

1



2

3

UNITED NATIONS

4

5

6

UNITED NATIONS ECONOMIC COMMISSION FOR EUROPE

7

8

UNITED NATIONS CENTRE FOR TRADE FACILITATION AND  
ELECTRONIC BUSINESS (UN/CEFACT)

9

***UML Profile for UN/CEFACT's Modeling Methodology (UMM)***

***Foundation Module***

***Version 2.0 Technical Specification***

***2011-04-01***

## 10 **Table of Contents**

11	Table of Contents .....	2
12	1 About this Document .....	4
13	1.1 Status of this Document .....	4
14	1.2 Revision History .....	4
15	1.3 Document Context.....	4
16	1.4 Conventions.....	5
17	2 Project Team.....	5
18	2.1 Disclaimer .....	5
19	2.2 Contact .....	5
20	2.3 Project Team Participants.....	6
21	3 Introduction.....	6
22	3.1 Audience.....	6
23	3.2 Related Documents .....	7
24	3.3 UN/CEFACT's Modeling Methodology (UMM): Overview .....	7
25	3.4 Objectives .....	9
26	3.4.1 Goals of the Technical Specification .....	9
27	3.4.2 Requirements .....	9
28	3.4.3 Caveats and Assumptions.....	9
29	3.5 Structure of the UMM Foundation Module .....	10
30	4 Dependencies on other UMM Modules (normative).....	11
31	4.1 Abbreviations of Stereotypes .....	11
32	4.2 Dependency between Base Module and Foundation Module.....	11
33	5 UMM Foundation Module.....	12
34	5.0 Foundation Module Management .....	12
35	5.0.1 Abbreviations of Stereotypes .....	12
36	5.0.2 Conceptual Description (informative) .....	13
37	5.0.3 Stereotypes and Tag Definitions (normative).....	14
38	5.0.4 Constraints (normative).....	16
39	5.1 Business Requirements View.....	16
40	5.1.0 Sub-Views in the Business Requirements View.....	16
41	5.1.1 Business Domain View.....	19
42	5.1.2 Business Partner View .....	35
43	5.1.3 Business Entity View .....	38

44	5.2	Business Choreography View .....	45
45	5.2.1	Sub-Views in the Business Choreography View .....	45
46	5.2.2	Business Transaction View .....	49
47	5.2.3	Business Collaboration View .....	69
48	5.2.4	Business Realization View.....	87
49	5.3	Business Information View .....	93
50	5.3.1	Abbreviations of Stereotypes .....	93
51	5.3.2	Conceptual Description (informative) .....	93
52	5.3.3	Stereotypes and Tag Definitions (normative).....	94
53	5.3.4	Constraints (normative).....	94
54	5.3.5	Example using UPCC (UML Profile for Core Components) (informative).....	95
55	I.	Business Transaction Patterns.....	97
56	II.	OCL Constraints .....	105
57		Copyright Statement .....	133
58			

60 **1 About this Document**

61 **1.1 Status of this Document**

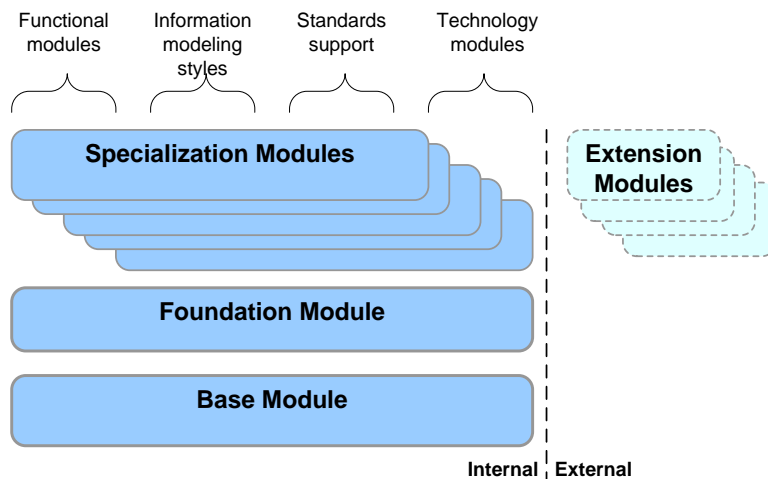
62 This document has completed the Open Development Process (ODP) of UN/CEFACT on 2011-04-01. It is a  
 63 UN/CEFACT Technical Specification

64 **1.2 Revision History**

Version	Release	Date	Comment
Candidate for 2.0	Internal Draft	2008-04-11	
Candidate for 2.0	Public Draft	2008-06-27	
Candidate for 2.0	Implementation Verification	2010-01-25	
Version 2.0	Technical Specification	2011-04-01	

65 **1.3 Document Context**

66 The UMM meta model is divided into a set of meta modules. This means that the UMM meta model is  
 67 partitioned into functional levels, ranging from core, minimal functionality, to complete functionality. The  
 68 following partition levels have been defined for meta modules:



69 **Figure 1 Module structure of the UMM meta model**

70 **Base:** Covers the fundamental principles that are shared across all the other modules.  
 71

72 **Foundation:** Includes the core concepts of the UMM. In addition, it defines all the concepts that are used as  
 73 part of the minimal methodology to produce a UMM compliant business collaboration model. Furthermore,  
 74 it provides fundamental principles which are shared across all other modules.  
 75

76 **Specialization:** Multiple specialization modules might define add-on concepts to the foundation. Each  
77 specialization module addresses a specialized type of analysis that extends the foundation module at a well-  
78 defined extension point for a specific topic. Specialization modules might become candidates for later  
79 inclusion into the foundation module.

80 **Extension:** Extension modules serve the same purpose as specialization modules. Whereas specialization  
81 modules are developed and maintained by UN/CEFACT, extension modules are adding features that are  
82 created and maintained by organization(s) which are external to UN/CEFACT.

83 This specification defines the foundation module of UMM 2.0.

## 84 1.4 Conventions

85 The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED,  
86 MAY and OPTIONAL, when they appear in this document, are to be interpreted as described in [RFC2119] as  
87 quoted here:

- 88 • MUST: This word, or the terms "REQUIRED" or "SHALL", means that the definition is an absolute  
89 requirement of the specification.
- 90 • MUST NOT: This phrase, or the phrase "SHALL NOT", means that the definition is an absolute  
91 prohibition of the specification.
- 92 • SHOULD: This word, or the adjective "RECOMMENDED", means that there may exist valid reasons in  
93 particular circumstances to ignore a particular item, but the full implications MUST be understood  
94 and carefully weighed before choosing a different course.
- 95 • SHOULD NOT: This phrase, or the phrase "NOT RECOMMENDED", means that there may exist valid  
96 reasons in particular circumstances when the particular behavior is acceptable or even useful, but  
97 the full implications should be understood and the case carefully weighed before implementing any  
98 behavior described with this label.
- 99 • MAY: This word, or the adjective "OPTIONAL", means that an item is truly optional. One vendor may  
100 choose to include the item because a particular marketplace requires it or because the vendor feels  
101 that it enhances the product while another vendor may omit the same item. An implementation that  
102 does not include a particular option MUST be prepared to interoperate with another  
103 implementation which does include the option, though perhaps with reduced functionality. In the  
104 same vein an implementation that does include a particular option MUST be prepared to  
105 interoperate with another implementation which does not include the option (except, of course, for  
106 the feature the option provides).

## 107 2 Project Team

### 108 2.1 Disclaimer

109 The views and specification expressed in this document are those of the authors and are not necessarily  
110 those of their employers. The authors and their employers specifically disclaim responsibility for any  
111 problems arising from correct or incorrect implementation or use of this technical specification.

### 112 2.2 Contact

113 Name: Christian Huemer  
114 Company: Vienna University of Technology

115 Street: Favoritenstrasse 9-11/188  
116 City, state, zip/other: 1040 Vienna  
117 Nation: Austria  
118 Phone: +43 1 58801 18882  
119 Email: huemer@big.tuwien.ac.at

## 120 2.3 Project Team Participants

121

122 Project Team Lead: Christian Huemer Austria  
123 Editing Team: Jens Dietrich Germany  
124 Birgit Hofreiter Austria  
125 Christian Huemer Austria  
126 Philipp Liegl Austria  
127 Glenn Miller Canada  
128 Harry Moyer Australia  
129 Rainer Schuster Austria  
130 Marco Zapletal Austria

131

132 Contributors: Steve Capell Australia  
133 Sylvie Colas France  
134 Nils Cordes Germany  
135 Barbara Flügge Switzerland  
136 William McCarthy USA  
137 Thomas Motal Austria  
138 Dennis Müller Germany  
139 Christian Senf Germany  
140 Nita Sharma USA  
141 Kees Sparreboom Nederlands  
142 Gunther Stuhec Germany

143 The Editing Team of this UMM foundation module likes to thank former members of TMG's Business Process  
144 Working Group (BPWG) who have spent enormous efforts in putting the UMM into a stage that we were  
145 able to build upon in order to create this foundation module.

## 146 3 Introduction

### 147 3.1 Audience

148 A reader of the document MUST have a deep understanding of UML 2.1.2. She or he MUST be able to  
149 understand meta models denoted as UML class diagrams. She or he SHOULD be familiar with the UML 2.1.2.  
150 meta model, at least she or he MUST be able to check back the UML 2.1.2. meta model. The reader SHOULD  
151 be familiar with OCL 2.0 in order to understand the OCL constraints of this UMM profile – those who are not  
152 familiar with OCL are provided with a plain text description of the constraint.

153 The information described in this manual is aimed at

- 154 • advanced business process modelers who check a UML model for UMM compliance (if not
- 155 supported by a tool)
- 156 • advanced business process modelers who train other business process modelers and business
- 157 process analysts
- 158 • software designers who want to produce UML tools providing support for this UMM foundation
- 159 module
- 160 • software designers who want to produce tools to transform UMM compliant business collaboration
- 161 models into specifications within an IT-layer (ebXML, Web Services, UN/EDIFACT, etc.).
- 162 • software designers who want to produce repositories to register UMM compliant business
- 163 collaboration models

## 164 3.2 Related Documents

- 165 • **UN/CEFACT**
  - 166 ○ UN/CEFACT Open Development Process (TRADE/R.650/Rev.4/Add.1/Rev.1 - 19 April 2007)
  - 167 [http://www.unece.org/cefact/cf\\_plenary/plenary07/trd\\_R650\\_Rev4\\_A1E.pdf](http://www.unece.org/cefact/cf_plenary/plenary07/trd_R650_Rev4_A1E.pdf)
  - 168 ○ UPCC: UML Profile for Core Components
  - 169 <http://unstandards.org:8080/display/public/UPCC++UML+Profile+for+Core+Components>
  - 170 ○ Core Component Technical Specification
  - 171 [http://www.unece.org/cefact/ebxml/CCTS\\_V2-01\\_Final.pdf](http://www.unece.org/cefact/ebxml/CCTS_V2-01_Final.pdf)
- 172 • **International Organization for Standardization (ISO)**
  - 173 ○ Open-edi Reference Model. ISO/IEC 14662
  - 174 [http://standards.iso.org/ittf/PubliclyAvailableStandards/c037354\\_ISO\\_IEC\\_14662\\_2004\(E\).zip](http://standards.iso.org/ittf/PubliclyAvailableStandards/c037354_ISO_IEC_14662_2004(E).zip)
- 175 • **Object Management Group (OMG)**
  - 176 ○ Unified Modeling Language Specification (UML), Version 2.1.2
  - 177 <http://www.omg.org/docs/formal/07-02-05.pdf>

## 178 3.3 UN/CEFACT's Modeling Methodology (UMM): Overview

179 UN/CEFACT's Modeling Methodology (UMM) is a UML modeling approach to design the business services  
 180 that each partner must provide in order to collaborate. It provides the business justification for the services  
 181 to be implemented in a service-oriented collaboration architecture. Thus, a primary vision of UN/CEFACT is  
 182 to capture the business knowledge that enables the development of low cost software based on service-  
 183 oriented architectures (SOA) helping the small and medium size companies (SMEs), and emerging economies  
 184 to engage in e-Business practices. UMM focuses on developing a global choreography of inter-organizational  
 185 business processes and their information exchanges. UMM models are notated in UML syntax and are  
 186 platform independent models. The platform independent UMM models identify which services need to be  
 187 realized in a service-oriented architecture, implementing the business collaboration. This approach provides  
 188 insurance against technical obsolescence.

189 The UMM, as described in this document, is the formal description technique for describing any Open-edi  
 190 scenario as defined in ISO/IEC 14662 "Open-edi reference model". An Open-edi scenario is a formal means  
 191 to specify a class of business transactions having the same business goal, such as, purchasing or inventory  
 192 management. The primary scope of UMM is the Business Operations View (BOV) and not the Functional  
 193 Service View (FSV) as defined in ISO/IEC IS 14662. The BOV is defined as "a perspective of business  
 194 transactions limited to those aspects regarding the making of business decisions and commitments among  
 195 organizations", while the FSV is focused on implementation specific, technological aspects of Open-edi. The  
 196 commitments of the BOV layer are reflected in the choreography of the inter-organizational business  
 197 processes and their information exchanges. At the FSV layer, this choreography must be implemented by a  
 198 set of composite services. Therefore it follows, that UMM, which targets the BOV layer, defines what the

199 business is about; and the technologies on the FSV layer define how to implement the business by a service-  
200 oriented architecture.

201 This version of the UMM consists of three views each covering a set of well defined artifacts:

- 202 • Business Requirements View (bRequirementsV)
- 203 • Business Choreography View (bChoreographyV)
- 204 • Business Information View (bInformationV)

205

206 **Business Requirements View (bRequirementsV):** The Business Requirements View is used to gather existing  
207 knowledge. It identifies the business processes in the domain and the business problems that are important  
208 to stakeholders. It is important at this stage that business processes are not constructed, but discovered.  
209 Stakeholders might describe intra-organizational as well as inter-organizational business processes. All of this  
210 takes place in the language of the business experts and stakeholders. The business requirements view results  
211 in a categorization of the business domain (manifested as a hierarchical structure of packages) and a set of  
212 relevant business processes (manifested as use cases). The result may be depicted in use case diagrams. In  
213 order to model the dynamics of each business process, one may use a Business Process Activity Model, or a  
214 Sequence Diagram, which would be placed beneath the Business Process Use Case. As a practical note, the  
215 Business Process Activity Model may depict a process or processes which involve one or more Business  
216 Partners. A Sequence Diagram will depict information exchanges between two or more Business Partners.  
217 The Business Partners are described within their own package (Business Partner View). A Business Process  
218 Activity Model may show state changes to Business Entities. Business Entities are “real-word” things having  
219 business significance and are shared among the business partners involved in the collaboration. The Business  
220 Entities and their lifecycles of state changes are modeled in the Business Entity View. Furthermore, the  
221 Business Entity View also contains one or more packages which represent the conceptual data structures of  
222 the Business Entities.

223 **Business Choreography View (bChoreographyV):** The Business Choreography View is used to define and  
224 document the global choreography between collaborating business partners in an inter-organizational  
225 business process. Within the Business Choreography View, the Business Transaction View contains and  
226 documents the requirements of Business Transaction Use Cases, and their participating Authorized Roles.  
227 The dynamics of a Business Transaction Use Case are described by a Business Transaction. A business  
228 transaction defines a simple choreography of exchanging business information between two authorized  
229 roles and an optional response. A business transaction identifies the business actions of each partner  
230 responsible for sending and receiving the business information. These actions correspond to the  
231 requirements of any solution that must be implemented on each business partner’s side in a service-  
232 oriented collaboration architecture. Within the Business Choreography View, the Business Collaboration  
233 View contains and documents the requirements of Business Collaboration Use Cases and their participating  
234 Authorized Roles. The dynamics of a Business Collaboration Use Case are described by a Business  
235 Collaboration Protocol. A Business Collaboration Protocol choreographs the flow among business  
236 transactions, and/or nested Business Collaboration Protocols. This flow depends on the states of business  
237 entities. When a Business Collaboration Use Case is identified, but different sets of parties may execute this  
238 collaboration, the different Realizations (executions) may be modeled within the Business Realization View,  
239 as a Business Realization Use Cases.



240 **Business Information View (bInformationV):** An execution of a business transaction usually results in the  
241 change of state of one or more business entities. Thus, the information exchanged in a transaction should be  
242 limited to the minimum information needed to change the state of a business entity. Nevertheless, UMM  
243 allows the definition of an information exchange in a document-centric approach – even if this is not  
244 recommended. A Business Information View contains Business Information Artifacts. UMM does not  
245 mandate a specific Business Information Modeling approach. However, UMM strongly recommends that  
246 Business Information is modeled in accordance to UN/CEFACT’s Core Components Technical Specification  
247 and Message Assembly Guidelines. In order to model Core Components by means of UML, UN/CEFACT  
248 provides the Profile for Core Components (UPCC).

## 249 **3.4 Objectives**

### 250 **3.4.1 Goals of the Technical Specification**

251 The goals of this specification are:

- 252 • To define the semantics of well-formed UMM business collaboration models, which describe a public  
253 choreography of an inter-organizational system. Local choreographies and private processes of a  
254 businesses partner are out of scope.
- 255 • To define the validation rules for UMM compliant business collaboration models.
- 256 • To clarify the basic concepts that a UMM-compliant business collaboration model is based on.
- 257 • To provide an unambiguous definition for UMM business collaboration models that allows an  
258 unambiguous mapping to artifacts for deployment in a service-oriented architecture. Note, that the  
259 mapping itself is not part of UMM.
- 260 • To define a UML profile for the UMM foundation module that allows UML tool vendors to customize  
261 their tools to be UMM compliant. Better UML 2.1.2. tool support will lead to a growing UMM user  
262 base.

### 263 **3.4.2 Requirements**

264 This specification is guided by the following key requirements derived from the above goals:

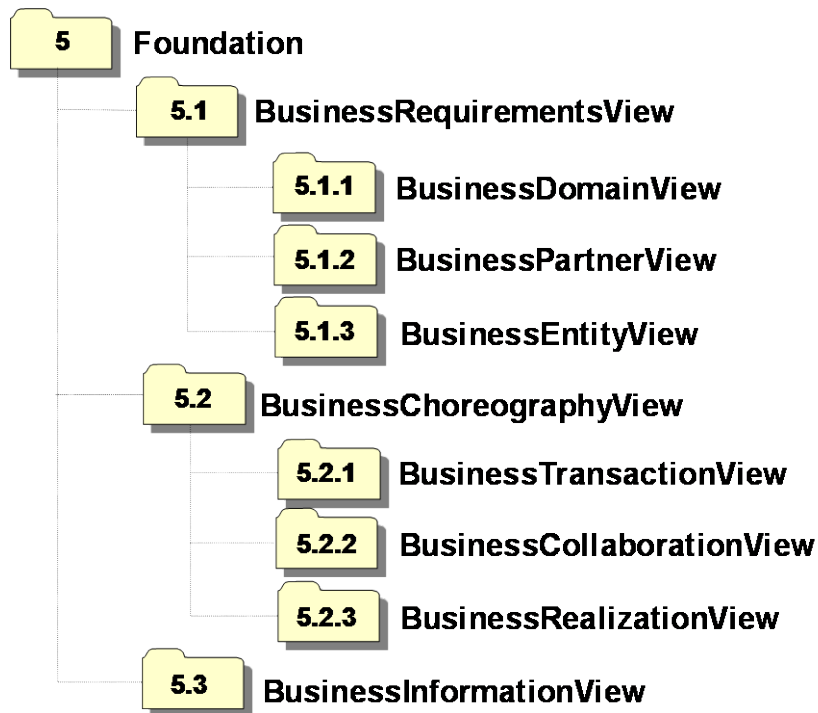
- 265 • The UMM foundation module defines only those modeling concepts that are considered as  
266 fundamental to deliver a UMM compliant model. Additional advanced modeling concepts shall be  
267 covered in specialization and extension modules.
- 268 • The UMM foundation module is directed towards the Business Operational View of Open-edi. This  
269 means it is independent of certain implementation technologies used in SOAs like Web Services and  
270 ebXML or other future technologies. However, the UMM compliant business collaboration models  
271 must be defined in a way that allows a mapping to an implementation technology of choice. Such a  
272 mapping is not part of the UMM foundation module. It is a candidate for a specialization/extension  
273 module.
- 274 • Today, the UML is the most commonly supported modeling language by modeling tools. In order to  
275 use the broad range of tools, a UMM business collaboration model must be a special kind of UML  
276 model. Thus, the UMM foundation module is based on the UML meta model. In fact, it provides a  
277 UML Profile consisting of stereotypes, tagged definitions and constraints.
- 278 • In order to support a broad adoption of the UMM-modeling approach, the formal descriptions of the  
279 UMM is supplemented by a set of examples that show UMM compliant artifacts.

### 280 **3.4.3 Caveats and Assumptions**

281 This specification makes the following assumptions:

- 282 • This UML profile is based on the UML meta-model version 2.1.2. This version is the current OMG  
283 version. Using another UML meta-model as a basis for the development of a UMM compliant  
284 business collaboration model may not deliver correct results.
- 285 • The basic concepts of the UMM and the way they relate to each other are described and explained  
286 by means of a meta model (to be found in the non-normative “conceptual description” sections of  
287 this document).
- 288 • Different specialization and extension modules might extend the foundation module in order to  
289 define additional semantics to the minimum semantics required to create a UMM compliant  
290 business collaboration models.

### 291 3.5 Structure of the UMM Foundation Module



292  
293 **Figure 2 Package overview of UMM Foundation Module meta model**

294 Section 5 defines the UML profile of the foundation module of the UMM meta model (Section 1, Figure 1).  
295 The figure above (Figure 2), shows the package structure of the foundation module of the UMM meta  
296 model. The numbers referring to the subsections are included in figure 2. Those numbers refer to the  
297 subsections containing the stereotypes, tag definitions and constraints of the corresponding package. The  
298 first level packages of the foundation module conform to the three views of the current UMM version:  
299 Business Requirements View (5.1), Business Choreography View (5.2), and Business Information View (5.3).

300 The Business Requirements View (5.1) comprises the Business Domain View (5.1.1), the Business Partner  
301 View (5.1.2), and the Business Entity View (5.1.3). The second top-level package, the Business Choreography  
302 View (5.2) consists of the Business Transaction View (5.2.1), the Business Collaboration View (5.2.2), and the  
303 Collaboration Realization View (5.2.3). The third top-level package is the Business Information View (5.3). It  
304 does not contain any sub packages.

305 Each section describing a package is structured in the same way. The first subsection is informative. It  
306 describes the conceptual model of the artefact that is addressed by the package. The second subsection is  
307 normative and defines all the stereotypes and associated tag definitions that are defined in the package. The  
308 third subsection is normative and includes all the constraints in plain text that apply to the respective

309 package. The constraints are also expressed using the Object Constraint Language (OCL) in section 6. The  
310 two remaining informative subsections cover on the one hand side worksheets used to gather information  
311 from business people in order to create the UMM models and on the other hand side examples to depict  
312 instances of the artefact type addressed by the package.

313 The

## 314 4 Dependencies on other UMM Modules (normative)

### 315 4.1 Abbreviations of Stereotypes

Stereotype Abbreviation	Full Stereotype Name
bInformation	BusinessInformation
bLibrary	BusinessLibrary
InfEnvelope	InformationEnvelope

### 316 4.2 Dependency between Base Module and Foundation Module

317

#### 318 **Figure 3 UMM Foundation Dependencies**

319 The UMM foundation module 2.0 is built on top of the UMM base module 2.0. This means that all  
320 stereotypes and tag definitions defined in the UMM base module 2.0 are imported into the UMM  
321 foundation module 2.0. The figure below shows the stereotypes defined in the UMM base module also used  
322 in the foundation module. Note that the stereotypes of the base module are identified with notes in all  
323 figures of this specification. The formal definition of the stereotypes *bInformation*, *InfEnvelope* and *bLibrary*  
324 is given in the UMM base module 2.0 specification. In the foundation module, packages - that are containers  
325 of stereotypes realizing main UMM artefacts - are defined as specializations of the base stereotype *bLibrary*.  
326 This means that such packages and their contents are candidates for registration in a registry. In the UMM  
327 foundation module 2.0 we do not define any stereotype that directly inherits from *bLibrary*. As a  
328 consequence, only packages are candidates for registration.

329 The concepts of *bInformation* and *InfEnvelope* are used to define the business document information being  
330 exchanged between authorized roles in a UMM business transaction.

331  
332  
333

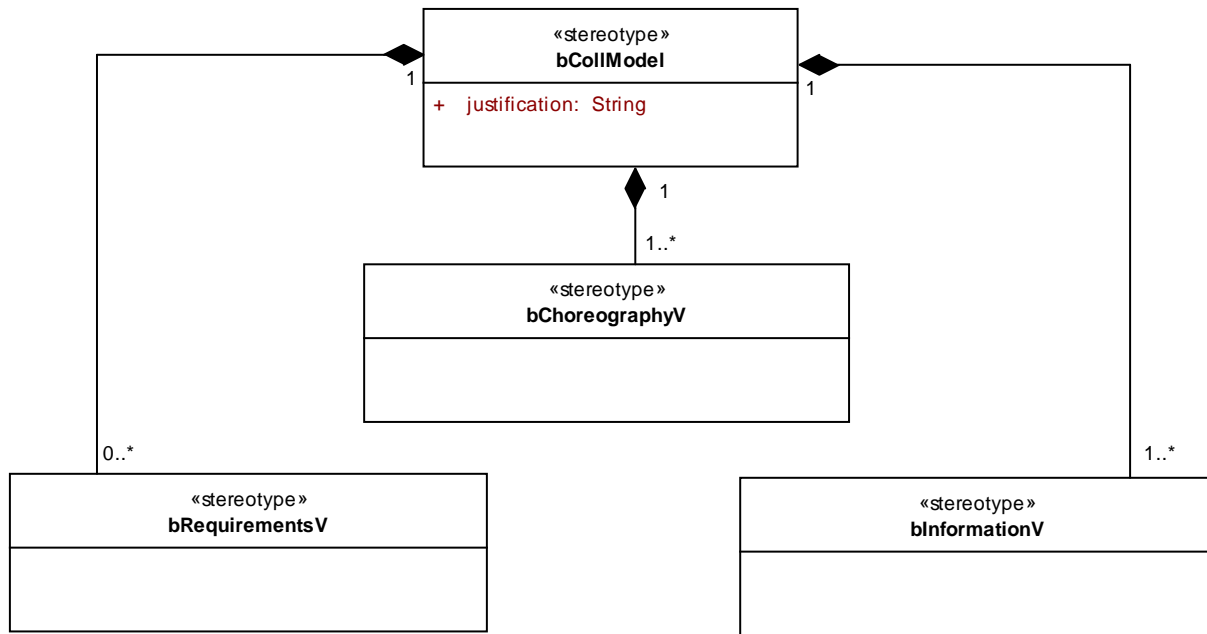
Figure 4 UMM Base Abstract Syntax

334 **5 UMM Foundation Module**  
335 **5.0 Foundation Module Management**  
336 **5.0.1 Abbreviations of Stereotypes**

Stereotype Abbreviation	Full Stereotype Name
bCollModel	BusinessCollaborationModel
bRequirementsV	BusinessRequirementsView
bChoreographyV	BusinessChoreographyView
bInformationV	BusinessInformationView
bLibrary	BusinessLibrary

337

338 5.0.2 Conceptual Description (informative)



339

340

**Figure 5 UMM Foundation Module Management - Conceptual Overview**

341

A project that follows the UMM approach leads to a business collaboration model. A business collaboration model that is UMM compliant is stereotyped as *bCollModel*. As described above, the UMM is built by three views. The business requirements view stereotyped as *bRequirementsV* is optional and may occur multiple times in a business collaboration model. The business choreography view (stereotyped as *bChoreographyV*) and the business information view (stereotyped as *bInformationV*) are mandatory parts of a business collaboration model and may also occur multiple times.

347

Within the business requirements view the specific requirements of the business collaboration between two or more business partners are captured. The collected information from the business collaboration is then further elaborated within the business choreography view. The information exchanged during the process is modeled in the business information view. For further information on the specific sub-views of the UMM, please see the relevant sub-chapters of this specification.

352

353 **5.0.3 Stereotypes and Tag Definitions (normative)**

from Base Module

354  
355

356

**Figure 6 UMM Foundation Module Management - Abstract Syntax**

<b>Stereotype</b>		<b>bCollModel (BusinessCollaborationModel)</b>	
<b>Base Class</b>	Package		
<b>Parent</b>	BusinessLibrary (from Base Module)		
<b>Description</b>	<p>A business collaboration model is a model that is compliant to the UMM meta model. It <b>MUST</b> be compliant to the base and foundation module, and it <b>MAY</b> be compliant to one or more specialization and/or extension modules.</p> <p>Since a business collaboration model is of base class package, a UML model <b>MAY</b> contain one to many business collaboration models. Therefore, either the root element of a UML model is stereotyped as business collaboration model or any of the packages beneath the root element.</p>		
<b>Tag Definition</b>	<b>justification</b>		
	<b>Type</b>	String	
	<b>Multiplicity</b>	1	
	<b>Description</b>	Explains the reason from a business perspective why the given business case is considered for possible business collaborations.	

