ICS 35.240.60
L 79

中华人民共和国国家标准

GB/T 39320—2020

电子商务　元模型　基本模块

Electronic business—Meta model—Foundation module

2020-11-19 发布　　2021-06-01 实施

国家市场监督管理总局
国家标准化管理委员会　发布

目　　次

前　言

本标准按照 GB/T 1.1—2009 给出的规则起草。

本标准由全国电子业务标准化技术委员会(SAC/TC 83)归口。

本标准起草单位：深圳市坤鑫国际货运代理有限公司、厦门吧哒文化传播有限公司、北京九星时代科技股份有限公司、浙江义境通电子商务有限公司、新疆高新技术项目开发研究院(有限公司)、中国标准化研究院、嘉兴市大地物流有限公司、北京中标纵横标准科技有限公司、重庆精驿行供应链管理有限公司、喀什金利达国际物流中心有限公司、和田恒扬对外贸易有限公司、图木舒克市海纳进出口有限公司、新疆德鲁亚国际物流有限公司、大连万发联合航贸科技有限公司、伊犁通瑞达运输有限公司、公安部第三研究所、重庆电子工程职业学院、成都理邦系统工程技术有限公司。

本标准主要起草人：林忠、张荫芬、朱彤、林婷、姚树红、李金华、林晓炜、曹建峰、金旭峰、杨希江、蒋啸冰、康树春、陈铭、蔡依荻、钟晶、雷雨峰、朱娇杨、向国伦、潘况、罗海莹。

引　言

联合国贸易便利化与电子业务中心(UN/CEFACT)建模方法(UMM)是一种统一建模语言(UML)建模方法,用于设计每一合作伙伴为了进行协作所应提供的业务服务,其为在面向服务的协作体系结构中付诸实施的服务提供业务理由。因此,UN/CEFACT 首先着眼于获取业务知识,使之能够开发基于面向服务的体系结构(SOA)的低成本软件,帮助中小企业(SME)以及新兴经济体投入电子商务的实际应用。UMM 致力于开发针对组织间业务过程及其信息交换的全球性的编排设计。UMM 模型以 UML 语法表示,并且是独立于平台的模型。独立于平台的 UMM 模型在面向服务的体系结构中辨别出需要实施业务协作的服务。这种方法可以防止技术过时。

本标准的 UMM 包括三个视图,每个包含一组定义明确的构件:

——业务领域视图(Business Domain View)(BDV);

——业务需求视图(Business Requirements View)(BRV);

——业务交易视图(Business Transaction View)(BTV) 。

本标准的目标在于:

——定义格式良好的 UMM 业务协作模型的语义;

——定义符合 UMM 的业务协作模型的验证规则;

——阐明符合 UMM 的业务协作模型所基于的基本概念;

——为 UMM 业务协作模型提供明确的定义,允许在面向服务的体系结构中进行非虚拟映射到构件;

——为 UMM 基础模块定义 UML 概要文件,允许 UML 工具供应商自定义其工具以符合 UMM。

电子商务　元模型　基本模块

1　范围

本标准规定了电子商务元模型的基本模块结构以及基本模块管理、业务领域视图、业务需求视图、业务交易视图等。

本标准适用于开放式电子数据交换业务和电子商务领域业务建模，允许 UML 工具供应商基于本标准的 UMM 基本模块建模方法定制与 UMM 兼容的工具。

2　规范性引用文件

下列文件对于本文件的应用是必不可少的。凡是注日期的引用文件，仅注日期的版本适用于本文件。凡是不注日期的引用文件，其最新版本(包括所有的修改单)适用于本文件。

GB/T 18811　电子商务基本术语

3　术语、定义和缩略语

3.1　术语和定义

GB/T 18811 界定的术语和定义适用于本文件。

3.2　缩略语

下列缩略语适用于本文件。

BCV：业务编排视图(Business Choreography View)

BEV：业务实体视图(Business Entity View)

BDV：业务领域视图(Business Domain View)

BIV：业务交互视图(Business Interaction View)

BPV：业务过程视图(Business Process View)

BRV：业务需求视图(Business Requirements View)

BTV：业务交易视图 (Business Transaction View)

OCL：对象约束语言 (Object Constraint Language)

UML：统一建模语言 (Unified Modeling Language)

UMM：联合国贸易便利化与电子业务中心建模方法 (UN/CEFACT Modeling Methodology)

4　UMM 元模型和基本模块

4.1　UMM 元模型

电子商务建模方法采用 UN/CEFACT 给出的方法。UMM 元模型可分为一组元模块。UMM 元模型按核心、基本功能到完整功能的层次分成基础模块(Base Module)、基本模块(Foundation Module)、专用模块(Specialization Module)和扩展模块(Extension Module)，见图 1。

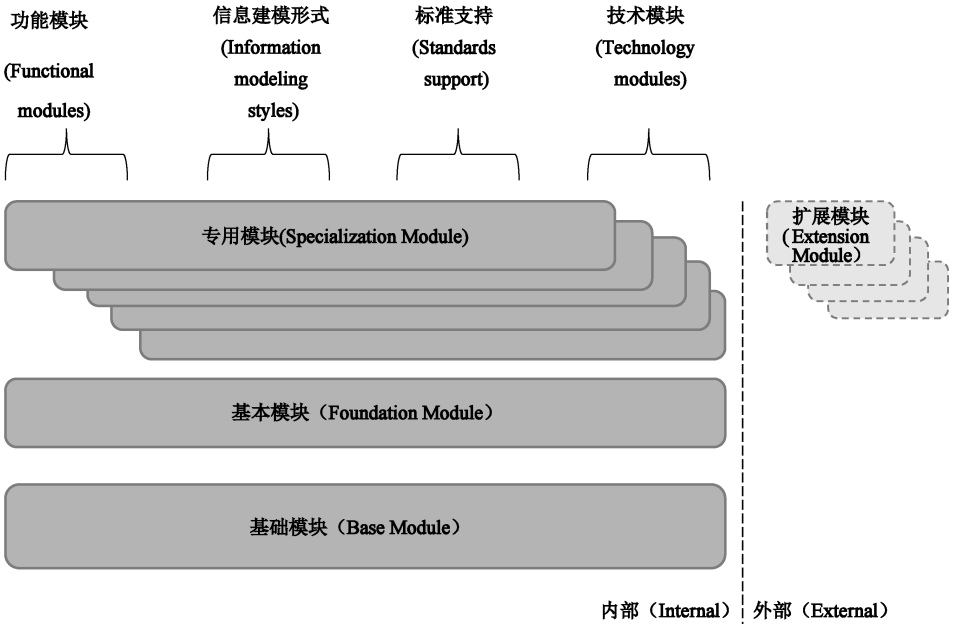

图 1　UMM 元模型的模块结构

基础模块(Base Module):所有其他高层模块共同适用的基本准则。

基本模块(Foundation Module):定义了构建符合 UMM 规则的业务协作模型的最基本方法论的所有概念。

专用模块(Specialization Module):定义了基本模块之上附加的专业化限定概念。每一专用模块都聚焦于特定领域的专业类型分析对基本模块进行扩充。专用模块可以作为未来候选的基本模块。

扩展模块(Extension Module):扩展模块与专用模块类似,但专用模块由 UN/CEFACT 开发和维护,扩展模块则可由其他机构基于 UMM 方法增加新特性创建并维护。

本标准所定义的 UML 概要文件以 UML 元模型版本 1.4.2 为基础,UMM 的基本概念及相互关联方式通过元模型描述和解释。本标准中 OCL(对象约束语言)约束规定的 UMM 语义验证将使用外部验证服务或定制插件。可以将基本模块扩展为不同的专业模块和扩展模块,以便在 UMM 业务协作模型所需的最小语义之上定义附加语义。

4.2　元模型的基本模块结构

4.2 中定义了 UMM 元模型的基本模块的 UML 概要文件。图 2 中显示 UMM 元模型基本模块的程序包结构,不同部分定义了相应程序包的构造型、标签定义和约束。

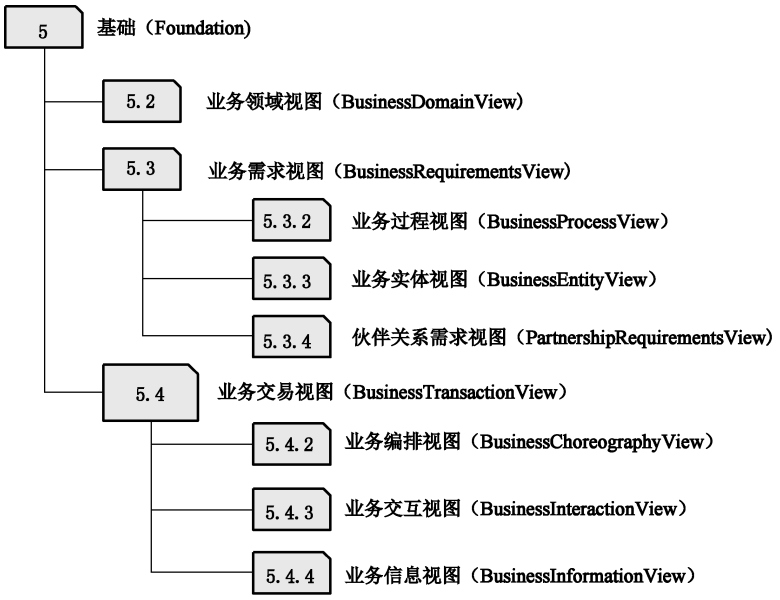

图 2　UMM 元模型基本模块程序包结构图

基本模块第一级程序包包含 UMM 三种视图：业务领域视图(BusinessDomainView)(5.2)、业务需求视图(BusinessRequirementsView)(5.3)和业务交易视图(BusinessTransactionView)(5.4)。

业务领域视图(BusinessDomainView)(5.2)不包含不同类型的构件，不再分为子程序包。

业务需求视图(BusinessRequirementsView) (5.3)包含三种不同类型的构件：业务过程活动图、业务实体生命周期和用例中定义的协作需求，对应的子程序包为业务过程视图(BusinessProcessView)(5.3.2)、业务实体视图(BusinessEntityView)(5.3.3)和伙伴关系需求视图(PartnershipRequirementsView)(5.3.4)。

业务交易视图(BusinessTransactionView)(5.4)包含三个构件：业务协作的流程、用于同步状态的业务交互流程，交互过程中交换的业务信息，对应的子程序包为业务编排视图(BusinessChoreography-View)(5.4.2)、业务交互视图(BusinessInteractionView)(5.4.3)和业务信息视图(BusinessInformation-View)(5.4.4)。

4.3　基本模块与元模型其他部分的依赖性(规范性)

4.3.1　UMM 基本模块 1.0 基于 UMM 基础模块 1.0 建立，见图 3。

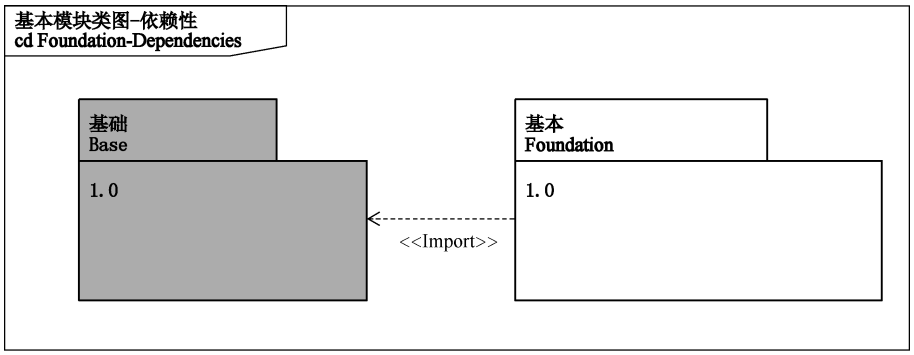

图 3　UMM 基本模块的依赖性

4.3.2　图 4 为 UMM 基础模块所定义并用于基本模块的构造型。基础模块的构造型均以灰色背景显示。UMM 基础模块 1.0 规定了注册对象(RegistryObject)和业务库包(BusinessLibraryPackage)的构

造型形式。基本模块中，程序包是实现主要 UMM 构件的构造型容器，被定义为基础构造型业务库包的专用构造型，该程序包及其内容可以作为注册的候选。

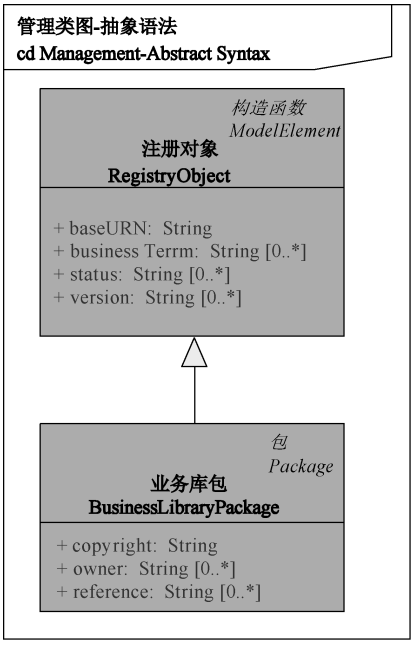

图 4　UMM 基础模块抽象语法

5　UMM 基本模块

5.1　基本模块管理

5.1.1　概念性描述（资料性）

基于 UMM 方法构建的业务协作模型，将作为构造型业务协作模型（BusinessCollaborationModel）。对于业务协作模型，业务领域视图是可选项，业务需求视图和业务交易视图是必选项。因此，业务协作模型（BusinessCollaborationModel）可由零或一个业务领域视图（BusinessDomainView）构成，同时业务协作模型（BusinessCollaborationModel）还应包括一个业务需求视图（BusinessRequirementsView），或一个业务交易视图（BusinessTransactionView）。

UMM 基础模块管理概念框架，见图 5。

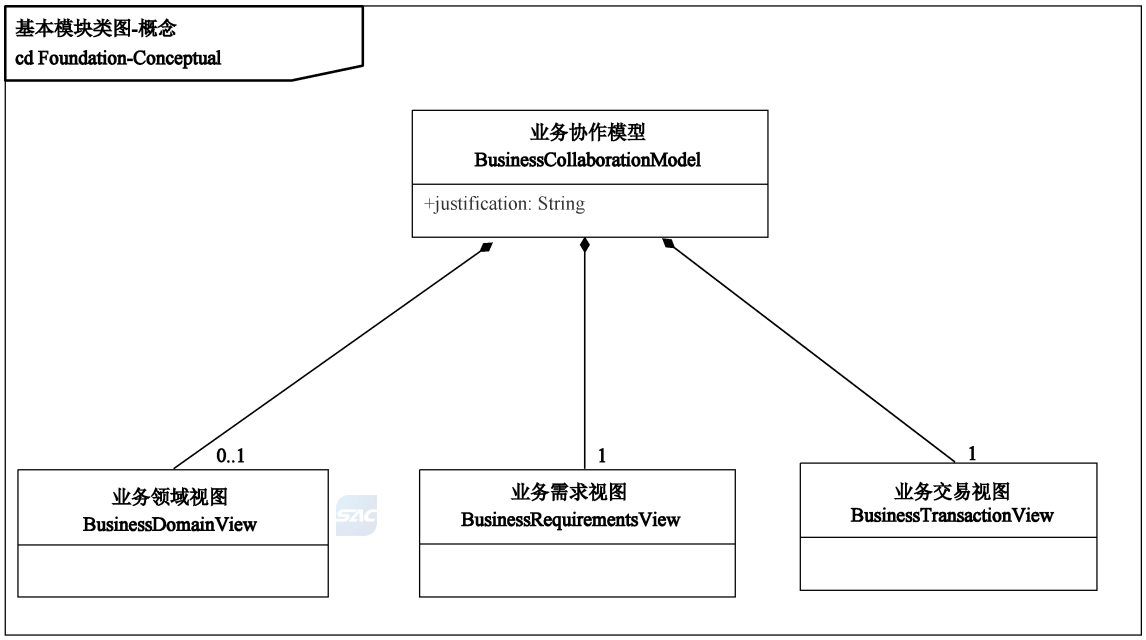

图 5　UMM 基础模块管理概念框架

5.1.2　构造型和标签定义(规范性)

5.1.2.1　UMM 基本模块管理抽象语法,见图 6。

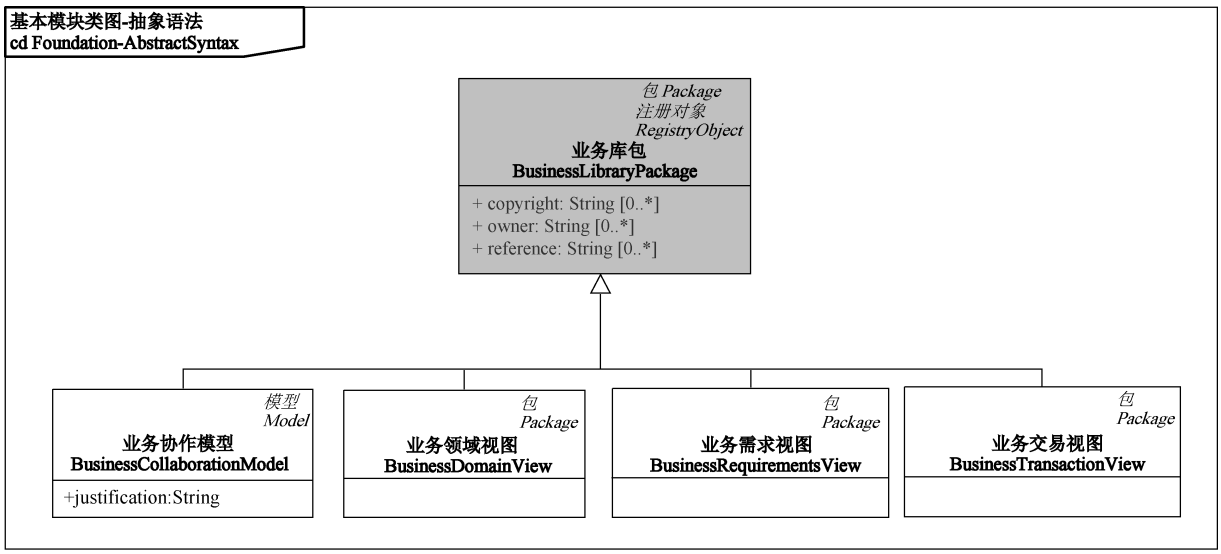

图 6　UMM 基本模块管理抽象语法

5.1.2.2　构造型和标签定义-业务协作模型,见表 1。

表 1　构造型和标签定义-业务协作模型

构造型(Stereotype)	业务协作模型(BusinessCollaborationModel)	
基类(Base Class)	模型(Model)	
父类(Parent)	业务库包(BusinessLibraryPackage)[来自基础模块(Base Module)]	
描述(Description)	业务协作模型符合 UMM 元模型,应与基础模块和基本模块一致,并可与一个或多个专业模块和/或扩展模块一致	
标签定义 (Tag Definition)	理由(justification)	
	类型(Type)	字符串(String)
	次数(Multiplicity)	1
	描述(Description)	从业务角度说明所确定的业务协作可能成为诸多业务协作的理由
	继承标签值(Inherited tagged value): ——基础统一资源名称(baseURN) ——所有者(owner) ——版权(copyright) ——参考号(reference) ——版本(version) ——状态(status) ——业务术语(businessTerm)	

5.1.2.3　构造型和标签定义-业务领域视图,见表 2。

表 2　构造型和标签定义-业务领域视图

构造型(Stereotype)	业务领域视图(BusinessDomainView)
基类(Base Class)	程序包(Package)
父类(Parent)	业务库包(BusinessLibraryPackage)[来自基础模块(Base Module)]
描述(Description)	业务领域是识别和理解业务过程,以及根据分类方案对其进行分类的框架。业务领域视图是包含分类方案以及被分类的业务过程的容器
标签定义 (Tag Definition)	继承标签值(Inherited tagged value): ——基础统一资源名称(baseURN) ——所有者(owner) ——版权(copyright) ——参考号(reference) ——版本(version) ——状态(status) ——业务术语(businessTerm)

5.1.2.4　构造型和标签定义-业务需求视图,见表 3。

表 3　构造型和标签定义-业务需求视图

构造型(Stereotype)	业务需求视图(BusinessRequirementsView)
基类(Base Class)	程序包(Package)
父类(Parent)	业务库包(BusinessLibraryPackage)[来自基础模块(Base Module)]
描述(Description)	所有元素的容器,用于确定和描述作为特定授权角色的业务伙伴类型之间的协作需求
标签定义 (Tag Definition)	**继承标签值(Inherited tagged value):** ——基础统一资源名称(baseURN) ——所有者(owner) ——版权(copyright) ——参考号(reference) ——版本(version) ——状态(status) ——业务术语(businessTerm)

5.1.2.5　构造型和标签定义-业务交易视图,见表 4。

表 4　构造型和标签定义-业务交易视图

构造型(Stereotype)	业务交易视图(BusinessTransactionView)
基类(Base Class)	程序包(Package)
父类(Parent)	业务库包(BusinessLibraryPackage)[来自基础模块(Base Module)]
描述(Description)	所有元素的容器,用于描述各级业务协作的流程以及流程中每一步需要交换的信息
标签定义 (Tag Definition)	**继承标签值(Inherited tagged value):** ——基础统一资源名称(baseURN) ——所有者(owner)-版权(copyright) ——参考号(reference) ——版本(version) ——状态(status) ——业务术语(businessTerm)

5.1.3　约束(规范性)

5.1.3.1　业务协作模型(BusinessCollaborationModel)可包含零或一个业务领域视图(BusinessDomainView)程序包。

示例:

```
package Model_Management
context Model

inv zeroToOneBusinessDomainView:
  self.isBusinessCollaborationModel() implies
  self.ownedElement->select(isBusinessDomainView())->size()<=1
```

5.1.3.2　业务协作模型(BusinessCollaborationModel)应包含一个业务需求视图(BusinessRequirementsView)程序包。

示例：

```
package Model_Management
context Model

inv oneBusinessRequirementsView：
  self.isBusinessCollaborationModel() implies
  self.ownedElement->one(isBusinessRequirementsView())
```

5.1.3.3 业务协作模型(BusinessCollaborationModel)应包含一个业务交易视图(BusinessTransaction-View)程序包。

示例：

```
package Model_Management
context Model

inv oneBusinessTransactionView：
  self.isBusinessCollaborationModel() implies
  self.ownedElement->one(isBusinessTransactionView())
```

5.1.3.4 业务领域视图(BusinessDomainView)、业务需求视图(BusinessRequirementsView)和业务交易视图(BusinessTransactionView)应直接位于业务协作模型(BusinessCollaborationModel)根目录下方。

示例：

```
package Model_Management
context Model

inv rootLevelPackages
  (self.isBusinessDomainView()orself.isBusinessRequirementsView()or
  self.isBusinessTransactionView()) implies
  self.namespace.isBusinessCollaborationModel()
```

5.1.4 用于 UMM 基本模块管理的 OCL 方法(规范性)

OCL 方法，参见示例。

示例：

```
package Foundation::Core
context ModelElement

--Predefined method which evaluates, if the given Modelelement
--has a stereotype equal to the passed name
def:
let hasStereotype (st : String) : Boolean =
  self.stereotype->select(cst | cst.name = st)->notEmpty()
--Predefined method which evaluates, if the given element
--has the stereotype 'BusinessCollaborationModel'
def:
let isBusinessCollaborationModel() : Boolean =
  self.oclIsKindOf(Model) and
  self.hasStereotype('BusinessCollaborationModel')
```

```
--Predefined method which evaluates, if the given element
--has the stereotype 'BusinessDomainView'
def :
let isBusinessDomainView() : Boolean =
  self.oclIsKindOf(Package) and
  self.hasStereotype('BusinessDomainView')
--Predefined method which evaluates, if the given element
--has the stereotype 'BusinessRequirementsView'
def :
let isBusinessRequirementsView() : Boolean =
  self.oclIsKindOf(Package) and
  self.hasStereotype('BusinessRequirementsView')
--Predefined method which evaluates, if the given element
--has the stereotype 'BusinessTransactionView'
def :
let isBusinessTransactionView() : Boolean =
  self.oclIsKindOf(Package) and
  self.hasStereotype('BusinessTransactionView')
```

5.2 业务领域视图

5.2.1 概念性描述(资料性)

5.2.1.1 业务领域视图用于发现一个项目中相关的业务过程。业务过程至少由一个(或多个)业务伙伴类型执行。一个业务伙伴类型可执行多个业务过程。业务伙伴类型 (BusinessPartnerType)和业务过程(BusinessProcess)之间的参与(participates)关联为(1..n)至(0..n)的关联。业务伙伴类型 (BusinessPartnerType)是利益相关方(Stakeholder)的专用构造型。业务过程(BusinessProcess)和利益相关方(Stakeholder)之间的关系以 UMM 中的利益关联(isOfInterestTo)依赖关系来描述。一个业务过程可使用包含(include)和扩展(extends)两个关联构造型分解为子过程,通过(0..1)至(0..*)来表示业务过程(BusinessProcess)的构成。

5.2.1.2 业务过程应先分类为业务类别,业务领域视图(BusinessDomainView)由一个或多个(1..n) 业务类别(BusinessCategories)构成,业务类别可具有层级关系,进一步由其他业务类别组成,因此业务类别(BusinessCategory)的构成应为(0..1) 至 (0..n)。业务类别层级中最低层的业务类别包含一个或多个业务过程,最高层的业务类别不包含任何业务过程。业务类别(BusinessCategory)和业务过程(BusinessProcess)之间的组合为 1 至(0..n)。

5.2.1.3 业务类别(BusinessCategory)的专用构造型为业务域(BusinessArea)和过程域(ProcessArea)。业务域对应于组织中的业务部门,过程域对应于业务域内的一组通用操作。业务域和过程域也可有多个层级。业务域层级中除了最底层之外各层级业务域都仅包含业务域,但最底层业务域则由一个或多个过程域构成。业务域(BusinessArea)和过程域(ProcessArea)之间组合由(0..1)至(0..n)表示。业务域不能包含业务过程。过程域层级中最底层包含一个或多个业务过程。较高层级中的过程域不包含任何业务过程。过程域(ProcessArea)和业务过程(BusinessProcess)之间的组合为 1 至(0..n)。一个构造型业务类别(BusinessCategory)和构造型过程域(ProcessArea)与业务域(BusinessArea)的组合之间可替代,但 UMM 模型不能同时使用业务类别(BusinessCategory)和后两者的组合。

