston, May 1997.
- [Chung1993] Chung, L., Representing and Using Non-Functional Requirements: A Process Oriented Approach," Ph.D. Thesis, Department of Computer Science, Univ. of Toronto, 1993.
- [Chung1995] Chung, L., Nixon, B. A., Dealing with Non-Functional Requirements: Three Experimental Studies of a Process-Oriented Approach, Proceedings of the 17th IEEE International Conference on Software Engineering, pp. 25-37, Seattle, 1995.
- [Chung2000] Chung, L., Nixon, B., Yu, E., Mylopoulos, J., Non-Functional Requirements in Software Engineering. Kluwer. 2000.
- [Classen2008] Classen, A., Heymans, P., Schobbens, P. Y., What's in a Feature: A Requirements Engineering Perspective, Proceedings of the 11th International Conference on Fundamental

Approaches to Software Engineering, 2008, pp. 16-30.

[Clements2006] P. Clements, Best Practices in Software Architecture. presentation given by Paul Clements, July 26, 2006.

<http://www.sei.cmu.edu/library/abstracts/presentations/bestpracticessoftwarearchitecture.cfm>.

[Cysneiros2004a] Cysneiros, L. M., Yu, E., Non-functional requirements elicitation, in Perspectives on Software Requirements, J. C. S. P. Leite, and J. H. Doorn, Ed. vol. 753, Springer US, 2004, pp. 115-138.

[Cysneiros2004b] Cysneiros, L.M., Leite, J.C.S.P., Nonfunctional Requirements: From Elicitation to Conceptual Models, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 30, NO. 5, MAY 2004.

[Darimont1996] Darimont, R. , Lamsweerde, A. , Formal Refinement Patterns for Goal-Driven Requirements Elaboration, Proceedings FSE-4 - 4th ACM Symp. on the Foundations of Software Engineering, San Francisco, Oct. 1996, pp179-190.

[Doerr2005] Doerr, J., Kerkow, D., Koenig, T., Olsson, T., Suzuki, T., Non-Functional Requirements in Industry – Three Case Studies Adopting an Experience-based NFR Method, Proceedings of the 2005 13th IEEE International Conference on Requirements Engineering (RE'05), 2005.

[Jansen2005] A. Jansen and J. Bosch, Software Architecture as a Set of Architectural Design Decisions, In Proceedings of WICSA 2005, 2005. [Dardenne1993] Dardenne, A., van Lamsweerde, A., and Fickas, S., Goal-Directed Requirements Acquisition, Science of Computer Programming, 20, pp. 3-50, 1993.

[Kavakli2002] Kavakli, E., *Goal-oriented requirements engineering: A unifying framework*, Requirements Engineering, vol.6, pp.237–251, 2002.

[Lamsweerde1995] van Lamsweerde, A., Darimont, R., and Massonet, Ph., Goal-Directed Elaboration of Requirements for a Meeting Scheduler: Problems and Lessons Learnt. Proc. RE'95 - 2 nd. IEEE Symp. on Requirements Engineering (York, UK), March 1995, 194-203.

[Lamsweerde1998a] Lamsweerde, A., Letier, E., Integrating Obstacles in Goal-Driven Requirements Engineering Proceedings ICSE'98 - 20th International Conference on Software Engineering, IEEE-ACM, Kyoto, April 1998.

[Lamsweerde1998b] Lamsweerde, A., Darimont, R., Letier, E., Managing Conflicts in Goal-Driven Requirements Engineering IEEE Transactions on Software Engineering, Special Issue on Managing Inconsistency in Software Development, November 1998.

[Lamsweerde2000a] Lamsweerde, A., *Requirements Engineering in the Year 00: A Research Perspective*, Invited Paper for ICSE'2000 - 22nd International Conference on Software Engineering, Limerick, ACM Press, June 2000.

- [Lamsweerde2000b] Lamsweerde,A., Letier , E., Handling Obstacles in Goal-Oriented Requirements Engineering IEEE Transactions on Software Engineering, Special Issue on Exception Handling, Vol. 26 No. 10, October 2000, 978-1005.
- [Lamsweerde2001] Lamsweerde, A., Goal-Oriented Requirements Engineering: A Guided Tour, Proceedings RE'01, 5th IEEE International Symposium on Requirements Engineering, Toronto, August 2001, 249-263.
- [Lamsweerde2004] Lamsweerde,A., Goal-Oriented Requirements Engineering: A Roundtrip from Research to Practice, RE '04, Kyoto, September 10, 2004.
- [Lapouchnian2005] Lapouchnian, A., Goal-Oriented Requirements Engineering: An Overview of the Current Research, Depth Report, University of Toronto, 2005.
- [Leffingwell1999] Leffingwell, D., Widrig, D., *Managing Software Requirements: A Unified Approach*, Addison-Wesley, 1999.
- [Mylopoulos1992] Mylopoulos, J., Chung, L., Nixon, B., Representing and Using Non-Functional Requirements: A Process-Oriented Approach. IEEE Transactions on Software Engineering, 18(6), June 1992.
- [Mylopoulos2006] Mylopoulos, J., Goal-Oriented Requirements Engineering, 14th IEEE Requirements Engineering Conference Minneapolis, September 15, 2006.
- [OMGUMLSuperstructure] OMG, UML 2.0 Superstructure Specification.
- [OMGBPMN] OMG,Business Process Model and Notation, <http://www.bpmn.org/>.
- [Oshiro2003] Oshiro, K., Watahiki, K., Saeki, M., Goal-Oriented Idea Generation Method for Requirements Elicitation, Proceedings of the 11th IEEE International Requirements Engineering Conference, 2003.
- [Regev2005] Regev, G., Wegmann, A., Where do Goals Come from: the Underlying Principles of Goal-Oriented Requirements Engineering, Presented at the 13th IEEE International Requirements Engineering Conference (RE'05), Paris, France, 2005.
- [Webster2005] Webster, I., Amaral, J., Cysneiros Filho, L.M., A Survey of Good Practices and Misuses for Modelling with i\* Framework,in WER2005, 2005.
- [Wiegiers2003] Wiegiers, K. *Software Requirements*, second edition. Redmond, WA: Microsoft Press, 2003.
- [Yu1997] Yu, E.,Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering, Proc. of the 3rd Interna. Symp. on Requirements Eng, Jan 1997, pp:226-235
- [Yu1998] Yu, E. and Mylopoulos, J., Why Goal-Oriented Requirements Engineering, Fourth International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'98), E. Dubois, A. Opdahl and K. Pohl (ed.), Pisa, Italy, 1998.
- [Zave1997] Zave, P., and Jackson, M., *Four Dark Corners of Requirements Engineering*, ACM

Transactions on Software Engineering and Methodology, 1997, 1-30.