357

	<p><b>Inherited tagged values:</b></p> <ul style="list-style-type: none"> <li>– businessTerm</li> <li>– copyright</li> <li>– owner</li> <li>– reference</li> <li>– status</li> <li>– uniqueIdentifier</li> <li>– versionIdentifier</li> </ul>
--	---

358

Stereotype	bRequirementsV (BusinessRequirementsView)
<b>Base Class</b>	Package
<b>Parent</b>	BusinessLibrary (from Base Module)
<b>Description</b>	<p>The business requirements view is a container for all elements needed to identify and describe the requirements of collaborations between business partners.</p> <p>It captures the relevant packages which are used for discovering relevant business processes and their business partners executing/participating in them, as well as the lifecycle and state changes of business entities which are important within a business process.</p>
<b>Tag Definition</b>	<p><b>Inherited tagged values:</b></p> <ul style="list-style-type: none"> <li>– businessTerm</li> <li>– copyright</li> <li>– owner</li> <li>– reference</li> <li>– status</li> <li>– uniqueIdentifier</li> <li>– versionIdentifier</li> </ul>

359

Stereotype	bChoreographyV (BusinessChoreographyView)
<b>Base Class</b>	Package
<b>Parent</b>	BusinessLibrary (from Base Module)
<b>Description</b>	<p>The business choreography view is a container for all elements needed to describe the choreography of business collaborations at various levels.</p>
<b>Tag Definition</b>	<p><b>Inherited tagged values:</b></p> <ul style="list-style-type: none"> <li>– businessTerm</li> <li>– copyright</li> <li>– owner</li> <li>– reference</li> <li>– status</li> <li>– uniqueIdentifier</li> <li>– versionIdentifier</li> </ul>

Stereotype	bInformationV (BusinessInformationView)
------------	---

<b>Base Class</b>	Package
<b>Parent</b>	BusinessLibrary (from Base Module)
<b>Description</b>	The business information view is a container for all elements representing the exchanged information in business collaborations.
<b>Tag Definition</b>	<b>Inherited tagged values:</b> <ul style="list-style-type: none"> <li>– businessTerm</li> <li>– copyright</li> <li>– owner</li> <li>– reference</li> <li>– status</li> <li>– uniqueIdentifier</li> <li>– versionIdentifier</li> </ul>

360 **5.0.4 Constraints (normative)**

361

362 Constraints with respect to the *BusinessCollaborationModel* (bCollModel)

363 C.1. A BusinessCollaborationModel MUST contain one to many BusinessChoreographyViews

364 C.2. A BusinessCollaborationModel MUST contain one to many BusinessInformationViews

365 C.3. A BusinessCollaborationModel MAY contain zero to many BusinessRequirementsViews

366 C.4. A BusinessRequirementsView, a BusinessChoreographyView and a BusinessInformationView MUST  
367 be directly located under a BusinessCollaborationModel

368 **5.1 Business Requirements View**

369 **5.1.0 Sub-Views in the Business Requirements View**

370 **5.1.0.1 Abbreviations and Stereotypes**

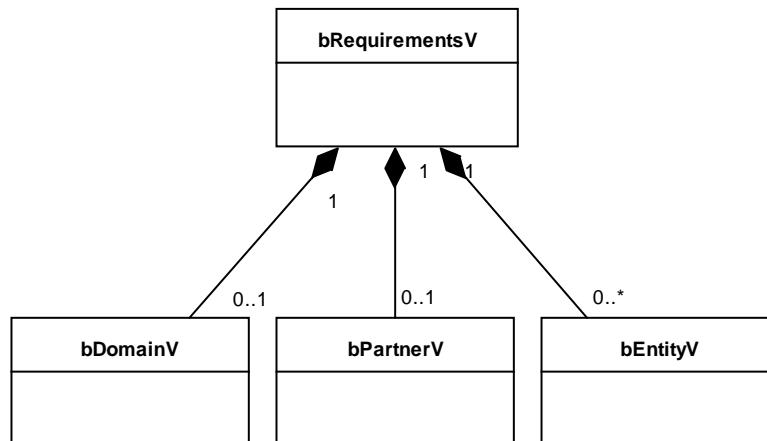
371

Stereotype Abbreviation	Full Stereotype Name
bRequirementsV	BusinessRequirementsView
bDomainV	BusinessDomainView
bPartnerV	BusinessPartnerView
bEntityV	BusinessEntityView

372



373 **5.1.0.2 Conceptual Description (informative)**



374

375 **Figure 7 BusinessRequirementsView Conceptual Overview**

376 The *BusinessRequirementsView* is composed by three significant sub-views which are all of optional use.  
377 Firstly, the *BusinessDomainView* captures all of the business processes which may be of interest for the  
378 domain under consideration. In order to enable users to readily identify business processes, these business  
379 processes are classified into business categories. This classification is done by creating business areas and  
380 process areas. However, UN/CEFACT recommends using this classification, but it is not mandatory. Secondly,  
381 the *BusinessPartnerView* specifies a list of business partners and stakeholders that are involved in the  
382 business processes defined in the *BusinessDomainView*. Furthermore the relationships between each others  
383 are defined in this package. Finally, the *BusinessEntityView* defines the business entities that are involved in  
384 a business process. A business entity is a real-world thing having business significance that is shared among  
385 two or more business partner in a collaborative business process (e.g. order, account, etc.). It is important to  
386 depict the possible state changes of such business entities within the *BusinessEntityView* in order to get an  
387 understanding of how a collaborative business process affects such real-world things during the execution of  
388 a business process.

from Base Module

390  
391 **Figure 8 BusinessRequirementsView Abstract Syntax**

<b>Stereotype bDomainV (BusinessDomainView)</b>	
<b>Base Class</b>	Package
<b>Parent</b>	BusinessLibrary (from Base Module)
<b>Description</b>	The business domain view is used to discover business processes that are of relevance in a project. A business domain is a framework for identification and understanding of business processes as well as for categorizing them according to a classification schema. The business domain view is a container capturing the categorization scheme and categorized business processes.
<b>Tag Definition</b>	<p><b>Inherited tagged values:</b></p> <ul style="list-style-type: none"> <li>– owner</li> <li>– copyright</li> <li>– reference</li> <li>– status</li> <li>– businessTerm</li> <li>– uniqueIdentifier</li> <li>– versionIdentifier</li> </ul>

392

<b>Stereotype bPartnerV (BusinessPartnerView)</b>	
<b>Base Class</b>	Package

<b>Parent</b>	BusinessLibrary (from Base Module)
<b>Description</b>	The business partner view captures a list of business partners and stakeholders in the domain under consideration as well as the relationships between them.
<b>Tag Definition</b>	<b>Inherited tagged values:</b> <ul style="list-style-type: none"> <li>– uniqueIdentifier</li> <li>– owner</li> <li>– copyright</li> <li>– reference</li> <li>– versionIdentifier</li> <li>– status</li> <li>– businessTerm</li> </ul>

393

<b>Stereotype bEntityV (BusinessEntityView)</b>	
<b>Base Class</b>	Package
<b>Parent</b>	BusinessLibrary (from Base Module)
<b>Description</b>	The business entity view is a container to describe the lifecycle of a business entity having business significance in the modelled domain including its' business entity states.
<b>Tag Definition</b>	<b>Inherited tagged values:</b> <ul style="list-style-type: none"> <li>– uniqueIdentifier</li> <li>– versionIdentifier</li> <li>– owner</li> <li>– copyright</li> <li>– reference</li> <li>– status</li> <li>– businessTerm</li> </ul>

394

395

#### 396 **5.1.0.4 Constraints (normative)**

397 Constraints with respect to a *BusinessRequirementsView*:

398 C.5. A *BusinessRequirementsView* MAY contain zero or one *BusinessDomainView*.

399 C.6. A *BusinessRequirementsView* MAY contain zero or one *BusinessPartnerView*.

400 C.7. A *BusinessRequirementsView* MAY contain zero to many *BusinessEntityViews*.

401 C.8. A *BusinessDomainView*, a *BusinessPartnerView*, and a *BusinessEntityView* MUST be located  
402 directly under a *BusinessRequirementsView*.

403

### 404 **5.1.1 Business Domain View**

#### 405 **5.1.1.1 Abbreviations of Stereotypes**

Stereotype Abbreviation	Full Stereotype Name
bDomainV	BusinessDomainView
bCategory	BusinessCategory
bArea	BusinessArea



410 The business domain view is used to discover business processes use cases that are of relevance in a project.  
411 A business process use case is executed by at least one (but possibly more) business partners. A business  
412 partner might execute multiple business process use cases. Thus, the *participates* association between  
413 *BusinessPartner* and *BusinessProcessUseCase* is a (1..n) to (0..n) association. A stakeholder does not need to  
414 participate in a business process use case. A stakeholder might have interest in multiple business process use  
415 cases and a business process use case might be of interest to multiple stakeholders. The relationship  
416 between a *BusinessProcessUseCase* and a *Stakeholder* is described by the *isOfInterestTo* dependency in  
417 UMM. A business process can be decomposed into sub-processes using the «include» and «extends»  
418 association stereotypes.

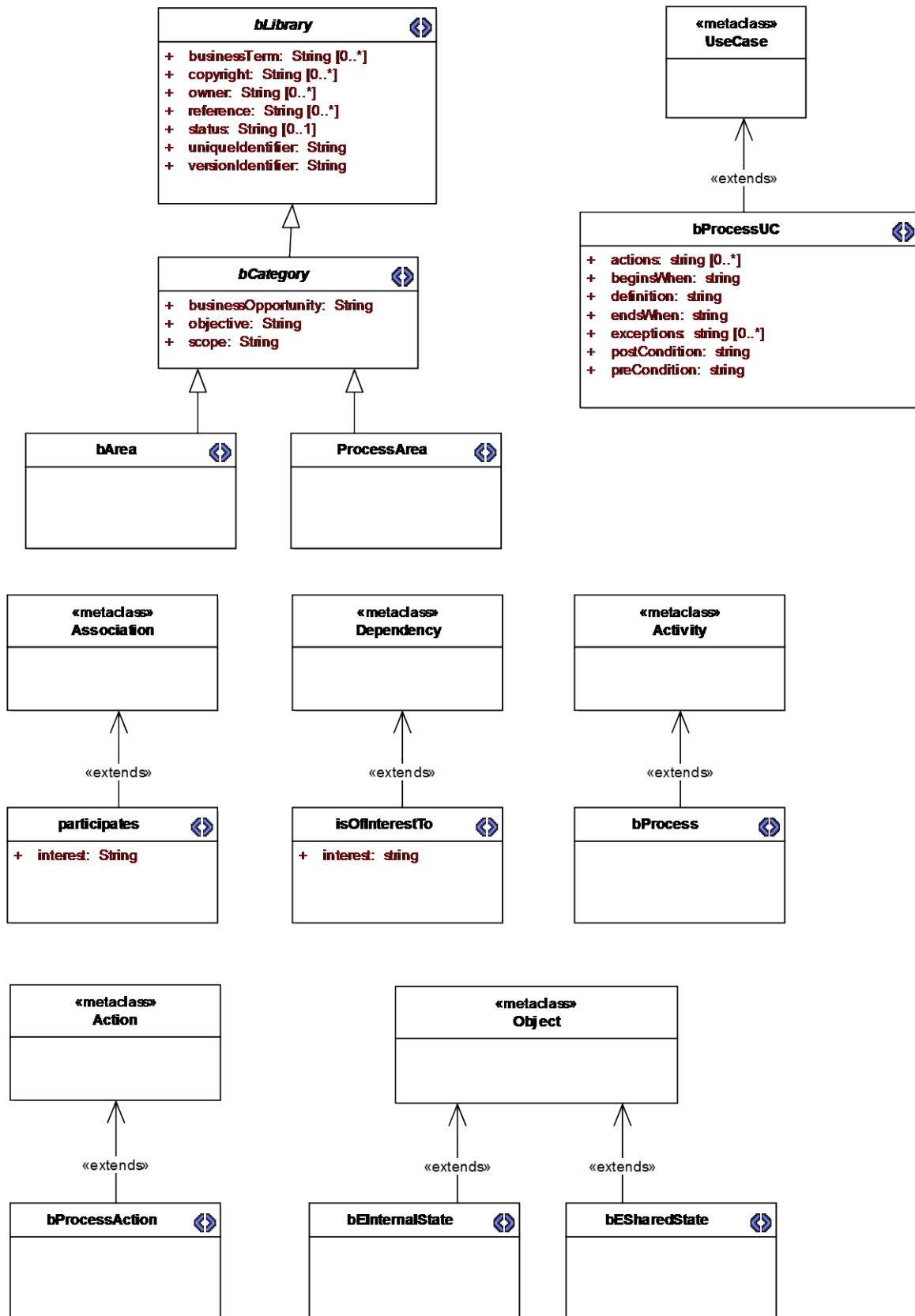
419 To enable users to readily identify business process use cases, they should be classified into business  
420 categories. A business category is an abstract concept, which has two concrete specializations – business  
421 area and process area. A business area corresponds to a division of an organization and a process area  
422 corresponds to a set of common operations within the business area. A business area might be composed of  
423 other business areas. This means, a business area may form a hierarchy. Thus, a unary (0..1) to (0..n)  
424 composition is defined for a *BusinessArea*. The lowest level of a business area hierarchy includes process  
425 areas or business processes use cases. Therefore, we have a (0..1) to (0..n) composition between  
426 *BusinessArea* and *ProcessArea*. Furthermore, a *BusinessArea* may also include 0..n *BusinessProcessUseCases*  
427 if no further classification using process areas is required. Similar to a business area, a process area may form  
428 a hierarchy. This means, a unary (0..1) to (0..n) composition is defined for a *ProcessArea*. Similar to a  
429 business area, a *ProcessArea* may contain zero to many *BusinessProcessUseCases*. On the lowest level of a  
430 *ProcessArea* hierarchy, at least one *BusinessProcessUseCase* must be present.

431 The flow of a business process use case may be described by business processes. Thus, a  
432 *BusinessProcessUseCase* is realized by zero to many *BusinessProcesses*. A business process represents the  
433 dynamic behavior of a business process use case. A business process corresponds to a flow of actions  
434 performed by one participant or even by more participants. If two or more business partners collaborate, a  
435 business process is divided into partitions – one for each business partner. In case of an internal business  
436 process, which is executed by one partner only, a single partition for that partner is optional. Consequently,  
437 a *BusinessProcess* is composed of zero or more UML *ActivityPartitions*. An *ActivityPartition* is assigned to one  
438 *BusinessPartner*; a *BusinessPartner* is assigned to one *ActivityPartition*. However, a *BusinessPartner* may be  
439 assigned to multiple *ActivityPartitions* – each one in a different *BusinessProcess*. Hence, there is a 1 to (0..n)  
440 association between *BusinessPartner* and *ActivityPartition*.

441 A business process is described as a flow of business process actions. In the case where no activity partition  
442 is used, the business process actions are directly included in the Activity Diagram of the business process. In  
443 case of activity partitions, a business process action is assigned to the partition of the business partner  
444 executing the action. The need for a collaborative business process is identified whenever a transition  
445 connecting two business process actions crosses activity partitions. It follows, that either a *BusinessProcess* is  
446 composed of one or more *BusinessProcessAction* or an *ActivityPartition* (which is part of a business process)  
447 is composed of one or more *BusinessProcessActions*. A business process action might be refined by another  
448 business process. Thus a *BusinessProcessAction* is composed of zero or one *BusinessProcess* which in turn is a  
449 composite of zero or one *BusinessProcessActions*.

450 A business process may also denote important states of business entities that are manipulated during the  
451 execution of a business process. A business entity state is the output from one business action and input to

452 another business action. There is a transition from a business process action to a business entity state  
453 signaling an output as well as a transition from a business entity state to a business process action signaling  
454 an input. Some business entity states are meaningful to one business partner only. These are internal  
455 business entity states. Business entity states that must be communicated to a business partner are shared  
456 business entity states. A business process may include both internal and shared business entity states.  
457 Hence, a *BusinessProcess* is composed of zero to many *InternalBusinessEntityStates* and of zero to many  
458 *SharedBusinessEntityStates*. If a business process uses activity partitions, the two business process actions  
459 creating and consuming an internal business entity state are in the same activity partition. In contrast, the  
460 two business process actions creating and consuming a shared business entity state are in different activity  
461 partitions. A shared business entity state signals the need for a collaborative business process.



463

464

Figure 10 BusinessDomainView Abstract Syntax

<b>Stereotype</b>		<b>bCategory (BusinessCategory, abstract)</b>	
<b>Base Class</b>	Package		
<b>Parent</b>	BusinessLibraryPackage (from Base Module)		
<b>Description</b>	<p>A business category is an abstract concept. Business categories are used to classify the business processes in the Business Domain View. The prime purpose of classifying the business processes is to enable potential users to readily identify processes that have been defined in the business category under consideration.</p> <p>Consequently a business category is used to group either other business categories or business processes that belong to the respective business category. The Business Domain View is structured by its specializations <i>BusinessArea</i> and <i>ProcessArea</i> (see below for these stereotype definitions).</p>		
<b>Tag Definition</b>	<b>objective</b>		
	<b>Type</b>	String	
	<b>Multiplicity</b>	1	
	<b>Description</b>	The purpose to be achieved by the business process within the business category under consideration.	
	<b>scope</b>		
	<b>Type</b>	String	
	<b>Multiplicity</b>	1	
	<b>Description</b>	Defines the boundaries of the business category under consideration.	
	<b>businessOpportunity</b>		
	<b>Type</b>	String	
	<b>Multiplicity</b>	1	
	<b>Description</b>	The strategic interest from a business perspective in order to address the business category under consideration.	
	<p><b>Inherited tagged values:</b></p> <ul style="list-style-type: none"> <li>– uniqueIdentifier</li> <li>– versionIdentifier</li> <li>– owner</li> <li>– copyright</li> <li>– reference</li> <li>– status</li> <li>– businessTerm</li> </ul>		

<b>Stereotype</b>		<b>bArea (BusinessArea)</b>	
<b>Base Class</b>	Package		
<b>Parent</b>	BusinessCategory		



<b>Description</b>	<p>A business area usually corresponds to a division of an enterprise. Business areas might be structured recursively. A business area is a category of decomposable business areas or process areas (on the lowest level of business area hierarchy). This means that a business area collates either other business areas, process areas or business process use case.</p> <p>The UMM does not mandate a specific classification schema. A classification schema that might be used is the Porter Value Chain. Based on the Porter Value Chain, the UN/CEFACT Common Business Process Catalog recommends a list of eight flat (i.e. non-recursive) categories: Procurement/Sales, Design, Manufacture, Logistics, Recruitment/Training, Financial Services, Regulation, and Health Care. This list of business areas is considered as non exhaustive.</p>
<b>Tag Definition</b>	<p><b>Inherited tagged values:</b></p> <ul style="list-style-type: none"> <li>– uniqueIdentifier</li> <li>– versionIdentifier</li> <li>– objective</li> <li>– scope</li> <li>– businessOpportunity</li> <li>– owner</li> <li>– copyright</li> <li>– reference</li> <li>– status</li> <li>– businessTerm</li> </ul>

467

<b>Stereotype</b>	<b>ProcessArea (ProcessArea)</b>
<b>Base Class</b>	Package
<b>Parent</b>	BusinessCategory
<b>Description</b>	<p>A process area corresponds to a set of common operations within a business area. Process areas might be structured recursively. A process area is a category of common business process use cases. This means a process area collates either other process areas or business process use cases.</p> <p>The UMM does not mandate a specific classification schema. The UN/CEFACT Common Business Process Catalog recommends a list of five flat (i.e. non-recursive) categories that correspond to the five successive phases of business collaborations as defined by the ISO Open-edi model: Planning, Identification, Negotiation, Actualization, Post-Actualization.</p>
<b>Tag Definition</b>	<p><b>Inherited tagged values:</b></p> <ul style="list-style-type: none"> <li>– uniqueIdentifier</li> <li>– versionIdentifier</li> <li>– objective</li> <li>– scope</li> <li>– businessOpportunity</li> <li>– owner</li> <li>– copyright</li> <li>– reference</li> <li>– status</li> <li>– businessTerm</li> </ul>

468

<b>Stereotype</b>	<b>bProcessUC (BusinessProcessUseCase)</b>
<b>Base Class</b>	UseCase

<b>Parent</b>	N/A
<b>Description</b>	A business process use case is a set of related activities that together create value for a business partner. A business process use case might be performed by a single business partner type or by multiple business partner types crossing organizational boundaries. In the case where organizations collaborate in a business process, the business process should create value for all of its participants. A business process use case can be decomposed into sub-processes using the «include» and «extends» association stereotypes defined in UML.
<b>Tag Definition</b>	<b>definition</b>
	<b>Type</b>   String
	<b>Multiplicity</b>   1
	<b>Description</b>   Gives a definition of the business process use case. This definition must describe the customer value to be created by the business process use case. In the case of a business process use case executed by multiple parties, it describes the value to be created to all participants.
	<b>beginsWhen</b>
	<b>Type</b>   String
	<b>Multiplicity</b>   1
	<b>Description</b>   Specifies a business event that triggers the initiation of the business process use case.
	<b>preCondition</b>
	<b>Type</b>   String
	<b>Multiplicity</b>   1
	<b>Description</b>   Specifies a condition that has to be fulfilled in order to execute a business process use case. This condition SHOULD refer to states in the life cycle of a business entity. A pre-condition statement MAY use Boolean operators specifying a combination of multiple business entity states.
	<b>endsWhen</b>
	<b>Type</b>   String
	<b>Multiplicity</b>   1
	<b>Description</b>   Specifies a business event that leads to the termination of the business process use case.
	<b>postCondition</b>
	<b>Type</b>   String
	<b>Multiplicity</b>   1
	<b>Description</b>   Specifies a condition that will be reached after executing the business process use case. Usually, this condition SHOULD refer to states in the life cycle of a business entity. A post-condition statement MAY use Boolean operators specifying a combination of multiple business entity states.

469

	<b>exceptions</b>	
	<b>Type</b>	String
	<b>Multiplicity</b>	0..*
	<b>Description</b>	Identifies situations leading to a deviation of the regular execution of the business process use case.
	<b>actions</b>	
	<b>Type</b>	String
<b>Multiplicity</b>	0..*	
<b>Description</b>	Lists the tasks that together make up a business process use case. In the case of a business process use case executed by multiple parties, a special emphasis on interface tasks is needed. An interface task is a step in the business process use case that requires communication with another business partner.	

470

<b>Stereotype</b>	<b>participates (participates)</b>	
<b>Base Class</b>	Association	
<b>Parent</b>	N/A	
<b>Description</b>	Describes the association between a business partner and a business process use case. This stereotype defines that the business partner provides input to and/or output from the associated business process use case.	
<b>Tag Definition</b>	<b>interest</b>	
	<b>Type</b>	String
	<b>Multiplicity</b>	1
	<b>Description</b>	Describes the vested interest of the business partner type in the business process associated by this participates-association.

471

<b>Stereotype</b>	<b>isOfInterestTo (isOfInterestTo)</b>	
<b>Base Class</b>	Dependency	
<b>Parent</b>	N/A	
<b>Description</b>	Describes a dependency from a business process use case to a stakeholder. This stereotype defines that a business process use case depends on the interest of the connected stakeholder.	
<b>Tag Definition</b>	<b>interest</b>	
	<b>Type</b>	String
	<b>Multiplicity</b>	1
	<b>Description</b>	Describes the vested interest of the stakeholder in the business process use case linked by this is of interest to dependency.

472

<b>Stereotype</b> <b>bProcess (BusinessProcess)</b>	
<b>Base Class</b>	Activity
<b>Parent</b>	N/A
<b>Description</b>	The business process describes the behavior of a business process use case between the involved business partners. It is a tool to identify requirements to collaborate between two or more business partners. A business process refines a business process use case by describing its dynamic behaviour.
<b>Tag Definition</b>	No tagged values.

473

<b>Stereotype</b> <b>bProcessAction (BusinessProcessAction)</b>	
<b>Base Class</b>	Action
<b>Parent</b>	N/A
<b>Description</b>	A business process action corresponds to a step in the execution of a business process. A business process action might be refined by another business process. In this case, a UML call behavior action <b>MUST</b> be used as base class for the business process action
<b>Tag Definition</b>	No tagged values.

474

<b>Stereotype</b> <b>bInternalState (InternalBusinessEntityState)</b>	
<b>Base Class</b>	ObjectNode
<b>Parent</b>	N/A
<b>Description</b>	The internal business entity state represents a state of a business entity that is internal to the business process of a business partner.
<b>Tag Definition</b>	No tagged values.

475

<b>Stereotype</b> <b>bSharedState (SharedBusinessEntityState)</b>	
<b>Base Class</b>	ObjectNode
<b>Parent</b>	N/A
<b>Description</b>	The shared business entity state represents a state of a business entity that is shared between the business processes between two involved business partners.
<b>Tag Definition</b>	No tagged values.

<b>Form for Business Domain View</b>	
<b>General</b>	
Name	
Description	
<b>Business Library Information</b>	
UniqueIdentifier	
BusinessTerm	
VersionIdentifier	
Status	
Owner	
Copyright	
Reference(s)	
<b>Business Area(s)</b>	
Business Area No 1	
Business Area No 2	
Business Area No 3	
Business Area No 4	
Business Area No 5	
Business Area No 6	(add columns as needed)

477 **5.1.1.4 Worksheets**

478

<b>Form for Business Area</b>	
<b>General</b>	
Name	

Description	
<b>Details</b>	
Objective	
Scope	
Business Opportunity	
Included in	(insert the parent Business Area or Business Domain View)
<b>Business Library Information</b>	
UniqueIdentifier	
BusinessTerm	
VersionIdentifier	
Status	
Owner	
Copyright	
Reference(s)	
<b>Business Area(s)</b>	(insert additional nested business areas if appropriate; otherwise fill process areas that apply)
Business Area No 1	
Business Area No 2	
Business Area No 3	
Business Area No 4	
Business Area No 5	(add columns as needed)
<b>Process Area(s)</b>	(you only fill process areas if you do not have completed a business area above)
Process Area No 1	
Process Area No 2	
Process Area No 3	
Process Area No 4	

Process Area No 5	(add columns as needed)
-------------------	-------------------------

479

<b>Form for Process Area</b>	
<b>General</b>	
Name	
Description	
<b>Details</b>	
Objective	
Scope	
Business Opportunity	
Included in	(insert the parent Business Area or Process Area)
<b>Business Library Information</b>	
UniqueIdentifier	
BusinessTerm	
VersionIdentifier	
Status	
Owner	
Copyright	
Reference(s)	
<b>Process Area(s)</b>	<b>(if present)</b>
Process Area No 1	
Process Area No 2	
Process Area No 3	
Process Area No 4	
Process Area No 5	(add columns as needed)

480

481

482

483

<b>Form for Business Process</b>	
<b>General</b>	
Name	
Description	
<b>Details</b>	
Classified to Business Areas and Process Areas	
Participants and their interests	
Stakeholders and their interests	
Reference(s)	
<b>Start/End Characteristics</b>	
Pre-condition	
Post-condition	
Begins When	
Ends When	
Actions	
Exceptions	
<b>Relationships</b>	
Included Business Processes	
Affected Business Entities and their states	

484

485 **5.1.1.5 Constraints (normative)**

486 C.9. A *BusinessDomainView* MUST include one to many *BusinessAreas*.

487 C.10. A *BusinessArea* MUST include one to many *BusinessAreas* or one to many *ProcessAreas* or  
 488 one to many *BusinessProcessUseCases*.

489 C.11. A *ProcessArea* MUST contain one to many other *ProcessAreas* or one to many  
 490 *BusinessProcessUseCases*



- 491 C.12. A *BusinessProcessUseCase* MUST be associated with one to many *BusinessPartners* using the  
492 *participates* relationship
- 493 C.13. A *BusinessProcessUseCase* MAY be associated with zero to many *Stakeholders* using the  
494 *isOfInterestTo* relationship
- 495 C.14. A *BusinessProcessUseCase* SHOULD be refined by zero to many *BusinessProcesses*. These  
496 relationships MAY also be visualized by realize relationships from each of the owned  
497 *BusinessProcesses* to the owning *BusinessProcessUseCase*
- 498 C.15. A *BusinessProcess* MUST be modeled as a child of a *BusinessProcessUseCase*
- 499 C.16. A *BusinessProcessUseCase* MAY be refined by zero to many UML Sequence Diagrams
- 500 C.17. A *BusinessProcess* MAY contain zero to many *ActivityPartitions*
- 501 C.18. A *BusinessProcess*, which has no *ActivityPartitions*, MUST contain one or more  
502 *BusinessProcessActions* and MAY contain zero to many *InternalBusinessEntityStates* and zero to  
503 many *SharedBusinessEntityStates*.
- 504 C.19. An *ActivityPartition* being part of a *BusinessProcess* MUST contain one to many  
505 *BusinessProcessActions* and MAY contain zero to many *InternalBusinessEntityStates*.
- 506 C.20. A *SharedBusinessEntityState* MUST NOT be located in an *ActivityPartition*. (They must be  
507 contained within the *BusinessProcess* even if this *BusinessProcess* contains *ActivityPartitions*.)