5.2.1.4 业务领域视图(BusinessDomainView)概念框架,见图 7。

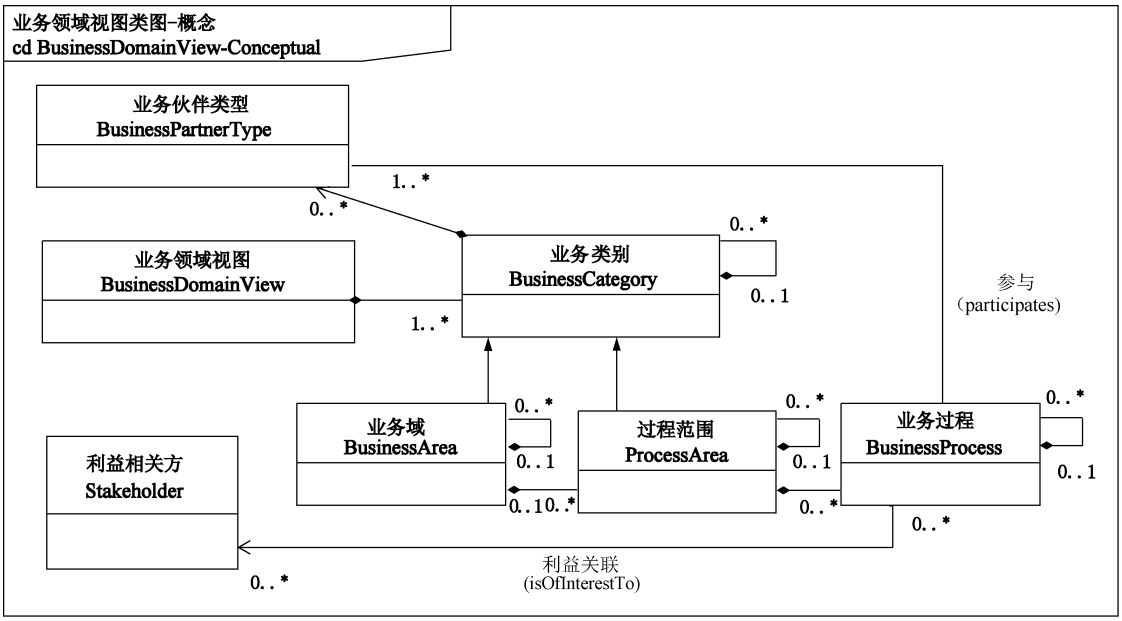

图 7　业务领域视图概念框架

5.2.2　构造型和标签定义(规范性)

5.2.2.1　业务领域视图(BusinessDomainView)抽象语法,见图 8。

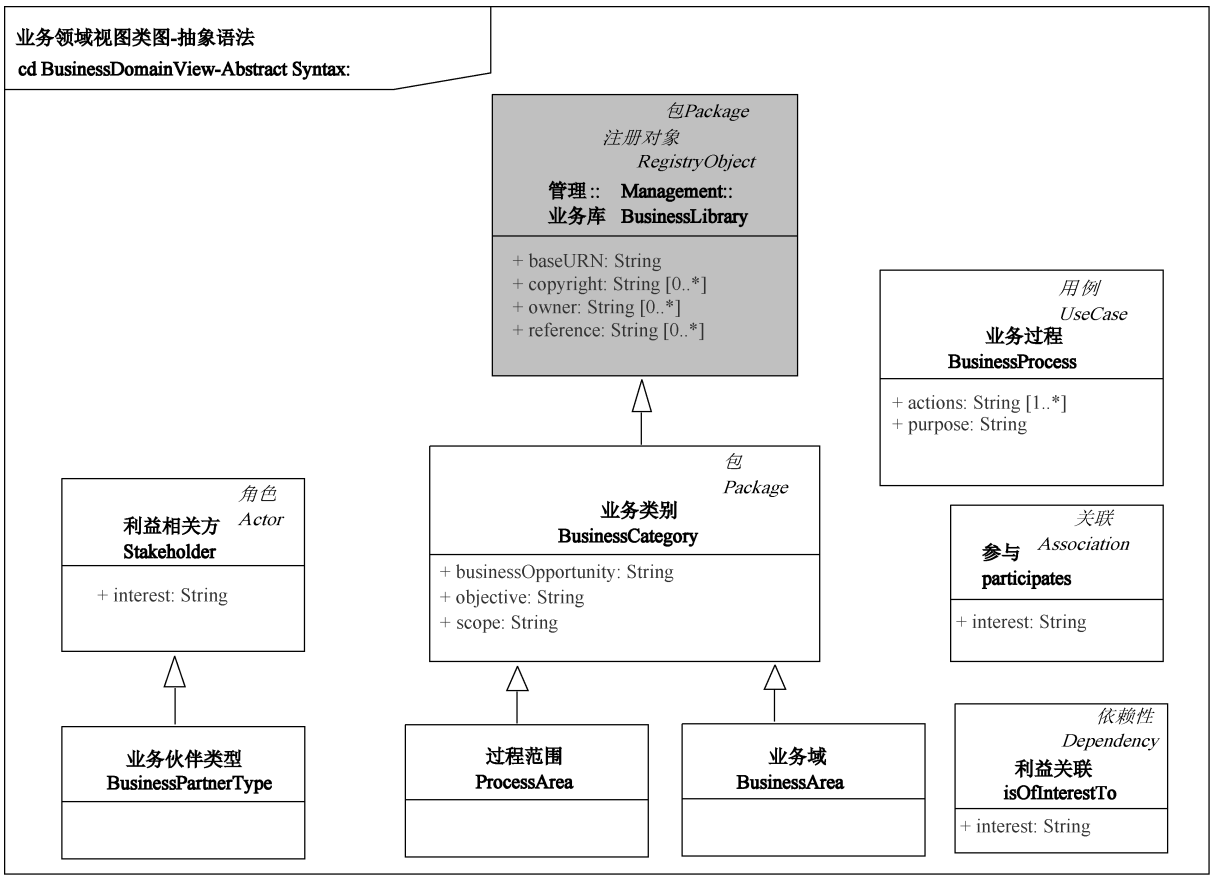

图 8　业务领域视图抽象语法

5.2.2.2 构造型和标签定义-业务类别，见表5。

表5 构造型和标签定义-业务类别

构造型(Stereotype)	业务类别(BusinessCategory)	
基类(Base Class)	程序包(Package)	
父类(Parent)	业务库包(BusinessLibraryPackage)[来自基础模块(Base Module)]	
描述(Description)	业务类别用于对业务领域视图中的业务过程进行分类。 业务领域视图可通过 *BusinessCategory* 构造型或其专用构造型 *BusinessArea* 和 *ProcessArea* 构成，*BusinessArea* 和 *ProcessArea* 构造型见其定义	
标签定义 (Tag Definition)	目标(objective)	
	类型(Type)	字符串(String)
	次数(Multiplicity)	1
	描述(Description)	通过业务类别中的业务过程所达到的目标
	范围(scope)	
	类型(Type)	字符串(String)
	次数(Multiplicity)	1
	描述(Description)	定义业务类别的范围
	商机(businessOpportunity)	
	类型(Type)	字符串(String)
	次数(Multiplicity)	1
	描述(Description)	从商业角度，描述业务类别的战略利益
	继承标签值(Inherited tagged value)： ——基础统一资源名称(baseURN) ——所有者(owner) ——版权(copyright) ——参考号(reference) ——版权(copyright) ——状态(status) ——业务术语(businessTerm)	

5.2.2.3 构造型和标签定义-业务域，见表6。

表6 构造型和标签定义-业务域

构造型(Stereotype)	业务域(BusinessArea)
基类(Base Class)	程序包(Package)
父类(Parent)	业务类别(BusinessCategory)
描述(Description)	业务域对应于组织中的一个业务部门。业务域可由其他业务域构成。业务域可以对其他业务域或过程域进行归类(业务域和过程域可以归类整理到一个业务域中)。 UMM不具体规定分类方案。UN/CEFACT通用业务过程目录(Common Business Process Catalog)推荐一个业务域分类列表，包括采购/销售、设计、制造、物流、招聘/培训、金融服务、法规、卫生保健。业务域分类应能无限扩展

表 6（续）

构造型(Stereotype)	业务域(BusinessArea)
标签定义 (Tag Definition)	**继承标签值(Inherited tagged value)：** ——目标(objective) ——范围(scope) ——参考号(reference) ——基础统一资源名称(baseURN) ——所有者(owner) ——版权(copyright) ——参考号(reference) ——版权(copyright) ——状态(status) ——业务术语(businessTerm)

5.2.2.4 构造型和标签定义-过程域，见表 7。

表 7 构造型和标签定义-过程域

构造型(Stereotype)	过程域(ProcessArea)
基类(Base Class)	程序包(Package)
父类(Parent)	业务类别(BusinessCategory)
描述(Description)	过程域对应于业务域内的一组常用操作，过程域可由其他过程域构成。过程域可以对其他过程域或业务域进行归类。 UMM 不具体规定分类方案。UN/CEFACT 通用业务过程目录(Common Business Process Catalog)推荐一个过程域分类列表，包括由 Open-edi 模型定义的规划、识别、谈判、实施、售后五个过程域
标签定义 (Tag Definition)	**继承标签值(Inherited tagged value)：** ——目标(objective) ——范围(scope) ——参考号(reference) ——基础统一资源名称(baseURN) ——所有者(owner) ——版权(copyright) ——参考号(reference) ——版权(copyright) ——状态(status) ——业务术语(businessTerm)

5.2.2.5 构造型和标签定义-利益相关方，见表 8。

表 8　构造型和标签定义-利益相关方

构造型(Stereotype)	利益相关方(Stakeholder)	
基类(Base Class)	参与者(Actor)	
父类(Parent)	无 (N/A)	
描述(Description)	利益相关方可以是个人也可以是组织代表,其在某一业务类别或业务过程结果中有既有利益。利益相关方不一定参与业务过程的执行	
标签定义 (Tag Definition)	利益(Interest)	
	类型(Type)	字符串(String)
	次数(Multiplicity)	1
	描述(Description)	描述在业务类别中定义的利益相关方的既有利益

5.2.2.6　构造型和标签定义-业务伙伴类型,见表 9。

表 9　构造型和标签定义-业务伙伴类型

构造型(Stereotype)	业务伙伴类型(BusinessPartnerType)
基类(Base Class)	参与者(Actor)
父类(Parent)	利益相关方(Stakeholder)
描述(Description)	业务伙伴类型为组织类型,即参与业务过程的组织部门类型或个人类型。业务伙伴类型通常向业务过程提供输入和/或接收来自业务过程输出
标签定义 (Tag Definition)	继承标签值(Inherited tagged value): ——利益(interest)

5.2.2.7　构造型和标签定义-业务过程,见表 10。

表 10　构造型和标签定义-业务过程

构造型(Stereotype)	业务过程(BusinessProcess)	
基类(Base Class)	用例(UseCase)	
父类(Parent)	无(N/A)	
描述(Description)	业务过程是业务伙伴共同创造价值的一组相关活动。业务过程可能由一个业务伙伴类型或跨组织间多个业务伙伴类型执行。可使用 UML 中定义的《include》和《extends》两个关联构造型将业务过程分为子过程	
标签定义 (Tag Definition)	定义(Definition)	
	类型(Type)	字符串(String)
	次数(Multiplicity)	1
	描述(Description)	定义业务过程,应描述业务过程所创造的客户价值。倘若业务过程由多个参与方执行,则应描述所有参与者创造的价值

表 10（续）

构造型(Stereotype)	业务过程(BusinessProcess)	
标签定义 (Tag Definition)	开始时间(beginsWhen)	
	类型(Type)	字符串(String)
	次数(Multiplicity)	1
	描述(Description)	规定触发业务过程开始的业务事件
	前置条件(preCondition)	
	类型(Type)	字符串(String)
	次数(Multiplicity)	1
	描述(Description)	规定执行业务过程应满足的条件。该条件应指业务实体生命周期中的状态。前置条件语句可使用布尔运算符表示多个业务实体状态的组合
	结束时间(endsWhen)	
	类型(Type)	字符串(String)
	次数(Multiplicity)	1
	描述(Description)	规定导致业务过程终止的业务事件
	后置条件(postCondition)	
	类型(Type)	字符串(String)
	次数(Multiplicity)	1
	描述(Description)	规定执行业务过程之后将达到的条件。该条件应指业务实体生命周期中的状态。后置条件语句可使用布尔运算符表示多个业务实体状态的组合
	异常(exceptions)	
	类型(Type)	字符串(String)
	次数(Multiplicity)	1.. *
	描述(Description)	识别导致业务过程正常执行时出现异常的情况
	动作(actions)	
	类型(Type)	字符串(String)
	次数(Multiplicity)	1.. *
	描述(Description)	构成业务过程的任务列表。在多参与方执行的业务过程中,需要特别强调接口任务。接口任务指需要与其他业务伙伴类型交互的业务过程步骤

5.2.2.8 构造型和标签定义-参与,见表 11。

表 11 构造型和标签定义-参与

构造型(Stereotype)	参与(Participates)
基类(Base Class)	关联(Association)

表 11（续）

构造型（Stereotype）	参与（Participates）	
父类（Parent）	无（N/A）	
描述（Description）	描述业务伙伴类型和业务过程之间的关联。该构造型定义业务伙伴类型向相关业务过程的输入和/或接收来自该业务过程的输出	
标签定义（Tag Definition）	利益（Interest）	
	类型（Type）	字符串（String）
	次数（Multiplicity）	1
	描述（Description）	描述由此参与-关联所关联的业务过程中业务伙伴类型的既有利益

5.2.2.9 构造型和标签定义-利益关联，见表 12。

表 12 构造型和标签定义-利益关联

构造型（Stereotype）	利益关联（isOfInterestTo）	
基类（Base Class）	依赖性（Dependency）	
父类（Parent）	无（N/A）	
描述（Description）	描述业务过程对利益相关方的依赖性。该构造型定义了业务过程依赖于其利益相关方的利益	
标签定义（Tag Definition）	利益（Interest）	
	类型（Type）	字符串（String）
	次数（Multiplicity）	1
	描述（Description）	描述由此参与-依赖性所连接的业务过程的利益相关方的既有利益

5.2.3 约束（规范性）

5.2.3.1 业务领域视图（BusinessDomainView）程序包应包含至少一个业务类别（BusinessCategory）程序包或至少一个业务域（BusinessArea）程序包。此外，业务领域视图（BusinessDomainView）可包含利益相关方（Stakeholders）和业务伙伴类型（BusinessPartnerTypes）。业务领域视图（BusinessDomainView）不应包含业务类别（BusinessCategory）和业务域（BusinessArea）程序包的组合。

示例：

```
package Model_Management
context Package

inv isBusinessDomainViewPackage:
  self.isBusinessDomainView() implies
  self.contents->notEmpty() and (
  self.contents->forAll(isJustBusinessCategory() or isStakeholderOrBusinessPartnerType()) or
  self.contents->forAll(isBusinessArea() or isStakeholderOrBusinessPartnerType()))
```

5.2.3.2 业务域（BusinessArea）程序包应包含一个或多个业务域（BusinessArea）程序包，或者一个或多个过程域（ProcessArea）程序包，不应包含业务域（BusinessArea）和过程域（ProcessArea）程序包组合。但可包含业务伙伴类型（BusinessPartnerTypes）和利益相关方（Stakeholders）。

示例：

```
package Model_Management
context Package

inv contentsOfBusinessArea:
  self.isBusinessArea() implies
  self.contents->notEmpty() and (
  self.contents->forAll(isProcessArea()
  or isStakeholderOrBusinessPartnerType())
  or self.contents->forAll(isBusinessArea() or isStakeholderOrBusinessPartnerType()))
```

5.2.3.3 过程域(ProcessArea)或包含一个或多个其他过程域(ProcessAreas)以及零或多个业务伙伴类型(BusinessPartnerTypes)和利益相关方(Stakeholders),或应包含至少一个业务过程(BusinessProcess)和可包含业务伙伴类型(BusinessPartnerTypes)、利益相关方(Stakeholders)、构造型关联参与(participates),以及构造型依赖性利益相关方(isOfInterestTo)。

示例：

```
package Model_Management
context Package

inv contentsOfProcessArea:
  self.isProcessArea() implies
  self.contents->notEmpty and
  (self.contents->forAll(isProcessArea() or isStakeholderOrBusinessPartnerType()) or
  (self.contents->forAll(isBusinessProcess() or isBusinessPartnerType() or
  isStakeholder() or isParticipates() or isIsOfInterestTo()) and
  self.contents->select(isBusinessProcess())->size()>= 1))
```

5.2.3.4 业务类别(BusinessCategory)或包含一个或多个其他业务类别(BusinessCategories)以及零或多个业务伙伴类型(BusinessPartnerTypes)和利益相关方(Stakeholders),或者应包含至少一个业务过程(BusinessProcess)和可包含业务伙伴类型(BusinessPartnerTypes)、利益相关方(Stakeholders)、构造型关联参与(participates),以及构造型依赖性利益相关方(isOfInterestTo)。

示例：

```
package Model_Management
context Package

inv contentsOfBusinessCategory:
  self.isBusinessCategory() implies
  self.contents->notEmpty and (self.contents->forAll(isBusinessCategory() or
  isStakeholderOrBusinessPartnerType()) or
  (self.contents->forAll(isBusinessProcess()
  or isBusinessPartnerType() or
  isStakeholder() or isParticipates() or isIsOfInterestTo()) and
  self.contents->select(isBusinessProcess())->size()>= 1))
```

5.2.3.5 作为业务类别(BusinessCategory)[或其专用构造型过程域(ProcessArea)]一部分的参与(participates)关联应始终连接业务伙伴类型(BusinessPartnerType)和业务过程(BusinessProcess)。

示例：

```
package Foundation::Core
context Association

inv isParticipatesConnector:
  (self.isParticipates() and self.namespace.isBusinessCategory())implies
  self.allConnections->size() = 2 and
  self.allConnections->one(isBusinessProcess()) and
  self.allConnections->one(isBusinessPartnerType())
```

5.2.3.6 利益相关方(isOfInterestTo)的依赖性(dependency)应从业务过程(BusinessProcess)向利益相关方(Stakeholder)而建立。

示例：

```
package Foundation::Core
context Dependency

inv isIsOfInterestTo:
  self.isIsOfInterestTo() implies
  self.client->one(isBusinessProcess()) and
  self.supplier->one(isStakeholder()) and
  self.client->size() = 1 and
  self.supplier->size() = 1
```

5.2.4 实例（资料性）

业务领域视图(BusinessDomainView)实例：协商(报价单)，见图 9。

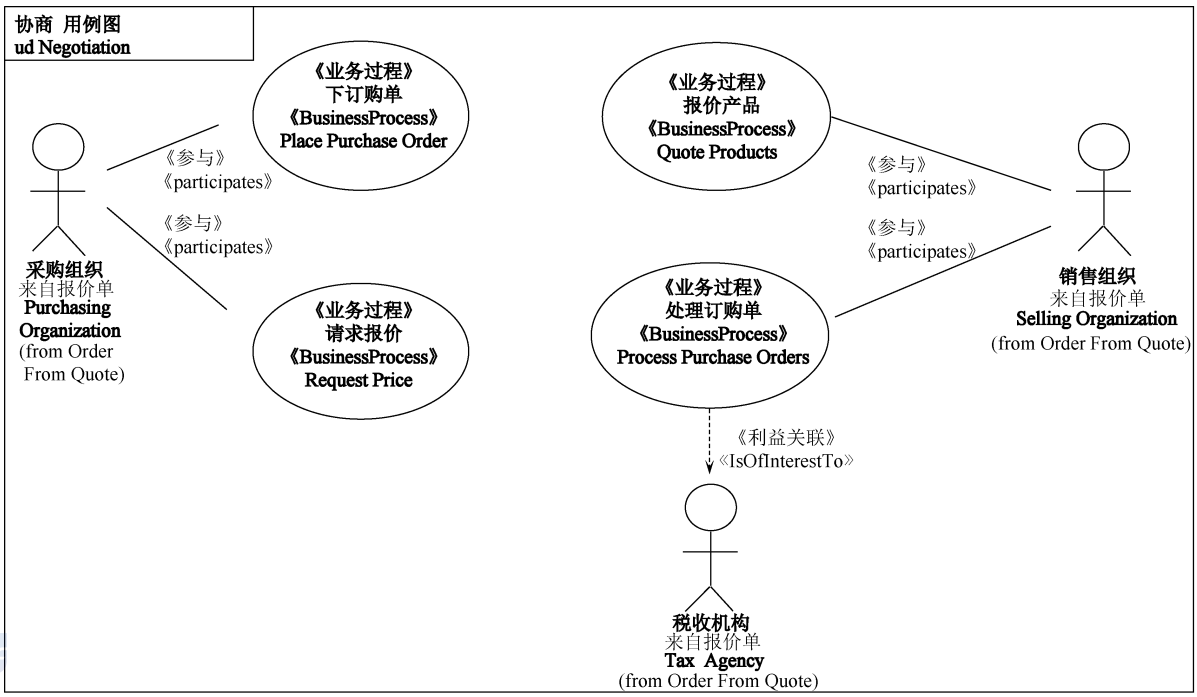

图 9 业务领域视图实例：协商(报价单)

5.2.5 BDV 所有程序包中使用的 OCL 方法(规范性)

OCL 方法,参见示例。

示例:

```
package Foundation::Core
context ModelElement

-- checks if a model element has a certain stereotype
def:
let hasStereotype (st : String) : Boolean =
  self.stereotype->select(self.name = st)->notEmpty()
-- checks if a Package is stereotyped as
-- BusinessDomainView
def:
let isBusinessDomainView() : Boolean =
  self.oclIsKindOf(Package) and
  self.hasStereotype('BusinessDomainView')
-- checks if a Package is a BusinessCategory. This includes
-- also BusinessAreas and ProcessAreas due to the inheritance hierachy
-- in the metamodel
def:
let isBusinessCategory() : Boolean =
  self.oclIsKindOf(Package) and (
  self.hasStereotype('BusinessCategory') or
  isBusinessArea() or
  isProcessArea()
  )
-- checks if an Association is stereotyped as participates
def:
let isParticipates() : Boolean =
  self.oclIsKindOf(Association) and
  self.hasStereotype('participates')
-- checks if an Association is stereotyped as isInterestOf
def:
let isIsOfInterestTo() : Boolean =
  self.oclIsKindOf(Dependency) and
  self.hasStereotype('isOfInterestTo')
-- checks if a package is a ProcessArea
def:
let isProcessArea() : Boolean =
  self.oclIsKindOf(Package) and
  self.hasStereotype('ProcessArea')
-- checks if a package is a BusinessArea
def:
let isBusinessArea() : Boolean =
  self.oclIsKindOf(Package) and
```

```
    self.hasStereotype('BusinessArea')
-- checks if an Actor is a BusinessPartnerType
def :
let isBusinessPartnerType() : Boolean =
    self.oclIsTypeOf(Actor) and
    self.hasStereotype('BusinessPartnerType')
-- checks if an Actor is a Stakeholder
def:
let isStakeholder() : Boolean =
    self.oclIsTypeOf(Actor) and (
    self.hasStereotype('Stakeholder') or
    isBusinessPartnerType()
)
--checks if an Actor is a BusinessPartnerType or a Stakeholder
def :
let isStakeholderOrBusinessPartnerType() : Boolean =
    self.isStakeholder() or self.isBusinessPartnerType()
-- checks if a UseCase is stereotyped as BusinessProcess
def :
let isBusinessProcess() : Boolean =
    self.oclIsTypeOf(UseCase) and
    self.hasStereotype('BusinessProcess')
```

5.3 业务需求视图

5.3.1 需求视图中的子视图

5.3.1.1 概念性描述(资料性)

5.3.1.1.1 业务需求视图用于确定不同业务伙伴类型间的协作业务过程并描述其需求,业务需求视图(BusinessRequirementsView)程序包包含三个不同构件,其中:

——业务过程视图描述业务领域视图内发现的业务过程的活动流和状态,虽然业务过程视图不是必备项,但业务需求视图可以由多个业务过程视图组成。因此业务需求视图(BusinessRequirementsView)包含零或多个业务过程视图(BusinessProcessView)。

——业务实体视图描述协作业务过程中操作的业务实体生命周期,同样地,业务实体视图也是可选择项可以重复。所以业务需求视图(BusinessRequirementsView)包含零或多个业务实体视图(BusinessEntityView)。

——业务需求视图包含伙伴关系需求视图,伙伴关系需求视图描述业务伙伴类型中的合作伙伴需求,最底层不可再分的伙伴关系是业务交易。业务协作是由业务交易和/或其他业务协作建立的伙伴关系。交易需求视图描述业务交易需求,协作需求视图描述业务协作需求。相同业务协作可由一组不同业务伙伴类型执行。协作实现视图描述一组特定业务伙伴类型实现业务协作用例的需求。伙伴关系需求视图(PartnershipRequirementsView)是一个抽象概念,通过交易需求视图(TransactionRequirementsView)、协作需求视图(CollaborationRequirementsView)或协作实现视图(CollaborationRealizationView)实现。业务需求视图至少描述一个业务协作,业务协作包含至少一个业务交易,并由一组业务伙伴类型执行。因此,业务需求视图(BusinessRequirementsView)包含一到多个协作需求视图(CollaborationRequirementsView),一个或多个交易需求视图(TransactionRequirementsView),一个或多个协作实现视图(CollaborationRealizationView)。

5.3.1.1.2 业务需求(BusinessRequirements)视图概念框架,见图 10。

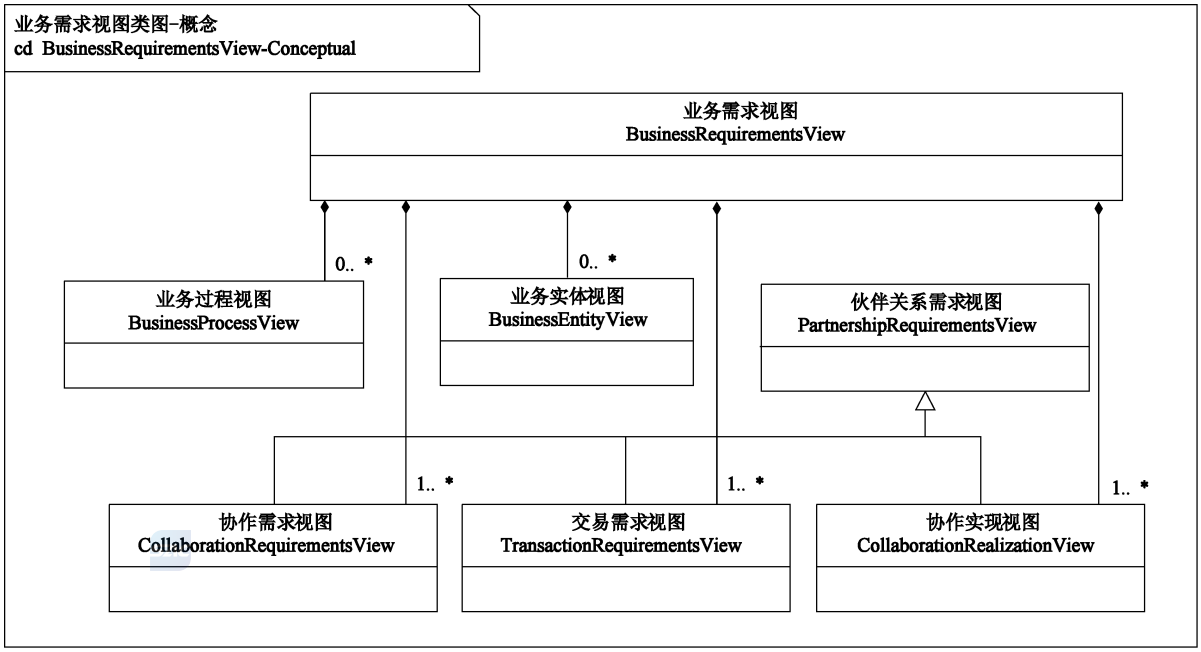

图 10 业务需求视图概念框架

5.3.1.2 构造型和标签定义(规范性)

5.3.1.2.1 业务需求视图(BusinessRequirementsView)抽象语法 ,见图 11。

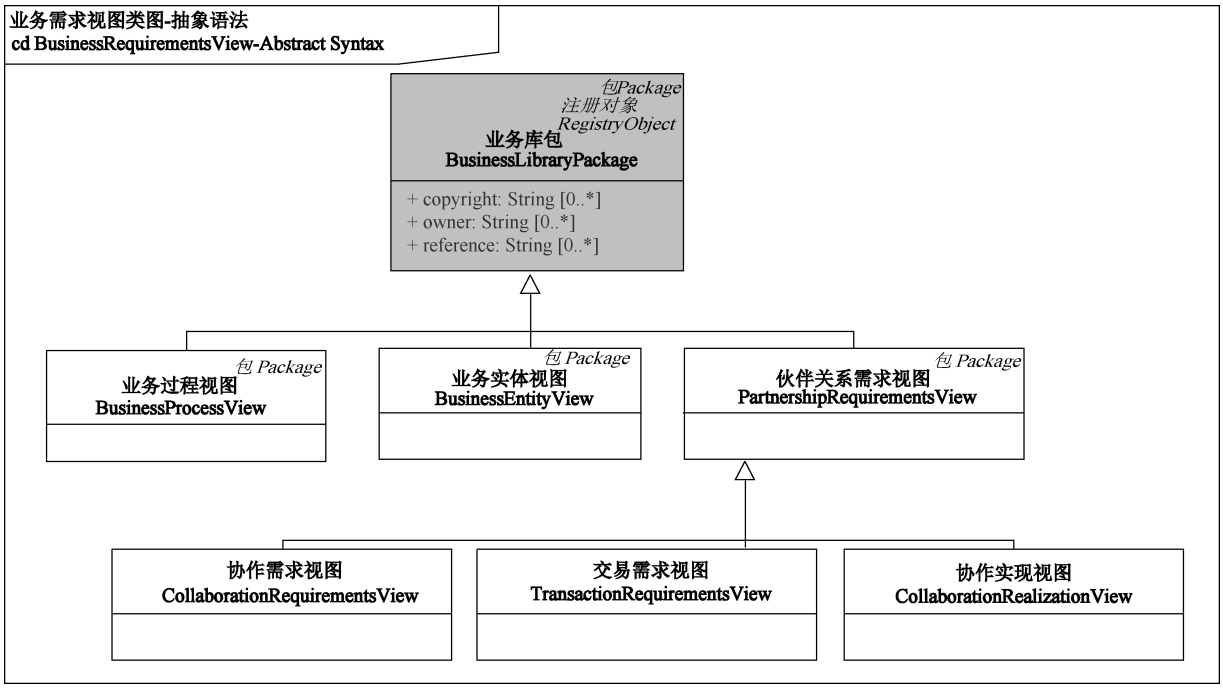

图 11 业务需求视图抽象语法

5.3.1.2.2 造型和标签定义-业务过程视图,见表 13。

表 13　构造型和标签定义-业务过程视图

构造型(Stereotype)	业务过程视图(BusinessProcessView)
基类(Base Class)	程序包(Package)
父类(Parent)	业务库包(BusinessLibraryPackage)[来自基础模块(Base Module)]
描述(Description)	业务过程视图是一个容器,用于描述一个业务伙伴类型的内部业务过程行为或连接不同业务伙伴类型内部过程的业务过程行为
标签定义 (Tag Definition)	**继承标签值(Inherited tagged value):** ——基础统一资源名称(baseURN) ——所有者(owner) ——版权(copyright) ——参考号(reference) ——版权(copyright) ——状态(status) ——业务术语(businessTerm)

5.3.1.2.3　构造型和标签定义-业务实体视图,见表 14。

表 14　构造型和标签定义-业务实体视图

构造型(Stereotype)	业务实体视图(BusinessEntityView)
基类(Base Class)	程序包(Package)
父类(Parent)	业务库包(BusinessLibraryPackage)[来自基础模块(Base Module)]
描述(Description)	业务实体视图是一个容器,用于描述在建模域内有业务意义的业务实体,包含其业务实体生命周期和业务实体状态
标签定义 (Tag Definition)	**继承标签值(Inherited tagged value):** ——基础统一资源名称(baseURN) ——所有者(owner) ——版权(copyright) ——参考号(reference) ——版权(copyright) ——状态(status) ——业务术语(businessTerm)

5.3.1.2.4　构造型和标签定义-伙伴关系需求视图(抽象),见表 15。

表 15　构造型和标签定义-伙伴关系需求视图(抽象)

构造型(Stereotype)	伙伴关系需求视图(PartnershipRequirementsView)
基类(Base Class)	程序包(Package)
父类(Parent)	业务库包(BusinessLibraryPackage)[来自基础模块(Base Module)]

表 15（续）

构造型(Stereotype)	伙伴关系需求视图(PartnershipRequirementsView)
描述(Description)	伙伴需求关系需求视图是一个容器，描述业务伙伴类型之间的伙伴关系需求。这些需求或应用于业务协作、业务交易，或应用于业务协作实现。伙伴关系需求视图分为三个专用构造型，即协作需求视图、交易需求视图和协作实现视图。伙伴关系需求视图是抽象构造型，应使用三者之一
标签定义 (Tag Definition)	**继承标签值(Inherited tagged value)：** ——基础统一资源名称(baseURN) ——所有者(owner) ——版权(copyright) ——参考号(reference) ——版权(copyright) ——状态(status) ——业务术语(businessTerm)

5.3.1.2.5 构造型和标签定义-协作需求视图，见表 16。

表 16 构造型和标签定义-协作需求视图

构造型(Stereotype)	协作需求视图(CollaborationRequirementsView)
基类(Base Class)	程序包(Package)
父类(Parent)	伙伴关系需求视图(PartnershipRequirementsView)
描述(Description)	协作需求视图是一个容器，描述授权角色之间业务协作的需求
标签定义 (Tag Definition)	**继承标签值(Inherited tagged value)：** ——基础统一资源名称(baseURN) ——所有者(owner)-版权(copyright) ——参考号(reference) ——版权(copyright) ——状态(status) ——业务术语(businessTerm)

5.3.1.2.6 构造型和标签定义-交易需求视图，见表 17。

表 17 构造型和标签定义-交易需求视图

构造型(Stereotype)	交易需求视图(TransactionRequirementsView)
基类(Base Class)	程序包(Package)
父类(Parent)	伙伴关系需求视图(PartnershipRequirementsView)
描述(Description)	交易需求视图是一个容器，描述授权角色之间业务交易需求
标签定义 (Tag Definition)	**继承标签值(Inherited tagged value)：** ——基础统一资源名称(baseURN) ——所有者(owner)- 版权(copyright) ——参考号(reference) ——版权(copyright) ——状态(status) ——业务术语(businessTerm)

5.3.1.2.7 构造型和标签定义-协作实现视图，见表18。

表18 构造型和标签定义-协作实现视图

构造型(Stereotype)	协作实现视图(CollaborationRealizationView)
基类(Base Class)	程序包(Package)
父类(Parent)	伙伴关系需求视图(PartnershipRequirementsView)
描述(Description)	协作实现视图是一个容器，描述业务协作用例中由业务伙伴类型实现的需求
标签定义 (Tag Definition)	继承标签值(Inherited tagged value)： ——基础统一资源名称(baseURN) ——所有者(owner)- 版权(copyright) ——参考号(reference) ——版权(copyright) ——状态(status) ——业务术语(businessTerm)

5.3.1.3 约束(规范性)

业务需求视图(BusinessRequirementsView)应至少包含一个协作需求视图(CollaborationRequirementsView)程序包，至少包含一个交易需求视图(TransactionRequirementsView)程序包，至少包含一个协作实现视图(TransactionRequirementsView)。此外，可包含业务过程视图(BusinessProcessView)程序包和业务实体视图(BusinessEntityView)程序包。除此之外，不应包含任何其他元素。

示例：

```
package Model_Management
context Package

inv packagesAllowedInBRV:
   self.isBusinessRequirementsView() implies
   self.contents->forAll(isBusinessProcessView() or
   isBusinessEntityView() or
   isCollaborationRequirementsView() or
   isTransactionRequirementsView() or
   isCollaborationRealizationView()) and
   self.contents->exists(isCollaborationRequirementsView) and
   self.contents->exists(isTransactionRequirementsView) and
   self.contents->exists(isCollaborationRealizationView)
```

5.3.2 业务过程视图

5.3.2.1 概念性描述(资料性)

5.3.2.1.1 业务过程视图描述了业务过程、业务过程活动和执行活动的业务伙伴类型。业务过程视图(BusinessProcessView)由一个或多个业务过程(BusinessProcess)构成，一个业务过程可能包含其他业务过程或是其他业务过程的扩展。

5.3.2.1.2 业务过程活动模型(BusinessProcessActivityModel)表示业务过程的动态行为，业务过程(BusinessProcess)中包含零或一个业务过程活动模型(BusinessProcessActivityModel)。业务过程活动

模型描述了由一个或多个参与方实施的活动流。涉及两个以上的业务伙伴类型协作的业务过程活动模型将分区。一个业务过程活动模型(BusinessProcessActivityModel)中包含零或多个分区(Partitions)(UML 共享元素),一个分区指定给一个业务伙伴类型。一个业务伙伴类型可对应于处于不同活动模型中的多个分区,业务伙伴类型 (BusinessPartnerType)和分区(Partitions)之间以 1 至(0..n)的关联来表示。

5.3.2.1.3 有多个分区的,一个业务过程活动将被分配给执行该活动的业务伙伴类型所处的分区。只要通过分区连接两种业务过程活动,就需要协作业务过程。一个业务过程活动模型(BusinessProcessActivityModel)是由一个或多个业务过程活动(BusinessProcessActivity)构成,或作为业务过程活动模型一部分的一个分区由一个或多个业务过程活动(BusinessProcessActivity)构成。一个业务过程活动可通过另一个业务过程活动模型细化。一项业务过程活动(BusinessProcessActivity)中包含零或含有一个业务过程活动模型(BusinessProcessActivityModel),反之是零个或一个业务过程活动(BusinessProcessActivity)的组合。

5.3.2.1.4 一个业务过程活动模型还可表示执行业务过程期间被影响的业务实体状态。业务实体状态是一个业务活动的输出,并输入到另一个业务活动。一个业务过程活动到业务实体状态的转换是标志一个输出,而一个业务实体状态到业务过程活动的转换则标志一个输入。业务过程活动模型包含内部业务实体状态和共享业务实体状态,内部实体状态只对一个业务伙伴类型有意义。业务过程活动模型(BusinessProcessActivityModel)中包含零或多个内部业务实体状态(InternalBusinessEntityStates)或共享业务实体状态(SharedBusinessEntityStates)。倘若一个业务过程活动模型有多个分区,则创建和使用内部业务实体状态的两个业务过程活动处于同一分区,反之,创建和使用共享业务实体状态的两个业务过程活动处于不同分区,共享业务实体状态标志着协作业务过程的需求。

5.3.2.1.5 业务过程视图(业务需求视图)[BusinessProcessView(BusinessRequirementsView)]概念框架,见图 12。

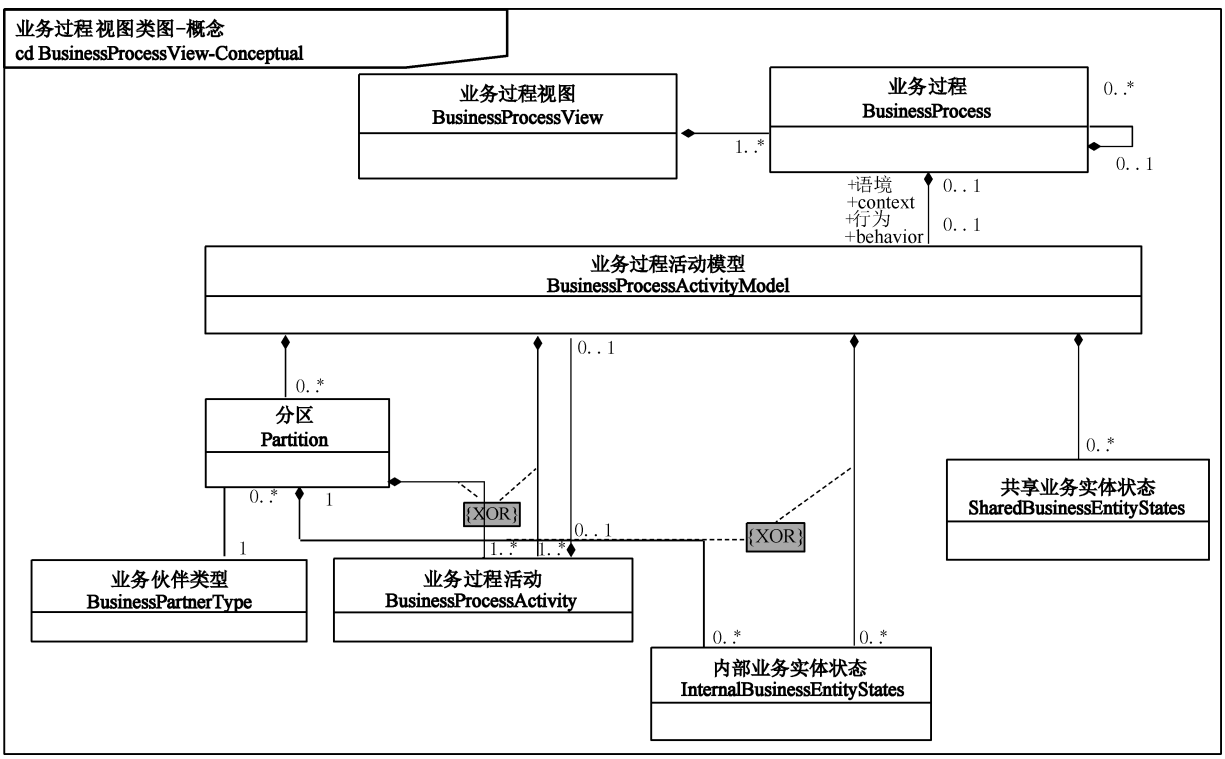

图 12 业务过程视图(业务需求视图)概念框架

5.3.2.2 构造型和标签定义(规范性)

5.3.2.2.1 业务过程视图(业务需求视图)[BusinessProcessView(BusinessRequirementsView)]抽象语法,见图13。

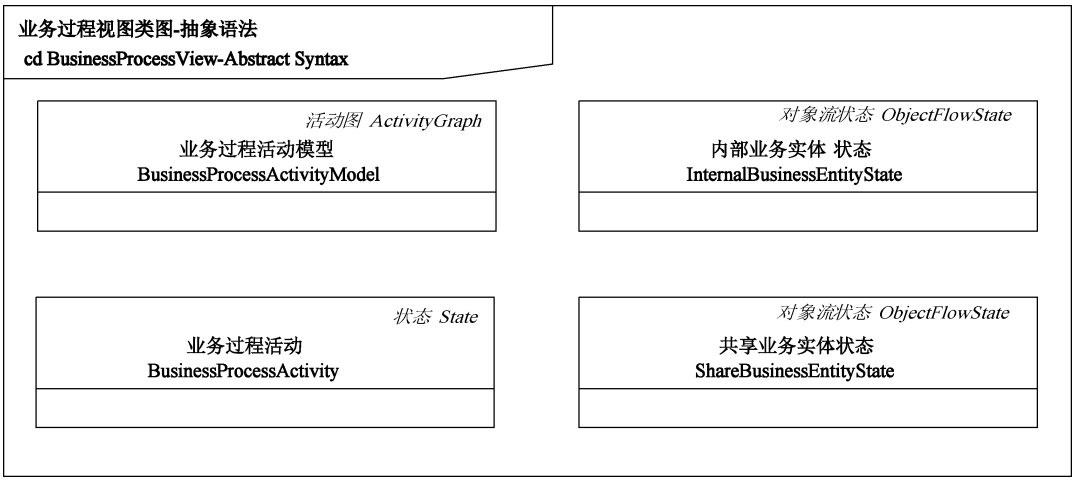

图 13 业务过程视图(业务需求视图)抽象语法

5.3.2.2.2 构造型和标签定义-业务过程活动模型,见表19。

表 19 构造型和标签定义-业务过程活动模型

构造型(Stereotype)	业务过程活动模型(BusinessProcessActivityModel)
基类(Base Class)	活动图(ActivityGraph)
父类(Parent)	无(N/A)
描述(Description)	业务过程活动模型描述了业务伙伴类型参与的业务过程行为,是确定两个或更多业务伙伴类型之间协作需求的工具,业务过程活动模型与业务领域视图中确定的业务过程相关联,并描述该业务过程的动态行为
标签定义(Tag Definition)	无标签值

5.3.2.2.3 构造型和标签定义-业务过程活动,见表20。

表 20 构造型和标签定义-业务过程活动

构造型(Stereotype)	业务过程活动(BusinessProcessActivity)
基类(Base Class)	状态(State)
父类(Parent)	无(N/A)
描述(Description)	业务过程活动对应于业务过程活动模型中执行的一个步骤,可以由另一个业务过程活动模型进一步细化。业务过程活动的UML基类不是最小级的动作状态,而是一个对动作状态和组合状态的一般性描述的状态
标签定义(Tag Definition)	无标签值

5.3.2.2.4 构造型和标签定义-内部业务实体状态，见表21。

表21 构造型和标签定义-内部业务实体状态

构造型(Stereotype)	内部业务实体状态(InternalBusinessEntityStates)
基类(Base Class)	对象流状态(ObjectFlowState)
父类(Parent)	无(N/A)
描述(Description)	内部业务实体状态表示业务实体的一种状态，对于某种业务伙伴类型的业务过程而言是内部的
标签定义(Tag Definition)	无标签值

5.3.2.2.5 构造型和标签定义-共享业务实体状态，见表22。

表22 构造型和标签定义-共享业务实体状态

构造型(Stereotype)	共享业务实体状态(SharedBusinessEntityStates)
基类(Base Class)	对象流状态(ObjectFlowState)
父类(Parent)	无(N/A)
描述(Description)	共享业务实体状态表示业务实体的一种状态，在涉及两个业务伙伴类型的业务过程间是共享
标签定义(Tag Definition)	无标签值

5.3.2.3 约束(规范性)

5.3.2.3.1 业务过程视图(BusinessProcessView)应仅包含业务过程活动模型(BusinessProcessActivityModel)、业务伙伴类型 (BusinessPartnerType)以及业务过程(BusinessProcesses)，且应为空。

示例：

```
package Model_Management
context Package

inv AllowedElementsInBusinessProcessView:
  self.isBusinessProcessView() implies
  self.contents->forAll(isBusinessProcessActivityModel() or
  isBusinessPartnerType() or
  isBusinessProcess()) and
  self.contents->notEmpty()
```

5.3.2.3.2 不含分区的业务过程活动模型(BusinessProcessActivityModel)应含有一个或多个业务过程活动(BusinessProcessActivity)，且可含有内部业务实体状态(InternalBusinessEntityStates)、共享业务实体状态(SharedBusinessEntityStates)、伪态(Pseudo states)、最终状态(Final states)以及转换态(Transitions)。

示例：

```
package Behavioral_Elements::State_Machines
context CompositeState

inv AllowedElementsInBusinessProcessActivityModelWithoutPartition:
  (self.stateMachine.isBusinessProcessActivityModel() and
  self.stateMachine.oclAsType(ActivityGraph).partition->isEmpty()) implies
  self.subvertex->notEmpty() and
  self.subvertex->exists(isBusinessProcessActivity()) and
  self.subvertex->forAll(isBusinessProcessActivity() or
  isInternalBusinessEntityState() or
  isSharedBusinessEntityState() or
  isPseudoStateOrFinalStateOrTransition())
```

5.3.2.3.3 业务过程活动模型(BusinessProcessActivityModel)中的分区应含有一个或多个业务过程活动(BusinessProcessActivity)，且可含有内部业务实体状态(InternalBusinessEntityStates)、伪态(Pseudo states)、最终状态(Final states)以及转换态(Transitions)。

示例：

```
package Behavioral_Elements::Activity_Graphs
context Partition

inv AllowedModelElementsInBusinessProcessActivityModelPartition:
self.isPartition() implies
  self.contents->forAll(isBusinessProcessActivity()
  or isInternalBusinessEntityState()
  or isPseudoStateOrFinalStateOrTransition()
  ) and
  self.contents->exists(isBusinessProcessActivity())
```

5.3.2.4 实例（资料性）

业务过程视图（业务需求视图)[BusinessProcessView(BusinessRequirementsView)]实例：采购产品-协作视图业务过程活动模型(活动图)[BusinessProcessActivityModel (ActivityGraph)]，见图 14。

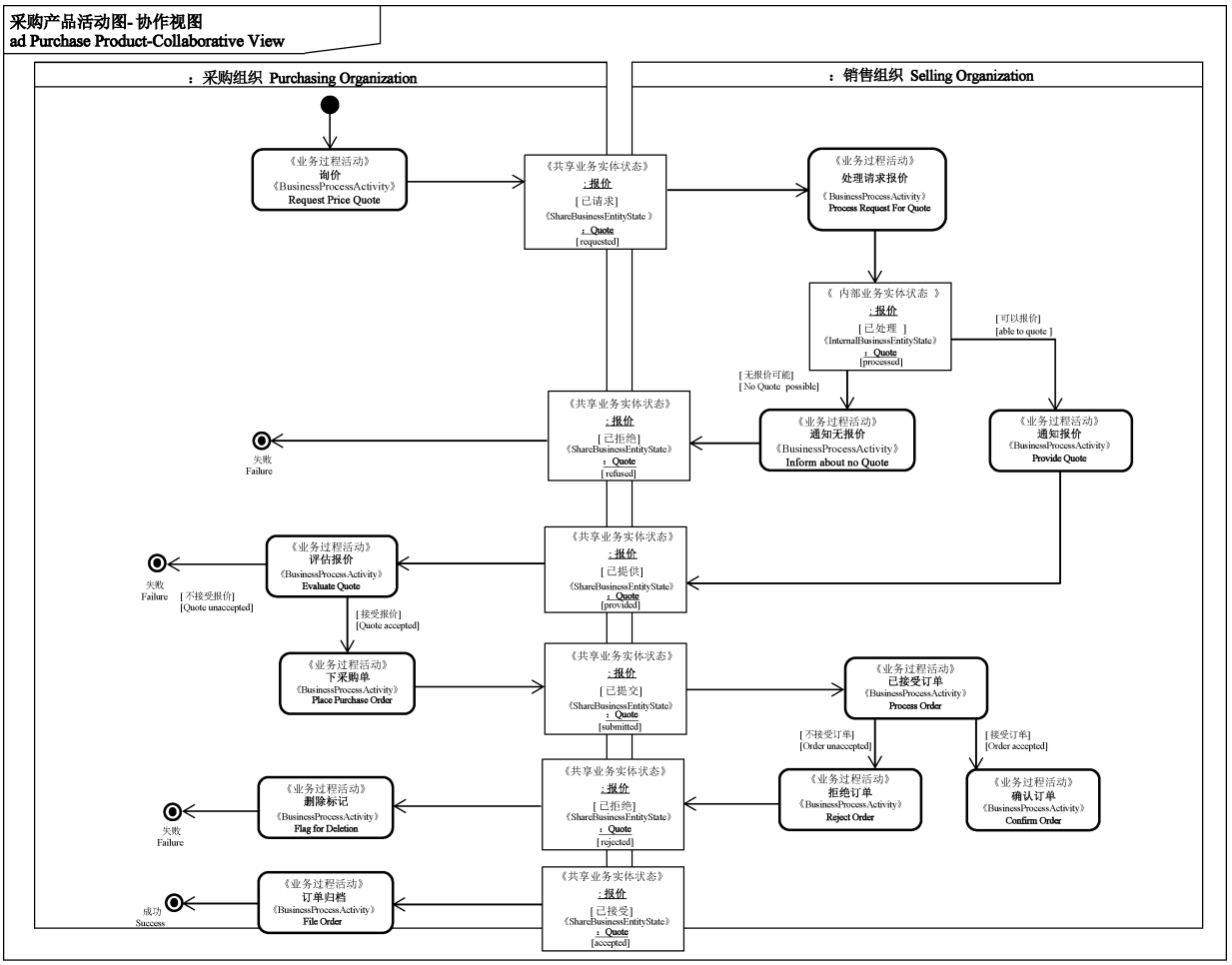

图 14　业务过程视图(业务需求视图)实例：采购产品-协作视图业务过程活动模型(活动图)

5.3.3　业务实体视图

5.3.3.1　概念性描述(资料性)

5.3.3.1.1　sinessEntity(业务实体)是实际应用中具有明确业务意义的实体，在协作业务过程(如“订单”“账目”等)中由两个或多个业务伙伴类型（BusinessPartnerType)共用。业务实体视图(BusinessEntityView)由一个至多个业务实体(BusinessEntity)构成。业务实体(BusinessEntity)可不包含业务实体生命周期(BusinessEntityLifecycle)，也可包含一个业务实体生命周期(BusinessEntityLifecycle)。业务实体生命周期表示一个业务实体在一个完整业务过程中可以不同状态存在，业务实体生命周期(BusinessEntityLifecycle)包含一个或多个业务实体状态(BusinessEntityState)。和其他 UML 状态机相似，业务实体生命周期也包含事件和状态转换触发业务实体从一个状态转换到另一个状态。

5.3.3.1.2　业务实体视图(业务需求视图)[BusinessEntityView(BusinessRequirementsView)]概念框架，见图 15。

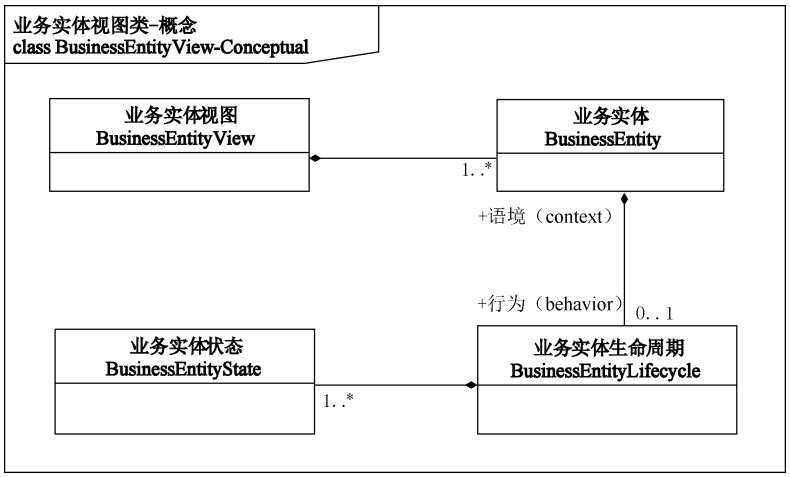

图 15　业务实体视图(业务需求视图)概念框架

5.3.3.2　构造型和标签定义(规范性)

5.3.3.2.1　业务实体视图(业务需求视图)[BusinessEntityView(BusinessRequirementsView)]抽象语法,见图 16。