508 **5.1.1.6 Example (informative)**

509

510 **Figure 11 Business Domain View Example: Negotiation In the Order from Quote Example (Use Case Diagram showing**  
511 **Business Process Use Cases)**

512

513  
514

**Figure 12 Business Domain View Example: Business Process of the internal Place Order Business Process Use Case (Activity Diagram)**

515

516 **Figure 13 Business Domain View Example: Business Process of the to-be-designed inter-organizational process called Purchase**  
517 **Product (Activity Diagram)**

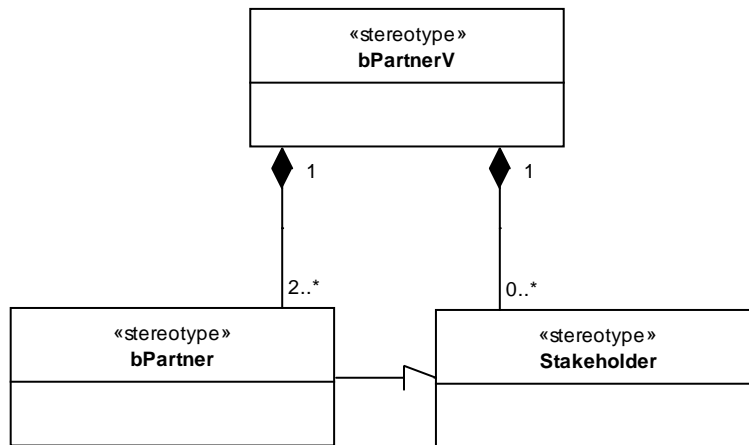
518

## 519 **5.1.2 Business Partner View**

### 520 **5.1.2.1 Abbreviations of Stereotypes**

Stereotype Abbreviation	Full Stereotype Name
bPartnerV	BusinessPartnerView
bPartner	BusinessPartner
Stakeholder	Stakeholder

521 **5.1.2.2 Conceptual Description (informative)**



522

523 **Figure 14 BusinessPartnerView – Conceptual Overview**

524 A business partner is an organization type, an organizational unit type or a person type that participates in a  
525 business process. A *BusinessPartnerView* must contain at least two *BusinessPartners*. A stakeholder is a  
526 person or representative of an organization who has a stake – a vested interest – in a certain business  
527 category or in the outcome of a business process. By definition, a business partner always has a vested  
528 interest in the business processes which they are participating in. Therefore, a *BusinessPartner* is a special  
529 type of a *Stakeholder*. In UML, specific relationships between Actors MAY be defined. The business partner  
530 view does not restrict the definition of those relationships between business partners and/or stakeholders.  
531 For example, generalizations between business partners MAY be defined.

532 **5.1.2.3 Stereotypes and Tag Definitions (normative)**

533

534 **Figure 15 BusinessPartnerView – Abstract Syntax**

Stereotype Stakeholder (Stakeholder)	
Base Class	Actor
Parent	N/A
Description	A stakeholder is a person or representative of an organization who has a stake – a vested interest – in a certain business category or in the outcome of a business process. A stakeholder does not necessarily participate in the execution of a business process.
Tag Definition	<b>interest</b>
	Type String
	Multiplicity 1
	Description Describes the vested interest of the stakeholder in the business category it is defined within.

535

Stereotype bPartner (BusinessPartner)	
Base Class	Actor
Parent	Stakeholder
Description	A business partner type is an organization type, an organizational unit type or a person type that participates in a business process. Business partner types typically provide input to and/or receive output from a business process. Due to the fact that a business partner type participates in a business process, they have, by default, a vested interest in the business process. Therefore, a business partner type is a special kind of stakeholder.
Tag Definition	<b>Inherited tagged values:</b> - interest

536

#### 5.1.2.4 Constraints (normative)

537

C.21. A *BusinessPartnerView* MUST contain at least two to many *BusinessPartners*. If the

538

*BusinessPartnerView* is hierarchically decomposed into sub packages these *BusinessPartners* MAY be

539

contained in any of these sub packages.

540

C.22. A *BusinessPartnerView* MAY contain zero to many *Stakeholders*

541

#### 5.1.2.5 Example (informative)

542

543

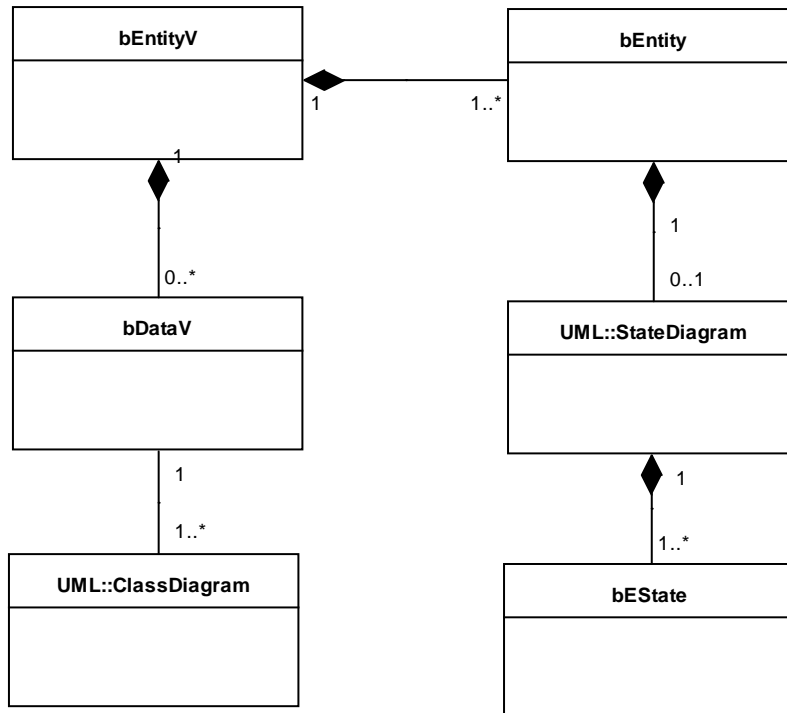
Figure 16 Business Partner View Example

544 **5.1.3 Business Entity View**

545 **5.1.3.1 Abbreviations of Stereotypes**

Stereotype Abbreviation	Full Stereotype Name
bEntityV	BusinessEntityView
bEntity	BusinessEntity
bEState	BusinessEntityState
bDataV	BusinessDataView

546 **5.1.3.2 Conceptual Description (informative)**



547

548 **Figure 17 BusinessEntityView (BusinessRequirementsView) Conceptual Overview**

549

550 A business entity is a real-world thing having business significance that is shared between two or more  
 551 business partners in a collaborative business process (e.g. “order”, “account”, etc.). Within the business  
 552 entity view at least one, but possibly more business entities are described. Thus, the *BusinessEntityView* is  
 553 composed of one to many *BusinessEntities*. The lifecycle of a business entity MAY be described as a flow of  
 554 business entity states. Depending on the importance of the business entity lifecycle, the lifecycle may or  
 555 may not be included. A lifecycle is described using a UML State Diagram. Hence, a *BusinessEntity* is  
 556 composed of zero to one UML State Diagram. The lifecycle represents the different business entity states a  
 557 business entity can exist in. The lifecycle of a business entity consists of at least one business entity state.  
 558 Therefore, the lifecycle of a business entity is composed of one or more *BusinessEntityStates*.

559 A business entity is a potential candidate for becoming a business document in later steps of the UMM. A  
 560 business data view MAY be used to elaborate a first conceptual design of a business entity. Hence, a  
 561 *BusinessEntity* is composed of zero to one *BusinessDataViews*. Within a business data view, A UML class

562 diagram is used to describe the assembly of a business entity. Thus, a *BusinessDataView* contains one to  
563 many UML Class Diagrams.

564 **5.1.3.3 Stereotypes and Tag Definitions (normative)**

565

566

567

**Figure 18 BusinessEntityView (BusinessRequirementsView) Abstract Syntax**

<b>Stereotype</b>	<b>bEntity (BusinessEntity)</b>
<b>Base Class</b>	Class
<b>Parent</b>	N/A
<b>Description</b>	A business entity is a real-world thing having business significance that is shared among two or more business partner types in a collaborative business process (e.g. order, account, etc.).
<b>Tag Definition</b>	No tagged values.

568

569

570

Stereotype <b>bEState (BusinessEntityState)</b>	
Base Class	State
Parent	N/A
Description	A business entity state represents a specific state a business entity can exists in during its lifecycle (an "order" can exist in the states "issued", "rejected", "confirmed", etc.)
Tag Definition	No tagged values.

571

Stereotype <b>bDataV (BusinessDataView)</b>	
Base Class	Package
Parent	BusinessLibrary
Description	The business data view is a container for all elements needed to describe the conceptual assembly of a business entity
Tag Definition	<b>Inherited tagged values:</b> <ul style="list-style-type: none"> <li>- uniqueIdentifier</li> <li>- versionIdentifier</li> <li>- owner</li> <li>- copyright</li> <li>- reference</li> <li>- status</li> <li>- businessTerm</li> </ul>

572

#### 5.1.3.4 Constraints (normative)

573

C.23. A *BusinessEntityView* MUST contain one to many *BusinessEntities*

574

C.24. A *BusinessEntity* SHOULD have one UML State Diagram that describe its lifecycle

575

C.25. A UML State Diagram describing the lifecycle of a *BusinessEntity* MUST contain one to many *BusinessEntityStates*. The parent of a *BusinessEntityState* MUST be a *BusinessEntity*.

576

577

C.26. A *BusinessEntityView* MAY contain zero to many *BusinessDataView* that describe its conceptual design

578

579

C.27. The parent of a *BusinessDataView* MUST be a *BusinessEntityView*

580

C.28. A *BusinessDataView* SHOULD use a UML Class Diagram to describe the conceptual design of a *BusinessEntity*

581

582

C.29. A *BusinessDataView* SHOULD contain one to many classes.

583

#### 5.1.3.5 Worksheets

### Form for Business Entity

#### General

Business Entity Name



Description

### **Business Library Information**

UniqueIdentifier

BusinessTerm

VersionIdentifier

Status

Owner

Copyright

Reference(s)

### **Lifecycle**

Pre-Condition

Post-Condition

Begins When

Ends When

Exceptions

### **Lifecycle States (add more Business Entity States if needed)**

Business Entity State

Name

Description

Preceding State(s) including  
events and transition  
conditions

Valid Actions

Business Entity State

Name

Description

Preceding State(s) including  
events and transition  
conditions

Valid Actions

Business Entity State

Name

Description

Preceding State(s) including  
events and transition  
conditions

Valid Actions

Business Entity State

Name

Description

Preceding State(s) including  
events and transition  
conditions

Valid Actions

Business Entity State

Name

Description

Preceding State(s) including  
events and transition  
conditions

Valid Actions

584

585 **5.1.3.6 Example (informative)**

586

587 **Figure 19 BusinessEntityView Example: BusinessEntities - CreditCheck, Registration, Quote and Order (Class Diagram)**

588

589 **Figure 20 BusinessEntityView Example: CreditCeck Lifecycle (State Diagram)**

590

591

**Figure 21 BusinessEntityView Example: Registration Lifecycle (State Diagram)**

592

593

**Figure 22 BusinessEntityView (BusinessRequirementsView) Example: Order BusinessEntityLifecycle (State Diagram)**

594

595 **Figure 23 BusinessEntityView (BusinessRequirementsView) Example: Quote BusinessEntityLifecycle (State Diagram)**

596

## 597 **5.2 Business Choreography View**

### 598 **5.2.1 Sub-Views in the Business Choreography View**

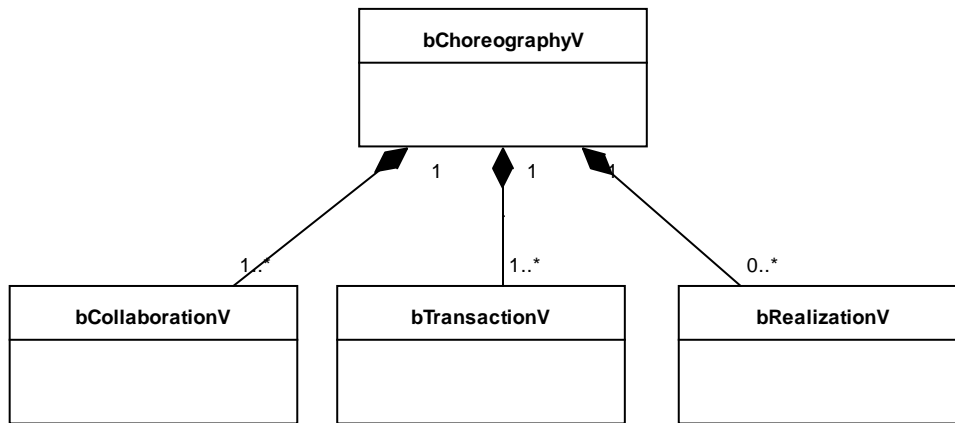
#### 599 *5.2.1.1 Abbreviations and Stereotypes*

600

Stereotype Abbreviation	Full Stereotype Name
bChoreographyV	BusinessChoreographyView
bCollaborationV	BusinessCollaborationView
bTransactionV	BusinessTransactionView
bRealizationV	BusinessRealizationView

601

602 **5.2.1.2 Conceptual Description (informative)**



603

604 **Figure 24 BusinessChoreographyView Conceptual Overview**

605 The *BusinessChoreographyView* is the second out of the 3 views of a UMM compliant business collaboration  
 606 model. The business choreography view describes the view how the business analyst sees the process to be  
 607 modeled. The requirements captured in the business requirements view serve as a basis for the definition of  
 608 a choreography of information exchanges. The business choreography view is a container for three different  
 609 artifacts that together describe the overall choreography of information exchanges. A  
 610 *BusinessTransactionView* is a container for artifacts that define a choreography leading to synchronized  
 611 states of business entities at both sides of the interaction. In fact, a business transaction view captures two  
 612 different artifacts that define the business transaction. First, the business analyst defines concrete  
 613 requirements specifying the business transaction on a more general level by using business transaction use  
 614 cases. Second, he defines the flow of information exchanges in accordance to the requirements specified in  
 615 this container. The business collaboration view is a container for artifacts describing the flow of a complex  
 616 business collaboration between business partner types that may involve many steps. Similar to the business  
 617 transaction view, the *BusinessCollaborationView* captures two different artifacts as well. Once the business  
 618 analyst has specified the concrete requirements for a business collaboration by using business collaboration  
 619 use cases, he is able to define the flow in accordance to the requirements defined in this container. Finally,  
 620 the *CollaborationRealizationView* describes the realization of a business collaboration use case for a specific  
 621 set of business partner types.

622 **5.2.1.3 Stereotypes and Tag Definitions (normative)**

623  
624 **Figure 25 BusinessChoreographyView Abstract Syntax**  
625

<b>Stereotype bChoreographyV (BusinessChoreographyView)</b>	
<b>Base Class</b>	Package
<b>Parent</b>	BusinessLibrary (from Base Module)
<b>Description</b>	The business choreography view is a container for all elements needed to describe the choreography of a business collaboration at various levels and the information exchanged in each step of the choreography.
<b>Tag Definition</b>	<p><b>Inherited tagged values:</b></p> <ul style="list-style-type: none"> <li>– uniqueIdentifier</li> <li>– versionIdentifier</li> <li>– owner</li> <li>– copyright</li> <li>– reference</li> <li>– status</li> <li>– businessTerm</li> </ul>

626

<b>Stereotype bCollaborationV (BusinessCollaborationView)</b>	
---	--

<b>Base Class</b>	Package
<b>Parent</b>	BusinessLibrary (from Base Module)
<b>Description</b>	The business collaboration view is a container for artifacts describing the flow of a complex business collaboration between business partner types that may involve many steps.
<b>Tag Definition</b>	<b>Inherited tagged values:</b> <ul style="list-style-type: none"> <li>– uniqueIdentifier</li> <li>– versionIdentifier</li> <li>– owner</li> <li>– copyright</li> <li>– reference</li> <li>– status</li> <li>– businessTerm</li> </ul>

627

<b>Stereotype</b>	<b>bTransactionV (BusinessTransactionView)</b>
<b>Base Class</b>	Package
<b>Parent</b>	BusinessLibrary (from Base Module)
<b>Description</b>	The transaction requirements view is a container for artifacts that define a choreography leading to synchronized states of business entities at both sides of the business transaction.
<b>Tag Definition</b>	<b>Inherited tagged values:</b> <ul style="list-style-type: none"> <li>– uniqueIdentifier</li> <li>– versionIdentifier</li> <li>– owner</li> <li>– copyright</li> <li>– reference</li> <li>– status</li> <li>– businessTerm</li> </ul>

628

<b>Stereotype</b>	<b>bRealizationV (BusinessRealizationView)</b>
<b>Base Class</b>	Package
<b>Parent</b>	BusinessLibrary (from Base Module)
<b>Description</b>	The business realization view is a container for all elements describing the realization of a business collaboration use case by business partner types.
<b>Tag Definition</b>	<b>Inherited tagged values:</b> <ul style="list-style-type: none"> <li>– uniqueIdentifier</li> <li>– versionIdentifier</li> <li>– owner</li> <li>– copyright</li> <li>– reference</li> <li>– status</li> <li>– businessTerm</li> </ul>

629

630



631 **5.2.1.4 Constraints (normative)**

632 Constraints with respect to a *BusinessChoreographyView*:

633 C.30. A *BusinessChoreographyView* MUST contain one to many *BusinessCollaborationViews*

634 C.31. A *BusinessChoreographyView* MUST contain one to many *BusinessTransactionViews*.

635 C.32. A *BusinessChoreographyView* MAY contain zero to many *BusinessRealizationViews*.

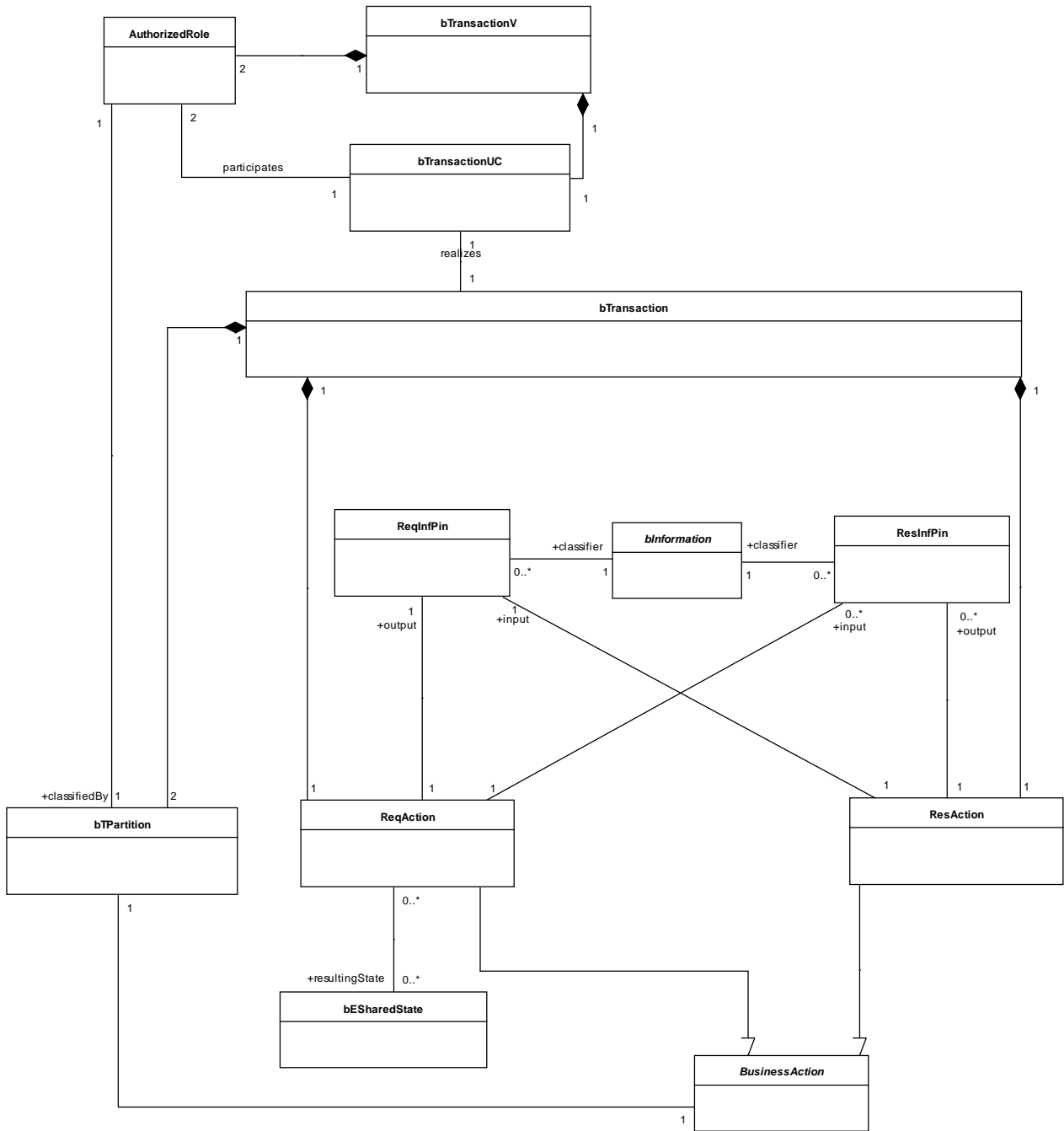
636 C.33. A *BusinessTransactionView*, a *BusinessCollaborationView*, and a *BusinessRealizationView* MUST be  
637 directly located under a *BusinessChoreographyView*

638 **5.2.2 Business Transaction View**

639 **5.2.2.1 Abbreviations of Stereotypes**

Stereotype Abbreviation	Full Stereotype Name
bTransactionView	BusinessTransactionView
AuthorizedRole	AuthorizedRole
bTransactionUC	BusinessTransactionUseCase
bTransaction	BusinessTransaction
bTPartition	BusinessTransactionPartition
bInformation	BusinessInformation
ReqInfPin	RequestingInformationPin
ResInfPin	RespondingInformationPin
InfPin	InformationPin
BusinessAction	BusinessAction
ReqAction	RequestingBusinessAction
ResAction	RespondingBusinessAction
bESharedState	SharedBusinessEntityState

640



642  
643 **Figure 26 Business Transaction View – Conceptual Overview**

644 Each *BusinessTransactionView* defines exactly one message exchange that leads to a synchronized business  
645 state between the two authorized roles executing it. The flow of messages is specified by the concept of a  
646 *BusinessTransaction*. The requirements of a *BusinessTransaction* are captured by a  
647 *BusinessTransactionUseCase*.

648 Each business transaction and its corresponding business transaction use case are defined in their own  
649 business transaction view package. Accordingly, the business transaction view is composed of exactly one  
650 *BusinessTransactionUseCase* and one *BusinessTransaction*.

#### 651 5.2.2.2.1 Business Transaction Use Case Diagram

652 Two authorized roles participate in a business transaction use case. These authorized roles must be defined  
653 in the same business transaction view package as the corresponding business transaction use case.  
654 Accordingly, a *BusinessTransactionView* is composed of exactly two *AuthorizedRoles*. This means, if a certain  
655 role (e.g. buyer, seller, etc.) participates in multiple business transactions, it requires a different authorized  
656 role for each business transaction use case. Each authorized role of the same role (i.e., with the same name)  
657 is in a different namespace of a corresponding business transaction view. Therefore, an authorized role  
658 participates in only one business transaction use case – it is the one in the same business transaction view.  
659 Accordingly, *BusinessTransactionUseCase* and *AuthorizedRole* are related by a 1 to 2 association. It is  
660 important to note, that the same authorized role is not associated twice to the same business transaction  
661 use case.

#### 662 5.2.2.2.2 Business Transaction Diagram

663 A *BusinessTransaction* choreographs the synchronization of business states and the required information  
664 exchange between two authorized roles. The business transaction follows exactly the requirements defined  
665 in the corresponding business transaction use case. The business transaction that describes the business  
666 transaction use case is defined as a child beneath. Accordingly, each *BusinessTransactionUseCase* has exactly  
667 one *BusinessTransaction* beneath. A business transaction is a “composite” UML *Activity*. The graph of a  
668 business transaction is described by a flow of UML *Actions*.

669 A business transaction is an atomic step in a collaborative business process between two authorized roles,  
670 which involves sending business information from one authorized role to the other and an optional reply.  
671 The business transaction is built by two partitions - one for each authorized role. Hence, a  
672 *BusinessTransaction* is composed of exactly two *BusinessTransactionPartitions*. Each  
673 *BusinessTransactionPartition* relates to one *AuthorizedRole*. An *Authorized Role* is assigned to exactly one  
674 *BusinessTransactionPartition*. It follows, that the two partitions of a business transaction must be assigned to  
675 different authorized roles.

676 Within a business transaction each authorized role performs exactly one business action – the requesting  
677 authorized role performs a requesting business action and the responding authorized role performs a  
678 responding business action. Each business action – no matter whether requesting or responding business  
679 action – is assigned to a swimlane, and each swimlane comprises exactly one business action. It follows that  
680 a *BusinessTransaction* is composed of exactly one *RequestingBusinessAction* and exactly one  
681 *RespondingBusinessAction*. Both, *RequestingBusinessAction* and *RespondingBusinessAction* are  
682 specializations of the abstract type *BusinessAction*. A *BusinessAction* is assigned to one  
683 *BusinessTransactionPartition*, and a *BusinessTransactionPartition* comprises one *BusinessAction*. Since a  
684 partition is dedicated to exactly one authorized role, it follows that the business action is executed by this  
685 authorized role. Furthermore an authorized role executes just one business action, because only one  
686 business action sits within a partition.

687 The requesting business action outputs the requesting information through the requesting information pin  
688 that is input to the responding business action’s requesting information pin. Business information created by  
689 the responding business action and returned to the requesting business action is optional. If business  
690 information is returned by the responding business action zero to many responding information pins might  
691 be specified. Multiple responding information pins may be used to describe different business intentions  
692 (e.g., a positive and a negative response to a purchase order).

693 It follows, that a *BusinessTransaction* is composed of exactly two *RequestingInformationPins* and zero to  
694 many *RespondingInformationPins*. Both *RequestingInformationPin* and *RespondingInformationPin* are  
695 instances of the type *BusinessInformation*. A *RequestingBusinessAction* has exactly one  
696 *RequestingInformationPin* and zero to many *RespondingInformationPins*.

697 A *RespondingBusinessAction* has exactly one *RequestingInformationPin* and zero to many  
698 *RespondingInformationPins*.

699 *RequestingInformationPin* and *RespondingInformationPin* are stereotypes of the UML base class *Pin*. The  
700 type of the *Pin* is defined by the *BusinessInformation* that is a stereotype of the UML base class *Class*.  
701 According to UML, multiple *Pins* might be instances of the same *Class*. It follows that different requesting or  
702 responding information pins might be instances of the same business information. In other words, business  
703 information might be reused in different business transactions. Action pins that specify output information  
704 from a business action MUST be stereotyped and classified accordingly, whereas action pins that specify  
705 input information to a business action MAY not be stereotyped and classified.

706 If multiple responding information pins are defined, those must be in an XOR relationship with each other.  
707 In order to specify an XOR relationship between multiple incoming or outgoing responding information pins,  
708 each of them has to be enclosed by an UML *ParameterSet* (c.f. Figure 32). If only one responding information  
709 pin is defined within a business transaction, *ParameterSets* SHOULD not be used.

710 In order to determine the outcome of a business transaction (success or failure) the contents of the  
711 responding business document SHOULD be evaluated. OCL constraints SHOULD be used for assessing the  
712 document's content. An OCL constraint may either check the responding business information's type (e.g.,  
713 positive or negative response to a quote) or directly investigate the document's content (e.g., if products  
714 were quoted or not). If the responding business information is checked, the constraints MUST be applied as  
715 condition guards to the transitions leading into the respective final states (e.g., success or response) of the  
716 business transaction. If the business transaction does not include a response, OCL constraints MAY not be  
717 used.