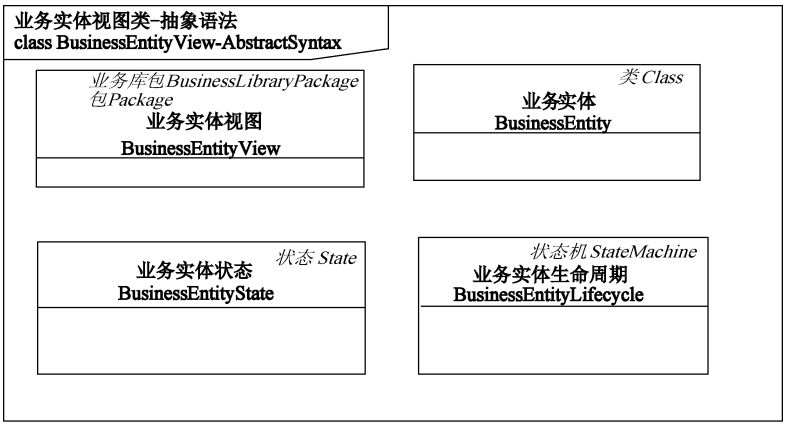

图 16　业务实体视图(业务需求视图)抽象语法

5.3.3.2.2　构造型和标签定义-业务实体,见表 23。

表 23　构造型和标签定义-业务实体

构造型(Stereotype)	业务实体(BusinessEntity)
基类(Base Class)	类(Class)
父类(Parent)	无(N/A)
描述(Description)	业务实体在协作业务过程(如“订单”“账目”等)中由两个或更多业务伙伴类型所共用
标签定义(Tag Definition)	无标签值

5.3.3.2.3　构造型和标签定义-业务实体生命周期,见表 24。

表 24　构造型和标签定义-业务实体生命周期

构造型(Stereotype)	业务实体生命周期(BusinessEntityLifecycle)
基类(Base Class)	状态机(StateMachine)
父类(Parent)	无(N/A)
描述(Description)	业务实体生命周期表示在一个完整业务过程中一个业务实体可以不同状态存在,且表示业务实体从一个状态至另一个状态转换的事件和过渡
标签定义(Tag Definition)	无标签值

5.3.3.2.4　构造型和标签定义-业务实体状态,见表 25。

表 25　构造型和标签定义-业务实体状态

构造型(Stereotype)	业务实体状态(BusinessEntityState)
基类(Base Class)	状态(State)
父类(Parent)	无(N/A)
描述(Description)	业务实体状态表示在生命周期中一个业务实体可以某种状态存在(如“订单”可以有“发出”“拒绝”“确认”等状态中)
标签定义(Tag Definition)	无标签值

5.3.3.3　约束(规范性)

5.3.3.3.1　业务实体视图(BusinessEntityView)应且仅包含业务实体(BusinessEntity)。

示例:

```
package Model_Management
context Package

inv AllowedElementsInBusinessEntityView:
  self.isBusinessEntityView() implies
  self.contents->notEmpty() and
  self.contents->forAll(isBusinessEntity())
```

5.3.3.3.2　业务实体(BusinessEntity)不包含或只含有一个业务实体生命周期(BusinessEntityLifecycle)来表示其行为。

示例:

```
package Foundation::Core
context Class

inv LifecyclesOfBusinessEntity:
  self.isBusinessEntity() implies
  self.behavior->select(isBusinessEntityLifecycle())->size()<=1
```

5.3.3.3.3　业务实体生命周期(BusinessEntityLifecycle)应且仅含有业务实体状态(BusinessEntityState)、伪态(PseudoState)、最终状态(FinalState)或转换(Transition)。

示例：

```
package Behavioral_Elements::State_Machines
context CompositeState

inv ContainsOnlyBusinessEntityStates:
  self.stateMachine.isBusinessEntityLifecycle() implies
  self.subvertex->forAll(isBusinessEntityState() or
  isPseudoStateOrFinalStateOrTransition())
  and self.subvertex->exists(isBusinessEntityState())
```

5.3.3.4 实例(资料性)

5.3.3.4.1 业务实体视图(业务需求视图)[BusinessEntityView (BusinessRequirementsView)]的实例：业务实体(BusinessEntities)报价和订单类图(ClassDiagram)，见图 17。

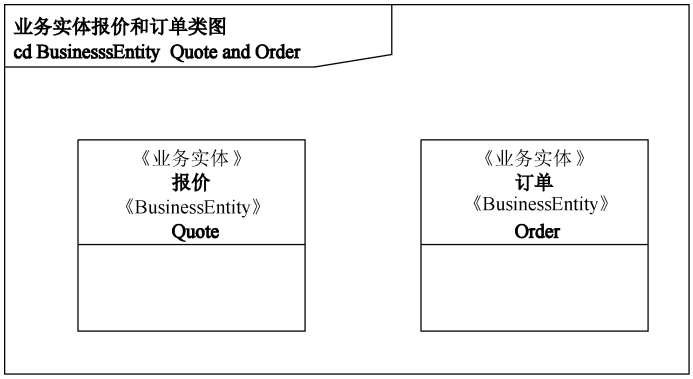

图 17 业务实体视图(业务需求视图)实例：业务实体报价和订单(类图)

5.3.3.4.2 业务实体视图(业务需求视图)[BusinessEntityView (BusinessRequirementsView)]的实例：报价业务实体生命周期(状态机)[BusinessEntityLifecycle (StateMachine)]，见图 18。

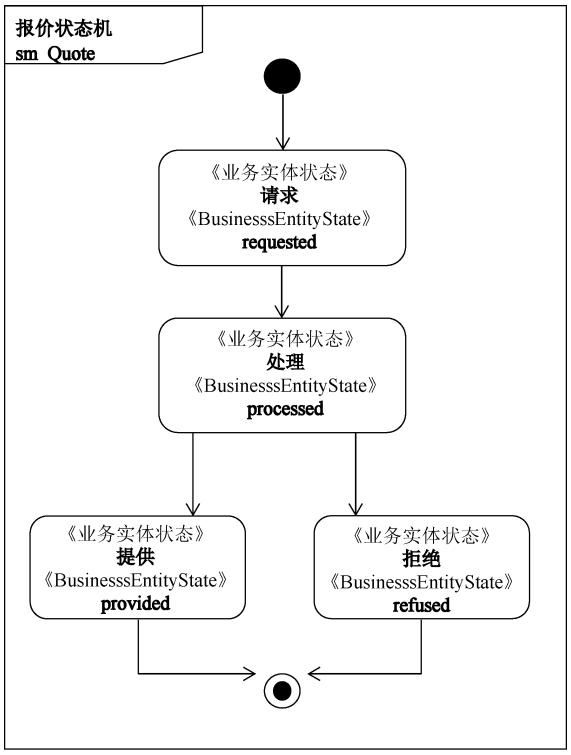

图 18　业务实体视图(业务需求视图)实例:报价[业务实体生命周期(状态机)]

5.3.3.4.3　业务实体视图(业务需求视图)[BusinessEntityView(BusinessRequirementsView)]的实例:订单业务实体生命周期(状态机)[BusinessEntityLifecycle (StateMachine)],见图 19。

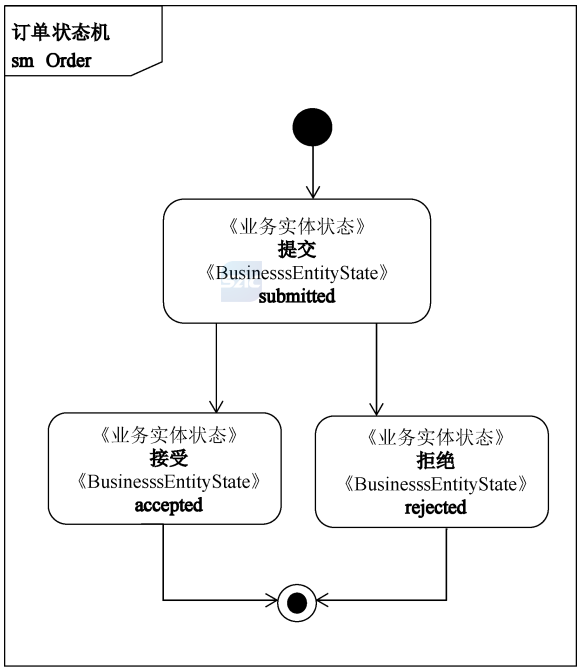

图 19　业务实体视图(业务需求视图)实例:订单 [业务实体生命周期(状态机)]

5.3.4 伙伴关系需求视图

5.3.4.1 概念性描述(资料性)

5.3.4.1.1 业务伙伴关系视图以用例方式描述业务伙伴类型之间已确定的协作需求,包含业务协作用例、业务交易用例以及业务协作实现。业务交易用例用于说明交易需求,是两种授权角色之间仅限于一次发起信息交换和一个可选响应的特定交互行为。业务协作用例描述了两个或更多授权角色之间执行的业务协作需求,由一个或多个业务交易或嵌套的业务协作构成。业务协作用例应由一组业务伙伴类型执行。两组不同的业务伙伴类型可能会实现相同的业务协作用例。业务协作实现是通过一组特定业务伙伴类型实现业务协作的过程。

5.3.4.1.2 交易需求视图(TransactionRequirementsView)仅由一个业务交易用例(BusinessTransactionUseCase)构成。同时,交易需求视图(TransactionRequirementsView)仅包含两个授权角色(AuthorizedRole),这两个授权角色(AuthorizedRole)参与该业务交易用例(BusinessTransactionUseCase),授权角色应与对应的业务交易用例在同一交易需求视图程序包中定义。假设某一角色(如卖方、收款人)参与多项业务交易,则每个业务协作用例需要该角色的不同授权角色,每个授权角色都处于对应交易需求视图中的不同命名空间。

每个业务交易用例(BusinessTransactionUseCase)和每个授权角色(AuthorizedRole)只能关联一次,用 1 表示,每个业务交易用例(BusinessTransactionUseCase)与两个授权角色(AuthorizedRole)关联,用 2 表示。一个业务协作用例(BusinesCollaborationUseCase)可嵌套在多个父类业务协作用例(BusinesCollaborationUseCase)中,以对应的(0…n)和(0….n)的包含(include)关系表示嵌套与被嵌套的业务协作用例的关联。业务协作用例可包含多个业务交易用例,业务交易用例应至少包含在一个业务协作用例中,以(1….n)和(0….n)表示业务协作用例(BusinessCollaborationUseCase)和业务交易用例(BusinessTransactionUseCase)之间的聚合关联,但 UMM 不在业务协作用例和业务交易用例中使用任何扩展(extend)关联。业务协作用例至少包含一个嵌套业务协作用例或一个业务交易用例。包含关系的业务协作用例的层级结构不能含有任何循环。业务交易用例不能进一步分解。

5.3.4.1.3 对于每个包含关系的用例,应将源用例的授权角色映射至目标用例授权角色,授权角色(AuthorizedRole)应是一一对应的映射关系,以(1..n)至(1..n)表示。每个来源用例的授权角色最多只映射一次到相同的目标用例的授权角色,同时可映射至不同目标用例的不同授权角色。业务伙伴类型不直接与业务协作用例和业务交易用例相关联,而是用到业务协作实现规定一组参与协作的具体业务伙伴类型。协作实现视图(CollaborationRealizationView)中仅包含一个业务协作实现(BusinessCollaborationRealization),业务协作实现仅实现一个业务协作用例。每个业务协作用例可通过多个业务协作实现来实现,但并非每个业务协作用例(如嵌套在另一业务协作中的用例)都需要一个相应的业务协作实现,业务协作用例(BusinessCollaborationUseCase)和业务协作实现(BusinessCollaborationRealization)之间的 realize(实现)关联可表示为 1 和(0…n)。参与一个业务协作实现的两个或多个授权角色与此业务协作实现都应在同一协作实现视图程序包中定义。协作实现视图(CollaborationRealizationView)含有两个或更多授权角色(AuthorizedRole)。参与业务协作用例和业务协作实现中的授权角色(如付款人和收款人)在不同命名空间中定义,即各自相对应的视图程序包中定义。与业务协作用例相类似,每个授权角色(AuthorizedRole)只能与业务协作实现(BusinessCollaborationRealization)关联一次,以 1 表示,两个或多个授权角色(AuthorizedRole)可参与到同一个业务协作实现(BusinessCollaborationRealization)中,以(2…n)表示,且参与业务协作用例中的角色数量应与参与业务协作实现中的角色数量相同。

5.3.4.1.4 业务伙伴类型需要映射为参与业务协作实现的授权角色。每个具体的授权角色是相应的某一业务伙伴类型映射关系唯一对应的一个目标。一个业务伙伴类型可以对应业务协作实现中的多个授

权角色，业务伙伴类型（BusinessPartnerType）和授权角色（AuthorizedRole）之间的映射至（mapsTo）关联为（0…1）和（0…n）。

5.3.4.1.5　伙伴关系需求视图（业务需求视图）[CollaborationRequirementsView（BusinessRequirementsView）]概念框架，见图 20。

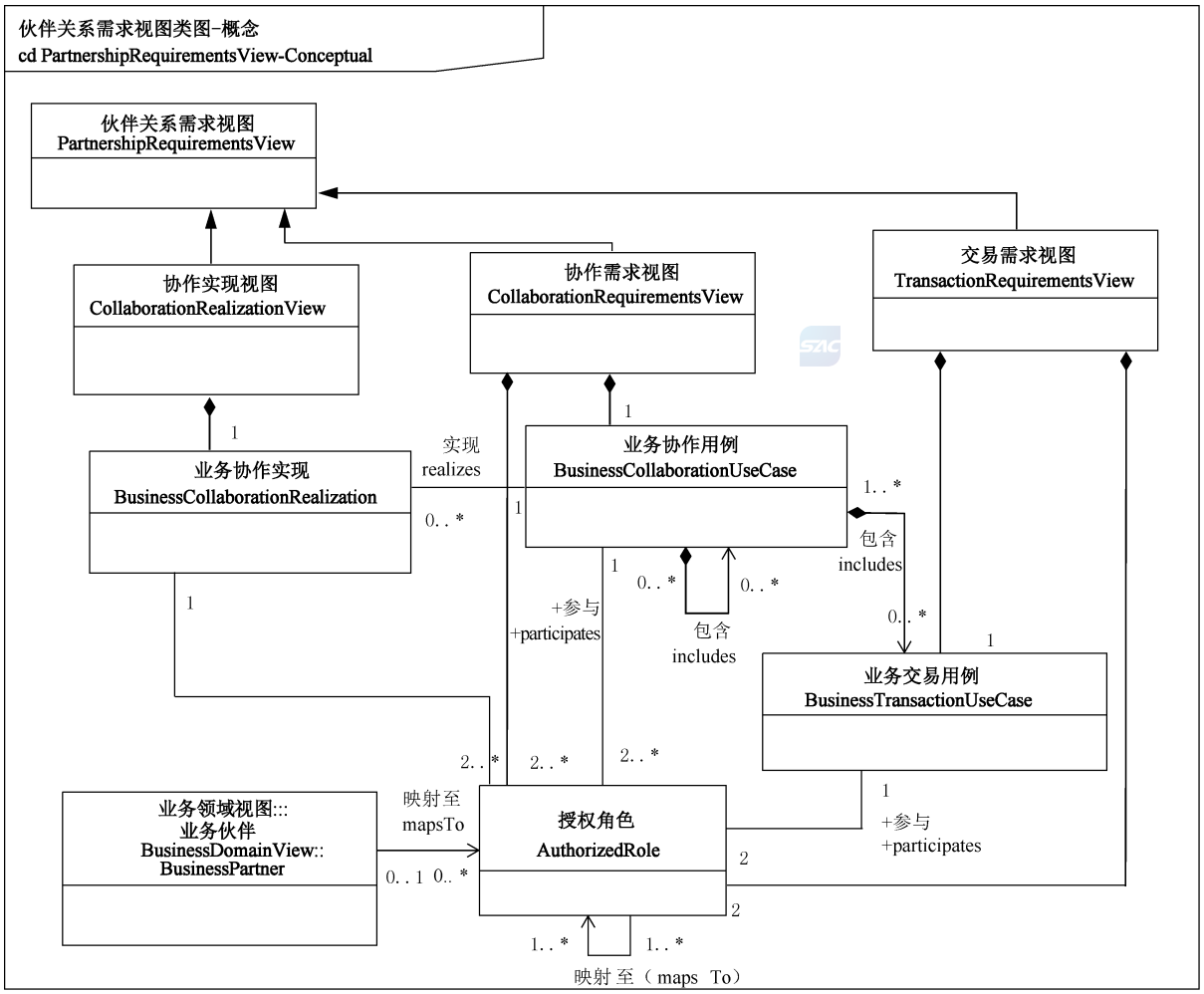

图 20　伙伴关系需求视图（业务需求视图）概念框架

5.3.4.2　构造型和标签定义（规范性）

5.3.4.2.1　协作需求视图（业务需求视图）[CollaborationRequirementsView（BusinessRequirementsView）]抽象语法，见图 21。

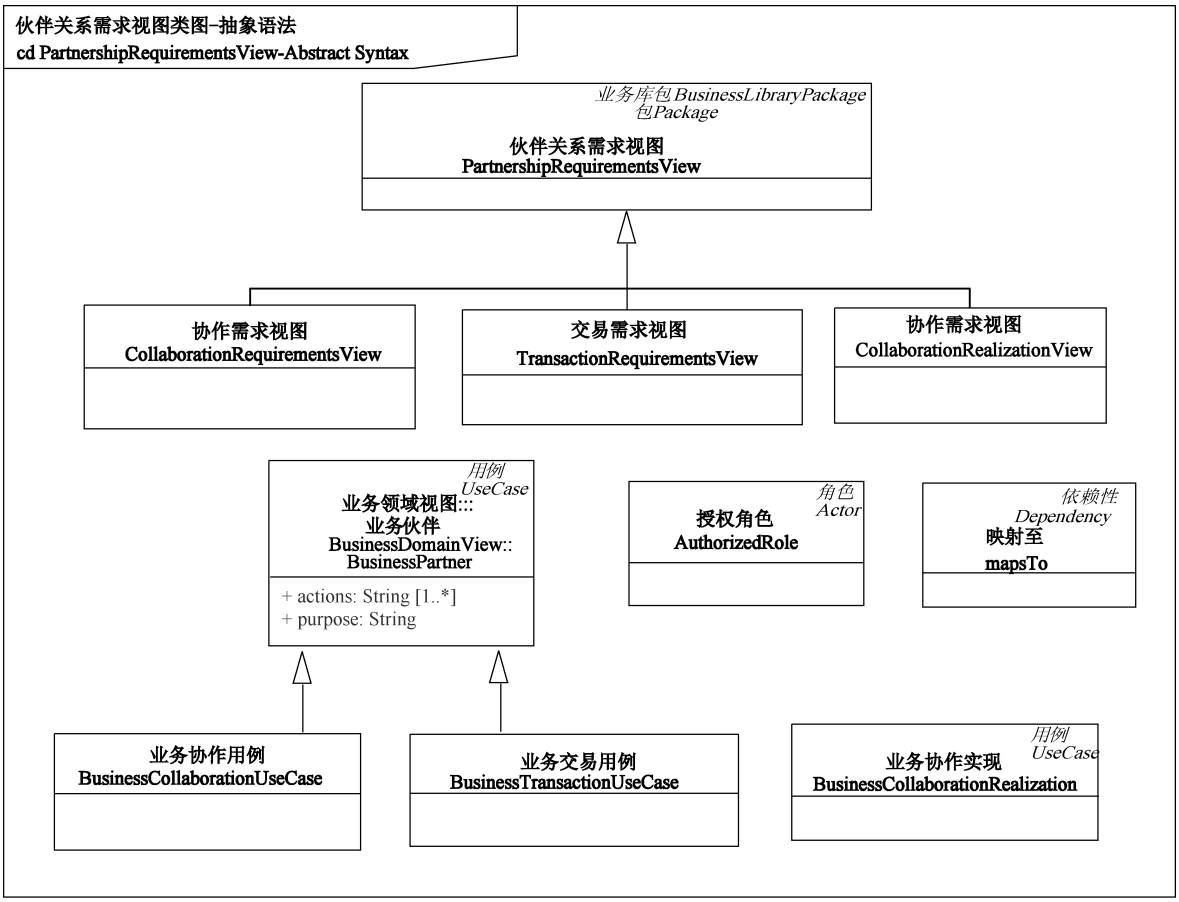

图 21　协作需求视图(业务需求视图)抽象语法

5.3.4.2.2　构造型和标签定义-业务协作用例,见表 26。

表 26　构造型和标签定义-业务协作用例

构造型(Stereotype)	业务协作用例(BusinesCollaborationUseCase)
基类(Base Class)	用例(UseCase)
父类(Parent)	业务过程(BusinessProcess)
描述(Description)	业务协作用例详细描述了两个或多个参与方之间协作需求,业务伙伴类型以授权角色身份参与到业务协作用例中。业务协作用例可进一步分解为业务协作用例和业务交易用例
标签定义 (Tag Definition)	继承标签值(Inherited tagged value): ——定义(definition) ——开始时间(beginsWhen) ——前置条件(preCondition) ——结束时间(endsWhen) ——后置条件(postCondition) ——例外(exceptions) ——动作(actions)

5.3.4.2.3　构造型和标签定义-业务交易用例,见表 27。

表 27　构造型和标签定义-业务交易用例

构造型(Stereotype)	业务交易用例(BusinessTransactionUseCase)
基类(Base Class)	用例(UseCase)
父类(Parent)	业务过程(BusinessProcess)
描述(Description)	业务交易用例详细描述了具体两个参与方之间的协作需求。业务交易用例不能进一步细分，描述了单向或双向的信息交换需求。业务伙伴类型以授权角色的身份参与到业务交易用例中
标签定义 (Tag Definition)	继承标签值(Inherited tagged value)： ——定义 (definition) ——开始时间(beginsWhen) ——前置条件(preCondition) ——结束时间(endsWhen) ——后置条件(postCondition) ——例外(exceptions) ——动作(actions)

5.3.4.2.4　构造型和标签定义-业务协作实现，见表 28。

表 28　构造型和标签定义-业务协作实现

构造型(Stereotype)	业务协作实现(BusinessCollaborationRealization)
基类(Base Class)	Collaboration(协作)
父类(Parent)	无(N/A)
描述(Description)	业务协作实现用于实现具体一组业务伙伴类型之间的业务协作用例。业务协作实现需求在相应的业务协作用例的标签中定义，业务协作实现不包含任何标签定义
标签定义(Tag Definition)	无标签值

5.3.4.2.5　构造型和标签定义-授权角色，见表 29。

表 29　构造型和标签定义-授权角色

构造型(Stereotype)	授权角色(AuthorizedRole)
基类(Base Class)	参与者(Actor)
父类(Parent)	无(N/A)
描述(Description)	授权角色(如买方)是比业务伙伴类型(如经纪人)更通用的概念，在给定业务场景中，可将授权角色映射至一个业务伙伴类型以便重用该协作。业务协作用例和业务交易用例被定义为授权角色之间发生的业务关系，在相同业务领域甚至不同业务领域的业务场景中，授权角色都可通过不同业务伙伴类型(“经纪人”或“托管人”)重用
标签定义(Tag Definition)	无标签值

5.3.4.2.6　构造型和标签定义-映射至，见表 30。

表 30　构造型和标签定义-映射至

构造型(Stereotype)	映射至(mapsTo)
基类(Base Class)	依赖性(Dependency)
父类(Parent)	无(N/A)
描述(Description)	映射至依赖对象表示(1)一个业务伙伴类型以某个授权角色身份出现在业务协作实现中,(2)源业务协作用例的授权角色在目标业务交易用例或业务协作用例中以某个授权角色身份出现
标签定义(Tag Definition)	无标签值

5.3.4.3　约束(规范性)

5.3.4.3.1　协作需求视图(CollaborationRequirementsView)应且仅含有一个业务协作用例(BusinessCollaborationUseCase)和至少两个授权角色(AuthorizedRoles),以及至少两个参与(participate)关联。

示例:

```
package Model_Management
context Package

inv AllowedElementsInCollaborationRequirementsView:
  self.isCollaborationRequirementsView() implies
  self.contents->notEmpty() and
  self.contents->select(isAuthorizedRole())->size()>=2 and
  self.contents->one(isBusinessCollaborationUseCase()) and
  self.contents->select(isParticipates())->size()>=2 and
  self.contents->forAll(isAuthorizedRole() or
  isBusinessCollaborationUseCase()
  or isParticipates())
```

5.3.4.3.2　交易需求视图(TransactionRequirementsView)应且仅含有一个业务交易用例(BusinessTransactionUseCase),并仅含两个授权角色(AuthorizedRole)和两个参与(participate)关联。

示例:

```
package Model_Management
context Package

inv AllowedElementsInTransactionRequirementsView:
  self.isTransactionRequirementsView() implies
  self.contents->notEmpty() and
  self.contents->select(isAuthorizedRole())->size()=2 and
  self.contents->one(isBusinessTransactionUseCase()) and
  self.contents->select(isParticipates())->size()=2 and
  self.contents->forAll(isAuthorizedRole() or
  isBusinessTransactionUseCase()
  or isParticipates())
```

5.3.4.3.3　协作实现视图(CollaborationRealizationView)应且仅含有一个业务协作实现(BusinessCollaborationRealization)、至少两个授权角色(AuthorizedRole),以及至少两个参与(participate)关联。

示例：

```
package Model_Management
context Package

inv AllowedElementsInRealizationView：
  self.isCollaborationRealizationView() implies
  self.contents->notEmpty() and
  self.contents->select(isAuthorizedRole())->size()>=2 and
  self.contents->one(isBusinessCollaborationRealization()) and
  self.contents->select(isParticipates())->size()>=2 and
  self.contents->forAll(isBusinessCollaborationRealization() or
  isParticipates() or isAuthorizedRole())
```

5.3.4.3.4 业务协作用例(BusinessCollaborationUseCase)应通过构造型参与(participate)关联与两个或更多的授权角色(AuthorizedRole)进行二元关联。

示例：

```
package Behavioral_ Elements::Use_Cases
context UseCase

inv BusinessCollaborationUCAssociatedWith2AuthorizedRoles：
  self.isBusinessCollaborationUseCase() implies
  self.associations->size() >= 2 and
  self.associations->forAll(a | a.isParticipates() and
  a.allConnections->exists(isAuthorizedRole())
  and a.connection->size=2)
```

5.3.4.3.5 业务交易用例(BusinessTransactionUseCase)应通过构造型参与(participate)来关联，且仅与两个授权角色(AuthorizedRole)进行二元关联。

示例：

```
package Behavioral_ Elements::Use_Cases
context UseCase

inv BusinessTransactionUCAssociatedWith2AuthorizedRoles：
  self.isBusinessTransactionUseCase() implies
  self.associations->size() = 2 and
  self.associations->forAll(a | a.isParticipates() and
  a.allConnections->exists(isAuthorizedRole())
  and a.connection->size=2)
```

5.3.4.3.6 业务协作实现(BusinessCollaborationRealization)应通过构造型参与(participate)关联与两个或更多授权角色(AuthorizedRole)进行二元关联。

示例：

```
package Behavioral_ Elements::Use_Cases
context UseCase

inv BusinessCollaborationRealizationAssociatedWith2AuthorizedRoles：
  self.isBusinessCollaborationRealization() implies
  self.associations->size() >= 2 and
  self.associations->forAll(a | a.isParticipates() and
  a.allConnections->exists(isAuthorizedRole())
  and a.connection->size=2)
```

5.3.4.3.7 业务协作实现(BusinessCollaborationRealization)应且仅是一个依赖于业务协作用例(BusinessCollaborationUseCase)的实现的客户端。

示例：

```
package Behavioral _Elements：: Use_ Cases
context UseCase

inv BusinessCollaborationRealizationRealizesOneBusinessCollaborationUseCase：
  self.isBusinessCollaborationRealization() implies
  self.clientDependency->size()=1 and
  self.clientDependency->forAll(d | d.isRealization() and
  d.supplier->size()=1 and
  d.supplier->forAll(isBusinessCollaborationUseCase()))
```

5.3.4.3.8 业务协作用例(BusinessCollaborationUseCase)应包含一个或多个其他业务协作用例(BusinessCollaborationUseCase)，或者一个或多个业务交易用例(BusinessTransactionUseCase)，但至少为业务协作用例(BusinessCollaborationUseCase)或业务交易用例(BusinessTransactionUseCase)之一。

示例：

```
package Behavioral_ Elements：: Use_Cases
context UseCase

inv AllowedIncludesOfBCUC：
  self.isBusinessCollaborationUseCase() implies
  self.include->notEmpty() and
  self.include->forAll(i | i.addition.isBusinessCollaborationUseCase() or
  i.addition.isBusinessTransactionUseCase())
```

5.3.4.3.9 业务交易用例(BusinessTransactionUseCase)不应包含其他用例。

示例：

```
package Behavioral_ Elements：: Use_Cases
context UseCase

inv NoIncludesOfBTUC：
  self.isBusinessTransactionUseCase() implies
  self.include->collect(addition)->isEmpty()
```

5.3.4.3.10 业务交易用例(BusinessTransactionUseCase)应被包含在至少一个业务协作用例(BusinessCollaborationUseCase)中。

示例：

```
package Behavioral_ Elements：: Use_Cases
context UseCase

inv BTUCIncludedAtLeastOnce：
  self.isBusinessTransactionUseCase() implies
  self.include->forAll(base.isBusinessCollaborationUseCase()) and
  self.include->collect(base)->notEmpty()
```

5.3.4.3.11 业务协作用例(BusinessCollaborationUseCase)和业务交易用例(BusinessTransactionUseCase)不应是一个扩展(extends)关联的来源或目标。

示例：

```
package Behavioral_ Elements::Use_Cases
context UseCase

inv BTUC_BCUC_IsNoExtendTarget:
  (self.isBusinessTransactionUseCase() or
  self.isBusinessCollaborationUseCase()) implies
  self.extend->isEmpty()
```

5.3.4.3.12 业务协作实现(BusinessCollaborationRealization)不应是一个包含(includes)关联或多个扩展(extends)关联的来源或目标。

示例：

```
package Behavioral_ Elements::Use_Cases
context UseCase

inv BusinessCollaborationRealizationNoIncludesAndExtends:
  self.isBusinessCollaborationRealization() implies
  self.extend->isEmpty() and
  self.include->isEmpty()
```

5.3.4.3.13 所有来自或目标为授权角色的授权角色(AuthorizedRole)依赖对象应映射至(mapsTo)其依赖对象。

示例：

```
package Behavioral_ Elements::Use_Cases
context Actor

inv AllDependenciesToAndFromAuthorizedRoleMustBeMapsTo:
  self.isAuthorizedRole() implies
  self.clientDependency->forAll(d | d.isMapsToDependency()) and
  self.supplierDependency->forAll(s | s.isMapsToDependency())
```

5.3.4.3.14 参与业务协作实现(BusinessCollaborationRealization)的授权角色(AuthorizedRole)应且仅是一个映射至(mapsTo)依赖于业务伙伴类型(BusinessPartnerType)的对象的提供方。进而,参与业务协作实现(BusinessCollaborationRealization)的授权角色(AuthorizedRole)应且仅是一个映射至(mapsTo)依赖于参与业务协作用例(BusinessCollaborationUseCase)的授权角色(AuthorizedRole)的对象的客户端。

示例：

```
package Behavioral_ Elements::Use_Cases
context Actor

inv BCRAuthorizedRoleIsMappedByOnlyOneBusinessPartnerType:
(self.isAuthorizedRole() and self.namespace.isCollaborationRealizationView()) implies
  self.supplierDependency->size()=1 and (
  self.supplierDependency->forAll(c | c.client->size()=1 and
  self.supplierDependency->forAll(c.client->
 forAll(isBusinessPartnerType()))))
  and self.clientDependency->size()=1 and (
```

```
    self.clientDependency->forAll(s | s.supplier->size()=1 and
    self.clientDependency->forAll(s|s.supplier->forAll(isAuthorizedRole()
    and s.namespace.isCollaborationRequirementsView))))
```

5.3.4.3.15 源业务协作用例(BusinessCollaborationUseCase)包含目标业务交易用例(BusinessTransaction-UseCase)和/或业务协作用例(BusinessCollaborationUseCase)。源用例的每个授权角色(AuthorizedRole),应最多一次映射至(mapsTo)同一目标用例的授权角色(AuthorizedRole)[但可映射至(mapsTo)不同目标用例的不同授权角色(AuthorizedRole)]。目标用例的每个授权角色(AuthorizedRole)是从源用例的授权角色(AuthorizedRole)所映射至(mapsTo)的对象的提供方。

示例:

```
package Behavioral_ Elements:: Use_Cases
context UseCase

inv AuthorizedRoleofBTUCisSupplierOfOnlyOneAuthorizedRoleOfBCUC:
  (self.isBusinessTransactionUseCase() or
  self.isBusinessCollaborationUseCase()) implies
  self.include->select(a | a.base <> self)->collect(base)->collect(x | x.associations)->
  collect(y|y.allConnections)->select(isAuthorizedRole)->forAll(x|self.associations->collect(allConnections)->
  select(isAuthorizedRole)->collect(supplierDependency)->collect(client) ->isUnique(x))
```

5.3.4.3.16 参与业务协作用例(BusinessCollaborationUseCase)的授权角色(AuthorizedRole)数量应与实现该用例的每个业务协作实现(BusinessCollaborationRealization)的授权角色(AuthorizedRole)数量相同。

示例:

```
package Behavioral_ Elements:: Use_Cases
context UseCase

inv AuthorizedRoleCountSameForBCUCandRealizingBCR:
  self.isBusinessCollaborationRealization() implies
  self.associations->collect(allConnections)->select(isAuthorizedRole)
  ->size() =
  (self.clientDependency->collect(supplier)->collect(associations) ->collect(allConnections)->
  select(isAuthorizedRole)->size())
```

5.3.4.3.17 交易需求视图(TransactionRequirementsView)、协作需求视图(CollaborationRequire-mentsView)或协作实现视图(CollaborationRealizationView)中的授权角色(AuthorizedRole)应在其所在的程序包内具有唯一名称。

示例:

```
package Model_Management
context Package

inv AuthorizedRolesMustHaveUniqueName:
  self.isTransactionRequirementsView() or
  self.isCollaborationRequirementsView() or
  self.isCollaborationRealizationView() implies
  self.contents->select(isAuthorizedRole())
  ->isUnique(element | element.name)
```

5.3.4.4 实例(资料性)

5.3.4.4.1 协作需求视图(业务需求视图)[CollaborationRequirementsView(BusinessRequirements-View)]实例:报价单(用例图)[OrderFromQuote(UseCase Diagram)],见图 22。

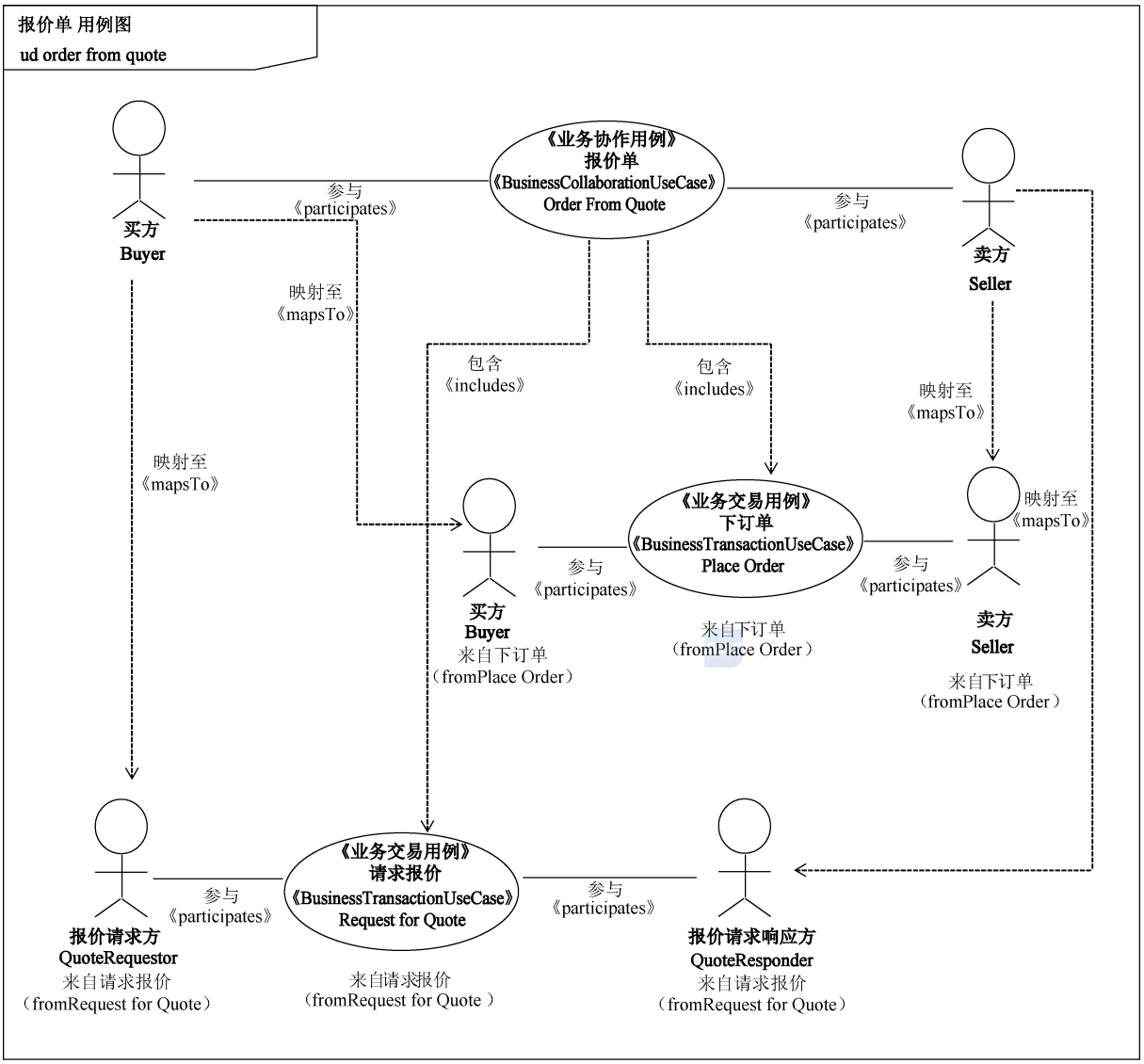

图 22 协作需求视图(业务需求视图)实例:报价单(用例图)

5.3.4.4.2 交易需求视图(业务需求视图)[TransactionRequirementsView(BusinessRequirementsView)]实例:下订单交易(PlaceOrder Transaction),见图 23。

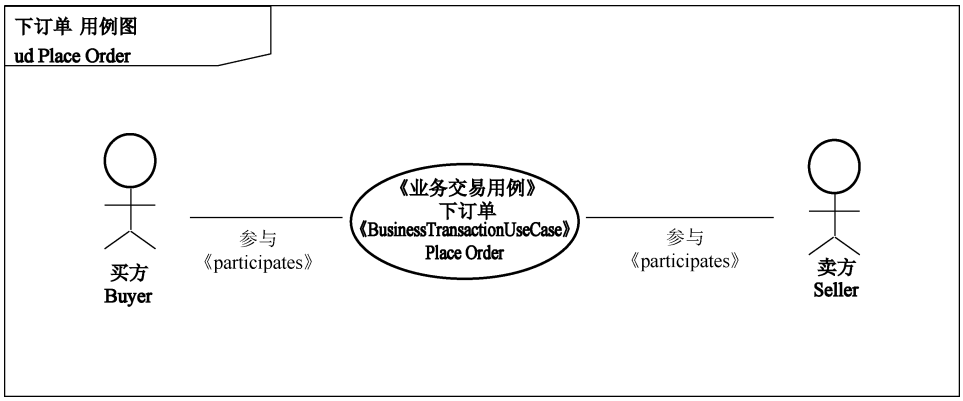

图 23　交易需求视图(业务需求视图)实例:下订单交易

5.3.4.4.3　协作实现视图(业务需求视图)[CollaborationRealizationView(BusinessRequirementsView)]实例:采购组织和销售组织之间实现报价单(OrderFromQuote)协作,见图 24。

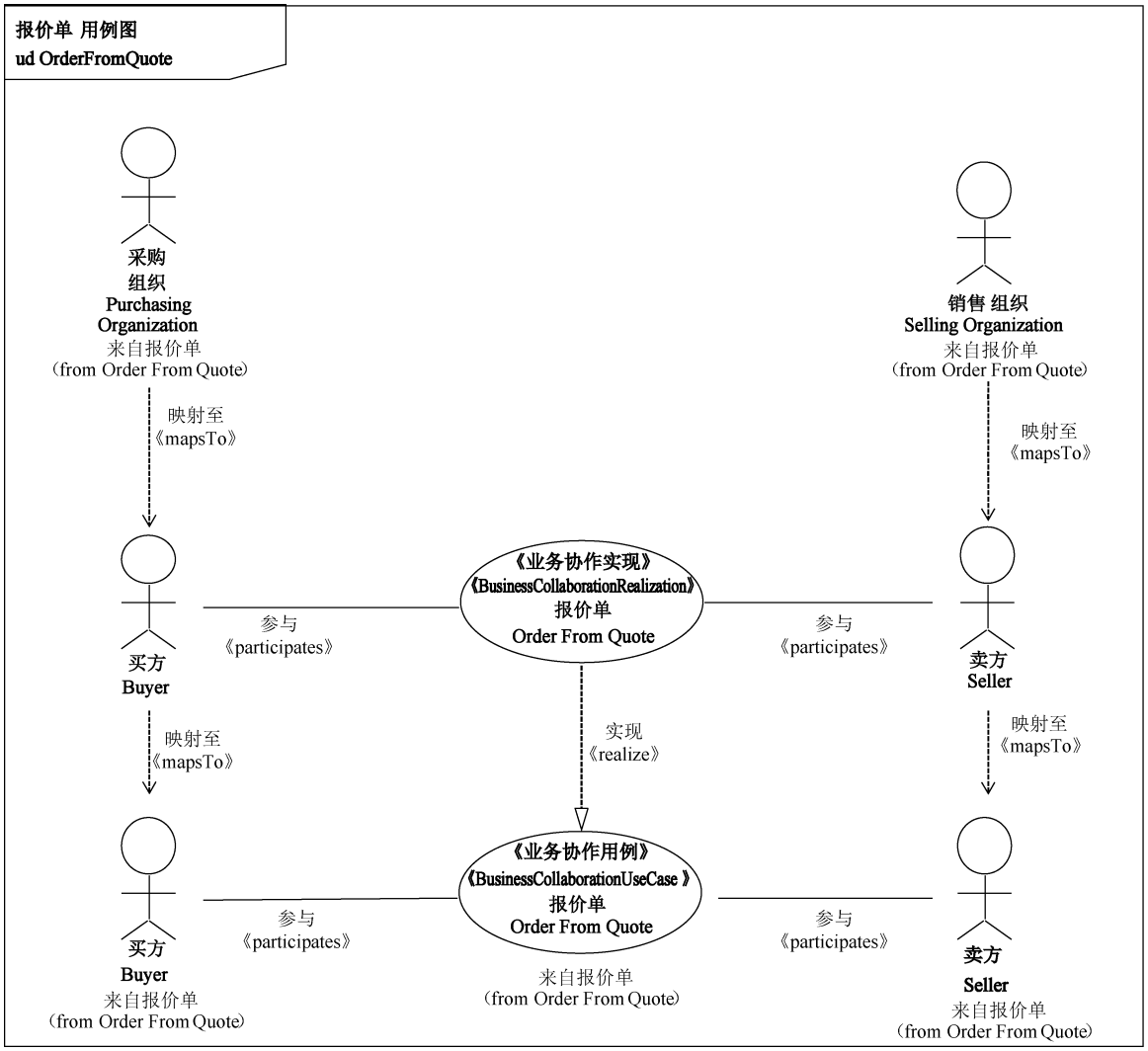

图 24　协作实现视图(业务需求视图)实例:采购组织和销售组织之间实现报价单协作

5.3.5 所有 BRV 程序包中采用的 OCL 方法(规范性)

OCL 方法,参见示例。

示例:

```
package Foundation::Core
context Model Element

--Predefined method which evaluates, if the given Modelelement
--has a stereotype equal to the passed name
def:
let hasStereotype (st : String) : Boolean =
  self.stereotype->select(cst | cst.name = st)->notEmpty()
--Predefined method which evaluates, if the given element
--has the stereotype 'InternalBusinessEntityState'
def:
let isInternalBusinessEntityState() : Boolean =
  self.oclIsKindOf(ObjectFlowState) and
  self.hasStereotype('InternalBusinessEntityState')
--Predefined method which evaluates, if the given element
--has the stereotype 'ShardedBusinessEntityState'
def:
let isSharedBusinessEntityState() : Boolean =
  self.oclIsKindOf(ObjectFlowState) and
  self.hasStereotype('SharedBusinessEntityState')
--Predefined method which evaluates, if the given element
--has the stereotype 'BusinessProcessActivity'
def:
let isBusinessProcessActivity() : Boolean =
  self.oclIsKindOf(ObjectFlowState) and
  self.hasStereotype('BusinessProcessActivity')
-- Returns true if the type of the element or one of the
-- supertypes is 'PseudoKindState' and its Pseudostatekind
-- is initial
def:
let isInitialState () : Boolean =
  self.oclAsType(Pseudostate).kind = PseudostateKind::initial and
  self.oclIsKindOf(Pseudostate)
-- Returns true if the type of the element or one of the
-- supertypes is 'PseudoKindState' and its Pseudostatekind
-- is choice
def:
let isChoice () : Boolean =
  self.oclAsType(Pseudostate).kind = PseudostateKind::choice and
  self.oclIsKindOf(Pseudostate)
-- Returns true if the type of the element or one of the
-- supertypes is 'PseudoKindState' and its Pseudostatekind
```

```
-- is fork
def:
let isFork () : Boolean =
  self.oclAsType(Pseudostate).kind = PseudostateKind::fork and
  self.oclIsKindOf(Pseudostate)
-- Returns true if the type of the element or one of the
-- supertypes is 'PseudoKindState' and its Pseudostatekind
-- is join
def:
let isJoin () : Boolean =
  self.oclAsType(Pseudostate).kind = PseudostateKind::join and
  self.oclIsKindOf(Pseudostate)
-- Returns true if the type of the element or is 'FinalState'
def:
let isFinalState () : Boolean =
  self.oclIsKindOf(FinalState)
-- Returns true if the type of the element 'Transition'
def:
let isTransition () : Boolean =
  self.oclIsKindOf(Transition)
--Returns true if the element is a standard—element of an ActivityGraph
def:
let isPseudoStateOrFinalStateOrTransition() : Boolean =
  isInitialState() or isChoice() or isFork() or isJoin() or isTransition()
  or isFinalState()
--Predefined method which evaluates, if the given element
--has the stereotype 'BusinessProcessView'
def :
let isBusinessProcessView() : Boolean =
  self.oclIsKindOf(Package) and
  self.hasStereotype('BusinessProcessView')
--Predefined method which evaluates, if the given element
--has the stereotype 'BusinessEntityView'
def :
let isBusinessEntityView() : Boolean =
  self.oclIsKindOf(Package) and
  self.hasStereotype('BusinessEntityView')
--Predefined method which evaluates, if the given element
--has the stereotype 'BusinessRequirementsView'
def :
let isBusinessRequirementsView() : Boolean =
  self.oclIsKindOf(Package) and
  self.hasStereotype('BusinessRequirementsView')
--Predefined method which evaluates, if the given element
--has the stereotype 'BusinessProcessActivityModel'
def:
let isBusinessProcessActivityModel() : Boolean =
```

```
    self.oclIsKindOf(ActivityGraph) and
    self.hasStereotype('BusinessProcessActivityModel')
  --return true if the given element is a partition
  def:
  let isPartition () : Boolean =
    self.oclIsKindOf(Partition)
  --Predefined method which evaluates, if the given element
  --has the stereotype 'BusinessEntity'
  def :
  let isBusinessEntity () : Boolean =
    self.oclIsKindOf(Class) and
    self.hasStereotype('BusinessEntity')
  --Predefined method which evaluates, if the given element
  --has the stereotype 'BusinessEntityState'
  def :
  let isBusinessEntityState() : Boolean =
    self.oclIsKindOf(State) and
    self.hasStereotype('BusinessEntityState')
  --Predefined method which evaluates, if the given element
  --has the stereotype 'BusinessEntityLifecycle'
  def :
  let isBusinessEntityLifecycle() : Boolean =
    self.oclIsKindOf(StateMachine) and
    self.hasStereotype('BusinessEntityLifecycle')
  --return true if the given element is a package
  def :
  let isPackage () : Boolean =
    self.oclIsKindOf(Package)
  --Predefined method which evaluates, if the given element
  --has the stereotype 'BusinessCollaborationUseCase'
  def :
  let isBusinessCollaborationUseCase() : Boolean =
    self.oclIsKindOf(UseCase) and
    self.hasStereotype('BusinessCollaborationUseCase')
  --Predefined method which evaluates, if the given element
  --has the stereotype 'BusinessTransactionUseCase'
  def :
  let isBusinessTransactionUseCase() : Boolean =
    self.oclIsKindOf(UseCase) and
    self.hasStereotype('BusinessTransactionUseCase')
  --Predefined method wich evaluates, if the given element
  --has the stereotype 'BusinesCollaborationRealization'
  def:
  let isBusinessCollaborationRealization() : Boolean =
    self.oclIsKindOf(Collaboration) and
    self.hasStereotype('BusinessCollaborationRealization')
  --Predefined method which evaluates, if the given element
```

```
--has the stereotype 'AuthorizedRole'
def :
let isAuthorizedRole () : Boolean =
  self.oclIsKindOf(Actor) and
  self.hasStereotype('AuthorizedRole')
--Predefined method which evaluates, if the given element
--has the stereotype 'BusinessPartnerType'
def :
let isBusinessPartnerType () : Boolean =
  self.oclIsKindOf(Actor) and
  self.hasStereotype('BusinessPartnerType')
--Predefined method which evaluates, if the given element
--has the stereotype 'mapsTo'
def :
let isMapsToDependency() : Boolean =
  self.oclIsKindOf(Dependency) and
  self.hasStereotype('mapsTo')
--Predefined method which evaluates, if the given element
--is a Realization dependency
def :
let isRealization() : Boolean =
  self.oclIsKindOf(Abstraction) and
  self.hasStereotype('realize')
-- checks if an Association is stereotyped as participates
def:
let isParticipates() : Boolean =
  self.oclIsKindOf(Association) and
  self.hasStereotype('participates')
--Predefined method which evaluates, if the given element
--is an Association
def:
let isAssociation() : Boolean =
  self.oclIsKindOf(Association)
--Predefined method which evaluates, if the given element
--has the stereotype 'CollaborationRequirementsView'
def :
let isCollaborationRequirementsView() : Boolean =
  self.oclIsKindOf(Package) and
  self.hasStereotype('CollaborationRequirementsView')
--Predefined method which evaluates, if the given element
--has the stereotype 'TransactionRequirementsView'
def :
let isTransactionRequirementsView() : Boolean =
  self.oclIsKindOf(Package) and
  self.hasStereotype('TransactionRequirementsView')
--Predefined method which evaluates, if the given element
--has the stereotype 'CollaborationRealizationView'
```

```
    def :
    let isCollaborationRealizationView() : Boolean =
      self.oclIsKindOf(Package) and
      self.hasStereotype('CollaborationRealizationView')
    -- checks if a UseCase is stereotyped a BusinessProcess
    def :
    let isBusinessProcess() : Boolean =
      self.oclIsTypeOf(UseCase) and
      self.hasStereotype('BusinessProcess')
```

5.4 业务交易视图

5.4.1 交易视图概述

5.4.1.1 概念性描述(资料性)

5.4.1.1.1 业务交易视图、业务编排视图、业务交互视图和业务信息视图都是包含各类构件的容器。业务交易视图程序包包含业务编排视图、业务交互视图和业务信息视图三个不同构件,共同描述信息交换的总体编排。业务编排视图和业务交互视图分别根据 BRV 中相应的协作需求视图和交易需求视图确定可定义流程的构件,这些构件主要描述协作中动态过程,而业务信息视图处理描述协作的结构方面的构件。业务编排视图包含的构件描述涉及业务伙伴类型之间的多个步骤的复杂业务协作。业务交互视图包含的构件定义交互的两个业务实体之间的编排进而同步其状态。业务信息视图包含的构件描述交互过程中信息交换。

5.4.1.1.2 业务交易视图(BusinessTransactionView)包含一个到多个业务编排视图(BusinessChoreographyView),一个到多个业务交互视图(BusinessInteractionView),以及一个到多个业务信息视图(BusinessInformationView)。

5.4.1.1.3 业务交易视图(BusinessTransactionView)概念框架,见图 25。

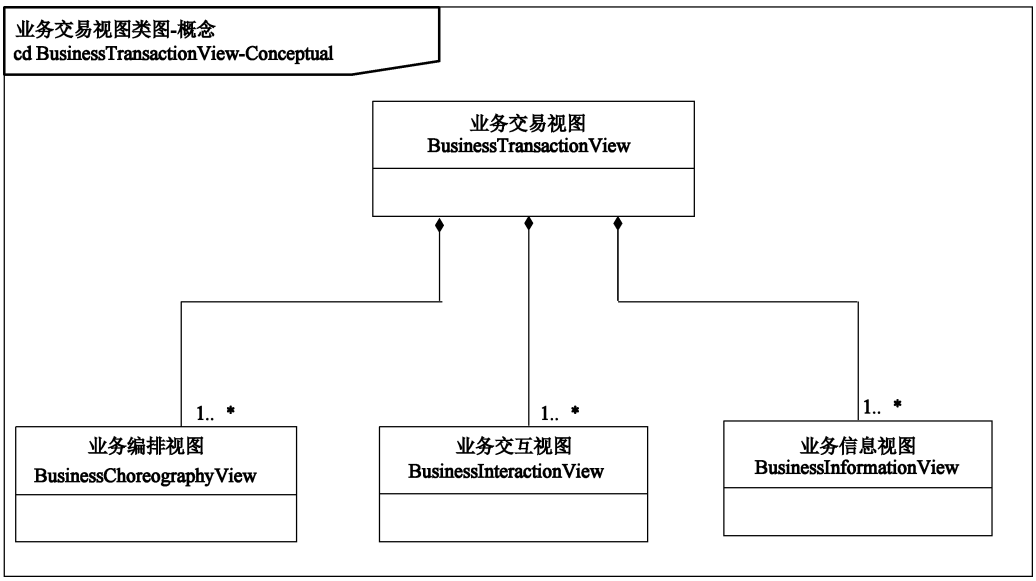

图 25 业务交易视图概念框架

5.4.1.2 构造型和标签定义(规范性)