718 A business transaction synchronizes the states between the two authorized roles executing it. Thus, the  
719 execution of a business transaction results in a certain business entity shared state (the concept of business  
720 entity shared states have already been introduced in section 5.1.1.2). In order to point out the state change,  
721 setting the resulting shared state of a business entity might be visualized on the diagram of a business  
722 transaction. A *SharedBusinessEntityState* MAY be included as a predecessor of a final state to indicate the  
723 resulting synchronized state. The example in Figure 33 illustrates this concept.

from BusinessDomainView

from Business Domain  
View

725

726

**Figure 27 Business Transaction View – Abstract Syntax**

<b>Stereotype bTransactionUC (BusinessTransactionUseCase)</b>	
<b>Base Class</b>	UseCase
<b>Parent</b>	bProcessUC
<b>Description</b>	A business transaction use case describes in detail the requirements on a collaboration between exactly two involved partners. A business transaction use case cannot be further refined and describes the requirements on a one-way or two-way information exchange. Business partners take part in a business transaction use case by playing an authorized role in it.
<b>Tag Definition</b>	<b>Inherited tagged values:</b>

727

	<ul style="list-style-type: none"> <li>- definition</li> <li>- beginsWhen</li> <li>- preCondition</li> <li>- endsWhen</li> <li>- postCondition</li> <li>- exceptions</li> <li>- actions</li> </ul>
--	--

728

Stereotype	AuthorizedRole (AuthorizedRole)
<b>Base Class</b>	Actor
<b>Parent</b>	N/A
<b>Description</b>	An authorized role (e.g. a “buyer”) is a concept which is more generic than a business partner (e.g. a “wholesaler”) and allows the reuse of collaborations by mapping an <i>AuthorizedRole</i> to a business partner within a given scenario. Since business collaboration use case and business transaction use case are defined as occurring between authorized roles, they might be reused by different business partners (a “wholesaler” or a “broker”) in different scenarios of the same domain or even in different domains.
<b>Tag Definition</b>	No tagged values.

Stereotype	bTransaction (BusinessTransaction)
<b>Base Class</b>	Activity
<b>Parent</b>	N/A
<b>Description</b>	<p>A business transaction is the basic building block to define choreography between authorized roles. If an authorized role recognizes an event that changes the state of a business object, it initiates a business transaction to synchronize with the collaborating authorized role. It follows that a business transaction is an atomic unit that leads to a synchronized state in both information systems. We distinguish one-way and two-way business transaction: In the former case, the initiating authorized role reports an already effective and irreversible state change that the reacting authorized role has to accept. Examples are the notification of shipment or the update of a product in a catalog. It is a one-way business transaction, because business information (not including business signals for acknowledgments) flows only from the initiating to the reacting authorized role. In the other case, the initiating partner sets the business object(s) into an interim state and the final state is decided by the reacting authorized role. Examples include request for registration, search for products, etc. It is a two-way transaction, because business information flows from the initiator to the responder to set the interim state and backwards to set the final and irreversible state change. In a business context irreversible means that returning to an original state requires another – compensating – business transaction. E.g., once a purchase order is agreed upon in a business transaction a rollback is not allowed anymore, but requires the execution of a cancel order business transaction compensating the before sent purchase order. We distinguish 2 one-way business transactions and four two-way business transactions. The type of transaction is indicated in the tagged value of business transaction type. The other tagged values provide quality of service parameters.</p> <p>A business transaction follows always the same pattern: A business transaction is performed between two authorized roles that are assigned to exactly one swimlane each. Each authorized role performs exactly one activity. An object flow between the requesting and the responding business activity is mandatory. Up to two object flows in the reverse direction are optional. If two object flows are defined as</p>

	<p>response, those are in an XOR relationship with each other (e.g., a purchase order is accepted or declined). According to the business transaction semantics, the requesting business activity does not end after sending the business information - it is still alive. The responding business activity may output the response which is returned to the still living requesting business activity.</p>	
Tag Definition	<b>businessTransactionType</b>	
	<b>Type</b>	<p>Enumeration with name <i>TransactionPatterns</i> consisting of the following values:</p> <ul style="list-style-type: none"> <li>• Commercial Transaction</li> <li>• Request/Confirm</li> <li>• Query/Response</li> <li>• Request/Response</li> <li>• Notification</li> <li>• Information Distribution</li> </ul>
	<b>Multiplicity</b>	1
	<b>Description</b>	<p>The business transaction type determines a corresponding business transaction pattern. A business transaction pattern provides a language and grammar for constructing business transactions. The business transaction type follows one of the following six property-value conventions:</p> <p>(1) Commercial Transaction - used to model the “offer and acceptance” business transaction process that results in a residual obligation between both parties to fulfill the terms of the contract</p> <p>(2) Query/Response – used to query for information that a responding partner already has e.g. against a fixed data set that resides in a database</p> <p>(3) Request/Response - used for business contracts when an initiating partner requests information that a responding partner already has and when the request for business information requires a complex interdependent set of results</p> <p>(4) Request/Confirm - used if an initiating partner asks for information that requires only confirmation with respect to previously established contracts or with respect to a responding partner’s business rules</p> <p>(5) Information Distribution - used to model an informal information exchange business transaction that therefore has no non-repudiation requirements</p> <p>(6) Notification - used to model a formal information exchange business transaction that therefore has non-repudiation requirements</p>
	<b>isSecureTransportRequired</b>	
	<b>Type</b>	Boolean
	<b>Multiplicity</b>	1
	<b>Description</b>	<p>Both partners must agree to exchange business information using a secure transport channel. The following security controls ensure that business document content is protected against unauthorized disclosure or modification and that business services are protected against unauthorized access. This is a point-to-point security requirement. Note that this requirement does not protect business information once it is off the network and inside an enterprise. The following are requirements for secure transport channels.</p>

729

		<p>Authenticate sender identity – Verify the identity of the sender (employee or organization) that is initiating the interaction (authenticate). For example, a driver’s license or passport document with a picture is used to verify an individual’s identity by comparing the individual against the picture.</p> <p>Authenticate receiver identity – Verify the identity of the receiver (employee or organization) that is receiving the interaction.</p> <p>Verify content integrity – Verify the integrity of the content exchanged during the interaction i.e. check that the content has not been altered by a 3rd party.</p> <p>Maintain content confidentiality – Confidentiality ensures that only the intended, receiver can read the content of the interaction. Information exchanged during the interaction must be encrypted when sent and decrypted when received. For example, you seal envelopes so that only the recipient can read the content.</p>
--	--	--

730

<b>Stereotype</b>		<b>bTPartition (BusinessTransactionPartition)</b>
<b>Base Class</b>		Partition
<b>Parent</b>		N/A
<b>Description</b>		A business transaction partition is used to define an area of responsibility. An authorized role is appointed to a business transaction swimlane to indicate that this authorized role takes on the responsibility for the business action that is allocated within that area.
<b>Tag Definition</b>		<b>No Tagged Values</b>

<b>Stereotype</b>		<b>BusinessAction (BusinessAction, abstract)</b>
<b>Base Class</b>		Action
<b>Parent</b>		N/A
<b>Description</b>		A business action is executed by an authorized role during a business transaction. Business action is an abstract stereotype. This means a business action is either a requesting business action or a responding business action.
<b>Tag Definition</b>	<b>isAuthorizationRequired</b>	
	<b>Type</b>	<b>Boolean</b>
	<b>Multiplicity</b>	<b>1</b>
	<b>Description</b>	If an authorized role needs authorization to request a business action or to respond to a business action then the sender must sign the business document exchanged and the receiver must validate this business control and approve the authorizer. A receiver must signal an authorization exception if the sender is not authorized to perform the business activity. A sender must send notification of failed authorization if a receiver is not authorized to perform the responding business activity.
	<b>isNonRepudiationRequired</b>	
<b>Type</b>	<b>Boolean</b>	



<b>Multiplicity</b>	<b>1</b>
<b>Description</b>	The <i>isNonRepudiationRequired</i> tag is used to indicate that an involved party must not be able to repudiate the execution of the business action that input/outputs business information.
<b>isNonRepudiationReceiptRequired</b>	
<b>Type</b>	<b>Boolean</b>
<b>Multiplicity</b>	<b>1</b>
<b>Description</b>	The <i>isNonRepudiationOfReceiptRequired</i> tag requires the receiver of a business information to send a signed receipt. The <i>isNonRepudiationOfReceiptRequired</i> tag indicates that an involved party must not be able to repudiate the execution of sending the signed receipt.
<b>timeToAcknowledgeReceipt</b>	
<b>Type</b>	<b>String (which must conform to a value of the W3C duration data type)</b>
<b>Multiplicity</b>	<b>1</b>
<b>Description</b>	<p>Both partners may agree to mutually verify receipt of business information within a specific time duration. Acknowledgements of receipt may be sent for both the requesting business information and the responding business information. This means the sender of the business information may be the requesting authorized role as well as the responding authorized role – it depends on whether a requesting or a responding business information is acknowledged. Similarly, the affirmant may be the requesting authorized role as well as the responding authorized role – again depending of which business information is acknowledged. Inasmuch we use the terms sender and affirmant in the explanation of acknowledgement of receipt semantics.</p> <p>An affirmant must exit the transaction if they are not able to verify the proper receipt of a business information within the agree timeout period. A sender must retry a business transaction if necessary or must send notification of failed business control (possibly revoking a contractual offer) if an affirmant does not verify properly receipt of a business information within the agreed time period. The time to acknowledge receipt is the maximum duration from the time a business information is sent by a sender until the time a verification of receipt is “properly received” by the sender (of the business information). Accordingly, the time to acknowledge receipt is always specified by the sender’s business action. This verification of receipt is an audit-able business signal and is instrumental in contractual obligation transfer during a contract formation process (e.g. offer/accept).</p>
<b>timeToAcknowledgeProcessing</b>	
<b>Type</b>	<b>String (which must conform to a value of the W3C duration data type)</b>
<b>Multiplicity</b>	<b>1</b>
<b>Description</b>	Similarly to the <i>timeToAcknowledgeReceipt</i> , the sender of a business information might be the requesting authorized role as well as the responding authorized role – depending whether a requesting or a responding business information is acknowledged. Also the affirmant may be one of the two authorized roles. Thus,

		<p>we use again the terms sender and affirmant in the explanation of the acknowledgment of processing semantics.</p> <p>Both partners may agree to the need for an acknowledgment of processing to be returned by a responding partner after the requesting business information passes a set of business rules and is handed over to the application for processing. The time to acknowledge processing of a business information is the duration from the time a sender sends a business information until the time an acknowledgement of processing is “properly received” by the sender (of the business information). Accordingly, the time to acknowledge processing is always specified by the sender’s business action. An affirmant must exit the transaction if they are not able to acknowledge processing of business information within the maximum timeout period. A sender must retry a business transaction if necessary or must send notification of failed business control (possibly revoking a contractual offer) if an affirmant does not acknowledge processing of business information within the agreed time period.</p>
	<b>isIntelligibleCheckRequired</b>	
	<b>Type</b>	<b>Boolean</b>
	<b>Multiplicity</b>	<b>1</b>
	<b>Description</b>	<p>In order to define the <i>isIntelligibleCheckRequired</i> semantics, we use again the terms sender and affirmant as introduced for the last two tag definitions.</p> <p>Both partners may agree that an affirmant must check that business information is not garbled (unreadable, unintelligible) before verification of proper receipt is returned to the sender (of the business information). Verification of receipt must be returned when a document is “accessible” but it is preferable to also check for garbled transmissions at the same time in a point-to-point synchronous business network where partners interact without going through an asynchronous service provider.</p>

731

<b>Stereotype</b>	<b>ReqAction (RequestingBusinessAction)</b>	
<b>Base Class</b>	Action	
<b>Parent</b>	BusinessAction	
<b>Description</b>	A requesting business action is a business action that is performed by an authorized role requesting business service from another authorized role.	
<b>Tag Definition</b>	<b>timeToRespond</b>	
	<b>Type</b>	<b>String (which must conform to a value of the W3C duration data type)</b>
	<b>Multiplicity</b>	1
	<b>Description</b>	<p>Both partners may agree in case of a two-way business transaction that the responding authorized role must return the responding information business information within a specific duration.</p> <p>A responding authorized role must exit the transaction if they are not able to return the responding business information within the agreed timeout period. A requesting authorized role must retry a business transaction if necessary or must</p>

	send notification of failed business control (possibly revoking a contractual offer) if a responding authorized role does not deliver the responding business information within the agreed time period. The time to perform is the maximum duration from the time a requesting business information is sent by a requesting authorized role until the time a responding business information is “properly received” by the requesting authorized role in return.							
<b>retryCount</b>								
<b>Type</b>	Integer							
<b>Multiplicity</b>	1							
<b>Description</b>	The requesting authorized role must re-initiate the business transaction so many times as specified by the retry count in case that a time-out-exception – by exceeding the time to acknowledge receipt, or the time to acknowledge processing, or the time to respond – is signaled. This parameter only applies to time-out signals and not document content exceptions or sequence validation exceptions – i.e., failed business control exceptions.							
<b>Inherited tagged values:</b>								
<ul style="list-style-type: none"> <li>- isAuthorizationRequired</li> <li>- isNonRepudiationRequired</li> <li>- isNonRepudiationReceiptRequired</li> <li>- timeToAcknowledgeReceipt</li> <li>- timeToAcknowledgeProcessing</li> <li>- isIntelligibleCheckRequired</li> </ul>								
Default assignment of tagged values for the requesting business action:								
	Time to Acknowledge Receipt	Time to Acknowledge Processing	Time to Respond	Is Authorization Required	Is Non Repudiation Required	Is Non Repudiation of Receipt Required	Retry Count	Is Intelligible Check Required
Commercial Transaction	2h	6h	24h	TRUE	TRUE	TRUE	3	TRUE
Request/Confirm	NULL	NULL	24h	FALSE	FALSE	FALSE	3	TRUE
Request/Response	NULL	NULL	4h	FALSE	FALSE	FALSE	3	TRUE
Query/Response	NULL	NULL	4h	FALSE	FALSE	FALSE	3	TRUE
Notification	24h	NULL	NULL	FALSE	TRUE	TRUE	3	TRUE
Information Distribution	NULL	NULL	NULL	FALSE	FALSE	FALSE	0	TRUE

Stereotype		ResAction (RespondingBusinessAction)						
Base Class	Action							
Parent	Business Action							
Description	A responding business activity is a business action that is performed by an authorized role responding to another authorized role's request for business service.							
Tag Definition	<b>Inherited tagged values:</b> - isAuthorizationRequired - isNonRepudiationRequired - isNonRepudiationReceiptRequired - timeToAcknowledgeReceipt - timeToAcknowledgeProcessing - isIntelligibleCheckRequired  Default assignment of tagged values for the responding business action:							
		Time to Acknowledge Receipt	Time to Acknowledge Processing	Is Authorization Required	Is Non Repudiation Required	Is Non Repudiation of Receipt Required	Is Intelligible Check Required	
	Commercial Transaction	2h	6hr	TRUE	TRUE	TRUE	TRUE	
	Request/Confirm	2h	NULL	TRUE	FALSE	TRUE	TRUE	
	Request/Response	NULL	NULL	FALSE	FALSE	FALSE	TRUE	
	Query/Response	NULL	NULL	FALSE	FALSE	FALSE	TRUE	
	Notification	NULL	NULL	FALSE	FALSE	FALSE	TRUE	
	Information Distribution	NULL	NULL	FALSE	FALSE	FALSE	TRUE	

Stereotype		InfPin (InformationPin, abstract)						
Base Class	Pin							
Parent	N/A							
Description	The abstract concept information pin represents the incoming/outgoing point for business information in a business action. Business information is sent from the requesting authorized role to the responding authorized role or the reverse way. The actual exchanged information is represented using the type							

	business information. Both concrete stereotypes requesting information pin and responding information pin inherit from the abstract stereotype information pin.	
<b>Tag Definition</b>	<b>isConfidential</b>	
	<b>Type</b>	Boolean
	<b>Multiplicity</b>	1
	<b>Description</b>	If the flag is set, the exchanged information is encrypted so that unauthorized parties cannot view the information.
	<b>isTamperProof</b>	
	<b>Type</b>	Boolean
	<b>Multiplicity</b>	1
	<b>Description</b>	If the flag is set, the exchanged information has an encrypted message digest that can be used to check if the message has been tampered with. This requires a digital signature (sender's digital certificate and encrypted message digest) associated with the document entity.
	<b>isAuthenticated</b>	
	<b>Type</b>	Boolean
<b>Multiplicity</b>	1	
<b>Description</b>	If the flag is set, there is a digital certificate associated with the document entity. This provides proof of the signer's identity.	

735

<b>Stereotype</b>	<b>ReqInfPin (RequestingInformationPin)</b>
<b>Base Class</b>	Pin
<b>Parent</b>	InformationPin
<b>Description</b>	The requesting information pin is a container for business information that is sent from the requesting authorized role to the responding authorized role to indicate a state change in one or more business entities. This business state change might be irreversible in the case of a one-way business transaction or an interim state of a two-way business transaction. It is important to note that the term requesting information pin does not mean that the exchanged business information refers to a request in a business sense. The term requesting information pin indicates that the execution of a transaction is requested from the requesting authorized role to the responding authorized role – no matter whether this is an information distribution, a notification, a request, or the offer in a commercial transaction.
<b>Tag Definition</b>	<b>Inherited tagged values:</b> - isConfidential - isAuthenticated - isTamperProof

736

<b>Stereotype</b>	<b>ResInfPin (RespondingInformationPin)</b>
<b>Base Class</b>	Pin
<b>Parent</b>	InformationPin

<b>Description</b>	The responding information pin is a container of business information that is sent in case of a two-way business transaction from the responding authorized role to the requesting authorized role in order to set one or more business entities in a final state (which were in an interim state before).
<b>Tag Definition</b>	<b>Inherited tagged values:</b> - isConfidential - isAuthenticated - isTamperProof

737

738 **5.2.2.4 Constraints (normative)**

- 739 C.34. A *BusinessTransactionView* MUST contain exactly one *BusinessTransactionUseCase*, exactly  
740 two *AuthorizedRoles*, and exactly two *participates* associations.
- 741 C.35. A *BusinessTransactionUseCase* MUST be associated with exactly two *AuthorizedRoles* via  
742 stereotyped binary *participates* associations.
- 743 C.36. A *BusinessTransactionUseCase* MUST NOT include further *UseCases*
- 744 C.37. A *BusinessTransactionUseCase* MUST be included in at least one  
745 *BusinessCollaborationUseCase*.
- 746 C.38. A *BusinessTransactionUseCase* MUST NOT be source or target of an extend association.
- 747 C.39. The two *AuthorizedRoles* within a *BusinessTransactionView* MUST NOT be named identically
- 748 C.40. Exactly one *BusinessTransaction* MUST be placed beneath each *BusinessTransactionUseCase*.  
749 This relationship MAY also be visualized by a realize relationship from the *BusinessTransaction* to the  
750 *BusinessTransactionUseCase*.
- 751 C.41. A *BusinessTransaction* MUST have exactly two partitions. Each of them MUST be stereotyped  
752 as *BusinessTransactionPartition*.
- 753 C.42. One of the two *BusinessTransactionPartitions* MUST contain one *RequestingBusinessAction*  
754 and the other one MUST contain one *RespondingBusinessAction*.
- 755 C.43. A *BusinessTransactionPartition* MUST have a classifier, which MUST be one of the associated  
756 *AuthorizedRoles* of the corresponding *BusinessTransactionUseCase*.
- 757 C.44. The two *BusinessTransactionPartitions* MUST have different classifiers.
- 758 C.45. The *BusinessTransactionPartition* containing the *RequestingBusinessAction* MUST contain  
759 two or more *FinalStates*. Each of the *FinalStates* MAY have a *SharedBusinessEntityState* as  
760 predecessor. One of the *FinalStates* SHOULD reflect a *ControlFailure* – this *FinalState* SHOULD NOT  
761 have a predecesing *SharedBusinessEntityState*.
- 762 C.46. A *RequestingBusinessAction* MUST embed exactly one *RequestingInformationPin*
- 763 C.47. A *RespondingBusinessAction* MUST embed exactly one *RequestingInformationPin*
- 764 C.48. If the tagged value *businessTransactionType* of the *BusinessTransaction* is either  
765 *Request/Response*, *Query/Response*, *Request/Confirm*, or *CommercialTransaction*, then the  
766 *RequestingBusinessAction* MUST embed one to many *RespondingInformationPins* and the  
767 *RespondingBusinessAction* MUST embed one to many *RespondingInformationPins*.
- 768 C.49. If the tagged value *businessTransactionType* of the *BusinessTransaction* is either *Notification*  
769 or *InformationDistribution*, then both, the *RequestingBusinessAction* and the  
770 *RespondingBusinessAction*, MUST NOT embed a *RespondingInformationPin*
- 771 C.50. A *RequestingBusinessAction* and a *RespondingBusinessAction* MUST embed same number of  
772 *RespondingInformationPins*.

- 773 C.51. The *RequestingInformationPin* of the *RequestingBusinessAction* MUST be connected with the  
 774 *RequestingInformationPin* of the *RespondingBusinessAction* using an object flow relationship leading  
 775 from the *RequestingBusinessAction* to the *RespondingBusinessAction*.
- 776 C.52. Each *RespondingInformationPin* of the *RespondingBusinessAction* MUST be connected with  
 777 exactly one *RespondingInformationPin* of the *RequestingBusinessAction* using an object flow  
 778 relationship leading from the *RespondingBusinessAction* to the *RequestingBusinessAction*
- 779 C.53. If a *BusinessTransactionPartition* contains *SharedBusinessEntityStates*, each  
 780 *SharedBusinessEntityState* MUST be the target of exactly one control flow relationship starting from  
 781 the *RequestingBusinessAction* and MUST be the source of exactly one control flow relationship  
 782 targeting a *FinalState*.
- 783 C.54. Each *FinalState* MUST be the target of one to many control flow relationships starting from  
 784 the *RequestingBusinessAction* or from a *SharedBusinessEntityState*.
- 785 C.55. Each *RequestingInformationPin* and each *RespondingInformationPin* MUST have a classifier,  
 786 this classifier MUST be an *InformationEnvelope* or a subtype defined in an extension/specialization  
 787 module.
- 788 C.56. Two *RequestingInformationPins* which are connected using an object flow MUST have the  
 789 same classifier.
- 790 C.57. Two *RespondingInformationPins* which are connected using an object flow MUST have the  
 791 same classifier.

792 **5.2.2.5 Worksheets**

Form for Business Transaction Use Case	
<b>General</b>	
Name	
Description	
<b>Business Library Information</b>	
UniqueIdentifier	
BusinessTerm	
VersionIdentifier	
Status	
Owner	
Copyright	
Reference(s)	
<b>Details</b>	
Requesting Role	

Responding Role	
Requesting Activity	
Responding Activity	
Is Included In (Name of Business Collaboration)	
<b>Start/End Characteristics</b>	
Affected Business Entities	
Pre-condition	
Post-condition	
Begins When	
Ends When	
Exceptions	

793

<b>Form for Business Transaction</b>	
<b>General</b>	
Name	
Description	
<b>Business Library Information</b>	
UniqueIdentifier	
BusinessTerm	
VersionIdentifier	
Status	
Owner	
Copyright	
Reference(s)	
<b>Details</b>	



Select Business Transaction Pattern	<input type="checkbox"/> Information Distribution <input type="checkbox"/> Notification <input type="checkbox"/> RequestResponse <input type="checkbox"/> RequestConfirm <input checked="" type="checkbox"/> QueryResponse <input type="checkbox"/> Commercial Transaction
Secure Transport	
<b>Requestor's Side</b>	
Requesting Role	
Requesting Business Action Name	
Time to Respond	
Time to Acknowledge Receipt	
Time to Acknowledge Processing	
Authorization Required	
Non Repudiation Required	
Non Repudiation of Receipt Required	
Intelligible Check Required	
Number of Retries	
<b>Responder's Side</b>	
Responding Role	
Responding Business Action Name	
Time to Acknowledge Receipt	
Time to Acknowledge Processing	

Authorization Required	
Non Repudiation Required	
Non Repudiation of Receipt Required	
Intelligible Check Required	
<b>Business Information Envelopes</b>	
<b>Requesting Information Envelope</b>	
Name	
Are Contents Confidential?	
Is the Envelope Tamperproof?	
Authentication Required?	
<b>Responding Information Envelope (add more Responding Information Envelopes if different response documents are possible)</b>	
Name	
Resulting Business Entity State (including transition condition)	
Are Contents Confidential?	
Is the Envelope Tamperproof?	
Authentication Required?	

794

795 **5.2.2.6 Example (informative)**

796

797  
798

**Figure 28 Business Transaction Use Case Example: Register Customer (including the optional realize relationship to the business transaction placed beneath)**

799

800

801

**Figure 29 Business Transaction Example: Register Customer**

802

803

**Figure 30 Business Transaction Use Case Example: Request For Quote (the optional realize relationship is not shown)**

804

805

**Figure 31 Business Transaction Example: Request For Quote**

806

807

**Figure 32 Business Transaction Use Case Example: Place Order (the optional realize relationship is not shown)**

808

809

810

**Figure 33 Business Transaction Example: Place Order**

811

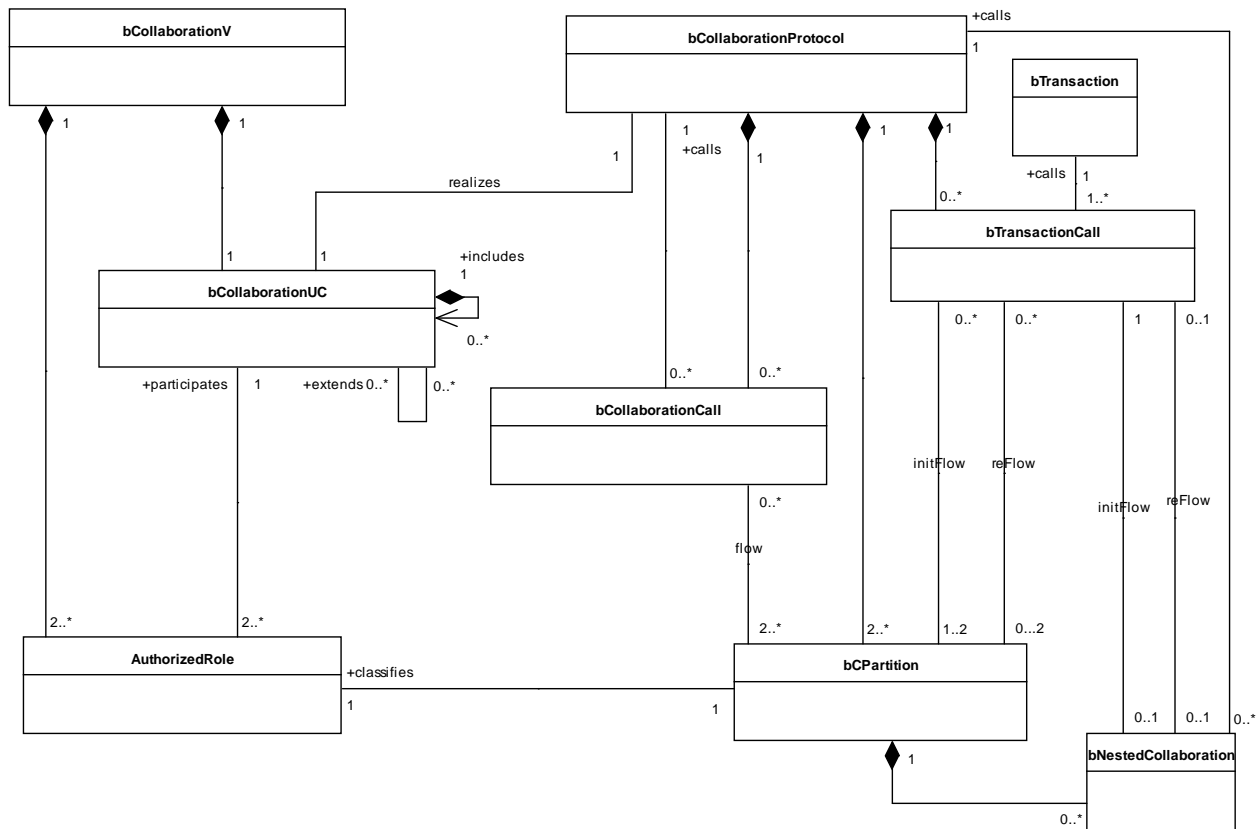
## 812 5.2.3 Business Collaboration View

### 813 5.2.3.1 Abbreviations of Stereotypes

Stereotype Abbreviation	Full Stereotype Name
bCollaborationV	BusinessCollaborationView
AuthorizedRole	AuthorizedRole
bCollaborationUC	BusinessCollaborationUseCase
bCollaborationProtocol	BusinessCollaborationProtocol
bCPartition	BusinessCollaborationPartition
bTransactionCall	BusinessTransactionCall
bCollaborationCall	BusinessCollaborationCall
bNestedCollaboration	NestedBusinessCollaboration

814

815 **5.2.3.2 Conceptual Description (informative)**



816

817 **Figure 34 Business Collaboration View - Conceptual Overview**

818 A *BusinessCollaborationView* is used to define the business choreography of exactly one business  
 819 collaboration. This business choreography is specified by the concept of a *BusinessCollaborationProtocol*.  
 820 The requirements of a *BusinessCollaborationProtocol* are captured by a *BusinessCollaborationUseCase*.

821 Each *BusinessCollaborationUseCase* and its corresponding *BusinessCollaborationProtocol* are defined in their  
 822 own business collaboration view package. Accordingly, the *BusinessCollaborationView* is composed of  
 823 exactly one *BusinessCollaborationUseCase* and one *BusinessCollaborationProtocol*.