5.4.1.2.1 业务交易视图(BusinessTransactionView)抽象语法,见图 26。

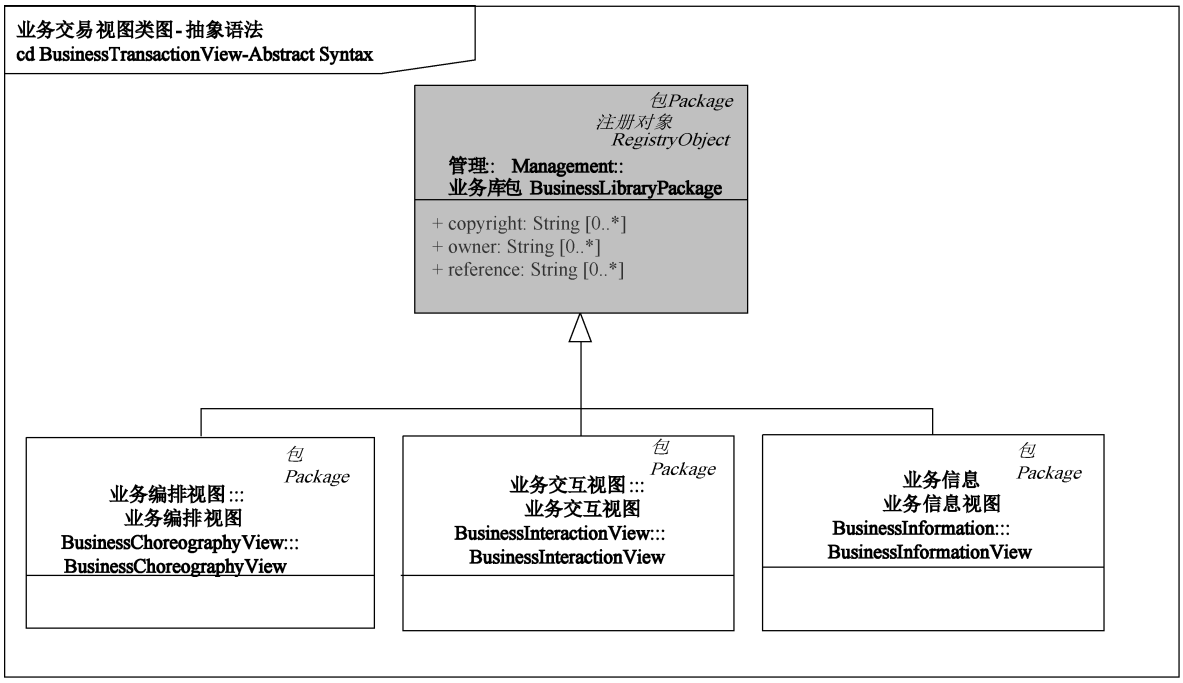

图 26 业务交易视图抽象语法

5.4.1.2.2 构造型和标签定义-业务编排视图,见表 31。

表 31 构造型和标签定义-业务编排视图

构造型(Stereotype)	业务编排视图(BusinessChoreographyView)
基类(Base Class)	程序包(Package)
父类(Parent)	业务库包(BusinessLibraryPackage)[来自基础模块(Base Module)]
描述(Description)	业务编排视图是一个包含构件的容器,其构件描述业务伙伴类型之间涉及多个步骤的复杂业务协作
标签定义 (Tag Definition)	继承标签值(Inherited tagged value): ——基础统一资源名称(baseURN) ——所有者(owner) ——版权(copyright) ——参考号(reference) ——版权(copyright) ——状态(status) ——业务术语(businessTerm)

5.4.1.2.3 构造型和标签定义-业务交互视图,见表 32。

表 32　构造型和标签定义-业务交互视图

构造型(Stereotype)	业务交互视图(BusinessInteractionView)
基类(Base Class)	程序包(Package)
父类(Parent)	业务库包(BusinessLibraryPackage)[来自基础模块(Base Module)]
描述(Description)	业务交互视图是一个包含构件的容器,其构件定义了交互的两个业务实体之间为同步双方状态的编排
标签定义 (Tag Definition)	**继承标签值(Inherited tagged value)**: ——基础统一资源名称(baseURN) ——所有者(owner) ——版权(copyright) ——参考号(reference) ——版权(copyright) ——状态(status) ——业务术语(businessTerm)

5.4.1.2.4　构造型和标签定义-业务信息视图,见表 33。

表 33　构造型和标签定义-业务信息视图

构造型(Stereotype)	业务信息视图(BusinessInformationView)
基类(Base Class)	程序包(Package)
父类(Parent)	业务库包(BusinessLibraryPackage)[来自基础模块(Base Module)]
描述(Description)	业务信息视图是一个包含构件的容器,其构件描述了交互过程中的信息交换
标签定义 (Tag Definition)	**继承标签值(Inherited tagged value)**: ——基础统一资源名称(baseURN) ——所有者(owner) ——版权(copyright) ——参考号(reference) ——版权(copyright) ——状态(status) ——业务术语(businessTerm)

5.4.1.3　约束(规范性)

一个业务交易视图(BusinessTransactionView)应包含至少一个业务编排视图(BusinessChoreographyView)程序包、至少一个业务交互视图(BusinessInteractionView)程序包,和至少一个业务信息视图(BusinessInformationView)程序包。

示例:

```
package Model_Management
context Package

inv packagesAllowedInBTV:
  self.isBusinessTransactionView() implies
  self.contents->exists(isBusinessChoreographyView()) and
  self.contents->exists(isBusinessInteractionView()) and
  self.contents->exists(isBusinessInformationView())
```

5.4.2 业务编排视图

5.4.2.1 概念性描述(资料性)

5.4.2.1.1 业务编排只定义一个业务协作的流程,因此业务编排视图(BusinessChoreographyView)仅由某一个业务编排(BusinessChoreography)组成,业务编排通过业务编排行为定义业务协作的执行顺序,每个业务编排(BusinessChoreography)仅由一个业务编排行为(BusinessChoreographyBehavior)组成。一个业务协作用例(BusinessCollaborationUseCase)和业务编排行为(BusinessChoreographyBehavior)具有一对一的映射至(mapsTo)关系。

5.4.2.1.2 一个有效的专用构造型业务编排行为(BusinessChoreographyBehavior)被定义为业务协作协议(BusinessCollaborationProtocol)。一个业务协作协议的活动为业务协作活动和/或业务交易活动。一个业务协作协议(BusinessCollaborationProtocol)包含零到多个业务协作活动(BusinessCollaborationActivities)或零到多个业务交易活动(BusinessTransactionActivities)。一个业务协作协议中应出现至少一个业务协作活动或业务交易活动。定义业务协作活动和/或业务交易活动的活动流转换可由业务实体的状态监控。

5.4.2.1.3 倘若一个业务协作协议嵌套于一个业务协作活动中,该业务协作活动将被此业务协作协议精确确定。一个业务协作协议可嵌套于不同的业务协作活动中。业务协作活动(BusinessCollabortionActivity)和业务协作协议(BusinessCollaborationProtocol)之间的聚合关系为(0…n)对 1。

5.4.2.1.4 一个业务交易活动由业务交易来精确确定。每个业务交易应至少有一次被用于确定业务交易活动。一个业务交易可被嵌套于不同的业务交易活动中,业务交易活动(BusinessTransactionActivity)和业务交易(BusinessTransaction)之间的聚合关系为(1…n)对 1。

5.4.2.1.5 业务编排视图(业务交易视图)[BusinessChoreographyView(BusinessTransactionView)]概念框架,见图 27。

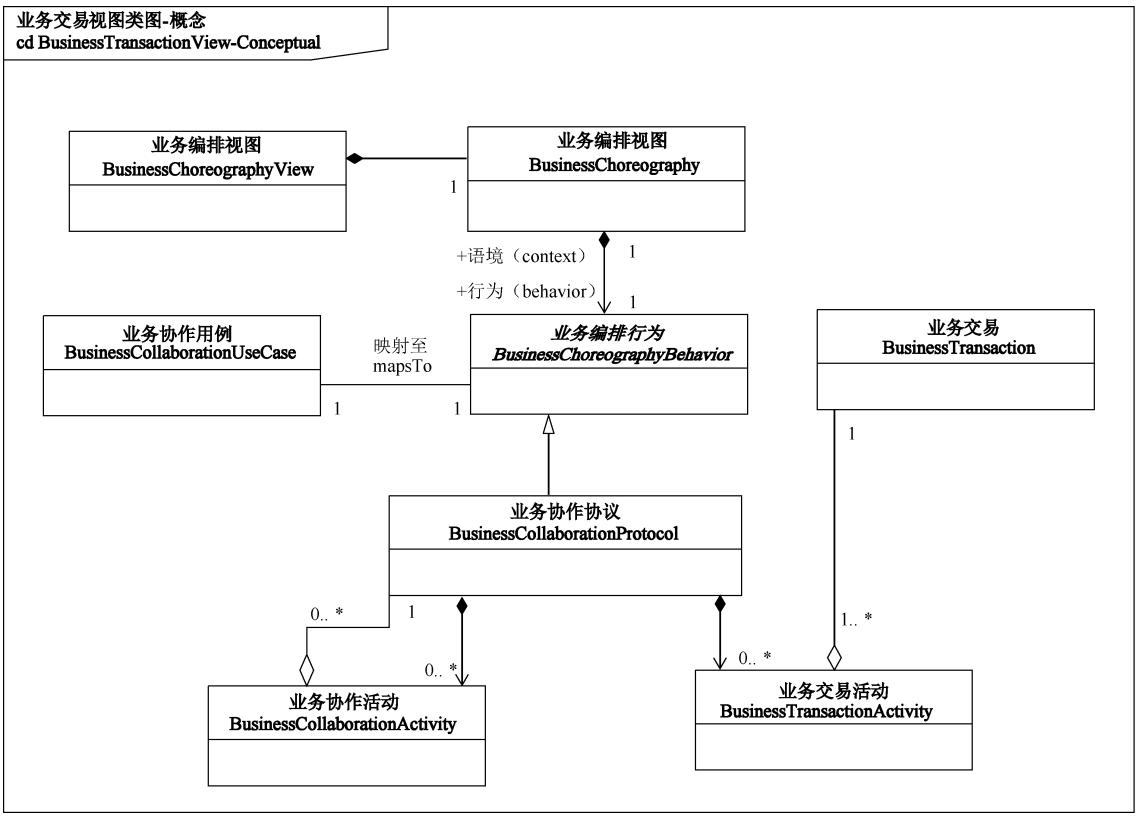

图 27 业务编排视图(业务交易视图)概念框架

5.4.2.2 构造型和标签定义(规范性)

5.4.2.2.1 业务编排视图(业务交易视图)[BusinessChoreographyView(BusinessTransactionView)]抽象语法,见图 28。

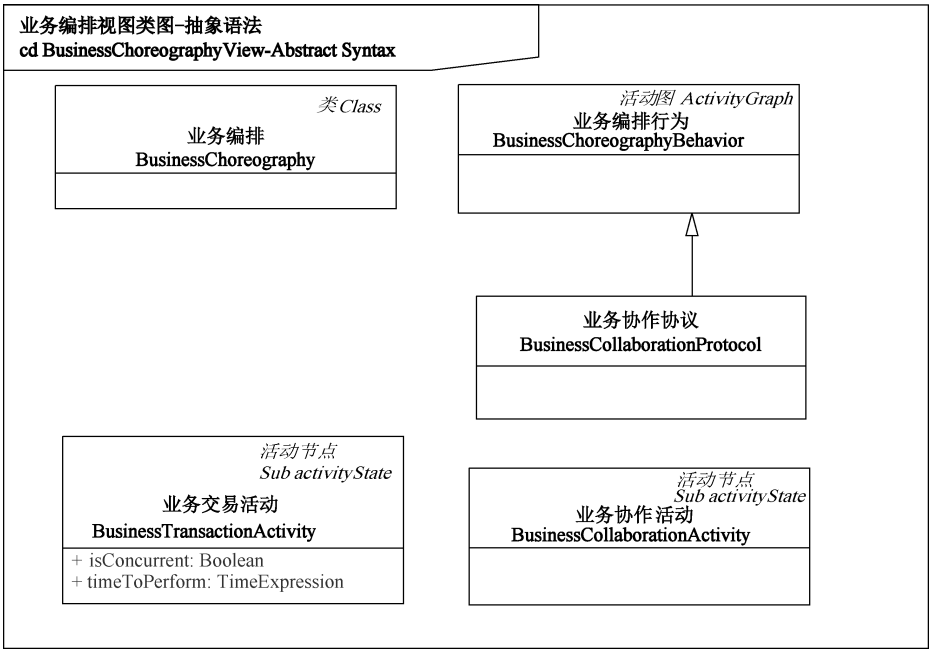

图 28 业务编排视图(业务交易视图)抽象语法

5.4.2.2.2 构造型和标签定义-业务编排,见表 34。

表 34 构造型和标签定义-业务编排

构造型(Stereotype)	业务编排(BusinessChoreography)
基类(Base Class)	类
父类(Parent)	无(N/A)
描述(Description)	业务编排是业务协作持续执行的顺序
标签定义(Tag Definition)	无标签值

5.4.2.2.3 构造型和标签定义-业务编排行为(抽象),见表 35。

表 35 构造型和标签定义-业务编排行为(抽象)

构造型(Stereotype)	业务编排行为(抽象)[BusinessChoreographyBehavior(abstract)]
基类(Base Class)	活动图(ActivityGraph)
父类(Parent)	无(N/A)
描述(Description)	业务编排行为定义了一项业务协作的动态行为,参考:一项业务协作的编排
标签定义(Tag Definition)	无标签值

5.4.2.2.4 构造型和标签定义-业务协作协议，见表36。

表36 构造型和标签定义-业务协作协议

构造型(Stereotype)	业务协作协议(BusinessCollaborationProtocol)
基类(Base Class)	活动图(ActivityGraph)
父类(Parent)	业务编排行为(BusinessChoreographyBehavior)
描述(Description)	业务协作协议是业务编排行为的专用构造型，用于编排业务交易活动和/或业务协作活动，两类活动应至少出现一个。业务协作协议是一项长期运行的交易，不满足交易的最小原则，用于不适于交易回滚的情况下
标签定义(Tag Definition)	无标签值

5.4.2.2.5 构造型和标签定义-业务交易活动见表37。

表37 构造型和标签定义-业务交易活动

构造型(Stereotype)	业务交易活动(BusinessTransactionActivity)	
基类(Base Class)	动作状态(ActionState)	
父类(Parent)	无(N/A)	
描述(Description)	一个业务交易活动是业务协作协议内的一项活动，是由一个嵌套业务交易确定的动作状态。业务交易活动执行嵌套的业务交易。假设“并行”为真，则可执行多次此业务交易活动	
标签定义(Tag Definition)	执行时间(timeToPerform)	
	类型(Type)	时间表达式(TimeExpression)
	次数(Multiplicity)	1
	描述(Description)	一个业务交易活动应在一个特定时间段中执行。发起方应在超时后向响应方发出失败通知，响应方即停止其活动。执行时间是指在请求授权角色发起交易活动(如发送请求业务信息)至请求授权角色收到实质性回应时的最大时间段。实质性回应是指倘若有任何业务信息则回应业务信息，在没有业务信息的情况下，倘若有任何处理确认，则回应的是处理确认，倘若也没有处理确认，则回应的是收到确认
	并行(isConcurrent)	
	类型(Type)	布尔值 (Boolean)
	次数(Multiplicity)	1
	描述(Description)	倘若业务交易活动并行发生，则与同一业务伙伴类型执行同一业务协作时，可在同一基础层的业务交易中同时启动多个业务交易活动。倘若业务交易活动不是并行发生，那么一次只能启动同一基础层的业务交易的一个业务交易活动

5.4.2.2.6 构造型和标签定义-业务协作活动见表38。

表38 构造型和标签定义-业务协作活动

构造型(Stereotype)	业务协作活动(BusinessCollaborationActivity)
基类(Base Class)	动作状态 (ActionState)
父类(Parent)	无(N/A)

表 38（续）

构造型（Stereotype）	业务协作活动（BusinessCollaborationActivity）
描述（Description）	一个业务协作活动是一个业务协作协议内的一个活动。该业务协作活动是由嵌套业务协作协议的活动图确定的一个动作状态。业务协作协议可递归嵌套，但业务协作活动只执行一次嵌套业务协作协议
标签定义（Tag Definition）	无标签值

5.4.2.3 约束（规范性）

5.4.2.3.1 业务编排行为（BusinessChorographyBehavior）应且仅是一个映射至（mapsTo）业务协作用例（BusinessCollaborationUseCase ）的依赖对象的客户端。

示例：

```
package Behavioral_Elements::Activity_Graphs
context Activity graph

inv BCBmapsToBCUseCase:
  self.isBusinessChoreographyBehavior() implies
  self.clientDependency->size()=1 and
  self.clientDependency->forAll(d | d.isMapsToDependency() and
  d.supplier->forAll(isBusinessCollaborationUseCase()) and
  d.supplier->size=1)
```

5.4.2.3.2 业务编排视图（BusinessChoreographyView）程序包应且仅包含一个业务编排（BusinessChoreography）且无其他元素。

示例：

```
package Model_Management
context Package

inv BCVcontainsExactlyOneBC:
  self.isBusinessChoreographyView() implies
  self.contents->one(isBusinessChoreography()) and
  self.contents->size()=1
```

5.4.2.3.3 一个业务编排（BusinessChoreography）的行为应且仅由一个业务编排行为（BusinessChoreographyBehavior）描述。

示例：

```
package Foundation:: Core
context Class

inv BCdescribedByOneBusinessChoreographyBehavior:
  self.isBusinessChoreography() implies
  self.behavior->one(isBusinessChoreographyBehavior()) and
  self.behavior->size()=1
```

5.4.2.3.4 一个业务协作协议（BusinessCollaborationProtocol）应包含至少一个业务交易活动（BusinessTransactionActivity）或业务协作活动（BusinessCollaborationActivity），可包含伪态（PseudoState）、

最终状态(FinalState)和转换(Transition)。

示例：

```
package Behavioral Elements::State_Machines
context CompositeState

inv AllowedModelElementsInBCP:
  self.stateMachine.isBusinessCollaborationProtocol() implies
  self.subvertex->forAll(isBusinessTransactionActivity()
  or isBusinessCollaborationActivity()
  or isPseudoStateOrFinalStateOrTransition()
  or isTransition()
  )
  and (self.subvertex->exists(isBusinessTransactionActivity()) or
  self.subvertex->exists(isBusinessCollaborationActivity()))
```

5.4.2.3.5 一个业务协作活动(BusinessCollaborationActivity)应且仅由一个业务协作协议(Business-CollaborationProtocol)通过与构造型映射至(mapsTo)相关的依赖对象来精确确定。

示例：

```
package Behavioral Elements::Activity_Graphs
context action state

inv BCArefinedByExactlyOneBCP:
  self.isBusinessCollaborationActivity() implies
  self.clientDependency->size() = 1 and
  self.clientDependency->forAll(d | d.isMapsToDependency() and
  d.supplier->forAll(isBusinessCollaborationProtocol()) and
  d.supplier->size=1)
```

5.4.2.3.6 一个业务交易活动(BusinessTransactionActivity)应仅由一个业务交易(BusinessTransaction)通过与构造型映射至(mapsTo)相关的依赖对象来精确确定。

示例：

```
package Behavioral_ Elements::Activity_Graphs
context ActionState

inv BTArefinedByExactlyOneBT:
  self.isBusinessTransactionActivity() implies
  self.clientDependency->size() = 1 and
  self.clientDependency->forAll(d | d.isMapsToDependency() and
  d.supplier->forAll(isBusinessTransaction()) and d.supplier->size=1)
```

5.4.2.4 实例(资料性)

业务编排视图(业务交易视图)[BusinessChoreographyView (BusinessTransactionView)]实例：报价单业务协作协议(活动图) [OrderFromQuote BusinessCollaborationProtocol (ActivityGraph)]，见图 29。

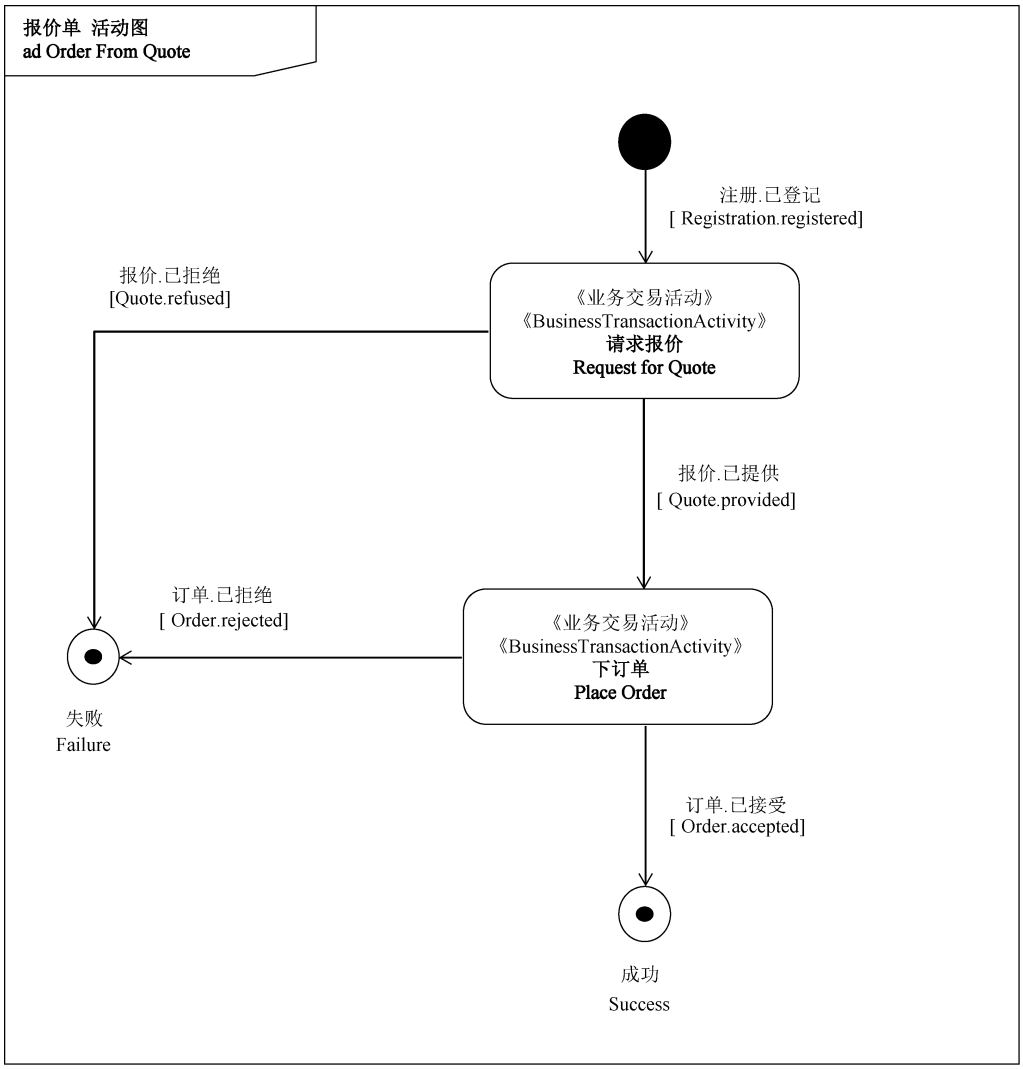

图 29　业务编排视图(业务交易视图)实例:报价单业务协作协议(活动图)

5.4.3　业务交互视图

5.4.3.1　概念性描述(资料性)

5.4.3.1.1　业务交互视图(BusinessInteractionView)仅包含一个业务交互(BusinessInteraction)。业务交互是两个授权角色之间持续进行业务状态同步的表现,由业务交互行为定义其同步过程的编排和需要进行的信息交换。每个业务交互(BusinessInteraction)仅由一个业务交互行为(BusinessInteractionBehavior)组成。每个业务交互行为(BusinessInteractionBehavior)仅从一个业务交易用例(BusinessTransactionUseCase)映射而来。

5.4.3.1.2　一个有效的专用构造型业务交互行为(BusinessInteractionBehavior)是一个业务交易(BusinessTransaction),两个授权角色之间最小单元的业务过程即为业务交易,涉及从一个授权角色向另一个授权角色发送业务信息以及一个可选回复。一个业务交易(BusinessTransaction)由两个业务交易通道(BusinessTransactionSwimlane)构成,一个业务交易通道(BusinessTransactionSwimlane)仅被指派一个授权角色(AuthorizedRole),业务交易的两个通道应分配给不同的授权角色。

5.4.3.1.3　业务交易中每个授权角色只完成一个业务动作,一个业务交易(BusinessTransaction)仅由一个请求业务活动(RequestingBusinessActivity)和一个响应业务活动(RespondingBusinessActivity)组成,请求业务活动(RequestingBusinessActivity)和响应业务活动(RespondingBusinessActivity)都是业务动作(BusinessAction)的专用构造型。一个业务动作(BusinessAction)被分配给一个业务交易通道

(BusinessTransactionSwimlane),一个业务交易通道(BusinessTransactionSwimlane)包含一个业务动作(BusinessAction)。一个通道只专用于一个授权角色执行一个业务动作。

5.4.3.1.4 请求业务活动输出请求信息信封作为响应业务活动的输入,响应业务活动创建业务信息且返回请求业务活动,业务信息是可选项。业务交易(BusinessTransaction)仅由一个请求信息信封(RequestingInformationEnvelope)和零或一个响应信息信封(RespondingInformationEnvelope)组成。请求信息信封(RequestingInformationEnvelope)和响应信息信封(RespondingInformationEnvelope)都是信息信封(InformationEnvelope)类型的实例。

5.4.3.1.5 一个请求业务活动(RequestingBusinessActivity)仅输出一个请求信息信封(RequestingInformationEnvelope),而一个请求信息信封(RequestingInformationEnvelope)也仅由一个请求业务活动(RequestingBusinessActivity)创建。一个请求业务活动(RequestingBusinessActivity)可接收到零或一个响应信息信封(RespondingInformationEnvelope)作为输入,而一个响应信息信封(RespondingInformatinEnvelope)也只输至一个请求业务活动(RequestingBusinessActivity)。一个响应业务活动(RespondingBusinessActivity)输出零或一个响应信息信封(RespondingInformationEnvelope),而一个响应信息信封(RespondingInformationEnvelope)也仅由一个响应业务活动(RespondingBusinessActivity)创建。一个响应业务活动(RespondingBusinessActivity)可接收一个请求信息信封(RequestingInformationEnvelope)作为输入,而一个请求信息信封(RequestingInformationEnvelope)也只输出至一个响应业务活动(RespondingBusinessActivity)。

5.4.3.1.6 请求信息信封(或响应信息信封)是基类对象流状态(ObjectFlowState)的构造型,对象流状态(ObjectFlowState)的类型是由基类类(Class)构造型的信息信封(InformationEnvelope)定义,在UML中多重的对象流状态(ObjectFlowState)应是同一类(Class)的实例,因此不同请求或响应信息信封可以是同一信息信封(InformationEnvelope),即一个信息信封可重用于不同的业务交易。