824

825 **5.2.3.2.1 Business Collaboration Use Case Diagram**

826 At least two authorized roles participate in a business collaboration use case. These authorized roles must be  
 827 defined in the same business collaboration view package as the corresponding business collaboration use  
 828 case. Accordingly, a *BusinessCollaborationView* is composed of two or more *AuthorizedRoles*. This means, if a  
 829 certain role (e.g. buyer, seller, etc.) participates in multiple business collaborations, it requires a different  
 830 authorized role for each business collaboration use case. Each authorized role of the same role (i.e., with the  
 831 same name) is in a different namespace of a corresponding business collaboration view. Therefore, an  
 832 authorized role participates in only one business collaboration use case – it is the one in the same business  
 833 collaboration view. Accordingly, *BusinessCollaborationUseCase* and *AuthorizedRole* are related by a 1 to  
 834 (2..n) association. This association is defined as a *participates* association. It is important to note, that the  
 835 same authorized role is not associated twice to the same business collaboration use case.

836 A business collaboration use case may include additional business collaboration use cases. A business  
837 collaboration use case may optionally have multiple parent business collaboration use cases. Hence,  
838 *BusinessCollaborationUseCase* has a unary (0..n) to (0..n) include-composition. A business collaboration use  
839 case may include multiple business transaction use cases. A business transaction use case must be included  
840 in at least one business collaboration use case. Consequently, an (1..n) to (0..n) aggregation between  
841 *BusinessCollaborationUseCase* and *BusinessTransactionUseCase* exists. It is important that a business  
842 collaboration use case includes at minimum one use case – no matter whether this is a business  
843 collaboration use case or a business transaction use case. A hierarchy of business collaboration use cases  
844 built by include-compositions must not include any cycles. A business transaction uses case cannot be  
845 further decomposed by an include-association.

846 A business collaboration use case may be extended by additional business collaboration use cases. A  
847 business collaboration use case may optionally extend multiple business collaboration use cases. Hence,  
848 *BusinessCollaborationUseCase* has a unary (0..n) to (0..n) extend-association.

#### 849 5.2.3.2.2 Business Collaboration Protocol Diagram

850 A business choreography view is used to define the business choreography of exactly one business  
851 collaboration. The *BusinessCollaborationProtocol* follows exactly the requirements defined by the  
852 corresponding *BusinessCollaborationUseCase*. The business collaboration protocol that describes the  
853 business collaboration use case is defined as a child beneath. Accordingly, each  
854 *BusinessCollaborationUseCase* has exactly one *BusinessCollaborationProtocol* beneath. A business  
855 collaboration protocol is a “composite” UML *Activity*. The diagram of a business collaboration protocol is  
856 described by a flow of UML *Actions*.

857 A business collaboration protocol defines the collaborative business process between two or more  
858 authorized roles. The business collaboration protocol must have a *BusinessCollaborationPartiton* for each of  
859 the authorized roles defined in the *BusinessCollaborationView*. Hence, a *BusinessCollaborationProtocol* is  
860 composed of two or more *BusinessCollaborationPartitions*. Each *BusinessCollaborationPartition* relates to  
861 one and only one *AuthorizedRole* defined in the *BusinessCollaborationView*. Each *AuthorizedRole* in the  
862 *BusinessCollaborationView* is assigned to exactly one *BusinessCollaborationPartition*. It follows, that the  
863 every *BusinessCollaborationPartition* of a *BusinessCollaborationProtocol* must be assigned to different  
864 authorized roles.

865 The collaborative actions of a business collaboration protocol are business collaboration calls and/or  
866 business transaction calls. Hence, a *BusinessCollaborationProtocol* is composed of zero to many  
867 *BusinessCollaborationCalls* and of zero to many *BusinessTransactionCalls*. However, at least one business  
868 collaboration call or at least one Business Transaction Call must be present in a business collaboration  
869 protocol. Transitions defining the flow among the business collaboration activities and/or business  
870 transaction activities may be guarded by the states of business entities.

871 A business collaboration call is characterized by the fact that it is executed by calling a business collaboration  
872 protocol. This calling behavior is accomplished by classifying the behavior of the business collaboration call  
873 by the desired business collaboration protocol. Not every business collaboration protocol is a called by a  
874 business collaboration call. A business collaboration protocol may be called by multiple business  
875 collaboration calls. Thus, the behavioral classifying relationship between *BusinessCollabortionAction* and  
876 *BusinessCollaborationProtocol* is (0..n) to 1.

877 A Business Transaction Call is characterized by the fact that it is executed by calling a business transaction.  
878 This calling behavior is accomplished by classifying the behavior of the Business Transaction Call by the  
879 desired business transaction. Since the business transaction is a concept of the business transaction view it is  
880 described in more detail above. Each business transaction must be at least once used to refine a Business  
881 Transaction Call. A business transaction may be called by many Business Transaction Calls. Hence, the  
882 behavioral classifying relationship between *BusinessTransactionCall* and *BusinessTransaction* is (1..n) to 1.

883 In many scenarios, there is a requirement for a nested business collaboration within the scope of execution  
884 of a given Business Transaction Call. In other words, before a responding authorized role can send a  
885 response as required by the Business Transaction Call that calls a two-way business transaction, the  
886 responding authorized role has to first participate in a business collaboration with other business partners.  
887 UMM supports this scenario by introducing the concept of a *NestedBusinessCollaboration*. Like the business  
888 collaboration call, the *NestedBusinessCollaboration* is characterized by the fact that it is executed by calling  
889 another business collaboration protocol. This calling behavior is accomplished by classifying the behavior of  
890 the *NestedBusinessCollaboration* by the desired business collaboration protocol. Not every business  
891 collaboration protocol is called by a *NestedBusinessCollaboration*. A business collaboration protocol may  
892 be called by multiple *NestedBusinessCollaborations*. Thus, the behavioral classifying relationship between  
893 *NestedBusinessCollaboration* and *BusinessCollaborationProtocol* is (0..n) to 1. However, unlike the business  
894 collaboration call, the *NestedBusinessCollaboration* must reside within a business collaboration partition  
895 representing the responding authorized role of a given Business Transaction Call. Accordingly, not every  
896 business collaboration partition representing a Business Transaction Call responding role includes a  
897 *NestedBusinessCollaboration*. Thus, there is a 1 to (0..n) composition between a  
898 *BusinessCollaborationPartiton* and a *NestedBusinessCollaboration*.

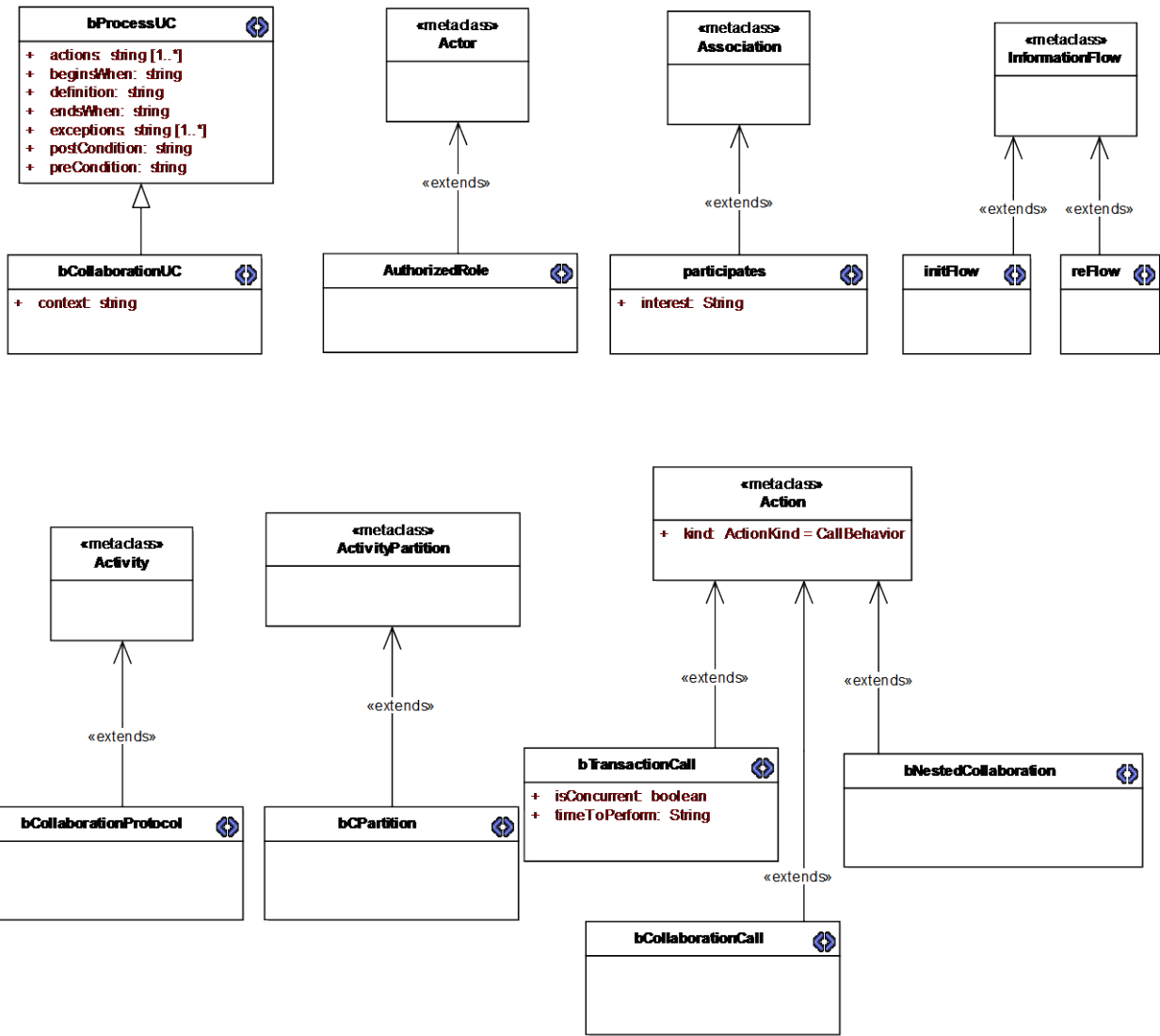
899 In UMM 2.0, role mapping between business collaboration authorized roles and either called business  
900 transaction authorized roles or business collaboration protocol authorized roles is defined in the business  
901 collaboration protocol and no longer by the business collaboration use case. This role mapping is  
902 accomplished by information flows and specializations of information flows, i.e. *InitiatingFlow* and  
903 *RespondingFlow*, between either business collaboration partitions or nested collaborations and either  
904 business collaboration calls or Business Transaction Calls. Using the approach also enhances the business  
905 collaboration protocol by graphically illustrating the relationships between authorized roles and the  
906 choreography of actions within a business collaboration protocol. This mapping approach is defined by the  
907 following cases:

908 1. *From business collaboration partitions to business collaboration calls*. The UML 2.0 Information flow  
909 is used. The source of Information flow must be the business collaboration partition and the target  
910 must be the business collaboration call. This means that the authorized role in this business  
911 collaboration protocol is participating in the called business collaboration protocol. Therefore the  
912 calling business collaboration protocol must have at least the same number of business collaboration  
913 partitions as the number of business collaboration partitions in the called business collaboration  
914 protocol. There are two cases for defining a role mapping:

- 915 • The authorized role name of the calling business collaboration protocol maps to an  
916 authorized role in the called business collaboration protocol which has exactly the same  
917 name. In this case, the information flow association explicitly defines the role mapping  
918 between these two roles.



- 919
- 920
- 921
- 922
- The authorized role name of the calling business collaboration protocol maps to an authorized role in the called business collaboration protocol which has a different name. In this case, the information flow association must be classified by the authorized role of the called business collaboration protocol.
- 923
- 924
- 925
- 926
- 927
- 928
- 929
- 930
2. *From business collaboration partitions to Business Transaction Calls.* In this case two specializations of the Information flow are used, *InitiatingFlow* and *RespondingFlow*. In all cases an authorized role of a business collaboration protocol initiates a Business Transaction Call, which calls a business transaction, which also has an initiating authorized role. To provide this role mapping the business collaboration partition classified by an authorized role is the source of the *InitiatingFlow* and the Business Transaction Call is the target. Likewise for two-way business transactions, a business collaboration partition is the source of the *RespondingFlow* and the Business Transaction Call is the target.
- 931
- 932
- 933
- 934
- 935
- 936
- 937
- 938
- 939
3. *From Business Transaction Calls to business collaboration partitions.* In this case two specializations of the Information Flow are used, *InitiatingFlow* and *RespondingFlow*. In all cases an authorized role of a business collaboration protocol responds to a Business Transaction Call which calls a business transaction which also has a responding authorized role. To provide this role mapping the Business Transaction Call is the source of an *InitiatingFlow* and a business collaboration partition is the target. Therefore the authorized role that classifies the business collaboration partition maps to the responding business transaction authorized role. Likewise, for two-way business transactions, a Business Transaction Call is the source of a *RespondingFlow* and a business collaboration partition is the target.
- 940
- 941
- 942
- 943
- 944
- 945
4. *From Business Transaction Calls to nested business collaborations.* In this case only one specialization of the Information Flow is used, *InitiatingFlow*. The source of the *InitiatingFlow* is a Business Transaction Call and the target is a nested business collaboration. This means that the responding authorized role of the business transaction initiates the business collaboration protocol called by the nested business collaboration and therefore maps to initiating authorized role of the called business collaboration protocol.
- 946
- 947
- 948
- 949
- 950
- 951
5. *From nested business collaborations to Business Transaction Calls.* In this case only one specialization of the Information Flow is used, *RespondingFlow* and only applies in the case of two-way business transactions. The *RespondingFlow* indicates that the called business collaboration has completed and the responding authorized role can now return the response envelope to the initiating authorized role. The *NestedBusinessCollaboration* is the source of the *RespondingFlow* and a Business Transaction Call is the target.



953  
954 **Figure 35 Business Collaboration View - Abstract Syntax**  
955

Stereotype	<b>bCollaborationUC (BusinessCollaborationUseCase)</b>	
Base Class	UseCase	
Parent	bProcessUC (Business Process Use Case)	
Description	A business collaboration use case describes in detail the requirements on the collaboration between two or more involved partners. Business partner types take part in a business collaboration use case by playing an authorized role in it. A business collaboration use case can be broken down into further business collaboration use cases and business transaction use cases. A business collaboration use case may extend another business collaboration use case.	
Tag Definition	<b>context</b>	
	Type	String

	<b>Multiplicity</b>	1
	<b>Description</b>	Describes the context (e.g., geo-political, industry, product, official constraints,...) of the given business collaboration
<b>Inherited tagged values:</b> - definition - beginsWhen - preCondition - endsWhen - postCondition - exceptions - actions		

956

<b>Stereotype</b>	<b>AuthorizedRole</b>
<b>Base Class</b>	Actor
<b>Parent</b>	N/A
<b>Description</b>	Already defined before in previous sub-section
<b>Tag Definition</b>	No tagged values.

957

<b>Stereotype</b>	<b>bCollaborationProtocol (BusinessCollaborationProtocol)</b>
<b>Base Class</b>	Activity
<b>Parent</b>	N/A
<b>Description</b>	A business collaboration protocol choreographs Business Transaction Calls and/or business collaboration calls. At least one action of either one must be present. A business collaboration protocol is a long running transaction that does not meet the atomic principle of transactions. It should be used in cases where transaction rollback is inappropriate.
<b>Tag Definition</b>	<b>No Tagged Values</b>

958

<b>Stereotype</b>	<b>bCPartition (BusinessCollaborationPartition)</b>
<b>Base Class</b>	ActivityPartition
<b>Parent</b>	N/A
<b>Description</b>	<p>A business collaboration partition is used to define an area of responsibility. The business collaboration partition is always classified by an authorized role defined as a participant in the corresponding business collaboration use case.</p> <p>A business collaboration partition may be empty. It is not empty in the special case of a nested collaboration. A nested collaboration must be placed within the business collaboration partition of the authorized role which is the responding authorized role in the triggering Business Transaction Call and which will initiate the nested collaboration.</p>

<b>Tag Definition</b>	No tagged values.
-----------------------	-------------------

<b>Stereotype</b>	<b>bTransactionAction (BusinessTransactionCall)</b>	
<b>Base Class</b>	CallBehaviorAction (Action with call behaviour action kind)	
<b>Parent</b>	N/A	
<b>Description</b>	A Business Transaction Call is an action within a business collaboration protocol. This action is refined by a using the call behaviour to classify the behaviour of this Business Transaction Call by one and only one business transaction. The Business Transaction Call executes the called business transaction. The Business Transaction Call can be executed more than once at the same time if the “isConcurrent” property is true.	
<b>Tag Definition</b>	<b>timeToPerform</b>	
	<b>Type</b>	<b>String (which must conform to a value of the W3C duration data type)</b>
	<b>Multiplicity</b>	1
	<b>Description</b>	A Business Transaction Call has to be executed within a specific duration. The initiating partner must send a failure notification to a responding partner on timeout. A responding partner simple terminates its activity. The time to perform is the maximum duration between the moment the requesting authorized role initiates the Business Transaction Call, i.e. sending the requesting business information envelope, and the moment the requesting authorized role receives a substantive response. The substantive response is the responding business information envelope if there is any. In case not, it is the acknowledgement of processing, if any. If not it is the acknowledgement of receipt, if any.
	<b>isConcurrent</b>	
	<b>Type</b>	Boolean
	<b>Multiplicity</b>	1
<b>Description</b>	If the Business Transaction Call is concurrent then more than one Business Transaction Call of the same underlying business transaction can be open at one time in executing the same business collaboration with the same business partner type. If the Business Transaction Call is not concurrent then only one Business Transaction Call of the same underlying business transaction can be open at one time.	

<b>Stereotype</b>	<b>bCollaborationAction (BusinessCollaborationCall)</b>	
<b>Base Class</b>	CallBehaviorAction (Action with call behaviour action kind)	
<b>Parent</b>	N/A	
<b>Description</b>	A business collaboration call is an action within a business collaboration protocol. This business collaboration call is refined by using the call behaviour to classify the behaviour of this business collaboration call by one and only one business collaboration protocol. The business choreography action executes the called business collaboration protocol exactly once. It follows, that business collaboration protocols might be recursively nested.	
<b>Tag Definition</b>	<b>No Tagged Values</b>	

<b>Stereotype</b> <b>NestedBCollaboration (NestedBusinessCollaboration)</b>	
<b>Base Class</b>	Action with call behaviour action kind (CallBehaviorAction)
<b>Parent</b>	N/A
<b>Description</b>	A nested business collaboration represents the case where the responding authorized role of a Business Transaction Call after receiving a requesting information envelope which is represented as an initFlow (InitiatingFlow) must carry out an additional business collaboration protocol with other business partners before responding to the initiating authorized role of a given Business Transaction Call indicated by a reFlow (RespondingFlow). This nested business collaboration is refined by using the call behaviour to classify the behaviour of this nested business collaboration by one and only one business collaboration protocol. The nested collaboration executes the called business collaboration protocol exactly once.
<b>Tag Definition</b>	No tagged values.

<b>Stereotype</b> <b>initFlow (InitiatingFlow)</b>	
<b>Base Class</b>	Information Flow
<b>Parent</b>	N/A
<b>Description</b>	<p>The initiating flow represents the following two cases:</p> <ul style="list-style-type: none"> <li>• The initiating flow of information that triggers the execution of a Business Transaction Call. The source of the initiating flow is the business collaboration partition that is classified by the authorized role initiating the Business Transaction Call and the target is the Business Transaction Call. In this case the initiating flow provides the authorized role mapping between the initiating role of the Business Transaction Call and the initiating authorized role of the business transaction that is called by this Business Transaction Call.</li> <li>• The initiating flow of information that triggers an execution on the responder's side. The source of this initiating flow is the Business Transaction Call and the target is the business collaboration partition which is classified by the authorized role responding in the Business Transaction Call. In this case the initiating flow provides the authorized role mapping between the responding role of the Business Transaction Call and the responding authorized role of the business transaction that is called by this Business Transaction Call. In the special case that the initiating flow triggers a nested business collaboration, the target of the initiating flow is not the business collaboration partition, but the nested business collaboration residing within this business collaboration partition.</li> </ul>
<b>Tag Definition</b>	No tagged values.

<b>Stereotype</b> <b>reFlow (RespondingFlow)</b>	
<b>Base Class</b>	Information Flow
<b>Parent</b>	N/A
<b>Description</b>	<p>The responding flow in case of two-way transactions represents the following two cases:</p> <ul style="list-style-type: none"> <li>• The responding flow of information that completes the execution of a Business Transaction Call. The source of the responding flow is the business collaboration partition that is classified by the</li> </ul>

	<p>authorized role responding in the Business Transaction Call and the target is the Business Transaction Call. In the special case that the responding flow is started after a nested business collaboration has completed, the source of the responding flow is not the business collaboration partition, but the nested business collaboration residing within this business collaboration partition.</p> <ul style="list-style-type: none"> <li>The responding flow of information that completes the Business Transaction Call on the initiator's side. The source of this initiating flow is the Business Transaction Call and the target is the business collaboration partition which is classified by the authorized role initiating the Business Transaction Call.</li> </ul>
<b>Tag Definition</b>	No tagged values.

964

#### 5.2.3.4 Constraints (normative)

965

C.58. A *BusinessCollaborationView* MUST contain exactly one *BusinessCollaborationUseCase*.

966

C.59. A *BusinessCollaborationView* MUST contain two to many *AuthorizedRoles*.

967

C.60. A *BusinessCollaborationUseCase* MUST have two to many *participates* associations to *AuthorizedRoles* contained in the same *BusinessCollaborationView*.

968

969

C.61. Each *AuthorizedRole* contained in the *BusinessCollaborationView* MUST have exactly one *participates* association to the *BusinessCollaborationUseCase* included in the same *BusinessCollaborationView*.

970

971

972

C.62. A *BusinessCollaborationUseCase* MUST have one to many *include* relationships to another *BusinessCollaborationUseCase* or to a *BusinessTransactionUseCase*.

973

974

C.63. Exactly one *BusinessCollaborationProtocol* MUST be placed beneath each *BusinessCollaborationUseCase*. This relationship MAY also be visualized by a realize relationship from the *BusinessCollaborationProtocol* to the *BusinessCollaborationUseCase*.

975

976

C.64. A *BusinessCollaborationProtocol* MUST contain one to many *BusinessTransactionCalls* and/or *BusinessCollaborationCall*.

977

978

C.65. Each *BusinessTransactionCall* MUST call exactly one *BusinessTransaction*

979

C.66. Each *BusinessTransaction* called by a *BusinessTransactionCall* MUST be placed beneath a *BusinessTransactionUseCase* which is included in the *BusinessCollaborationUseCase* that covers the corresponding *BusinessCollaborationProtocol*.

980

981

982

C.67. Each *BusinessCollaborationProtocol* called by a *BusinessCollaborationCall* MUST be placed beneath a *BusinessCollaborationProtocolUseCase* which is included in the *BusinessCollaborationUseCase* that covers the corresponding *BusinessCollaborationProtocol*.

983

984

C.68. A *BusinessCollaborationProtocol* MUST contain two to many *BusinessCollaborationPartions*.

985

986

C.69. The number of *AuthorizedRoles* in the *BusinessCollaborationView* MUST match the number of *BusinessCollaborationPartitions* in the *BusinessCollaborationProtocol* which is placed beneath the *BusinessCollaborationUseCase* of the same *BusinessCollaborationView*.

987

988

C.70. Each *AuthorizedRole* in the *BusinessCollaborationView* MUST be assigned to a *BusinessCollaborationPartition* in the *BusinessCollaborationProtocol* which is placed beneath the *BusinessCollaborationUseCase* of the same *BusinessCollaborationView*.

989

990

C.71. Each *BusinessCollaborationPartition* MUST be classified by exactly one *AuthorizedRole* included in the same *BusinessCollaborationView* as the *BusinessCollaborationUseCase* covering the *BusinessCollaborationProtocol* containing this *BusinessCollaborationPartition*.

991

992

993

C.72. A *BusinessCollaborationPartition* MUST be either empty or contain one to many *NestedBusinessCollaborations*.

994

995

996

997

- 998 C.73. Each *BusinessTransactionCall* MUST be the target of exactly one *InitialFlow* which source  
999 MUST be a *BusinessCollaborationPartition*.
- 1000 C.74. Each *BusinessTransactionCall* MUST be the source of exactly one *InitialFlow* which target  
1001 MUST be either a *BusinessCollaborationPartition* or a *NestedBusinessCollaboration*.
- 1002 C.75. The *InitialFlow* sourcing from a *BusinessTransactionCall* and the *InitialFlow* targeting a  
1003 *BusinessTransactionCall* MUST NOT be targeting to / sourcing from the same  
1004 *BusinessCollaborationPartition*, nor targeting to a *NestedBusinessCollaboration* within the same  
1005 *BusinessCollaborationPartition*.
- 1006 C.76. If a *BusinessTransactionCall* calls a two-way *BusinessTransaction*, this  
1007 *BusinessTransactionCall* MUST be the source of exactly one *RespondingFlow* which target MUST be a  
1008 *BusinessCollaborationPartition*.
- 1009 C.77. If a *BusinessTransactionCall* calls a two-way *BusinessTransaction*, this  
1010 *BusinessTransactionCall* MUST be the target of exactly one *RespondingFlow* which source MUST be  
1011 either a *BusinessCollaborationPartition* or a *NestedBusinessCollaboration*.
- 1012 C.78. The *RespondingFlow* sourcing from a *BusinessTransactionCall* and the *RespondingFlow*  
1013 targeting a *BusinessTransactionCall* MUST NOT be targeting to /sourcing from the same  
1014 *BusinessCollaborationPartition*, nor targeting to a *NestedBusinessCollaboration* within the same  
1015 *BusinessCollaborationPartition*.
- 1016 C.79. If a *BusinessTransactionCall* calls a one-way *BusinessTransaction*, this  
1017 *BusinessTransactionCall* MUST NOT be the source of a *RespondingFlow* and MUST NOT be the target  
1018 of a *RespondingFlow*.
- 1019 C.80. The *RespondingFlow* targeting a *BusinessTransactionCall* must start from the  
1020 *BusinessCollaborationPartition* / *NestedBusinessCollaboration* which is the target of the *InitialFlow*  
1021 starting from the same *BusinessTransactionCall*.
- 1022 C.81. The *RespondingFlow* starting from a *BusinessTransactionCall* must target the  
1023 *BusinessCollaborationPartition* which is the source of the *InitialFlow* targeting to the same  
1024 *BusinessTransactionCall*.
- 1025 C.82. A *NestedBusinessCollaboration* MUST be the target of exactly one *InitialFlow*.
- 1026 C.83. A *NestedBusinessCollaboration* MAY be the source of a *RespondingFlow*, but MUST NOT be  
1027 the source of more than one *RespondingFlow*.
- 1028 C.84. A *BusinessCollaborationCall* MUST be the target of two to many *InformationFlows* (UML  
1029 standard: <<flow>>).
- 1030 C.85. A *BusinessCollaborationCall* MUST not be the source of an *InformationFlow*.
- 1031 C.86. A *BusinessCollaborationCall* MUST not be the source and MUST not be the target of an  
1032 *InitialFlow*.
- 1033 C.87. A *BusinessCollaborationCall* MUST not be the source and MUST not be the target of a  
1034 *RespondingFlow*.
- 1035 C.88. A *BusinessTransactionCall* MUST not be the source and MUST not be the target of an  
1036 *InformationFlow* (<<flow>>) that is neither stereotyped as *InitialFlow* nor as *RespondingFlow* nor is  
1037 of type <<flow>>.
- 1038 C.89. A *NestedBusinessCollaboration* MUST not be the source and MUST not be the target of an  
1039 *InformationFlow* that targets to / sources from a *BusinessCollaborationCall*.
- 1040 C.90. The number of *InformationFlows* targeting a *BusinessCollaborationCall* MUST match the  
1041 number of *BusinessCollaborationPartitions* contained in the *BusinessCollaborationProtocol* that is  
1042 called by this *BusinessCollaborationCall*.

1043 C.91. Either an AuthorizedRole classifying a *BusinessCollaborationPartition* that is the source of an  
 1044 *InformationFlow* (UML standard: <<flow>>) targeting a *BusinessCollaborationCall* MUST match an  
 1045 AuthorizedRole classifying a *BusinessCollaborationPartition* in the *BusinessCollaborationProtocol*  
 1046 that is called by this *BusinessCollaborationCall* or the *InformationFlow* must be classified by an  
 1047 AuthorizedRole classifying a *BusinessCollaborationPartition* in the *BusinessCollaborationProtocol*  
 1048 that is called by this *BusinessCollaborationCall*.

1049 **5.2.3.5 Worksheets**

<b>Form for Business Collaboration Use Case</b>	
<b>General</b>	
Name	
Description	
<b>Business Library Information</b>	
UniqueIdentifier	
BusinessTerm	
VersionIdentifier	
Status	
Owner	
Copyright	
Reference(s)	
<b>Participants</b>	
Participating Role	
Participating Role	
[add more participating roles in case of a multiparty collaboration]	
Is Included In (Name of parent Business Collaboration – if there is any)	
<b>Start/End Characteristics</b>	
Affected Business Entities	



Pre-condition	
Post-condition	
Begins When	
Ends When	
Exceptions	
<b>Included Business Transaction Use Cases (add more Business Transaction Use Cases if needed)</b>	
Business Transaction Use Case Name	
Business Transaction Use Case Name	
Business Transaction Use Case Name	
Business Transaction Use Case Name	

1050

<b>Form for Business Collaboration Protocol</b>	
<b>General</b>	
Name	
Description	
<b>Participants (copy from Business Collaboration Use Case Worksheet)</b>	
Participating Role	
Participating Role	
[add more participating roles if elicited in the Business Collaboration Use Case Worksheet]	
<b>Included Business Transaction Calls / Business Collaboration Calls</b>	
Business Transaction Call	
Name	

Preceding Action(s) including transition condition		
Initiating Role	[select one participating role from above]	
Reacting Role	[select one participating role from above]	
Business Transaction Call		
Name		
Preceding Action(s) including transition condition		
Initiating Role	[select one participating role from above]	
Reacting Role	[select one participating role from above]	
Business Transaction Call [add more if needed]		
Name		
Preceding Action(s) including transition condition		
Initiating Role	[select one participating role from above]	
Reacting Role	[select one participating role from above]	
Business Collaboration Call [delete if not required or add more if needed]		
Name		
Preceding Action(s) including transition condition		
Role Mapping	Role in this Business Collaboration	Role in the nested Business Collaboration
Role Mapping	Role in this Business Collaboration	Role in the nested Business Collaboration
[add more Role Mappings if required]		

1051

1052 **5.2.3.6 Normal Business Collaboration View Example (informative)**

1053  
1054 **Figure 36 Business Collaboration Use Case Example: Register Customer (including the optional realize relationships)**

1055  
1056 **Figure 37 Business Collaboration Protocol Example: Register Customer**  
1057

1058

1059

**Figure 38 Business Collaboration Use Case Example: Order From Quote (the optional realize relationships are not shown)**

1060

1061

**Figure 39 Business Collaboration Protocol Example: Order From Quote**

1062

#### ***5.2.3.7 Nested Business Collaboration View Example (informative)***

1063

In this example the Order from Quote example is modified to show an example of a Nested Collaboration.

1064

Before the Seller can either accept or reject the Buyer's order, the seller must confirm the order with his/her

1065

business partners. If this confirmation is unsuccessful, it follows that the Seller will respond by sending a

1066

negative response message (e.g. *OrderRejectEnvelope*) to the Buyer.

1067

1068

1069

**Figure 40 Business Collaboration Use Case Example: Order From Quote (the optional realize relationships are not shown)**

1070

1071

1072

**Figure 41 Business Collaboration Protocol Example: Order from Quote (with nested collaboration)**

1073

1074  
1075

**Figure 42 Business Collaboration Use Case Example: Confirm Order With Partners the optional realize relationships are not shown)**

1076

1077

**Figure 43 Business Collaboration Protocol Example: Confirm Order with Partners**

1078

## **5.2.4 Business Realization View**

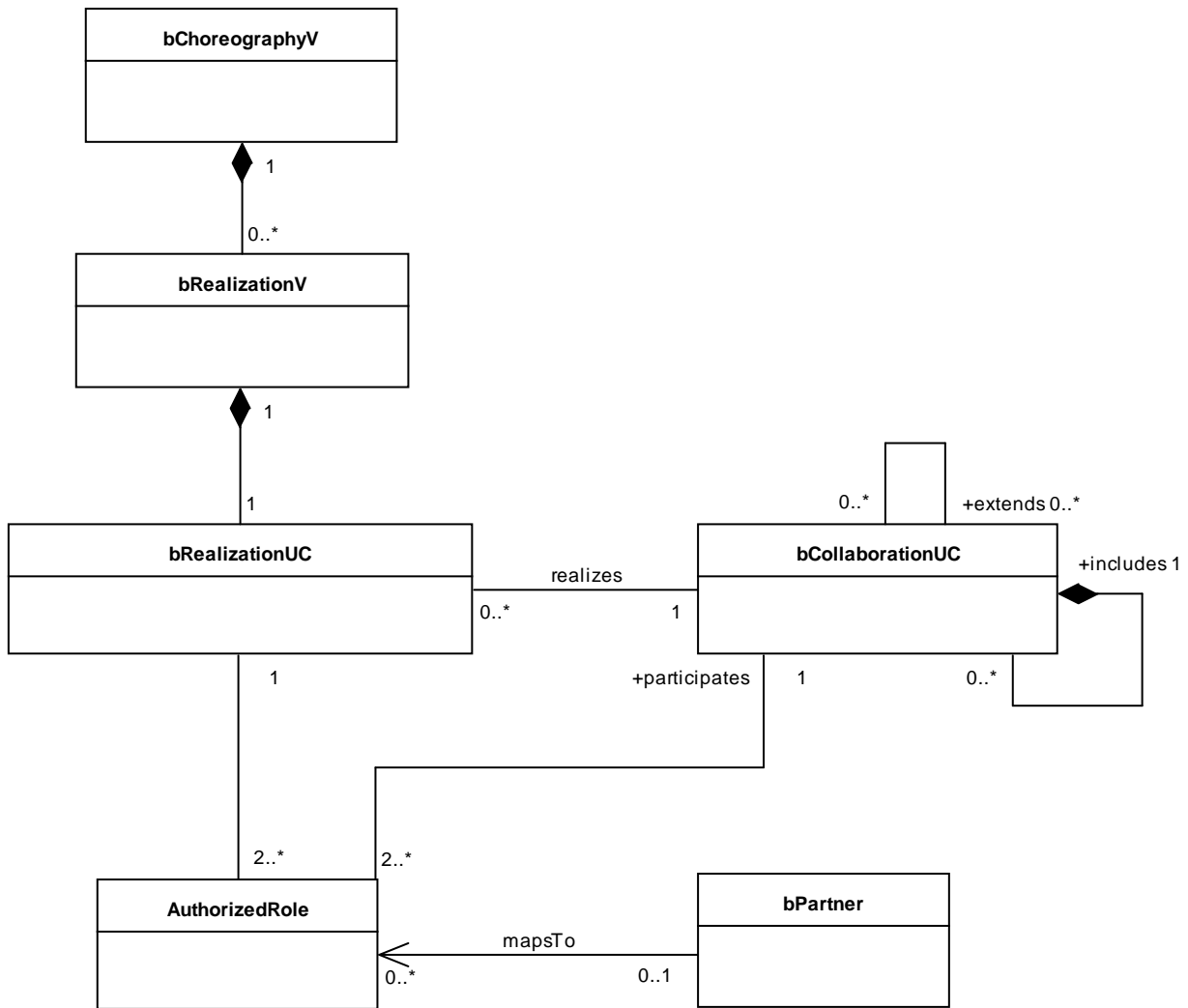
1079

### *5.2.4.1 Abbreviations and Stereotypes*

1080

Stereotype Abbreviation	Full Stereotype Name
bChoreographyV	BusinessChoreographyView
bRealizationV	BusinessRealizationView
bRealizationUC	BusinessRealizationUseCase
bCollaborationUC	BusinessCollaborationUseCase
bPartner	BusinessPartner
bProcessUC	BusinessProcessUseCase

1081



1083  
 1084 **Figure 44 BusinessRealizationView - Conceptual Overview**

1085 Business partners identified in the previous business requirements view must not directly be associated with  
 1086 business collaboration use cases and business transaction use cases.

1087 In order to specify that a specific set of business partners collaborate, we use the concept of a business  
 1088 realization use case. Each business realization use case is defined in its own business realization view.  
 1089 Accordingly, the *BusinessRealizationView* is composed of exactly one *BusinessRealizationUseCase*. A business  
 1090 realization use case realizes exactly one business collaboration use case. Each business collaboration use case  
 1091 may be realized by multiple business realization use cases. Not each business collaboration use case  
 1092 (e.g. one that is nested within another one) needs to have a corresponding business realization use case. As  
 1093 a consequence, the *realizes*-association between a *BusinessCollaborationUseCase* and  
 1094 *BusinessRealizationUseCase* is a 1 to (0..n).

1095 Two or more authorized roles participate in a business realization use case. These authorized roles (e.g.  
 1096 seller, payee) must be defined in the same business realization view package as the corresponding business  
 1097 realization use case. Accordingly a *BusinessRealizationView* is composed of two or more *AuthorizedRoles*.  
 1098 Usually, the names of the authorized roles participating in the business collaboration use case (e.g. payer



1099 and payee) will be the names of the authorized roles in the business realization use case (e.g. payer and  
1100 payee) realizing it. However, the authorized roles participating in the business collaboration use case and the  
1101 business realization use case will be defined in different namespaces – each in the package of the  
1102 corresponding view. In **Figure 46** the authorized role *Buyer* on the lower left hand side participates in the  
1103 business collaboration use case. It is defined in a different namespace than the *Buyer* participating in the  
1104 business realization use case.

1105 Similar to the business collaboration use case, the *BusinessRealizationUseCase* and *AuthorizedRole* are  
1106 related by an 1 to (2..n) association. Furthermore, the number of actors participating in a business  
1107 collaboration use case must be the same as the number of actors participating in the business realization use  
1108 case realizing it.

1109 In order to bind a business realization use case to the business partners executing it, business partners are  
1110 mapped to the authorized roles participating in the business realization use case. It is required that each  
1111 authorized role of a business realization use case (but not an authorized role in general) is target of exactly  
1112 one *mapsTo*-association from a business partner. A business partner may play multiple authorized roles of a  
1113 business realization use case. Consequently, there is a (0..1) to (0..n) *mapsTo*-association between  
1114 *BusinessPartner* and *AuthorizedRole*.

1115 **5.2.4.3 Stereotypes and Tag Definitions (normative)**

from BusinessDomainView

from BusinessPartnerView

1116  
1117 **Figure 45 BusinessRealizationView - Abstract Syntax**  
1118

<b>Stereotype</b>	<b>bRealizationUC (BusinessRealizationUseCase)</b>
<b>Base Class</b>	UseCase
<b>Parent</b>	N/A

<b>Description</b>	A business realization use case realizes a business collaboration use case between a specific set of business partners. The requirements of the business realization use case are the ones defined in the tags of the corresponding business collaboration use case. Thus, the business realization use case does not include any tag definitions for capturing requirements.
<b>Tag Definition</b>	No tagged values

1119

<b>Stereotype</b>	<b>AuthorizedRole (AuthorizedRole)</b>
<b>Base Class</b>	Actor
<b>Parent</b>	N/A
<b>Description</b>	Already defined before in previous sub-section
<b>Tag Definition</b>	No tagged values.

1120

<b>Stereotype</b>	<b>mapsTo (mapsTo)</b>
<b>Base Class</b>	Dependency
<b>Parent</b>	N/A
<b>Description</b>	A mapsTo dependency represents (1) the fact, that a business partner plays a certain authorized role in a business realization use case and (2) the fact, that an authorized role of a source business collaboration use case takes on a certain authorized role in a target business transaction use case or business collaboration use case.
<b>Tag Definition</b>	No tagged values.

1121

#### 1122 5.2.4.4 Constraints (normative)

- 1123 C.92. A *BusinessRealizationView* MUST contain exactly one *BusinessRealization*, two to many  
1124 *AuthorizedRoles*, and two to many *participates* associations.
- 1125 C.93. A *BusinessRealization* MUST be associated with two to many *AuthorizedRoles* via  
1126 stereotyped binary *participates* associations.
- 1127 C.94. A *BusinessRealization* MUST be the source of exactly one realization dependency to a  
1128 *BusinessCollaborationUseCase*.
- 1129 C.95. A *BusinessRealization* MUST NOT be the source or target of an *include* or *extends*  
1130 association.
- 1131 C.96. All dependencies from/to an *AuthorizedRole* must be stereotyped as *mapsTo*.
- 1132 C.97. An *AuthorizedRole*, which participates in a *BusinessRealization*, must be the target of exactly  
1133 one *mapsTo* dependency starting from a *BusinessPartner*. Furthermore the *AuthorizedRole*, which  
1134 participates in the *BusinessRealization* must be the source of exactly one *mapsTo* dependency  
1135 targeting an *AuthorizedRole* participating in a *BusinessCollaborationUseCase*.
- 1136 C.98. *AuthorizedRoles* in a *BusinessRealizationView* must have a unique name within the scope of  
1137 the package, they are located in

1138 C.99. The number of *AuthorizedRoles* participating in a *BusinessCollaborationUseCase* MUST  
1139 match the number of *AuthorizedRoles* participating in the *BusinessRealization* realizing this  
1140 *BusinessCollaborationUseCase*  
1141

1142 **5.2.4.5 Example (informative)**

1143  
1144 **Figure 46 BusinessRealizationView - Example: Realization of the OrderFromQuote Collaboration between Purchasing**  
1145 **Organization and SellingOrganization**  
1146

1147 **5.3 Business Information View**

1148 **5.3.1 Abbreviations of Stereotypes**

Stereotype Abbreviation	Full Stereotype Name
bInformationV	BusinessInformationView
bInformation	BusinessInformation
InfEnvelope	Information Envelope

1149

1150 **5.3.2 Conceptual Description (informative)**

from base module

1151

1152 **Figure 47 BusinessInformationView - Conceptual Overview**

1153 A *BusinessInformationView* is a container of artifacts that describe the information exchanged in a  
1154 *BusinessTransaction*. As previously mentioned; *RequestingInformationPin* and  
1155 *RespondingInformationPin* are classified by an *InformationEnvelope* which is a subclass of a  
1156 *BusinessInformation*. A *BusinessInformation* serves as an abstract container for all of the information  
1157 exchanged between the *RequestingAction* and the *RespondingAction* or vice versa, respectively. The  
1158 stereotypes *BusinessInformation* and *InformationEnvelope* is part of the UMM base module and  
1159 imported into the UMM foundation module.

1160 The current UMM foundation module does not mandate a specific business information modeling  
1161 approach. All methodologies and rules to build quality class diagrams can be used in order to model the  
1162 exchanged information, as long as the root element of the data structure generalizes the  
1163 *InformationEnvelope* class being part of the UMM base module.

1164 However, UMM strongly suggests using UN/CEFACT's Core Components and Core Components Message  
1165 Assembly artifacts to model the business information. Because Core Components are syntax  
1166 independent and stereotyped, the usage of the UML Profile for Core Components is suggested within  
1167 the *BusinessInformationView*.

1168

1169 **5.3.3 Stereotypes and Tag Definitions (normative)**

from base module

from base module

1170  
1171

1172 **Figure 48 BusinessInformationView - Abstract Syntax**

1173

Stereotype <b>bInformation (BusinessInformation)</b>	
<b>Base Class</b>	Class
<b>Parent</b>	N/A
<b>Description</b>	A <i>BusinessInformation</i> realizes abstract business document information that is exchanged between authorized roles performing activities in a business transaction. Since a <i>BusinessInformation</i> is defined as abstract it cannot be used directly in order to set the type of exchanged information in a <i>BusinessInformation</i> . Instead the concept of an <i>InformationEnvelope</i> is used.

1174

Stereotype <b>InfEnvelope (InformationEnvelope)</b>	
<b>Base Class</b>	Class
<b>Parent</b>	BusinessInformation
<b>Description</b>	An <i>InformationEnvelope</i> is a subtype of a <i>BusinessInformation</i> and represents a concrete business message which is exchanged in a UMM business transaction. Any business document artifacts are connected to an <i>InformationEnvelope</i> using associations.

1175

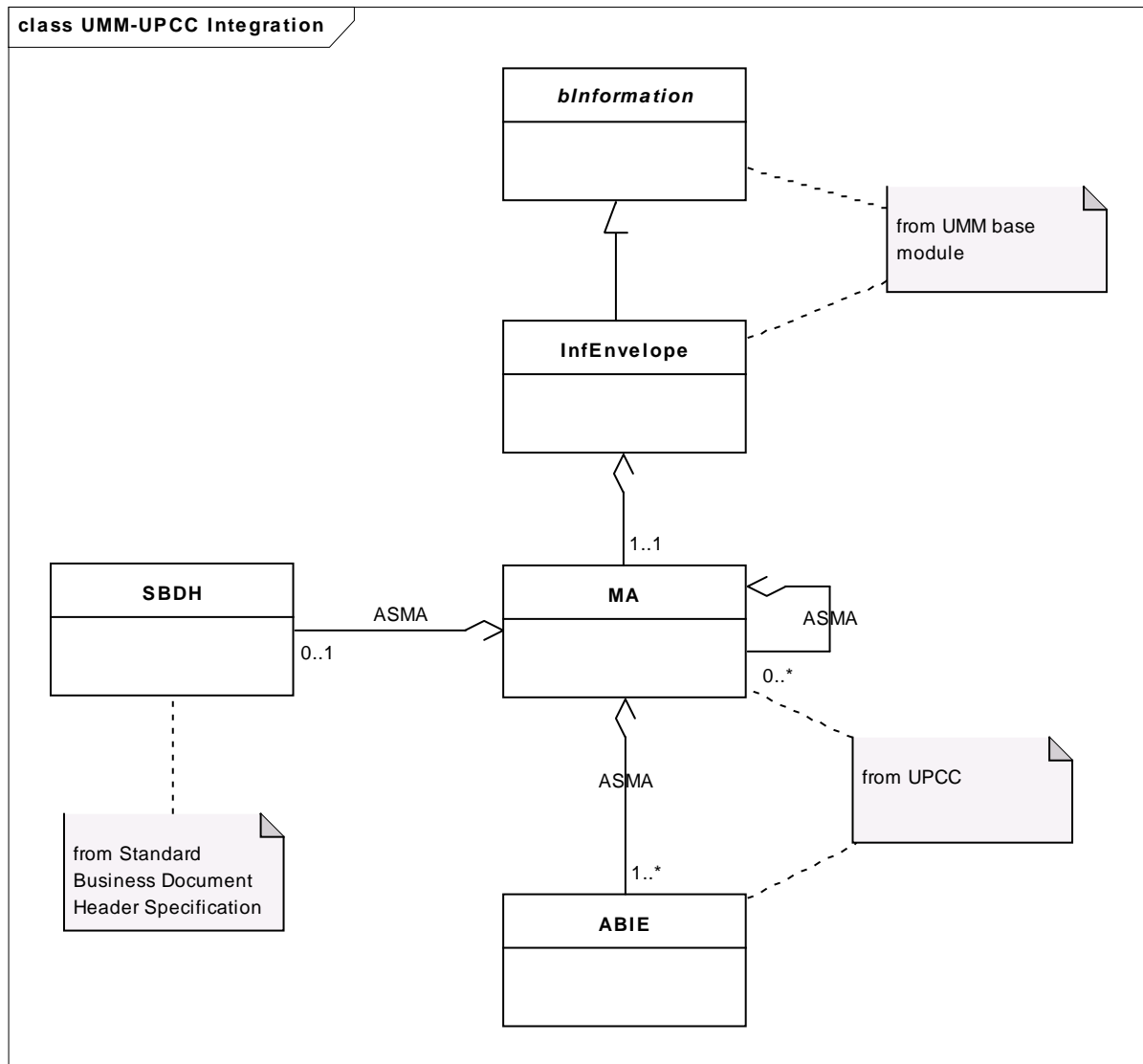
1176

1177 **5.3.4 Constraints (normative)**

1178 C.100. A *BusinessInformationView* MUST contain one to many *InformationEnvelopes* or  
 1179 subtypes thereof defined in any other extension/specialization module. Furthermore, it MAY  
 1180 contain any other document modeling artifacts.

1181 **5.3.5 Example using UPCC (UML Profile for Core Components) (informative)**

1182 The following example shows how to model the information exchanged in a *BusinessTransaction* using  
 1183 the UML Profile for Core Components (UPCC). **Figure 49** shows how UMM and UPCC are related to each  
 1184 other on the meta-level.



1185  
 1186 **Figure 49 Conceptual example for using UPCC artifacts to model business information**

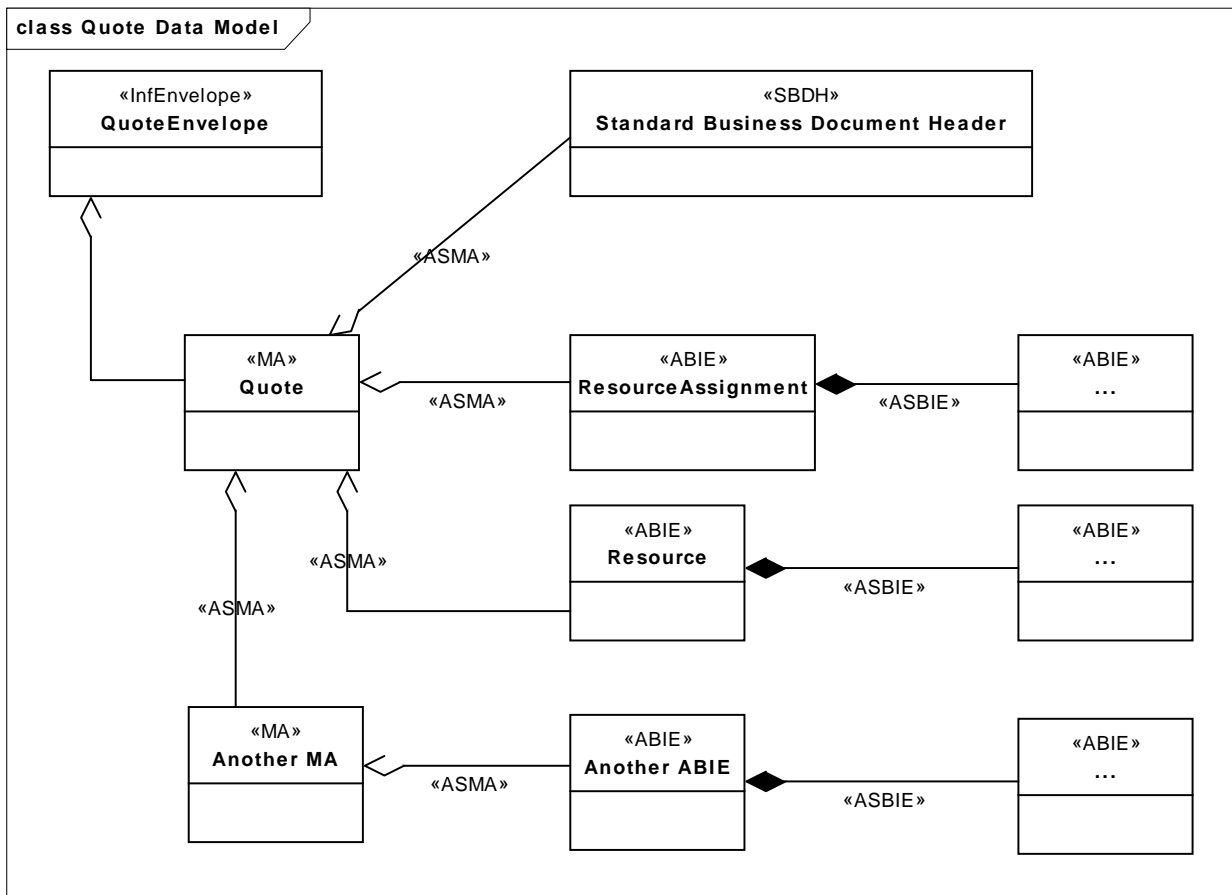
1187 An *information envelope* has exactly one message assembly (MA) which serves as the root element of  
 1188 the business document. The root message assembly (MA) has an optional *standard business document*  
 1189 *header* (SBDH) which serves for identification purposes of technical sender and receiver, document type  
 1190 etc. A standard business document header is defined in the Standard Business Document Header  
 1191 specification of UN/CEFACT.

1192 The root *message assembly* is connected to the *information envelope* using a standard UML  
 1193 *aggregation*. *Message assemblies* are used to assemble different *aggregate business information*  
 1194 *entities* (ABIE) to a specific business document. *Association message assemblies* (ASMA) are used to  
 1195 connect different *message assemblies* to each other and to connect *aggregate business information*  
 1196 *entities* to *message assemblies*. Additionally, *association message assemblies* are used to connect an  
 1197 optional *standard business document header* to the root *message assembly*. ABIEs, MAs, and ASMAs are  
 1198 part of the UML Profile for Core Components (UPCC) standard. A comprehensive description of the

1199 different core component compliant UML modeling artifacts and their relationships is given in the UPCC  
1200 standard.

1201 **Figure 50** shows an example for a *QuoteEnvelope* modeled using concepts of the UML Profile for Core  
1202 Components Standard.

1203



1204

1205

**Figure 50 UPCC example**

1206 The information envelope *QuoteEnvelope* has exactly one root message assembly *Quote*. Attached to a  
1207 *Quote* there are a standard business document header and multiple association message assemblies  
1208 (ASMA), leading to aggregate business information entities (ABIE) and other message assemblies (MA).



1209

## 1210 I. Business Transaction Patterns

1211 A UMM business transaction follows one of six business transaction patterns. The business transaction  
1212 pattern defines the type of a legally binding interaction between two decision making applications as  
1213 defined in Open-edi. In the following, the six business transaction patterns in UMM are described in  
1214 detail:

### 1215 i. Commercial transaction (two-way):

1216

#### 1217 **Figure 51 Commercial Transaction Pattern**

1218 Represents the typical „offer and acceptance“ business interaction. A *commercial transaction* (Figure 51)  
1219 results in a residual obligation between two parties to fulfill the terms of a contract. In other words both  
1220 parties enter into a commitment to fulfill their part of the contract. An example would be the  
1221 submission of an order and receipt of a purchase order response. The *commercial transaction* pattern  
1222 constitutes that the responding party has to return an *acknowledgement of receipt* when receiving the  
1223 requesting *information envelope*. The time frame within the *acknowledgement of receipt* has to be sent  
1224 is specified by *time to acknowledge receipt* (of the *requesting business action*). If the document passes a  
1225 set of business rules and is handed over to the business application the responder has to send an  
1226 *acknowledgement of processing*. The corresponding timeframe is specified by the *time to acknowledge*  
1227 *processing* (of the *requesting business action*). Furthermore, the responding party has to return the  
1228 *responding information envelope* within the period defined by *time to respond* (of the *requesting*  
1229 *business action*). The requesting party has to re-initiate the *business transaction* in case the *time to*  
1230 *acknowledge receipt*, *time to acknowledge processing* or *time to respond* is exceeded. The number of  
1231 attempts is defined by the *retry count*. When the responding party answers with the responding

1232 *information envelope* the requestor has to issue an *acknowledgement of receipt* within the *time to*  
1233 *acknowledge receipt* (specified in the *responding business action*). If the responding *information*  
1234 *envelope* passes again the business rules (e.g. grammar validation, sequence validation...) the requestor  
1235 has to transmit an *acknowledgement of processing* to the responder. The allowed period is set by the  
1236 *time to acknowledge processing* of the *responding business action*. Both parties are required to  
1237 authorize themselves (authorization is required by both business actions) and have to the sign their  
1238 envelopes and business signals (as defined by *non repudiation required* and *non repudiation of receipt*  
1239 *required* of both business actions).

1240

1241 **ii. Query/Response (two-way):**

1242

1243

#### **Figure 52 Query/Response Pattern**

1244 This pattern (Figure 52 )describes the request of information that is available to the responder prior to  
1245 the request. This might be a fixed data set inside a database or any kind of static information (e.g. a  
1246 catalog). The requestor initiates the transaction by submitting the request within a requesting  
1247 *information envelope* to the responder. The responder has to provide the information within the period  
1248 specified by *time to respond*. The requestor has to re-initiate the transaction as defined by the *retry*  
1249 *count* if the responder is not answering within the given *time to respond*. No business signals and no  
1250 non-repudiation requirements are necessary in the *query/response* pattern.

1251 **iii.** Request/Response (two-way):

1252

1253

**Figure 53 Request/Response Pattern**

1254 A transaction follows the *request/response* pattern (Figure 53) if the requestor asks for information that  
1255 requires some business processing on the responder's side. This includes information that needs to be  
1256 dynamically assembled and hence cannot be returned immediately (i.e. non-static information). An  
1257 example would be the request for a product quote. The *request/response* pattern results in no residual  
1258 obligation between the two parties to fulfill the terms of a contract. Concerning the *request for quote*  
1259 example, this inquiry leads to no commitment of the requestor to buy the quoted product. Similarly the  
1260 responder does not pledge himself to have the quoted product available in case of a further order. The  
1261 *request/response* pattern specifies the exchange of a requesting and a responding *information envelope*.  
1262 Non-Repudiation requirements as well as requiring business signals are optional, but not recommended  
1263 using the *request/response* pattern. If either business signals or nonrepudiation are required, they  
1264 follow the same semantics as specified for the *commercial transaction* pattern.