5.4.3.1.7 业务交互视图(业务交易视图)[BusinessInteractionView(BusinessTransactionView)]概念框架,见图30。

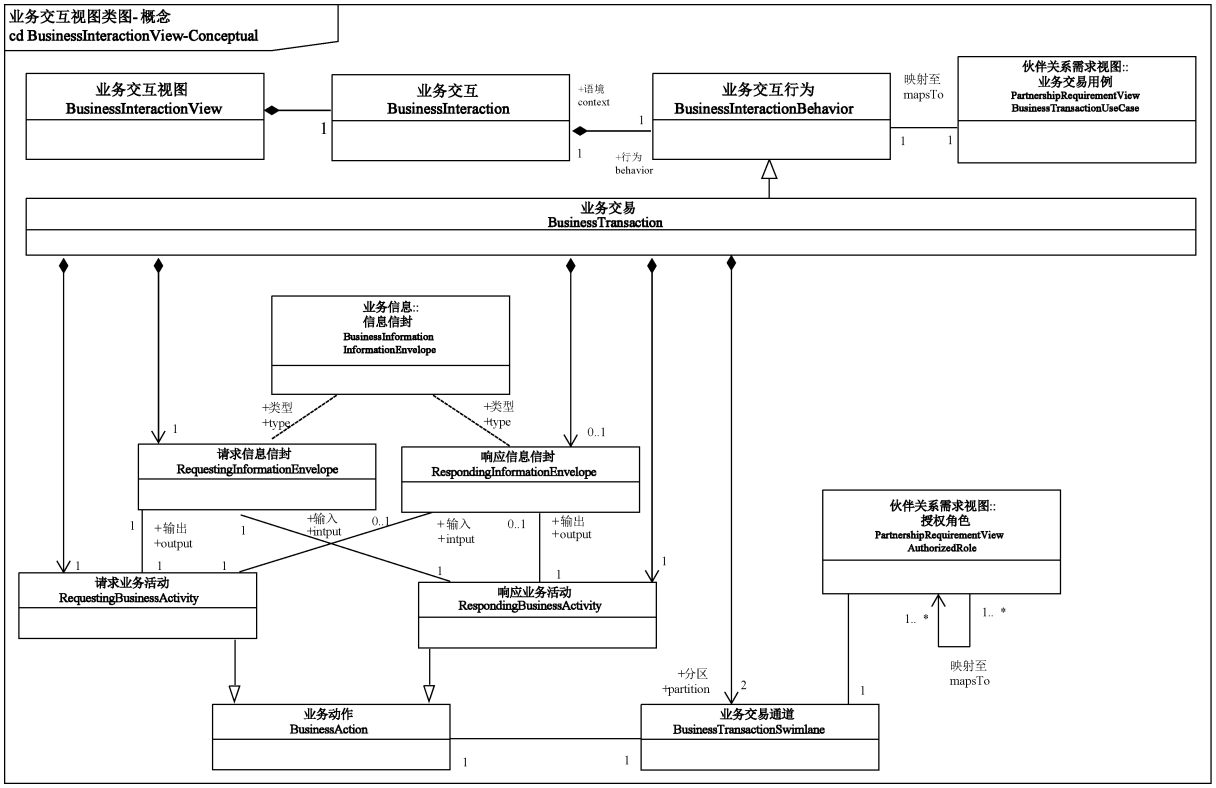

图30 业务交互视图(业务交易视图)概念框架

5.4.3.2 构造型和标签定义(规范性)

5.4.3.2.1 业务交互视图(业务交易视图)[BusinessInteractionView(BusinessTransactionView)]抽象语法,见图 31。

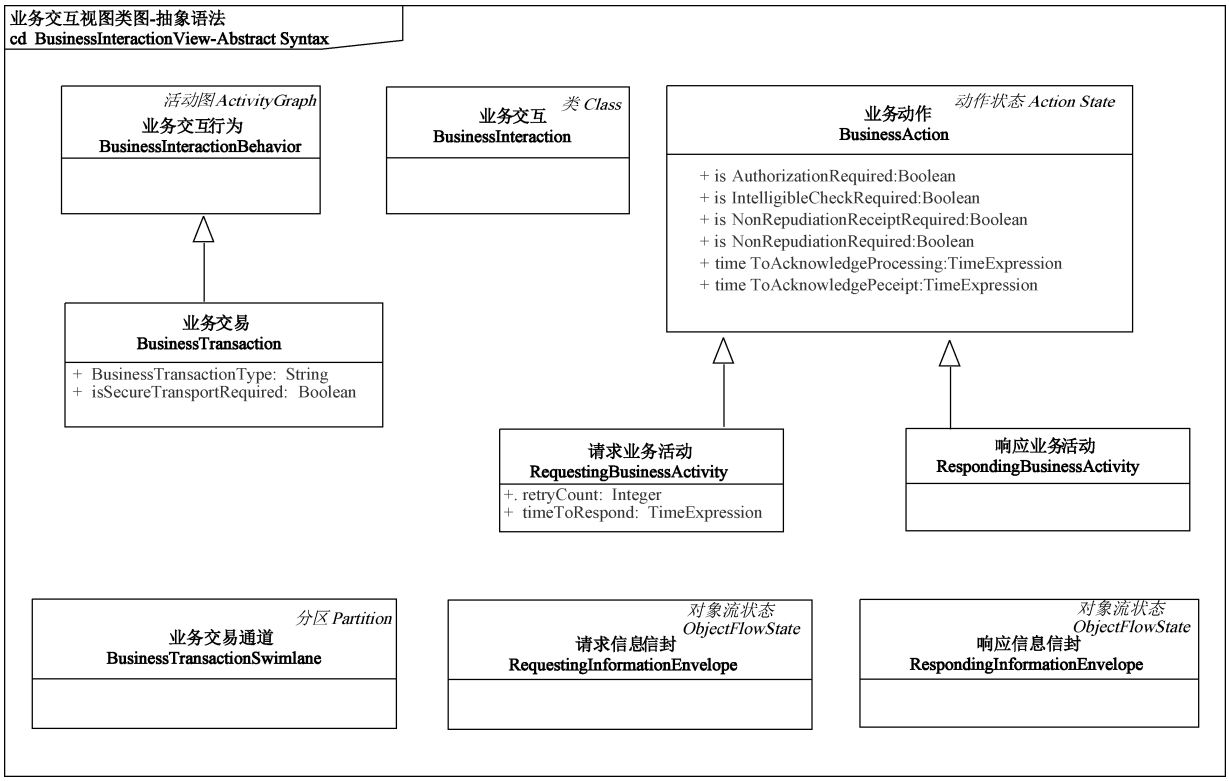

图 31 业务交互视图(业务交易视图)抽象语法

5.4.3.2.2 构造型和标签定义-业务交互,见表 39。

表 39 构造型和标签定义-业务交互

构造型(Stereotype)	业务交互(BusinessInteraction)
基类(Base Class)	类(Class)
父类(Parent)	无(N/A)
描述(Description)	业务交互是两个授权角色之间持续进行业务同步过程的表示,是允许回滚的一个工作单位
标签定义(Tag Definition)	无标签值

5.4.3.2.3 构造型和标签定义-业务交互行为(抽象),见表 40。

表 40 构造型和标签定义-业务交互行为(抽象)

构造型(Stereotype)	业务交互行为(抽象)[BusinessInteractionBehavior(abstract)]
基类(Base Class)	活动图(ActivityGraph)
父类(Parent)	无(N/A)

表 40（续）

构造型(Stereotype)	业务交互行为(抽象)[BusinessInteractionBehavior(abstract)]
描述(Description)	业务交互行为定义了动作编排以及所涉及的业务信息和两个授权角色之间的业务信号交换，该信息交换将触发业务状态同步
标签定义(Tag Definition)	无标签值

5.4.3.2.4 构造型和标签定义-业务交易，见表 41。

表 41 构造型和标签定义-业务交易

构造型(Stereotype)	业务交易(BusinessTransaction)	
基类(Base Class)	活动图(ActivityGraph)	
父类(Parent)	业务交互行为(BusinessInteractionBehavior)	
描述(Description)	业务交易是定义两个授权角色之间业务编排的基本构造单元，一个授权角色识别可改变业务对象状态时启动业务交易与另一相关授权角色同步，该业务交易是触发交互的两个信息系统间的状态同步最小单元，业务交易分单向和双向业务交易。单向交易中，发起的授权角色报告已经生效且不可逆的状态改变，而响应的授权角色应接受这一状态改变，因此业务信息流只从发起方到响应方，如发货通知或目录中产品更新。双向交易中，从发起方发起的信息流将业务对象设定在过渡状态，而反向的信息流来自响应方并决定不可逆的最终状态改变；如注册请求、产品搜索等。 在业务语境中，不可逆意味着要返回到一个原始状态需要另一个业务交易；若一旦业务交易中达成一个购买订单协议后不再允许业务回滚，需要执行另一个取消订单的业务交易。交易类型分两个单向业务交易和四个双向业务交易，在业务交易类型的标签值中表示。另一个标签值提供服务质量参数。 业务交易遵循相同模式：一个业务交易在两个授权角色之间进行，每个授权角色仅被分配到一个业务交易通道，且仅执行一项活动。请求和响应业务活动之间的对象流是必选的，反向对象流则是可选的。根据业务交易语义，请求业务活动在发送信息信封后不结束，响应业务活动可输出响应返回给仍然存续的请求业务活动	
	业务交易类型(businessTransactionType)	
	类型(Type)	字符串(String) Enumeration(枚举)：“Commercial Transaction(商业交易)”“Request/Confirm(请求/确认)”“Query/Response(查询/回复)”“Request/Response(请求/回复)”“Notification(通知)”“Information Distribution(信息发布)”
	次数(Multiplicity)	1
	描述(Description)	业务交易类型决定相应的业务交易模式。业务交易模式提供建立业务交易的语言和语法。业务交易类型遵循下列六个约定的属性值之一： a)商业交易—— 用于对“报价和接受”的业务交易过程的建模，该过程确立了双方履行合同条款的义务。 b)查询/回复——用于查询响应伙伴已有的信息，如数据库中固有数据。 c)请求/回复——用于业务合同中，当一个发起方请求响应方已有信息时，以及当业务信息的请求需要相互影响和依存的一组结果时。 d)请求/确认——用于当一个发起方所请求的信息仅需要先前确立的合同相关的确认时，或与响应方的业务规则相关的确认。 e)信息发布——用于非正式信息交换的业务交易建模，此模式不需要考虑不可抵赖性。 f)通知——用于正式信息交换的业务交易建模，此模式需要考虑不可抵赖性

表 41（续）

构造型（Stereotype）	业务交易（BusinessTransaction）	
描述（Description）	安全传输要求（isSecureTransportRequired）	
	类型（Type）	布尔值（Boolean）
	次数（Multiplicity）	1
	描述（Description）	双方均应同意通过安全的传输通道进行业务信息交换。下列安全控制策略防止业务文件内容的非授权披露或修改，防止业务服务的非授权访问。这是点对点安全性要求。注意一旦业务信息离开网络并进入企业内部，则不在此保护要求范围。以下是对安全传输通道的要求。 验证发送方身份—— 核实发起交互的发送方（员工或组织）身份的真实性。如通过比较个人和照片，有照片的驾驶执照或护照文件可用于验证个人的身份。 验证接收人的身份——核实接收交互的接收人（员工或组织）身份真实性。 验证内容完整性——验证交互中所交换内容的完整性。如检查内容是否被第三方修改过。 维护内容的机密性——机密性确保只有预先确定的接收人才能够读取交互内容。交互中交换的信息在发送时应加密，接收时进行解密。如密封信封为保证只有接收方才能够读取内容

5.4.3.2.5 构造型和标签定义-业务协交易通道，见表 42。

表 42 构造型和标签定义-业务协交易通道

构造型（Stereotype）	业务交易通道（BusinessTransactionSwimlane）
基类（Base Class）	分区（Partition）
父类（Parent）	无（N/A）
描述（Description）	业务交易通道用于定义责任区。一个授权角色被指定给一个业务交易通道分区。该授权角色将对其责任区中分配的业务动作负责
标签定义（Tag Definition）	无标签值

5.4.3.2.6 构造型和标签定义-业务动作（抽象），见表 43。

表 43 构造型和标签定义-业务动作（抽象）

构造型（Stereotype）	业务动作（抽象）[BusinessAction（abstract）]
基类（Base Class）	动作状态（ActionState）
父类（Parent）	无（N/A）
描述（Description）	业务动作由授权角色在业务交易中执行。业务动作是一个抽象构造型，意指业务动作或者是一个请求业务活动或者是一个响应业务活动

表 43（续）

构造型（Stereotype）	业务动作（抽象）[BusinessAction（abstract）]	
标签定义（Tag Definition）	授权要求（IsAuthorizationRequired）	
	类型（Type）	布尔值（Boolean）
	次数（Multiplicity）	1
	描述（Description）	倘若授权角色需要授权以请求一个业务动作或响应一个业务动作，则发送方应签署交换的业务文件并且接收方应验证业务控制的有效性并认可授权方。倘若发送方未被授权执行业务活动，那么接收方应发送一个授权异常的信号。倘若接收方未被授权执行响应业务活动，那么发送方应发送授权失败的通知
	不可抵赖要求（isNonRepudiationRequired）	
	类型（Type）	布尔值（Boolean）
	次数（Multiplicity）	1
	描述（Description）	此不可抵赖要求标签用于指示参与方不得拒绝执行输入/输出业务信息的业务动作
	不可抵赖收据要求（isNonRepudiationReceiptRequired）	
	类型（Type）	布尔值（Boolean）
	次数（Multiplicity）	1
	描述（Description）	不可抵赖收据要求标签要求信息信封的接收方发送一个已签名的收据。不可抵赖收据要求标签指示参与方不可拒绝执行发送已签名收据
	确认接收时间（timeToAcknowledge Receipt）	
	类型（Type）	时间表达式（TimeExpression）
	次数（Multiplicity）	1
	描述（Description）	双方可同意在一定期限内互相核对接收业务信息。接收确认既可以为接收到请求业务信息而发送，也可为接收到响应业务信息而发送，因此发送方既可能是请求授权角色，也可能是响应授权角色。同样，也可使用确认方表示请求授权角色或响应授权角色，取决于经确认接收的是请求业务信息还是响应业务信息。 倘若确认方不能够在协议期限内确认业务信息的接收无误则应退出交易。若有必要，如确认人不能在协议期限内确认已接收业务信息，则发送方应重启业务交易，或发送业务控制失败通知（可能取消合同要约）。确认接收的最长时间为从发送方发送业务信息到发送方“确切收到”接收确认之时。接收确认是一个可审核的业务信号，而且在合同形成过程中有助于合同义务转移（如要约/接受）

表 43（续）

构造型(Stereotype)	业务动作(抽象)[BusinessAction(abstract)]	
标签定义 (Tag Definition)	确认处理时间(timeToAcknowledgeProcessing)	
	类型(Type)	时间表达式(TimeExpression)
	次数(Multiplicity)	1
	描述(Description)	依据需要确认的是请求业务信息还是响应业务信息，发送方既可以是请求授权角色或者响应授权角色。同样确认方也可是这两个角色之一。因此，可使用发送方和确认方两个术语来表示处理确认这个语义。 在请求业务信息通过一组业务规则并提请处理后，双方可能同意需要响应方返回的处理确认。业务信息处理的确认时间为从发送方发送业务信息到发送方“确切收到”处理确认之时。倘若确认方不能在最大期限内确认处理业务信息，则应退出业务交易。倘若确认方不能在约定期限内确认处理业务信息，若有必要，发送人应重启业务交易或者应发送业务控制失败通知(可能撤销一个合同要约)
	清晰度检查要求(isIntelligibleCheckRequired)	
	类型(Type)	布尔值(Boolean)
	次数(Multiplicity)	1
	描述(Description)	为定义清晰度检查要求的语义，仍使用术语发送方和确认方。 在接收确认返回给业务信息发送方之前，双方同意确认方应检查业务信息是否明晰易懂。当文档“可访问”时应返回接收确认，但在业务伙伴不通过异步服务提供商进行交互时，最好在点对点同步业务网络的同时也检查乱码传输

5.4.3.2.7 构造型和标签定义-请求业务活动，见表 44。

表 44 构造型和标签定义-请求业务活动

构造型(Stereotype)	请求业务活动(RequestingBusinessActivity)	
基类(Base Class)	动作状态(ActionState)	
父类(Parent)	业务动作(BusinessAction)	
描述(Description)	请求业务活动是一个授权角色向另一个授权角色请求业务服务时所执行的业务动作	
标签定义 (Tag Definition)	响应时间(timeToRespond)	
	类型(Type)	时间表达式(TimeExpression)
	次数(Multiplicity)	1
	描述(Description)	在双向业务交易中，双方同意响应授权角色应在规定期限内返回响应业务信息 响应授权角色倘若不能在约定期限内返回响应业务信息则应退出交易。视需要，响应授权角色在不能在约定期限内交付响应业务信息的情况下，请求授权角色应重启业务交易，或发送业务控制失败通知(可能取消一项合同要约)。执行时间为从请求授权角色发送请求业务信息到请求授权角色“确切接收”到返回的响应业务信息之时的最长时间

表 44（续）

构造型(Stereotype)	请求业务活动(RequestingBusinessActivity)	
标签定义 (Tag Definition)	重试计数(retryCount)	
	类型(Type)	整数(Integer)
	次数(Multiplicity)	1
	描述(Description)	倘若系统给出超时异常信号，包括接收确认超时、处理确认超时或响应超时等，请求授权角色应按规定的重启次数多次重启业务交易。该参数仅适用于超时信号而非文档内容异常或序列验证异常的情况
	继承标签值(Inherited tagged value)： ——授权要求(isAuthorizationRequired) ——不可抵赖要求(isNonRepudiationRequired) ——不可抵赖收据要求(isNonRepudiationReceiptRequired) ——确认处理时间(timeToAcknowledgeReceipt) ——确认接收时间(timeToAcknowledgeAcceptance) ——清晰度检查要求(isIntelligibleCheckRequired)	

5.4.3.2.8 构造型和标签定义-响应业务活动，见表 45。

表 45 构造型和标签定义-响应业务活动

构造型(Stereotype)	响应业务活动(RespondingBusinessActivity)
基类(Base Class)	动作状态(ActionState)
父类(Parent)	业务动作(BusinessAction)
描述(Description)	响应业务活动是由一个授权角色响应另一个授权角色的业务服务要求而执行的业务动作
标签定义 (Tag Definition)	继承标签值(Inherited tagged value)： ——授权要求(isAuthorizationRequired) ——不可抵赖要求(isNonRepudiationRequired) ——不可抵赖收据要求(isNonRepudiationReceiptRequired) ——确认处理时间(timeToAcknowledgeReceipt) ——确认接收时间(timeToAcknowledgeAcceptance) ——清晰度检查要求(isIntelligibleCheckRequired)

5.4.3.2.9 构造型和标签定义-请求信息信封，见表 46。

表 46 构造型和标签定义-请求信息信封

构造型(Stereotype)	请求信息信封(RequestingInformationEnvelope)
基类(Base Class)	对象流状态(ObjectFlowState)
父类(Parent)	无(N/A)
描述(Description)	请求信息信封是一个业务信息的容器，从请求授权角色发送给响应授权角色以表示一个或多个业务实体的状态变化。这种业务状态变化在单向业务交易中是不可逆的，而在双向业务交易中是临时状态。 请求信息信封这个术语不代表此业务信息是商业意义上的请求。请求信息信封是指请求授权角色对响应授权角色执行交易的请求，无论它是信息发布、通知、请求或商业交易中的要约
标签定义(Tag Definition)	无标签值

5.4.3.2.10 构造型和标签定义-响应信息信封，见表47。

表47 构造型和标签定义-响应信息信封

构造型(Stereotype)	响应信息信封(RespondingInformationEnvelope)
基类(Base Class)	对象流状态(ObjectFlowState)
父类(Parent)	无(N/A)
描述(Description)	响应信息信封是一个业务信息的容器，在双向业务交易中由响应授权角色发送给请求授权角色，以便将一个或多个业务实体设定到最终状态(之前处于中间临时状态)
标签定义(Tag Definition)	无标签值

5.4.3.3 约束(规范性)

5.4.3.3.1 业务交互视图(BusinessInteractionView)程序包应且仅包含一个业务交互(BusinessInteraction)，无其他元素。

示例：

```
package model_management
context Package

inv BIVcontainsExactlyOneBI:
  self.isBusinessInteractionView() implies
  self.contents->one(isBusinessInteraction())
  and self.contents->size()=1
```

5.4.3.3.2 业务交互行为(BusinessInteractionBehavior)应通过与构造型映射至(mapsTo)的依赖对象与一个业务交易用例(BusinessTransactionUseCase)相联系。

示例：

```
package Behavioral_ Elements:: Activity_ Graphs
context ActivityGraph

inv BIBmapsToExactlyOneBusinessTransactionUseCase:
  self.isBusinessInteractionBehavior() implies
  self.clientDependency->size() = 1 and
  self.clientDependency->forAll(d | d.isMapsToDependency() and
  d.supplier->forAll(isBusinessTransactionUseCase()) and
  d.supplier->size=1)
```

5.4.3.3.3 业务交互(BusinessInteraction)的行为应仅由一个业务交互行为(BusinessInteractionBehavior)进行描述。

示例：

```
package Foundation:: Core
context Class

inv BehaviorOfBIdescribedByExactlyOneBusinessInteractionBehavior:
  self.isBusinessInteraction() implies
  self.behavior->one(isBusinessInteractionBehavior()) and
  self.behavior->size()=1
```

5.4.3.3.4 业务交易(BusinessTransaction)应仅有两个分区,每个分区以业务交易通道(BusinessTransactionSwimlane)为构造型。一个分区应包含请求业务活动(RequestingBusinessActivity),另一个应包含响应业务活动(RespondingBusinessActivity)。

示例:

```
package Behavioral_ Elements::Activity_ Graphs
context ActivityGraph

inv BusinessTransactionHasExactlyTwoBTSwimlanes:
  self.isBusinessTransaction() implies
  self.oclAsType(ActivityGraph).partition->size() = 2
  and self.oclAsType(ActivityGraph).partition->forAll(part | part.isUMMTransactionSwimlane()
  and (part.contents->one(isRequestingBusinessActivity()) xor part.contents
  ->one(isRespondingBusinessActivity())))
  and self.oclAsType(ActivityGraph).partition->collect(part | part.contents)->one(isRequestingBusinessActivity())
  and self.oclAsType(ActivityGraph).partition->collect(part | part.contents)->one(isRespondingBusinessActivity())
```

5.4.3.3.5 业务交易通道(BusinessTransactionSwimlane)应有一个分类器,其应是相应的业务交易用例(BusinessTransactionUseCase)的关联授权角色(AuthorizedRole)之一。

示例:

```
package Behavioral_ Elements::Activity_ Graphs
context Partition

inv BusinessTransactionSwimlaneClassifier:
  self.isUMMTransactionSwimlane() implies
  self.classifierRole.base->size()=1 and self.activityGraph.clientDependency->
  collect(s | s.supplier)->collect(a | a.oclAsType(UseCase).associations)-> collect(allConnections)
  ->select(isAuthorizedRole())->one(x | x = (self.classifierRole.base-> asSequence->first()))
```

5.4.3.3.6 请求授权角色的分区应仅包含一个请求业务活动(RequestingBusinessActivity),一个请求信息信封(RequestingInformationEnveleope)和一个初始状态(InitialState)。另外此业务交易通道(BusinessTransactionSwimlane)中应有至少两个最终状态(FinalState)。

示例:

```
package Behavioral_ Elements::Activity_ Graphs
context Partition

inv ContentsOfRequestingPartition:
  self.isUMMTransactionSwimlane() implies
  self.contents->one(isRequestingBusinessActivity()) implies
  self.contents->forAll(isRequestingBusinessActivity()
  or isRequestingInformationEnvelope()
  or isInitialState()
  or isFinalState()
  or isTransition()
  )
  and
  self.contents->one(isRequestingInformationEnvelope()) and
  self.contents->select(isFinalState())->size()>1 and
  self.contents->one(isInitialState())
```

5.4.3.3.7 响应授权角色的分区应仅包含一个响应业务活动(RespondingBusinessActivity)。另外,倘若是双向业务交易,那么分区也应包含一个响应信息信封(RespondingInformationEnvelope)。倘若是单向业务交易,那么响应者分区不应含有响应信息信封(RespondingInformationEnvelope)。

示例:

```
package Behavioral_ Elements:: Activity_ Graphs
context Partition

inv ContentsOfResponderPartition :
  self.isUMMTransactionSwimlane() implies
  self.contents->one(isRespondingBusinessActivity()) implies
  self.contents->forAll(isRespondingBusinessActivity()
  or isRespondingInformationEnvelope()
  or isTransition()
  )
  and if
  self.activityGraph.isTwoWayTransaction()
  then
  self.contents->one(isRespondingInformationEnvelope())
  else
  not self.contents->exists(isRespondingInformationEnvelope())
  endif
```

5.4.3.3.8 应仅有一个引发从初始状态(InitialState)到请求业务活动(RequestingBusinessActivity)的转换(Transition)。

示例:

```
package Behavioral_ Elements:: Activity_ Graphs
context Partition

inv TrInitialState2RequestingBusinessActivity:
  self.isUMMTransactionSwimlane() implies
  self.contents->one(isRequestingBusinessActivity()) implies
  self.contents->select(isInitialState())->
```

5.4.3.3.9 应有一个引发从请求业务活动(RequestingBusinessActivity)到请求信息信封(RequestingInformationEnvelope)的转换(Transition)。

示例:

```
package Behavioral_ Elements:: Activity_ Graphs
context Partition

inv TrRequestingBusinessActivity2RequInfEnvelope:
  self.isUMMTransactionSwimlane() implies
  self.contents->one(isRequestingBusinessActivity()) implies
  self.contents->select(isRequestingBusinessActivity())->
  forAll(oclAsType(ActionState).outgoing->size()=1 and
  oclAsType(ActionState).outgoing->asSequence()
  ->first().target.isRequestingInformationEnvelope())
```

5.4.3.3.10 应仅有一个引发从请求信息信封(RequestingInformationEnvelope)到响应业务活动(Re-

spondingBusinessActivity)的转换(Transition)。

示例：

```
package Behavioral_ Elements::Activity_ Graphs
context Partition

inv TrRequestingInformationEnvelope2RespondingBusinessActivity:
  self.isUMMTransactionSwimlane() implies
  self.contents->one(isRequestingBusinessActivity()) implies
  self.contents->select(isRequestingInformationEnvelope())->
  forAll(oclAsType(ObjectFlowState).outgoing->size()=1 and
  oclAsType(ObjectFlowState).outgoing->asSequence
  ->first().target.isRespondingBusinessActivity())
```

5.4.3.3.11 应仅有一个引发从响应业务活动(RespondingBusinessActivity)到响应信息信封(RespondingInformationEnvelope)的转换(Transition)。(只针对双向业务交易)

示例：

```
package Behavioral_ Elements::Activity_ Graphs
context Partition

inv TrRespondingBusinessActivity2RespondingInformationEnvelope:
  self.activityGraph.isTwoWayTransaction() implies
  self.contents->one(isRespondingBusinessActivity()) implies
  self.contents->select(isRespondingBusinessActivity())->
  forAll(oclAsType(ActionState).outgoing->size()=1 and
  oclAsType(ActionState).outgoing->asSequence
  ->first().target.isRespondingInformationEnvelope())
```