1265 **iv.** Request/Confirm (two-way):

1266

1267

**Figure 54 Request/Confirm Pattern**

1268 This pattern (Figure 54) should be used if the requesting partner asks for information that requires only  
1269 confirmation in respect to previously agreed business contracts. An example might be the request of  
1270 status information. The requestor initiates the transaction by submitting the request document to  
1271 confirm to the responder. Business signals or non-repudiation are not required by the responder.  
1272 Anyway, the requestor re-initiates the transaction as defined by the *retry count* if the responder misses  
1273 answering within the *time to respond*. Regarding the responding *information envelope*, the responder  
1274 might require that the requestor sends an *acknowledgement of receipt* when he receives the  
1275 confirmation response (within the timeframe specified by *time to acknowledge receipt*). Furthermore,  
1276 the responder might require the requestor to authenticate himself and to guarantee the non-  
1277 repudiation of the *acknowledgement of receipt*.

1278 v. Information distribution (one-way):

1279

1280

1281

**Figure 55 Information Distribution Pattern**

1282 This pattern (**Figure 55**) represents an informal, unidirectional information transmission. An example  
1283 would be information about price discounts to customers. Neither business signals nor non-repudiation  
1284 or authorization requirements are allowed in the information distribution pattern. Since the receipt of  
1285 the distributed information is not guaranteed no *retry count* must be claimed.

1286 **vi.** Notification (one-way):

1287

1288

**Figure 56 Notification Pattern**

1289 The *notification* pattern (**Figure 56**) represents a formal, unidirectional sending of information. This  
1290 pattern is applied if the requesting side has to inform the responding side about an irreversible business  
1291 state. An example is the notification of a product shipment. Since the notification transmittal is a formal  
1292 action the requestor has to claim for an *acknowledgement of receipt* with the specified *time*  
1293 *to acknowledge receipt*. Furthermore, the non-repudiation of a receipt is required. If the reacting party is  
1294 not sending the business signal within the agreed *time to acknowledge receipt* the requesting party has  
1295 to reinitiate the transaction as specified by the *retry count*.

1296

1297

1298 **vii. Default assignments of tagged values**

1299 Furthermore each business document has to be checked for readability by the receiver as defined by the  
1300 value of *is intelligible check required* which is by default set to true for every document. Table 1 shows  
1301 the requirements on the responding party within the different transaction patterns. These requirements  
1302 are specified in the *requesting business action* (because the requestor demands the responder to fulfill  
1303 these requirements). Similarly Table 2 shows the requirements posed by the responding party to the  
1304 requesting party. We specify them using the *tagged values* of the *responding business action*, because  
1305 the responder demands them to be fulfilled by the requestor.

1306 **Default assignment of tagged values for a requesting business action**

1307 The following table (Table 1) shows the default assignment of *tagged values* for a *requesting business*  
1308 *action*. They denote the requirements on the responder in context of the six business transaction  
1309 patterns.

	Time to Acknowledge Receipt	Time to Acknowledge Processing	Time to Respond	Is Authorization Required	Is Non Repudiation Required	Is Non Repudiation of Receipt Required	Retry Count	Is Intelligible Check Required
Commercial Transaction	2h	6h	24h	TRUE	TRUE	TRUE	3	TRUE
Request/Confirm	NULL	NULL	24h	FALSE	FALSE	FALSE	3	TRUE
Request/Response	NULL	NULL	4h	FALSE	FALSE	FALSE	3	TRUE
Query/Response	NULL	NULL	4h	FALSE	FALSE	FALSE	3	TRUE
Notification	24h	NULL	NULL	FALSE	TRUE	TRUE	3	TRUE
Information Distribution	NULL	NULL	NULL	FALSE	FALSE	FALSE	0	TRUE

1310 **Table 1 Default assignment of tagged values for a requesting business action**

1311 **Default assignment of tagged values for a responding business action**

1312 The following table (Table 2) shows the default assignment of *tagged values* for a *responding business*  
1313 *action*. They denote the requirements on the requestor in context of the six business transaction  
1314 patterns.

	Time to Acknowledge Receipt	Time to Acknowledge Processing	Is Authorization Required	Is Non Repudiation Required	Is Non Repudiation of Receipt Required	Is Intelligible Check Required
Commercial Transaction	2h	6hr	TRUE	TRUE	TRUE	TRUE
Request/Confirm	2h	NULL	TRUE	FALSE	TRUE	TRUE
Request/Response	NULL	NULL	FALSE	FALSE	FALSE	TRUE
Query/Response	NULL	NULL	FALSE	FALSE	FALSE	TRUE
Notification	NULL	NULL	FALSE	FALSE	FALSE	TRUE
Information Distribution	NULL	NULL	FALSE	FALSE	FALSE	TRUE

**Table 2 Default assignment of tagged values for a responding business action**

1315

1316



1317

## 1318 II. OCL Constraints

1319

```
1320 -- Constraint 1
1321 -- A BusinessCollaborationModel MUST contain one to many
1322 BusinessChoreographyViews.
1323 context Package inv: self.isBCollModel()
1324 implies self.nestedPackage->exists(a|a.isBChoreographyV())
1325
```

```
1326 -- Constraint 2
1327 -- A BusinessCollaborationModel MUST contain one to many
1328 BusinessInformationViews.
1329 context Package inv: self.isBCollModel()
1330 implies self.nestedPackage->exists(a|a.isBInformationV())
1331
```

```
1332 -- Constraint 3
1333 -- A BusinessCollaborationModel MAY contain zero to many
1334 BusinessRequirementsViews
1335 context Package inv: self.isBCollModel()
1336 implies self.nestedPackage->select(a|a.isBRequirementsV())->
1337 size>=0
1338
```

```
1339 -- Constraint 4
1340 -- A BusinessRequirementsView, a BusinessChoreographyView and a
1341 BusinessInformationView
1342 -- MUST be directly located under a BusinessCollaborationModel
1343 context Package inv: (self.isBChoreographyV() or
1344 self.isBInformationV() or self.isBRequirementsV())
1345 implies self.hlpOwningPackage().isBCollModel()
1346
```

```
1347 -- Constraint 5
1348 --A BusinessRequirementsView MAY contain zero or one
1349 BusinessDomainViews.
1350 context Package inv: self.isBRequirementsV()
1351 implies self.nestedPackage->select(a|a.isBDomainV())->size <=1
1352
```

```
1353 -- Constraint 6
1354 -- A BusinessChoreographyView MAY contain zero or one
1355 BusinessPartnerViews.
1356 context Package inv: self.isBRequirementsV()
1357 implies self.nestedPackage->select(a | a.isBPartnerV())->size <=1
1358
```

```
1359 -- Constraint 7
1360 -- A BusinessRequirementsView MAY contain zero to many
1361 BusinessEntityViews.
1362 context Package inv: self.isBRequirementsV()
```

```
1363 implies self.nestedPackage->select( a | a.isBEntityV()->size >= 0
1364
```

```
1365 -- Constraint 8
1366 -- A BusinessDomainView, a BusinessPartnerView, and a
1367 BusinessEntityView
1368 -- MUST be located directly under a BusinessRequirementsView
1369 context Package inv:
1370     self.isBDomainV() or self.isBPartnerV() or self.isBEntityV()
1371     implies self.hlpOwningPackage().isBRequirementsV()
1372
```

```
1373 -- Constraint 9
1374 -- A BusinessDomainView MUST include one to many BusinessAreas.
1375 context Package inv: self.isBDomainV()
1376     implies self.nestedPackage->exists( a | a.isBArea() )
1377
```

```
1378 -- Constraint 10
1379 -- A BusinessArea MUST include one to many BusinessAreas or one to
1380 -- many ProcessAreas or one to many BusinessProcessUseCases.
1381 context Package inv: self.isBArea()
1382     implies ( self.nestedPackage->exists( a | a.isBArea()
1383     or a.isProcessArea() ) or ( self.ownedElement ->
1384     exists( b | b.oclAsType(UseCase).isBProcessUseCase() ) ) )
1385
```

```
1386 -- Constraint 11
1387 -- A ProcessArea MUST contain one to many other ProcessAreas or one to
1388 -- many BusinessProcessUseCases
1389 context Package inv: self.isProcessArea()
1390     implies (self.nestedPackage -> exists( a | a.isProcessArea() )) or
1391     (self.ownedElement ->
1392     exists( b | b.oclAsType(UseCase).isBProcessUseCase() ))
1393
```

```
1394 -- Constraint 12
1395 --A BusinessProcessUseCase MUST be associated with one to many
1396 BusinessPartners using the participates relationship
1397 context UseCase inv: self.isBProcessUseCase() and
1398     not(self.isBCollaborationUC())
1399     and not(self.isBTransactionUC())
1400     implies self.owner.ownedElement->exists(a|
1401     a.oclAsType(Association).isParticipates()
1402     and a.oclAsType(Association).ownedEnd.type->
1403     exists(t|t.oclAsType(Actor).isBPartner())
1404     and a.oclAsType(Association).ownedEnd.type->
1405     exists(t|t.oclAsType(UseCase)=self))
1406
```

```
1407 -- Constraint 13
1408 -- A BusinessProcessUseCase may be associated with zero to many
1409 Stakeholders using the isOfInterestedTo relationship
1410 context UseCase inv: self.isBProcessUseCase()
1411     implies if self.owner.ownedElement->
1412     exists(a|a.oclAsType(Actor).isStakeholder())
```

```
1413 then self.oclasType(UseCase).clientDependency->
1414 exists(p|p.oclasType(Dependency).isOfInterestTo()
1415 and p.client->exists(t|t.oclasType(Actor).isStakeholder() ))
1416 else true endif
```

1417

```
1418 -- Constraint 14
1419 -- A BusinessProcessUseCase SHOULD be refined by zero to many
1420 BusinessProcesses. These relationships MAY also be visualized by
1421 realize relationships from each of the owned BusinessProcesses to
1422 the owning BusinessProcessUseCase
1423 context UseCase inv: self.isBProcessUseCase()
1424 implies self.ownedElement->select( process |
1425 process.oclasType(Activity).isBProcess()->size())>=0
```

1426

```
1427 -- Constraint 15
1428 -- A BusinessProcess MUST be modeled as a child of a
1429 BusinessProcessUseCase
1430 context Activity inv: self.isBProcess()
1431 implies self.owner.oclasType(UseCase).isBProcessUseCase()
```

1432

```
1433 Constraint 16 refers to a Diagram and is therefore not represented in
1434 OCL
```

1435

```
1436 -- Constraint 17
1437 -- A BusinessProcess MAY contain zero to many ActivityPartitions
1438 context Activity inv: self.isBProcess()
1439 implies self.ownedElement->select( partitions |
1440 partitions.oclasType(ActivityPartition))->size())>=0
```

1441

```
1442 -- Constraint 18
1443 -- A BusinessProcess, which has no ActivityPartitions, MUST contain
1444 one or more BusinessProcessActions and
1445 -- MAY contain zero to many InternalBusinessEntityStates and zero to
1446 many SharedBusinessEntityStates.
1447 context Activity inv: self.isBProcess()
1448 implies if self.ownedElement->
1449 select(b|b.oclasType(ActivityPartition).isActivityPartition()->
1450 size)=0
1451 then self.ownedElement-
1452 exists(bp|bp.oclasType(Action).isBProcessAction()) else true endif
```

1453

```
1454 -- Constraint 19
1455 -- An ActivityPartition being part of a BusinessProcess MUST contain
1456 one to many BusinessProcessActions and
1457 -- MAY contain zero to many InternalBusinessEntityStates.
1458 context ActivityPartition inv: self.isActivityPartition() and
1459 self.owner.oclasType(Activity).isBProcess()
1460 implies self.ownedElement->exists( act |
1461 act.oclasType(Action).isBProcessAction())
```

1462

```
1463 -- Constraint 20
1464 -- A SharedBusinessEntityState MUST NOT be located in an
1465 ActivityPartition. (They must be contained within
1466 -- the BusinessProcess even if this BusinessProcess contains
1467 ActivityPartitions.)
1468 context ObjectNode inv: self.oclAsType(ObjectNode).isBESharedState()
1469 and self.owner.oclAsType(Activity).isBProcess()
1470 implies
1471 not(self.owner.oclAsType(ActivityPartition).isActivityPartition())
1472
```

```
1473 -- Constraint 21
1474 -- A BusinessPartnerView MUST contain at least two to many
1475 BusinessPartners. If the BusinessPartnerView is hierarchically
1476 decomposed into subpackages these BusinessPartners MAY be contained
1477 in any of these subpackages.
1478 Missing on purpose
1479
```

```
1480 -- Constraint 22
1481 -- A BusinessPartnerView MAY contain zero to many Stakeholders
1482 Missing on purpose
1483
```

```
1484 -- Constraint 23
1485 -- A BusinessEntityView must contain one to many BusinessEntities
1486 context Package inv: self.isBEntityV()
1487 implies self.ownedElement->
1488 exists(a|a.oclAsType(Class).isBEntity())
1489
```

```
1490 Constraint 24 refers to a diagram and is therefore not represented in
1491 OCL
1492
```

```
1493 -- Constraint 25 (Since the first part of the constraint refers to a
1494 diagram, only the second part of the constraint is represented in
1495 OCL)
1496 -- A UML State Diagram describing the lifecycle of a BusinessEntity
1497 MUST contain one to many BusinessEntityStates. The parent of a
1498 BusinessEntityState MUST be a BusinessEntity
1499 context Class inv: self.isBEntityState()
1500 and self.owner.isBEntity()
1501
```

```
1502 -- Constraint 26
1503 -- A BusinessEntity MAY contain zero to many BusinessDataViews that
1504 describes its conceptual design
1505 context Package inv: self.isBEntityV()
1506 implies self.nestedPackage->select( package |
1507 package.oclAsType(Package).isBDataV()->size>=0
1508
```

```
1509 -- Constraint 27
1510 -- The parent of a BusinessDataView MUST be a BusinessEntityView
1511 context Package inv: self.isBDataV()
```

```
1512     implies self.owner.isBEntityV()  
1513
```

```
1514 -- Constraint 29  
1515 -- A BusinessDataView SHOULD contain one to many classes.  
1516 context Package inv: self.isBDataV()  
1517     implies self.ownedElement->select( elem |  
1518     elem.oclIsTypeOf(Class))->size()>=1  
1519
```

```
1520 -- Constraint 30  
1521 -- A BusinessChoreographyView MUST contain one to many  
1522     BusinessCollaborationViews  
1523 context Package inv: self.isBChoreographyV()  
1524     implies self.nestedPackage->exists(c|c.isBCollaborationV())  
1525
```

```
1526 -- Constraint 31  
1527 --A BusinessChoreographyView MUST contain one to many  
1528     BusinessTransactionViews  
1529 context Package inv: self.isBChoreographyV()  
1530     implies self.nestedPackage->exists(c|c.isBTransactionV())  
1531
```

```
1532 -- Constraint 32  
1533 -- A BusinessChoreographyView MAY contain zero to many  
1534     BusinessRealizationViews  
1535 context Package inv: self.isBChoreographyV()  
1536     implies self.nestedPackage->select( package |  
1537     package.oclAsType(Package).isBRealizationV())->size()>=0  
1538
```

```
1539 -- Constraint 33  
1540 -- A BusinessTransactionView, a BusinessCollaborationView, and a  
1541     BusinessRealizationView  
1542 -- MUST be directly located under a BusinessChoreographyView  
1543 context Package inv: self.isBTransactionV() or  
1544     self.isBCollaborationV() or self.isBRealizationV()  
1545     implies self.owner.isBChoreographyV()  
1546
```

```
1547 -- Constraint 34  
1548 -- A BusinessTransactionView MUST contain exactly one  
1549     BusinessTransactionUseCase, exactly two AuthorizedRoles,  
1550     and exactly two participates associations.  
1551 context Package inv: self.isBTransactionV()  
1552     implies self.ownedElement->  
1553     select(a|a.oclAsType(UseCase).isBTransactionUC())->size=1  
1554     and self.ownedElement->  
1555     select(b|b.oclAsType(Actor).isAuthorizedRole())->size=2  
1556     and self.ownedElement->select(c|c.oclIsTypeOf(Association)  
1557     and c.oclAsType(Association).isParticipates())->size=2  
1558     and self.ownedElement->size=5  
1559
```

```
1560 -- Constraint 35
```

```

1561 -- A BusinessTransactionUseCase MUST be associated with exactly two
1562     AuthorizedRoles via stereotyped binary participate associations.
1563 context UseCase inv: self.isBTransactionUC()
1564     implies self.owner.ownedElement->
1565     select(q|q.ocIsKindOf(Association))->
1566     forAll(a|a.ocIsType(Association).isParticipates()
1567     and a.ocIsType(Association).ownedEnd.type->
1568     forAll(t|t.ocIsType(Actor).isAuthorizedRole()
1569     or t.ocIsType(UseCase)=self))
1570

```

```

1571 -- Constraint 36
1572 -- A BusinessTransactionUseCase MUST NOT include further UseCases
1573 context UseCase inv: self.isBTransactionUC()
1574     implies self.include->size=0
1575

```

```

1576 -- Constraint 37
1577 -- A BusinessTransactionUseCase MUST be included in at least one
1578     BusinessCollaborationUseCase.
1579 context UseCase inv: self.isBTransactionUC()
1580     implies self.owner.owner.ocIsType(Package).nestedPackage
1581     ->select(collV|collV.isBCollaborationV())->
1582     exists(c|c.ownedElement
1583     ->exists(k|k.ocIsType(UseCase).isBCollaborationUC()
1584     and k.ocIsType(UseCase).include.addition
1585     ->exists(s|s.ocIsType(UseCase)=self)))
1586

```

```

1587 -- Constraint 38
1588 --A BusinessTransactionUseCase MUST NOT be source or target of an
1589     extend association.
1590 context UseCase inv: self.isBTransactionUC()
1591     implies self.extend->size=0 and UseCase.allInstances->
1592     forAll(u|u.ocIsType(UseCase).extend.extendedCase->
1593     forAll(t|t.ocIsType(UseCase).isNotBTransactionUC()))
1594

```

```

1595 -- Constraint 39
1596 -- The two AuthorizedRoles within a BusinessTransactionView MUST NOT
1597     be named identically
1598 context Actor inv: self.isAuthorizedRole() and
1599     self.owner.ocIsType(Package).isBTransactionV()
1600     implies self.owner.ownedElement->
1601     select(k|k.ocIsType(Actor).isAuthorizedRole())->
1602     collect(p|p.ocIsType(Actor).name)->asSet->size=2
1603

```

```

1604 -- Constraint 40
1605 -- A BusinessTransactionUseCase MUST be described by exactly one
1606     BusinessTransaction defined as a child element of this
1607     BusinessTransactionUseCase.
1608 context UseCase inv: self.isBTransactionUC()
1609     implies self.ownedElement->select( act |
1610     act.ocIsType(Activity).isBTransaction()) -> size = 1
1611

```

```

1612 -- Constraint 41
1613 -- A BusinessTransaction MUST have exactly two partitions.
1614 -- Each of them MUST be stereotyped as BusinessTransactionPartition.
1615 context Activity inv: self.isBTransaction()
1616     implies self.ownedElement->select( part |
1617         part.oclassType(ActivityPartition).isBTPartition())->size=2
1618

```

```

1619 -- Constraint 42
1620 -- One of the two BusinessTransactionPartitions MUST contain one
1621     RequestingBusinessAction and
1622 -- the other one MUST contain one RespondingBusinessAction.
1623 context Activity inv:
1624     let ReqAct : Action = self.ownedElement->
1625     select( act | act.oclassType(Action).isReqAction())->
1626     asSequence->first().oclassType(Action)
1627     in let ResAct : Action = self.ownedElement->
1628     select( act | act.oclassType(Action).isResAction())->
1629     asSequence->first().oclassType(Action)
1630     in let Part1 : ActivityPartition = self.ownedElement->
1631     select( part | part.oclassType(ActivityPartition).isBTPartition())
1632     ->asSequence->first().oclassType(ActivityPartition)
1633     in let Part2 : ActivityPartition = self.ownedElement->
1634     select( part | part.oclassType(ActivityPartition).isBTPartition())
1635     ->asSequence->last().oclassType(ActivityPartition)
1636     in self.isBTransaction()
1637     implies self.ownedElement->
1638     select( act | act.oclassType(Action).isReqAction())->size=1
1639     and self.ownedElement->
1640     select( act | act.oclassType(Action).isResAction())->size=1
1641     and ReqAct.inPartition->size=1 and ResAct.inPartition->size=1
1642     and ( (ReqAct.inPartition->asSequence
1643     ->first().oclassType(ActivityPartition)=Part1
1644     and ResAct.inPartition->asSequence
1645     ->first().oclassType(ActivityPartition)=Part2)
1646     or (ResAct.inPartition->asSequence
1647     ->first().oclassType(ActivityPartition)=Part1
1648     and ReqAct.inPartition->asSequence
1649     ->first().oclassType(ActivityPartition)=Part2 ) )
1650

```

```

1651 -- Constraint 43
1652 -- A BusinessTransactionPartition MUST have a classifier, which MUST
1653     be one of the associated
1654 -- AuthorizedRoles of the corresponding BusinessTransactionUseCase.
1655 context ActivityPartition inv:
1656     let authRoles : Sequence(Element) =
1657     self.owner.owner.owner.ownedElement->
1658     select(roles | roles.oclassType(Actor).isAuthorizedRole())
1659     ->asSequence in self.isBTPartition()
1660     implies self.name=authRoles->first().oclassType(Actor).name
1661     or self.name=authRoles->last().oclassType(Actor).name
1662

```

```

1663 -- Constraint 44
1664 -- The two BusinessTransactionPartitions MUST have different
1665     classifiers.

```

```

1666 context Activity inv: self.isBTransaction()
1667     implies not(self.ownedElement->
1668     select( part | part.oclAsType(ActivityPartition).isBTPartition())
1669     -> asSequence->first().oclAsType(ActivityPartition).name =
1670     self.ownedElement->
1671     select( part | part.oclAsType(ActivityPartition).isBTPartition())
1672     -> asSequence->last().oclAsType(ActivityPartition).name)
1673

```

```

1674 -- Constraint 45
1675 -- The BusinessTransactionPartition containing the
1676 -- RequestingBusinessAction MUST contain two or more FinalStates.
1677 -- Each of the FinalStates MAY have a SharedBusinessEntityState as
1678 -- predecessor.
1679 -- One of the FinalStates SHOULD reflect a ControlFailure - this
1680 -- FinalState SHOULD NOT have a
1681 -- predecesing SharedBusinessEntityState.
1682 context ActivityPartition inv: self.isBTPartition() and
1683     self.containedNode->exists( action |
1684     action.oclAsType(Action).isReqAction())
1685     implies self.containedNode->select( finNode |
1686     finNode.oclIsTypeOf(ActivityFinalNode))->size()>=2
1687

```

```

1688 -- Constraint 46
1689 -- A RequestingBusinessAction MUST embed exactly one
1690 -- RequestingInformationPin
1691 context Action inv: self.isReqAction()
1692     implies
1693     self.ownedElement->select( pin | pin.oclAsType(Pin).isReqInfPin())
1694     ->size=1
1695

```

```

1696 -- Constraint 47
1697 -- A RespondingBusinessAction MUST embed exactly one
1698 -- RequestingInformationPin
1699 context Action inv: self.isResAction()
1700     implies
1701     self.ownedElement->select( pin | pin.oclAsType(Pin).isReqInfPin())
1702     ->size=1
1703

```

```

1704 -- Constraint 48
1705 -- If the tagged value businessTransactionType of the
1706 -- BusinessTransaction is either Request/Response, Query/Response,
1707 -- Request/Confirm, or CommercialTransaction, then the
1708 -- RequestingBusinessAction must embed one to many
1709 -- RespondingInformationPins and the RespondingBusinessAction must
1710 -- embed one to many RespondingInformationPins.
1711 context Action inv: self.isBTransaction() and
1712     self.hlpMustHaveResInfPin() implies
1713     self.ownedElement-> select( action |
1714     action.oclAsType(Action).isReqAction())
1715     ->forall( actions | actions.ownedElement->
1716     exists( pin | pin.oclAsType(Pin).isResInfPin()))
1717

```



```

1718 -- Constraint 49
1719 -- If the tagged value businessTransactionType of the
1720 BusinessTransaction is either Notification or
1721 InformationDistribution, then both, the RequestingBusinessAction
1722 and the RespondingBusinessAction, MUST NOT embed a
1723 RespondingInformationPin
1724 context Action inv: self.isBTransaction() and
1725 self.hlpMustNotHaveResInfPin()
1726 implies
1727 not(self.ownedElement-> select( action |
1728 ction.oclaSType(Action).isReqAction()
1729 or action.oclaSType(Action).isResAction()->
1730 forAll( actions | actions.ownedElement->
1731 exists( pin | pin.oclaSType(Pin).isResInfPin()))))
1732

```

```

1733 -- Constraint 50
1734 -- A RequestingBusinessAction and a RespondingBusinessAction MUST
1735 embed same
1736 -- number of RespondingInformationPins.
1737 let ReqAct : Action = self.ownedElement->select( act |
1738 act.oclaSType(Action).isReqAction()->
1739 asSequence->first().oclaSType(Action) in
1740 let ResAct : Action = self.ownedElement->select( act |
1741 act.oclaSType(Action).isResAction()->
1742 asSequence->first().oclaSType(Action) in
1743 self.isBTransaction() implies
1744 ReqAct.ownedElement->select( resPin |
1745 resPin.oclaSType(Pin).isResInfPin()->size
1746 =
1747 ResAct.ownedElement->select(resPin |
1748 resPin.oclaSType(Pin).isResInfPin()->size
1749

```

```

1750 -- Constraint 51
1751 --
1752 The RequestingInformationPin of the RequestingBusinessAction MUST b
1753 e connected
1754 -- with the
1755 RequestingInformationPin of the RespondingBusinessAction using an o
1756 bject
1757 -- flow relationship leading from the RequestingBusinessAction
1758 -- to the RespondingBusinessAction.
1759 Missing on purpose
1760

```

```

1761 -- Constraint 52
1762 -- Each RespondingInformationPin of the RespondingBusinessAction MUST
1763 be
1764 -- connected with exactly one RespondingInformationPin of the
1765 RequestingBusinessAction
1766 -- using an object flow relationship leading from the
1767 RespondingBusinessAction
1768 -- to the RequestingBusinessAction
1769 Missing on purpose
1770

```

```

1771 -- Constraint 53
1772 -- If a BusinessTransactionPartition contains
1773   SharedBusinessEntityStates, each SharedBusinessEntityState
1774 -- MUST be the target of exactly one control flow relationship
1775   starting from the RequestingBusinessAction and
1776 -- MUST be the source of exactly one control flow relationship
1777   targeting a FinalState.
1778 context ActivityPartition inv: self.isBTPartition()
1779   implies self.owner.ownedElement->
1780   select( nodes | nodes.oclIsTypeOf(CentralBufferNode)
1781   and nodes.oclAsType(CentralBufferNode).isBESharedState() ) ->
1782   forAll( states | states.oclAsType(CentralBufferNode).incoming
1783   ->size()=1 and states.oclAsType(CentralBufferNode).incoming->
1784   forAll( income | income.oclAsType(ObjectFlow).source.
1785   oclAsType(Action).isReqAction()
1786   and states.oclAsType(CentralBufferNode).outgoing->
1787   forAll( outgo | outgo.oclAsType(ObjectFlow).target.
1788   oclIsTypeOf(ActivityFinalNode)
1789   and states.oclAsType(CentralBufferNode).outgoing->size()=1 ) )
1790

```

```

1791 -- Constraint 54
1792 -- Each FinalState MUST be the target of one to many control flow
1793   relationships starting
1794 -- from the RequestingBusinessAction or from a
1795   SharedBusinessEntityState.
1796 context ActivityFinalNode inv: self.oclIsTypeOf(ActivityFinalNode)
1797   and self.owner.oclAsType(Activity).isBTransaction()
1798   implies self.incoming->forAll( income |
1799   income.oclAsType(ControlFlow).source.
1800   oclAsType(Action).isReqAction() or
1801   income.oclAsType(ObjectFlow).source.oclAsType(CentralBufferNode).is
1802   BESharedState() )
1803

```

```

1804 -- Constraint 55
1805 -- Each RequestingInformationPin and each RespondingInformationPin
1806   MUST have a classifier, this classifier MUST be an
1807   InformationEnvelope or a subtype defined in an
1808   extension/specialization module.
1809
1810 Missing on purpose
1811

```

```

1812 -- Constraint 56
1813 -- Two RequestingInformationPins which are connected using an
1814 -- object flow MUST have the same classifier.
1815
1816 Missing on purpose
1817

```

```

1818 -- Constraint 57
1819 -- Two RespondingInformationPins which are connected using an
1820 -- object flow MUST have the same classifier.
1821
1822 Missing on purpose