5.4.3.3.12 应仅有一个引发从响应信息信封(RespondingInformationEnvelope)到请求业务活动(RequestingBusinessActivity)的转换(Transition)。(只针对双向业务交易)

示例：

```
package Behavioral_ Elements::Activity_ Graphs
context Partition

inv TrRespondingInformationEnvelope2RequestingBusinessActivity:
  self.activityGraph.isTwoWayTransaction() implies
  self.contents->one(isRespondingBusinessActivity()) implies
  self.contents->select(isRespondingInformationEnvelope())->
  forAll(oclAsType(ObjectFlowState).outgoing->size()=1 and
  oclAsType(ObjectFlowState).outgoing->asSequence
  ->first().target.isRequestingBusinessActivity())
```

5.4.3.3.13 可以有从响应业务活动(RespondingBusinessActivity)到请求业务活动(RequestingBusinessActivity)的转换(Transition)。(仅用于单向业务交易)

示例：

```
package Behavioral_Elements::Activity_Graphs
context Partition
```

```
inv TrPossibleRespondingInformationEnvelope2RequestingBusinessActivity:
self.activityGraph.isOneWayTransaction() implies
self.contents->one(isRespondingBusinessActivity()) implies
self.contents->select(isRespondingBusinessActivity())->
forAll(oclAsType(ActionState).outgoing->size()=1 and
(oclAsType(ActionState).outgoing->asSequence
->first().target.isRequestingBusinessActivity() or
oclAsType(ActionState).outgoing->isEmpty()))
```

5.4.3.3.14 应有引发从请求业务活动(RequestingBusinessActivity)到每个最终状态(FinalState)的转换(Transition)。

示例:

```
package Behavioral_Elements::Activity_Graphs
context Partition

inv TrRequestingBusinessActivity2FinalState:
  self.isUMMTransactionSwimlane() implies
  self.contents->one(isRequestingBusinessActivity()) implies
  self.contents->select(isRequestingBusinessActivity())->
  forAll(oclAsType(ActionState).outgoing->size()=1 and
  oclAsType(ActionState).outgoing->asSequence
  ->first().target.isFinalState())
```

5.4.3.3.15 每个请求信息信封(RequestingInformationEnvelope)和每个响应信息信封(RespondingInformationEnvelope)应有一个分类器,分类器自身应为一个类并为信息信封(InformationEnvelope)构造型。

示例:

```
package Behavioral_Elements::Activity_Graphs
context ObjectFlowState

inv ObjectFlowStateHasClassifier:
  (self.isRequestingInformationEnvelope() or
  self.isRespondingInformationEnvelope()) implies
  self.type.oclAsType(ClassifierInState).type.isInformationEnvelope()
```

5.4.3.4 实例(资料性)

业务交互视图(业务交易视图)[BusinessInteractionView (BusinessTransactionView)]实例:下订单业务交易)(活动图)[PlaceOrder BusinessTransaction (ActivityGraph)],见图 32。

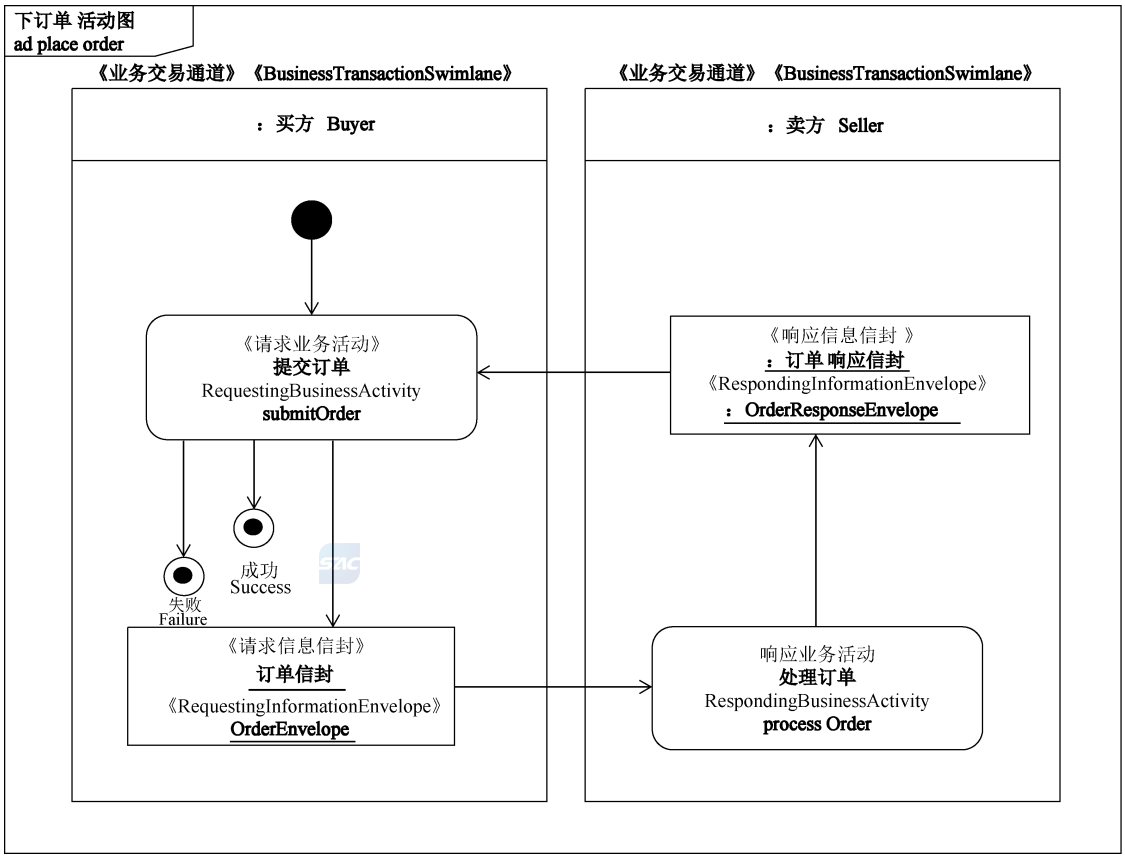

图 32 业务交互视图(业务交易视图)实例:下订单业务交易(活动图)

5.4.4 业务信息视图

5.4.4.1 概念性描述(资料性)

5.4.4.1.1 业务信息视图描述了一个业务交互中的信息交换。请求信息信封(RequestingInformation-Envelope)和响应信息信封(RequestingInformationEnvelope)是信息信封(InformationEnvelope)这一类中的两个类型。

5.4.4.1.2 信息信封的作用是业务活动和响应业务活动之间相互交互时各自交换信息的信封。信封中所包含的信息由以信息实体(InformationEntity)为构造型的类构成。信息实体可以递归嵌套,以添加多层信息实体构成的层次结构。信息信封由从零个或一个头(header)和一个或多个体(body)构成。头和体都表示为信息实体。一个信息信封(InformationEnvelope)仅由零或一个带有角色名的头(header)的信息实体(InformationEntity)以及一或多个带有角色名的体(body)的信息实体(InformationEntitie)构成。信息信封(InformationEnvelope)是满足了信息信封所有规则的信息实体(InformationEntity)的专用构造型。UMM 基本模块不定义构建信息实体的具体规则,其他可高质量构建类图的建模方法和规则也适用于信息信封以及内容的建模,如 UN/CEFACT 的核心构件(Core Component)方法。

5.4.4.1.3 无论使用何种类型的核心构件(Core Component)作为信息实体(InformationEntity)的构造型,所产生的类很重要。为了更好地支持核心构件对业务信息建模,还需专用模块-核心构件的 UML 概要文档(Core Component UML Profile)。

5.4.4.1.4 业务信息视图(业务交易视图)[BusinessInformationView(BusinessTransactionView)]概念框架,见图 33。

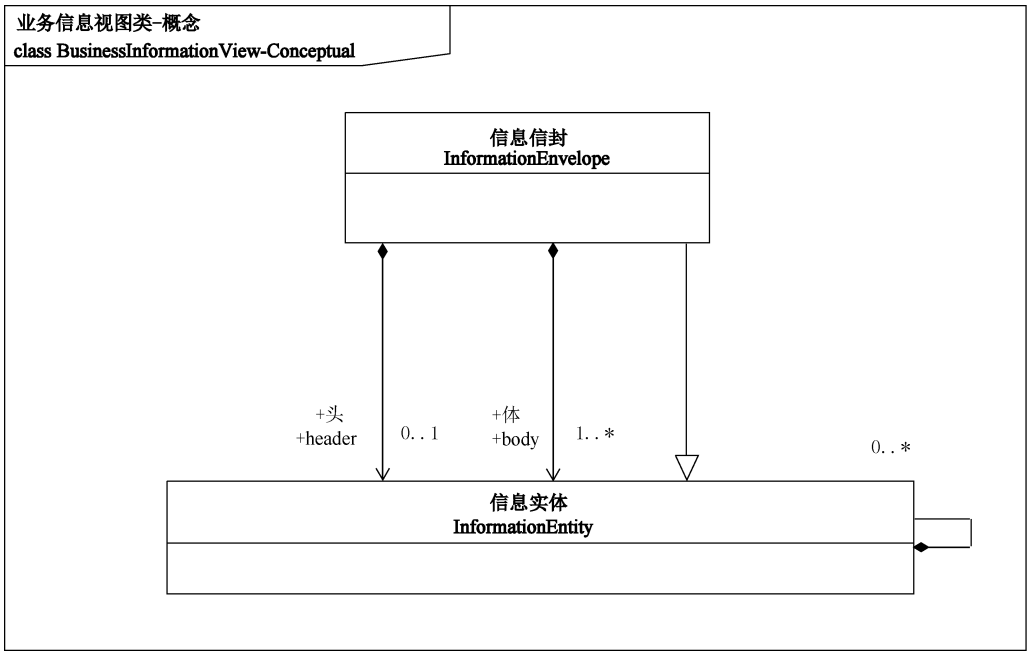

图 33　业务信息视图(业务交易视图)概念框架

5.4.4.2　构造型和标签定义(规范性)

5.4.4.2.1　业务信息视图(业务交易视图)[BusinessInformationView(BusinessTransactionView)]抽象语法,见图 34。

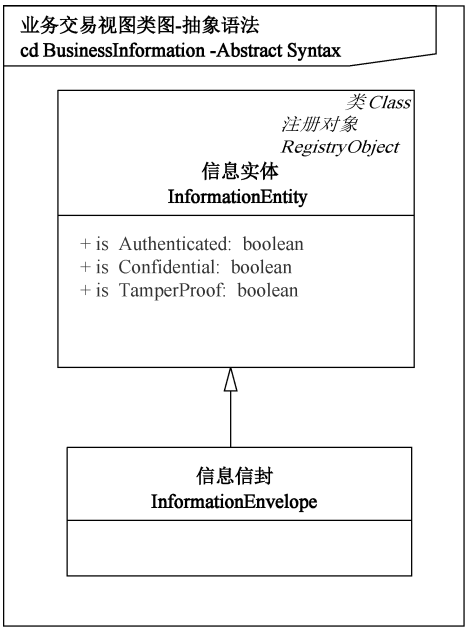

图 34　业务信息视图(业务交易视图)抽象语法

5.4.4.2.2　构造型和标签定义-信息实体,见表 48。

表 48　构造型和标签定义-信息实体

构造型(Stereotype)	信息实体(InformationEntity)	
基类(Base Class)	类(Class)	
父类(Parent)	无(N/A)	
描述(Description)	信息实体实现了业务交易中执行活动的授权角色之间交换的结构化业务信息。信息实体通过关联来包含或引用其他信息实体	
标签定义(Tag Definition)	机密要求(isConfidential)	
	类型(Type)	布尔值(Boolean)
	次数(Multiplicity)	1
	描述(Description)	倘若设置了此标志,表示信息实体被加密,因而未经授权的参与方不能查看该信息
	防篡改要求(isTamperProof)	
	类型(Type)	布尔值(Boolean)
	次数(Multiplicity)	1
	描述(Description)	倘若设置了此标志,表示信息实体有加密的信息摘要,可用于检查信息是否被篡改。这要求与文档实体相关的数字签名(发送方的数字证书和加密信息摘要)
	认证要求(isAuthenticated)	
	类型(Type)	布尔值(Boolean)
	次数(Multiplicity)	1
	描述(Description)	倘若设置了此标志,表示有与文档实体相关的数字证书。为签名人的身份提供证明

5.4.4.2.3　构造型和标签定义-信息信封,见表 49。

表 49　构造型和标签定义-信息信封

构造型(Stereotype)	信息信封(InformationEnvelope)
基类(Base Class)	类(Class)
父类(Parent)	信息实体(InformationEntity)
描述(Description)	信息信封是一个信息实体的容器,是信息实体的具体化规范。信息信封通过包含一个头(header)角色的信息实体和至少一个体(body)角色的信息实体,扩展了信息实体的概念。此外,业务交易中所交换的信息一般是信息信封类型,如请求业务信息和响应业务信息
标签定义(Tag Definition)	**继承标签值(Inherited tagged value)**: ——机密要求(isConfidential) ——防篡改要求(isTamperProof) ——认证要求(isAuthenticated)

5.4.4.3 限制(规范性)

5.4.4.3.1 业务信息视图(BusinessInformationView)程序包应只包含信息实体(InformationEntity)和信息信封(InformationEnvelope)而无其他元素。

示例:

```
package Foundation::Core
context Class

inv AllowedElementsInBusinessInformationView:
  self.isBusinessInformationView() implies
  self.contents->forAll(a | a.isInformationEntity() or a.isInformationEnvelope())
```

5.4.4.3.2 信息信封(InformationEnvelope)应与一个带有角色名头(header)的信息实体(InformationEntity)有零或一个关联。

示例:

```
package Foundation::Core
context Class

inv InformationEnvelopeHasHeader:
  self.isInformationEnvelope() implies
  self.associations->size() < 1 and
  self.associations->forAll(a | a.connection->size() = 2 and
  a.allConnections->one(participant.isInformationEntity() and
  AssociationEndRole.name = 'header'))
```

5.4.4.3.3 信息信封(InformationEnvelope)应至少有一个关联信息实体(InformationEntity),该信息实体带有角色名体(body)。

示例:

```
package Foundation::Core
context Class

inv InformationEnvelopeHasBodies:
  self.isInformationEnvelope() implies
  self.associations->forAll(a | a.connection->size() = 2 and
  a.allConnections->exists(participant.isInformationEntity() and
  AssociationEndRole.name = 'body'))
```

5.4.4.3.4 一个信息实体(InformationEntity)可包含其他信息实体(InformationEntity)。

示例:

```
package Foundation::Core
context Class

inv contentsOfInformationEntitiy:
  self.isInformationEntity() implies
  self.associations->
  forAll(a | a.allConnections->exists(isAggregate()) and
  a.allConnections->exists(participant.isInformationEntity()))
```

5.4.4.4 实例(资料性)

业务信息视图(业务交易视图)[BusinessInformationView (BusinessTransactionView)]实例:订单信封(类图)[OrderEnvelope (ClassDiagram)]概念,见图 35。

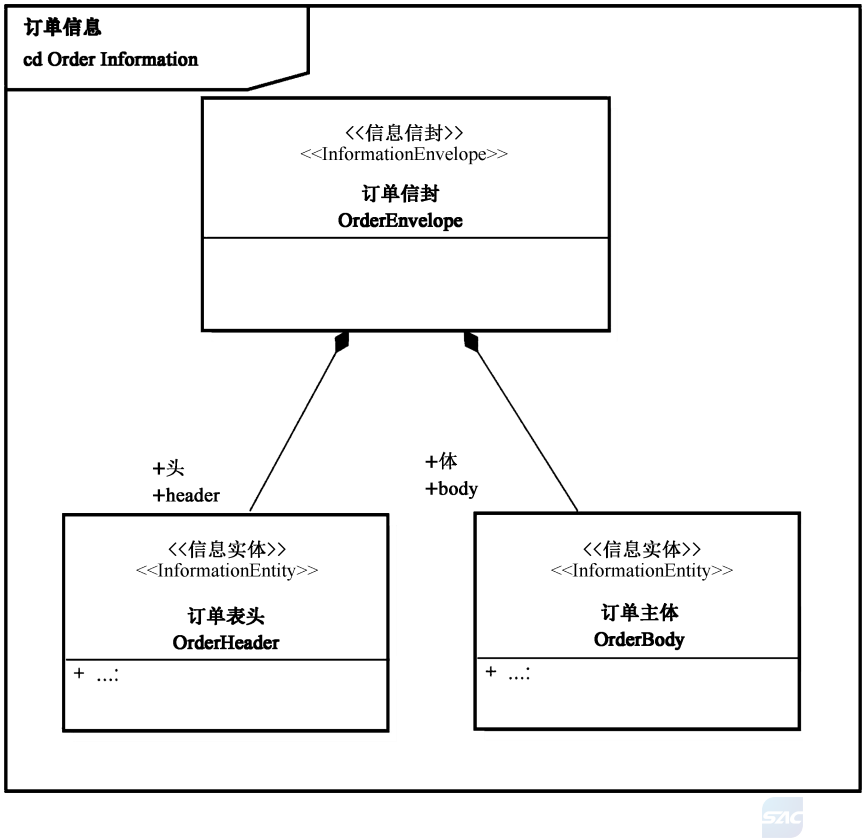

图 35 业务信息视图(业务交易视图)实例:订单信封(类图)概念

5.4.5 BTV 所有程序包中使用的 OCL 方法(规范性)

OCL 方法,参见示例。

示例:

```
package Foundation::Core
context ModelElement

--Predefined method whichs evaluates, if the given Modelelement
--has a stereotype equal to the passed name
def:
let hasStereotype (st : String) : Boolean =
   self.stereotype->select(self.name = st)->notEmpty()
--Predefined method whichs evaluates, if the given element
--has the stereotype 'BusinessTransaction'
def :
let isBusinessTransaction() : Boolean =
   self.oclIsKindOf(ActivityGraph) and
   self.hasStereotype('BusinessTransaction')
```

```
--Predefined method whichs evalutes, if the given element
--has the stereotype 'BusinessInteraction'
def :
let isBusinessInteraction() : Boolean =
  self.oclIsKindOf(Class) and
  self.hasStereotype('BusinessInteraction')
--Predefined method whichs evaluates, if the given element
--is a subtype of 'BusinessInteractionBehavior'
def :
let isBusinessInteractionBehavior() : Boolean =
  self.oclIsKindOf(ActivityGraph) and
  self.hasStereotype('BusinessTransaction')
--Predefined method whichs evaluates, if the given element
--is a 'BusinessChoreography'
def :
let isBusinessChoreography() : Boolean =
  self.oclIsKindOf(Class) and
  self.hasStereotype('BusinessChoreography')
--Predefined method which evaluates, if the
--ActivityGraph is a BusinessCollaborationProtocol
def:
let isBusinessCollaborationProtocol() : Boolean =
  self.oclIsKindOf(ActivityGraph) and
  self.hasStereotype('BusinessCollaborationProtocol')
--Predefined method which evaluates, if the
--ActivityGraph is a subtype of
--BusinessChoreographyBehavior
def:
let isBusinessChoreographyBehavior() : Boolean =
  self.oclIsKindOf(ActivityGraph) and
  self.hasStereotype('BusinessCollaborationProtocol')
--Predefined method which evaluates, if the given element
--has the stereotype 'RequestingBusinessActivity' and
--if its type is ActionState
def :
let isRequestingBusinessActivity() : Boolean =
  self.oclIsKindOf(ActionState) and
  self.hasStereotype('RequestingBusinessActivity')
--Predefined method which evaluates, if the given element
--has the stereotype 'RespondingBusinessActivity' and
--if its type is ActionState
def :
let isRespondingBusinessActivity() : Boolean =
  self.oclIsKindOf(ActionState) and
  self.hasStereotype('RespondingBusinessActivity')
-- Returns true if the element is located in a partition and
-- its stereotype is 'BusinessTransactionSwimlane'
```

```
def :
let isBusinessTransactionSwimlane() : Boolean =
  self.hasStereotype('BusinessTransactionSwimlane')
  and self.oclIsKindOf(Partition)
-- Returns true if the type of the element
-- is 'PseudoKindState' and its Pseudostatekind is pk_initial
def :
let isInitialState() : Boolean =
  self.oclIsKindOf(Pseudostate) and
  self.oclAsType(Pseudostate).kind = PseudostateKind::initial
-- Returns true if the type of the element is 'FinalState'
def:
let isFinalState() : Boolean =
  self.oclIsKindOf(FinalState)
-- Returns true if the type of the element
-- is 'PseudoKindState' and its Pseudostatekind
-- is pk_choice
def:
let isChoice() : Boolean =
  self.oclIsKindOf(Pseudostate) and
  self.oclAsType(Pseudostate).kind = PseudostateKind::choice
-- Returns true if the type of the element
-- is 'PseudoState' and its Pseudostatekind
-- is pk_fork
def:
let isFork() : Boolean =
  self.oclIsKindOf(Pseudostate) and
  self.oclAsType(Pseudostate).kind = PseudostateKind::fork
-- Returns true if the type of the element
-- is 'PseudokindState' and its Pseudostatekind
-- is pk_choice
def:
let isJoin() : Boolean =
  self.oclIsKindOf(Pseudostate) and
  self.oclAsType(Pseudostate).kind = PseudostateKind::join
--Returns true if the given element has a tagged value named 'tag' with
--a value 'value'
def :
let hasTaggedValue (tag : String, value : String) : Boolean =
  self.taggedValue->select(name = tag)->select(dataValue = value)->notEmpty()
--Returns true if the element has a tagged value named 'BusinessTransaction'
--with a value 'NotificationActivity' or 'InformationDistributionActivity'
def :
let isOneWayTransaction() : Boolean =
  self.hasTaggedValue('BusinessTransactionType','NotificationActivity')
  or
  self.hasTaggedValue('BusinessTransactionType','InformationDistributionActivity')
```

```
--Returns true if the element has a tagged value name 'BusinessTransaction'
--with a value 'QueryResponseActivity' or 'RequestResponseActivity' or
--'CommercialTransactionActivity' or 'RequestConfirmActivity'
def :
let isTwoWayTransaction() : Boolean =
  self.hasTaggedValue('BusinessTransactionType','QueryResponseActivity')
  or
  self.hasTaggedValue('BusinessTransactionType','RequestResponseActivity')
  or
  self.hasTaggedValue('BusinessTransactionType','CommercialTransactionActivity')
  or
  self.hasTaggedValue('BusinessTransactionType','RequestConfirmActivity')
-- Returns true if the stereotype of the given element is
--'BusinessCollaborationActivity'
-- and if the type of the element is ActionState
def:
let isBusinessCollaborationActivity() : Boolean =
  self.hasStereotype('BusinessCollaborationActivity') and
  self.oclIsKindOf(ActionState)
-- Returns true if the type of the element is Transition
def:
let isTransition() : Boolean =
  self.oclIsKindOf(Transition)
-- Returns true if the given element is an element of an Acitivity Graph
-- (InitialState, Choice, Fork, Join, Transition or FinalState)
def:
let isPseudoStateOrFinalStateOrTransition() : Boolean =
  isInitialState() or
  isChoice() or
  isFork() or
  isJoin() or
  isFinalState()
--Returns true if a package is stereotyped as BusinessTransactionView
def:
let isBusinessTransactionView() : Boolean =
  self.hasStereotype('BusinessTransactionView') and
  oclIsKindOf(Package)
--Returns true if a package is stereotyped as BusinessChoroeographyView
def:
let isBusinessChoreographyView() : Boolean =
  self.hasStereotype('BusinessChoreographyView') and
  oclIsKindOf(Package)
-- Returns true if the stereotype of the given element is
--'BusinessInformationView'
-- and if the type of the element is Package
def :
let isBusinessInformationView() : Boolean =
```

```
    self.hasStereotype('BusinessInformationView') and
    self.oclIsKindOf(Package)
-- Returns true if the stereotype of the given element is
-- 'BusinessInteractionView'
-- and if the type of the element is Package
def :
let isBusinessInteractionView() : Boolean =
    self.hasStereotype('BusinessInteractionView') and
    self.oclIsKindOf(Package)
-- Returns true if the stereotype of the given element is 'InformationEntitiy'
-- and if the type of the element is Class
def :
let isInformationEntity() : Boolean =
    self.hasStereotype('InformationEntity') and
    self.oclIsKindOf(Class)
-- Returns true if the association type of an association end is composite
def:
let isComposition() : Boolean =
    self.oclIsKindOf(AssociationEnd) and
    self.oclAsType(AssociationEnd).aggregation = AggregationKind::composite
-- Returns true if the association type of an association end is aggregation
def:
let isAggregate() : Boolean =
    self.oclIsKindOf(AssociationEnd) and
    self.oclAsType(AssociationEnd).aggregation = AggregationKind::aggregate
-- Returns true if the element is a partition
-- and stereotyped as BusinessTransactionSwimlane
def :
let isUMMTransactionSwimlane() : Boolean =
    self.oclIsKindOf(Partition) and
    self.hasStereotype('BusinessTransactionSwimlane')
-- Returns true if the stereotype of the element is
-- 'InformationEnvelope' and its type is Class
def :
let isInformationEnvelope() : Boolean =
    self.hasStereotype('InformationEnvelope') and
    oclIsKindOf(Class)
-- Returns true if the stereotype of the element
-- is 'RequestingInformationEnvelope'
def :
let isRequestingInformationEnvelope() : Boolean =
    self.hasStereotype('RequestingInformationEnvelope') and
    oclIsKindOf(ObjectFlowState)
-- Returns true if the stereotype of the element
-- is 'RespondingInformationEnvelope'
def :
let isRespondingInformationEnvelope() : Boolean =
```

```
    self.hasStereotype('RespondingInformationEnvelope') and
    oclIsKindOf(ObjectFlowState)
-- Predefined method which evaluates, if the given element
-- has the stereotype 'mapsTo'
def :
let isMapsToDependency() : Boolean =
    self.oclIsKindOf(Dependency) and
    self.hasStereotype('mapsTo')
-- Predefined method which evaluates, if the given element
-- has the stereotype 'BusinessCollaborationUseCase'
def :
let isBusinessCollaborationUseCase() : Boolean =
    self.oclIsKindOf(UseCase) and
    self.hasStereotype('BusinessCollaborationUseCase')
-- Predefined method which evaluates, if the given element
-- has the stereotype 'BusinessTransactionUseCase'
def :
let isBusinessTransactionUseCase() : Boolean =
    self.oclIsKindOf(UseCase) and
    self.hasStereotype('BusinessTransactionUseCase')
-- Predefined method which evaluates, if the given element
-- has the stereotype 'AuthorizedRole'
def :
let isAuthorizedRole() : Boolean =
    self.oclIsKindOf(Actor) and
    self.hasStereotype('AuthorizedRole')
```

参 考 文 献

[1] ISO/IEC 14662:2004 Open-edi reference model
[2] UN/CEFACT UML Profile for Core Components,UPCC (V 1.0)
[3] UN/CEFACT Core Component Technical Specification(CCTS)(V 2-01)
[4] UN/CEFACT Open Development Process,ODP
[5] UN/CEFACT UMM Meta Model Foundation Module Version 1.0—Technical Specification
[6] Object Management Group,OMG—Unified Modeling Language,UML(V 1.4.2)