```

1823

```
1824 -- Constraint 58
1825 -- A BusinessCollaborationView MUST contain exactly one
1826 BusinessCollaborationUseCase.
1827 context Package inv: self.isBCollaborationV()
1828     implies self.ownedElement
1829     ->select(k|k.oclasType(UseCase).isBCollaborationUC())->size=1
1830
```

```
1831 -- Constraint 59
1832 -- A BusinessCollaborationView MUST contain two to many
1833 AuthorizedRoles.
1834 context Package inv: self.isBCollaborationV()
1835     implies self.ownedElement
1836     ->select(k|k.oclasType(Actor).isAuthorizedRole())->size>=2
1837
```

```
1838 -- Constraint 60
1839 -- A BusinessCollaborationUseCase MUST have two to many participates
1840 associations
1841 -- to AuthorizedRoles contained in the same BusinessCollaborationView.
1842 context UseCase inv: self.isBCollaborationUC()
1843     implies self.owner.ownedElement
1844     ->select(c|c.oclasTypeOf(Association)
1845     and c.oclasType(Association).isParticipates()
1846     and c.oclasType(Association).ownedEnd.type
1847     ->exists(t|t.oclasType(Actor).isAuthorizedRole()))->size>=2
1848
```

```
1849 -- Constraint 61
1850 -- Each AuthorizedRole contained in the BusinessCollaborationView MUST
1851 have exactly
1852 -- one participates association to the BusinessCollaborationUseCase
1853 included in the same BusinessCollaborationView.
1854 context Actor inv: self.isAuthorizedRole() and
1855     self.owner.oclasType(Package).isBCollaborationV()
1856     implies self.owner.ownedElement->
1857     select(as|as.oclasTypeOf(Association))->
1858     select(ass| ass.oclasType(Association).ownedEnd.type->
1859     exists(end|end.oclasType(Actor)=self)
1860     and ass.oclasType(Association).ownedEnd.type->
1861     exists(oend|oend.oclasType(UseCase).isBCollaborationUC())) ->size=1
1862
```

```
1863 -- Constraint 62
1864 -- A BusinessCollaborationUseCase MUST have one to many include
1865 relationships to another BusinessCollaborationUseCase
1866 -- or to a BusinessTransactionUseCase.
1867 context UseCase inv: self.isBCollaborationUC()
1868     implies self.include.addition->size>0
1869     and self.include.addition->
1870     forAll(Uc |Uc.oclasType(UseCase).isBCollaborationUC()
1871     or Uc.oclasType(UseCase).isBTransactionUC())
1872
```

```
1873 -- Constraint 63
```

```
1874 -- Exactly one BusinessCollaborationProtocol MUST be placed beneath
1875     each BusinessCollaborationUseCase.
1876 context Activity inv: self.isBCollaborationProtocol()
1877     implies self.owner.oclAsType(UseCase).isBCollaborationUC()
1878
```

```
1879 -- Constraint 64
1880 -- A BusinessCollaborationProtocol MUST contain one to many
1881     BusinessTransactionCalls and/or BusinessCollaborationCall.
1882 context Activity inv: self.isBCollaborationProtocol()
1883     implies self.ownedElement->exists( actions |
1884     actions.oclAsType(CallBehaviorAction).isBTransactionAction()
1885     or actions.oclAsType(CallBehaviorAction).isBCollaborationAction())
1886
```

```
1887 -- Constraint 65
1888 -- Each BusinessTransactionCall MUST call exactly one
1889     BusinessTransaction
1890 context Action inv:
1891     self.oclAsType(CallBehaviorAction).isBTransactionAction()
1892     implies
1893     self.oclAsType(CallBehaviorAction).behavior.oclAsType(Activity).isB
1894     Transaction()
1895
```

```
1896 -- Constraint 66
1897 -- Each BusinessTransaction called by a BusinessTransactionCall MUST
1898     be placed beneath a
1899     BusinessTransactionUseCase which is included in the
1900     BusinessCollaborationUseCase that covers
1901     the corresponding BusinessCollaborationProtocol.
1902 context Activity inv: self.oclAsType(Activity).isBTransaction()
1903     and CallBehaviorAction.allInstances->
1904     exists( node | node.oclAsType(CallBehaviorAction).behavior.
1905     oclAsType(Activity)=self)
1906     implies self.owner.oclAsType(UseCase).isBTransactionUC()
1907     and UseCase.allInstances->
1908     select( uc | uc.oclAsType(UseCase).isBCollaborationUC()->
1909     exists( anInclude | anInclude.oclAsType(UseCase).include.addition
1910     -> exists( elem |
1911     elem.oclAsType(UseCase)=self.owner.oclAsType(UseCase))
1912     and anInclude.oclAsType(UseCase).ownedElement->exists( protocol |
1913     protocol.oclAsType(Activity).isBCollaborationProtocol() ) )
1914
```

```
1915 -- Constraint 67
1916 -- Each BusinessCollaborationProtocol called by a
1917     BusinessCollaborationCall MUST be placed beneath a
1918     BusinessCollaborationProtocolUseCase which is included in the
1919     BusinessCollaborationUseCase that covers the corresponding
1920     BusinessCollaborationProtocol.
1921 context CallBehaviorAction inv: let protocol : Activity =
1922     self.behavior.oclAsType(Activity) in self.isBCollaborationAction()
1923     implies protocol.owner.oclAsType(UseCase).isBCollaborationUC()
1924     and self.owner.owner.oclAsType(UseCase).isBCollaborationUC()
1925     and self.owner.owner.oclAsType(UseCase).include.addition->
```

```
1926     exists( inc |
1927     inc.oclAsType(UseCase)=protocol.owner.oclAsType(UseCase))
1928
```

```
1929 -- Constraint 68
1930 -- A BusinessCollaborationProtocol MUST contain two to many
1931     BusinessCollaborationPartions.
1932 context Activity inv: self.isBCollaborationProtocol()
1933     implies self.ownedElement->select( partitions |
1934     partitions.oclAsType(ActivityPartition).
1935     isBCollaborationPartition()->size())>=2
1936
```

```
1937 -- Constraint 69
1938 -- The number of AuthorizedRoles in the BusinessCollaborationView MUST
1939     match the number of BusinessCollaborationPartitions
1940 -- in the BusinessCollaborationProtocol which is placed beneath the
1941     BusinessCollaborationUseCase of the same BusinessCollaborationView.
1942 context Package inv:
1943     let authRoles : Set(Element) = self.ownedElement->
1944     select( role | role.oclAsType(Actor).isAuthorizedRole())
1945     in let collUC : Set(Element) = self.ownedElement->
1946     select( bCollUC | bCollUC.oclAsType(UseCase).isBCollaborationUC())
1947     in let protocol : Set(Element) = collUC.ownedElement
1948     ->select(prot|prot.oclAsType(Activity).isBCollaborationProtocol())
1949     ->asSet in let partitions : Set(Element) = protocol.ownedElement
1950     ->select(part|part.oclAsType(ActivityPartition).
1951     isBCollaborationPartition()->
1952     asSet in self.isBCollaborationV()
1953     implies authRoles->size() = partitions->size()
1954
```

```
1955 -- Constraint 70
1956 -- Each AuthorizedRole in the BusinessCollaborationView MUST be
1957     assigned to a BusinessCollaborationPartition
1958 -- in the BusinessCollaborationProtocol which is placed beneath the
1959     BusinessCollaborationUseCase
1960 -- of the same BusinessCollaborationView.
1961 context Actor inv:
1962     let bCollUC : Set(Element) = self.owner.ownedElement->
1963     select ( uc | uc.oclAsType(UseCase).isBCollaborationUC())
1964     in let bCollProtocol : Set(Element) = bCollUC.ownedElement->
1965     select ( protocol |
1966     protocol.oclAsType(Activity).isBCollaborationProtocol()->asSet
1967     in let bCollPartition : Set(Element) =
1968     bCollProtocol.ownedElement -> select( partition |
1969     partition.oclAsType(ActivityPartition).isBCollaborationPartition())
1970     ->asSet in self.isAuthorizedRole() and
1971     self.owner.oclAsType(Package).isBCollaborationV()
1972     implies bCollPartition.oclAsType(ActivityPartition).represents->
1973     select( part | part.oclAsType(Actor)=self.oclAsType(Actor))
1974     ->size())=1
1975
```

```
1976 -- Constraint 71
1977 -- Each BusinessCollaborationPartition MUST be classified by exactly
1978     one AuthorizedRole included in the same
```

```
1979 -- BusinessCollaborationView as the BusinessCollaborationUseCase
1980 covering the BusinessCollaborationProtocol
1981 -- containing this BusinessCollaborationPartition.
1982 context ActivityPartition inv: self.isBCollaborationPartition()
1983 implies self.hlpOwningPackage().ownedElement->
1984 exists( actor | actor.oclAsType(Actor).isAuthorizedRole()
1985 and actor.oclAsType(Actor)=self.represents.oclAsType(Actor) )
1986
```

```
1987 -- Constraint 72
1988 -- A BusinessCollaborationPartition MUST be either empty or contain
1989 one to many NestedBusinessCollaborations.
1990 context ActivityPartition inv: self.isBCollaborationPartition()
1991 implies self.owner.ownedElement->select( action |
1992 action.oclAsType(Action).inPartition->
1993 exists(element | element.oclAsType(ActivityPartition)=self))
1994 ->forall( elem | elem.oclAsType(Action).isBNestedCollaboration() )
1995 and (self.ownedElement->size()=0 or self.ownedElement->forall(
1996 elem | elem.oclAsType(Action).isBNestedCollaboration() ) )
1997
```

```
1998 -- Constraint 73
1999 -- Each BusinessTransactionCall MUST be the target of exactly one
2000 InitialFlow which source MUST be a BusinessCollaborationPartition.
2001 context CallBehaviorAction inv:
2002 self.oclAsType(CallBehaviorAction).isBTransactionAction()
2003 implies Dependency.allInstances->select( dependency |
2004 dependency.oclAsType(Dependency).supplier->
2005 exists( ends | ends.oclAsType(CallBehaviorAction)=self) and
2006 dependency.oclAsType(Dependency).isInitFlow()
2007 and dependency.oclAsType(Dependency).client->exists( end |
2008 end.oclAsType(ActivityPartition).isBCollaborationPartition() )->
2009 size()=1
2010
```

```
2011 -- Constraint 74
2012 -- Each BusinessTransactionCall MUST be the source of exactly one
2013 InitialFlow which target MUST be either a
2014 -- BusinessCollaborationPartition or a NestedBusinessCollaboration.
2015 context CallBehaviorAction inv:
2016 self.oclAsType(CallBehaviorAction).isBTransactionAction()
2017 and self.owner.oclAsType(Activity).isBCollaborationProtocol()
2018 implies Dependency.allInstances->select( dependency |
2019 dependency.oclAsType(Dependency).client->
2020 exists( ends | ends.oclAsType(CallBehaviorAction)=self)
2021 and dependency.oclAsType(Dependency).isInitFlow()
2022 and dependency.oclAsType(Dependency).supplier->exists(end |
2023 end.oclAsType(ActivityPartition).isBCollaborationPartition()
2024 or end.oclAsType(Action).isBNestedCollaboration() )->size()=1
2025
```

```
2026 -- Constraint 75
2027 -- The InitialFlow sourcing from a BusinessTransactionCall and the
2028 InitialFlow targeting a BusinessTransactionCall
2029 -- MUST NOT be targeting to / sourcing from the same
2030 BusinessCollaborationPartition, nor targeting to a
2031 NestedBusinessCollaboration
```

```

2032 -- within the same BusinessCollaborationPartition.
2033 context Dependency inv: self.isInitFlow()
2034     implies not(Dependency.allInstances->exists( dep |
2035 (dep.oclAsType(Dependency).supplier=self.client
2036 or ( dep.oclAsType(Dependency).supplier->forAll(end |
2037 end.oclAsType(Action).isBNestedCollaboration())
2038 and
2039 dep.oclAsType(Dependency).supplier.oclAsType(Action).inPartition
2040 ->includesAll(self.client.oclAsType(ActivityPartition)) ) )
2041 and dep.oclAsType(Dependency).client=self.supplier and
2042 dep.oclAsType(Dependency).isInitFlow() ) )
2043

```

```

2044 -- Constraint 76
2045 -- If a BusinessTransactionCall calls a two-way BusinessTransaction,
2046 this BusinessTransactionCall MUST be the source of exactly one
2047 RespondingFlow which target MUST be a
2048 BusinessCollaborationPartition.
2049 context CallBehaviorAction inv: self.isBTransactionAction() and
2050 self.behavior.oclAsType(Activity).isTwoWayBTransaction()
2051 implies self.owner.ownedElement->
2052 select( dep | dep.oclAsType(Dependency).isReFlow() and
2053 dep.oclAsType(Dependency).client->
2054 exists( end | end.oclAsType(CallBehaviorAction)=self)
2055 and dep.oclAsType(Dependency).supplier->
2056 exists( end |
2057 end.oclAsType(ActivityPartition).isBCollaborationPartition())
2058 ->size()=1
2059

```

```

2060 -- Constraint 77
2061 -- If a BusinessTransactionCall calls a two-way BusinessTransaction,
2062 this BusinessTransactionCall MUST be the target of exactly one
2063 RespondingFlow which source MUST be either a
2064 BusinessCollaborationPartition or a NestedBusinessCollaboration.
2065 context CallBehaviorAction inv: self.isBTransactionAction() and
2066 self.behavior.oclAsType(Activity).isTwoWayBTransaction()
2067 implies self.owner.ownedElement->
2068 select( dep | dep.oclAsType(Dependency).isReFlow()
2069 and dep.oclAsType(Dependency).supplier->
2070 exists( end | end.oclAsType(CallBehaviorAction)=self)
2071 and dep.oclAsType(Dependency).client->
2072 exists( end |
2073 end.oclAsType(ActivityPartition).isBCollaborationPartition()
2074 or end.oclAsType(Action).isBNestedCollaboration()))->size()=1
2075

```

```

2076 -- Constraint 78
2077 -- The RespondingFlow sourcing from a BusinessTransactionCall and the
2078 RespondingFlow targeting a BusinessTransactionCall
2079 -- MUST NOT be targeting to /sourcing from the same
2080 BusinessCollaborationPartition, nor targeting to a
2081 NestedBusinessCollaboration
2082 -- within the same BusinessCollaborationPartition.
2083 context Dependency inv: self.isReFlow()
2084     implies not(Dependency.allInstances->exists( dep |
2085 (dep.oclAsType(Dependency).supplier=self.client

```

```

2086     or dep.oclAsType(Dependency).supplier.owner.
2087     oclAsType(ActivityPartition)=self.client.
2088     oclAsType(ActivityPartition) )
2089     and ( dep.oclAsType(Dependency).client=self.supplier
2090     or dep.oclAsType(Dependency).supplier.owner.
2091     oclAsType(ActivityPartition)=self.supplier.
2092     oclAsType(ActivityPartition) )
2093     and dep.oclAsType(Dependency).isReFlow() ) )
2094

```

```

2095 -- Constraint 79
2096 -- If a BusinessTransactionCall calls a one-way BusinessTransaction,
2097 -- this BusinessTransactionCall MUST NOT be the source of a
2098 -- RespondingFlow and MUST NOT be the target of a RespondingFlow.
2099 context CallBehaviorAction inv: self.isBTransactionAction() and
2100 self.behavior.oclAsType(Activity).isOneWayBTransaction()
2101 implies not(self.owner.ownedElement->
2102 exists( dep | dep.oclAsType(Dependency).isReFlow() and (
2103 dep.oclAsType(Dependency).client->
2104 exists( end | end.oclAsType(CallBehaviorAction)=self or
2105 dep.oclAsType(Dependency).supplier->
2106 exists( end | end.oclAsType(CallBehaviorAction)=self )))
2107

```

```

2108 -- Constraint 80
2109 -- The RespondingFlow targeting a BusinessTransactionCall must start
2110 -- from the
2111 -- BusinessCollaborationPartition / NestedBusinessCollaboration which
2112 -- is the target of the InitialFlow starting
2113 -- from the same BusinessTransactionCall.
2114 context Dependency inv: self.isReFlow()
2115 and self.supplier->forall( end |
2116 end.oclAsType(CallBehaviorAction).isBTransactionAction())
2117 implies Dependency.allInstances->
2118 exists( flow | flow.oclAsType(Dependency).isInitFlow() and (
2119 flow.oclAsType(Dependency).supplier.oclAsType(ActivityPartition)=
2120 self.oclAsType(Dependency).client.oclAsType(ActivityPartition) or
2121 flow.oclAsType(Dependency).supplier.oclAsType(Action)=
2122 self.oclAsType(Dependency).client.oclAsType(Action) ) and
2123 flow.oclAsType(Dependency).client.oclAsType(CallBehaviorAction)=
2124 self.supplier.oclAsType(CallBehaviorAction) )
2125

```

```

2126 -- Constraint 81
2127 -- The RespondingFlow starting from a BusinessTransactionCall must
2128 -- target the BusinessCollaborationPartition which is
2129 -- the source of the InitialFlow targeting to the same
2130 -- BusinessTransactionCall.
2131 context Dependency inv: self.isReFlow()
2132 and self.oclAsType(Dependency).client->forall( end |
2133 end.oclAsType(CallBehaviorAction).isBTransactionAction())
2134 implies Dependency.allInstances->exists( flow |
2135 flow.oclAsType(Dependency).isInitFlow() and
2136 flow.oclAsType(Dependency).client.oclAsType(ActivityPartition)=
2137 self.oclAsType(Dependency).supplier.oclAsType(ActivityPartition)
2138 and flow.oclAsType(Dependency).supplier.
2139 oclAsType(CallBehaviorAction)=

```



```
2140 self.client.oclAsType(CallBehaviorAction) )
2141
```

```
2142 -- Constraint 82
2143 -- A NestedBusinessCollaboration MUST be the target of exactly one
2144 InitialFlow.
2145 context Action inv: self.isBNestedCollaboration()
2146     implies Dependency.allInstances->
2147     select( dep | dep.oclAsType(Dependency).supplier->
2148     exists(end | end.oclAsType(Action)=self.oclAsType(Action)))
2149     ->size()=1
2150
```

```
2151 -- Constraint 83
2152 -- A NestedBusinessCollaboration MAY be the source of a
2153 RespondingFlow,
2154 -- but MUST NOT be the source of more than one RespondingFlow.
2155 context Action inv:
2156     self.isBNestedCollaboration()
2157     implies Dependency.allInstances->
2158     select( dep | dep.oclAsType(Dependency).isReFlow()
2159     and dep.oclAsType(Dependency).client->exists(end |
2160     end.oclAsType(Action)=self.oclAsType(Action)))->size()<=1
2161
```

```
2162 -- Constraint 84
2163 -- A BusinessCollaborationCall MUST be the target of two to many
2164 InformationFlows
2165 context CallBehaviorAction inv: self.isBCollaborationAction()
2166     implies Dependency.allInstances->select( dep |
2167     dep.oclAsType(Dependency).supplier->
2168     exists( end | end.oclAsType(CallBehaviorAction)=self))->size()>=2
2169
```

```
2170 -- Constraint 85
2171 -- A BusinessCollaborationCall MUST not be the source of an
2172 InformationFlow.
2173 context CallBehaviorAction inv: self.isBCollaborationAction()
2174     implies not(Dependency.allInstances->exists( dep |
2175     dep.oclAsType(Dependency).client->
2176     exists( end | end.oclAsType(CallBehaviorAction)=self)))
2177
2178
```

```
2179 -- Constraint 86
2180 -- A BusinessCollaborationCall MUST not be the source and MUST not be
2181 the target of an InitialFlow.
2182 context CallBehaviorAction inv: self.isBCollaborationAction()
2183     implies not(Dependency.allInstances->exists( dep |
2184     dep.oclAsType(Dependency).isInitFlow()
2185     and (dep.oclAsType(Dependency).client->exists( end |
2186     end.oclAsType(CallBehaviorAction).isBCollaborationAction())
2187     or dep.oclAsType(Dependency).supplier->exists( end |
2188     end.oclAsType(CallBehaviorAction).isBCollaborationAction()))))
2189
```

```

2190 -- Constraint 87
2191 -- A BusinessCollaborationCall MUST not be the source and MUST not be
2192 the target of a RespondingFlow.
2193 context CallBehaviorAction inv: self.isBCollaborationAction()
2194     implies not(Dependency.allInstances->exists( dep |
2195     dep.oclcAsType(Dependency).isReFlow()
2196     and (dep.oclcAsType(Dependency).client->exists( end |
2197     end.oclcAsType(CallBehaviorAction).isBCollaborationAction()
2198     or dep.oclcAsType(Dependency).supplier->exists( end |
2199     end.oclcAsType(CallBehaviorAction).isBCollaborationAction()))))
2200

```

```

2201 -- Constraint 88
2202 -- A BusinessTransactionCall MUST not be the source and MUST not be
2203 the target of an InformationFlow that is neither
2204 -- stereotyped as InitialFlow nor as RespondingFlow.
2205 context CallBehaviorAction inv: self.isBTransactionAction()
2206     implies not( Dependency.allInstances-> exists( dep |
2207     dep.oclcAsType(Dependency).client->
2208     exists(end | end.oclcAsType(CallBehaviorAction)=self)
2209     and not(dep.oclcAsType(Dependency).isInitFlow()
2210     or dep.oclcAsType(Dependency).isReFlow() ) )
2211     and not( Dependency.allInstances-> exists( dep |
2212     dep.oclcAsType(Dependency).supplier->
2213     exists(end | end.oclcAsType(CallBehaviorAction)=self)
2214     and not(dep.oclcAsType(Dependency).isInitFlow()
2215     or dep.oclcAsType(Dependency).isReFlow() ) )
2216

```

```

2217 -- Constraint 89
2218 -- A NestedBusinessCollaboration MUST not be the source and MUST not
2219 be the target of an InformationFlow that
2220 -- targets to / sources from a BusinessCollaborationCall.
2221 context Action inv: self.oclcAsType(Action).isBNestedCollaboration()
2222     implies not(Dependency.allInstances->exists( dep |
2223     dep.oclcAsType(Dependency).supplier->exists(end |
2224     end.oclcAsType(Action)=self)
2225     and dep.oclcAsType(Dependency).client->exists(end |
2226     end.oclcAsType(CallBehaviorAction).isBCollaborationAction()))
2227     and not(Dependency.allInstances->exists( dep |
2228     dep.oclcAsType(Dependency).client->exists(end |
2229     end.oclcAsType(Action)=self)
2230     and dep.oclcAsType(Dependency).supplier->exists(end|
2231     end.oclcAsType(CallBehaviorAction).isBCollaborationAction()))))
2232

```

```

2233 -- Constraint 90
2234 -- The number of InformationFlows targeting a
2235 BusinessCollaborationCall MUST match the number of
2236 BusinessCollaborationPartitions
2237 -- contained in the BusinessCollaborationProtocol that is called by
2238 this BusinessCollaborationCall.
2239 context CallBehaviorAction inv: self.isBCollaborationAction()
2240     implies self.owner.ownedElement->select( dep |
2241     dep.oclcAsType(Dependency).supplier->
2242     exists( end | end.oclcAsType(CallBehaviorAction)=self ))->size() =
2243     self.behavior.oclcAsType(Activity).ownedElement->

```

```

2244     select( partitions |
2245     partitions.oclAsType(ActivityPartition).
2246     isBCollaborationPartition()->size()
2247

```

```

2248 -- Constraint 91
2249 -- Either an AuthorizedRole classifying a
2250 BusinessCollaborationPartition that is the source of an
2251 InformationFlow
2252 -- targeting a BusinessCollaborationCall MUST match an AuthorizedRole
2253 classifying a BusinessCollaborationPartition in the
2254 BusinessCollaborationProtocol that is called by this
2255 BusinessCollaborationCall or the InformationFlow must be classified
2256 -- by an AuthorizedRole classifying a BusinessCollaborationPartition
2257 in the BusinessCollaborationProtocol that is called
2258 -- by this BusinessCollaborationCall.
2259
2260 context ActivityPartition inv: self.isBCollaborationPartition()
2261     and self.represents.oclAsType(Actor).isAuthorizedRole()
2262     and Dependency.allInstances->exists( dep |
2263     dep.oclAsType(Dependency).client->exists( end |
2264     end.oclAsType(ActivityPartition)=self)
2265     and dep.oclAsType(Dependency).supplier-> exists( end |
2266     end.oclAsType(CallBehaviorAction).isBCollaborationAction()))
2267     implies Dependency.allInstances->select( dep |
2268     dep.oclAsType(Dependency).client->
2269     exists( end | end.oclAsType(ActivityPartition)=self)
2270     and dep.oclAsType(Dependency).supplier-> exists( end |
2271     end.oclAsType(CallBehaviorAction).isBCollaborationAction())) ->
2272     forAll(dependency | dependency.oclAsType(Dependency).supplier->
2273     exists(end|end.oclAsType(CallBehaviorAction).behavior.
2274     oclAsType(Activity).ownedElement->
2275     exists( partition | partition.oclAsType(ActivityPartition).
2276     isBCollaborationPartition()
2277     and partition.oclAsType(ActivityPartition).represents.
2278     oclAsType(Actor)=self.represents.oclAsType(Actor)))
2279

```

```

2280 -- Constraint 92
2281 -- A BusinessRealizationView MUST contain exactly one
2282 BusinessRealizationUC, two to many AuthorizedRoles,
2283 -- and two to many participates associations.
2284 context Package inv: let ownEl : Set(Element) = self.ownedElement in
2285     self.isBRealizationV()
2286     implies ownEl -> select ( ass |
2287     ass.oclAsType(Association).isParticipates()->size>=2
2288     and ownEl -> select ( bRealUC |
2289     bRealUC.oclAsType(UseCase).isBRealizationUC()
2290     and not(bRealUC.oclAsType(UseCase).isBCollaborationUC()))
2291     ->size()==1 and ownEl
2292     -> select ( roles |roles.oclAsType(Actor).isAuthorizedRole())
2293     ->size()>=2
2294

```

```

2295 -- Constraint 93
2296 -- A BusinessRealization MUST be associated with two to many
2297 AuthorizedRoles via stereotyped binary participates associations.

```

```

2298 context UseCase inv: self.isBRealizationUC() and
2299     not(self.isBCollaborationUC())
2300     implies self.owner.ownedElement->
2301     select(a| a.oclAsType(Association).isParticipates()
2302     and a.oclAsType(Association).ownedEnd.type
2303     ->exists(t|t.oclAsType(Actor).isAuthorizedRole())
2304     and a.oclAsType(Association).ownedEnd.type
2305     ->exists(t|t.oclAsType(UseCase)=self) )->size()>=2
2306

```

```

2307 -- Constraint 94
2308 -- A BusinessRealization MUST be the source of exactly one realization
2309 -- dependency to a BusinessCollaborationUseCase.
2310 context UseCase inv: self.isBRealizationUC()
2311     implies self.owner.ownedElement->exists( ass |
2312     ass.oclAsType(Realization).isRealizes()
2313     and ass.oclAsType(Realization).client->exists(uc |
2314     uc.oclAsType(UseCase)=self)
2315     and ass.oclAsType(Realization).supplier->exists(uc |
2316     uc.oclAsType(UseCase).isBCollaborationUC()))
2317

```

```

2318 -- Constraint 95
2319 -- A BusinessRealization MUST NOT be the source or target of an
2320 -- include or extends association.
2321 context UseCase inv: self.isBRealizationUC()
2322     implies self.include->size()=0
2323     and UseCase.allInstances
2324     ->forall(k|k.oclAsType(UseCase).include.addition
2325     ->forall(s|not(s.oclAsType(UseCase)=self)))
2326     and UseCase.allInstances
2327     ->forall(u|u.oclAsType(UseCase).extend.extendedCase
2328     ->forall(t|not(t.oclAsType(UseCase)=self)))
2329     and self.extend->size=0
2330

```

```

2331 -- Constraint 96
2332 -- All dependencies from/to an AuthorizedRole must be stereotyped as
2333 -- mapsTo.
2334 context Package inv: self.isBRealizationV()
2335     implies self.ownedElement->select( dep |
2336     dep.oclAsType(Dependency).client->
2337     exists( k | k.oclAsType(Actor).isAuthorizedRole() )
2338     and dep.oclAsType(Dependency).supplier->
2339     exists( k | k.oclAsType(Actor).isAuthorizedRole()))->
2340     forall(dep| dep.oclAsType(Dependency).isMapsTo())
2341

```

```

2342 -- Constraint 97
2343 -- An AuthorizedRole, which participates in a BusinessRealization,
2344 -- must be the target of exactly one mapsTo dependency
2345 -- starting from a BusinessPartner. Furthermore the AuthorizedRole,
2346 -- which participates in the BusinessRealization must
2347 -- be the source of exactly one mapsTo dependency targeting an
2348 -- AuthorizedRole participating in a BusinessCollaborationUseCase.
2349 context Actor inv: self.isAuthorizedRole()
2350     and self.owner.ownedElement ->

```

```

2351     exists( ass | ass.oclasType(Association).isParticipates()
2352     and ass.oclasType(Association).ownedEnd.type->exists( end |
2353     end.oclasType(Actor)=self)
2354     and ass.oclasType(Association).ownedEnd.type->exists( end |
2355     end.oclasType(UseCase).isBRealizationUC()
2356     and not(end.oclasType(UseCase).isBCollaborationUC()))
2357     implies self.owner.ownedElement->select( dep |
2358     dep.oclasType(Dependency).isMapsTo()
2359     and dep.oclasType(Dependency).client
2360     ->exists(t|t.oclasType(Actor).isBPartner()
2361     and not(t.oclasType(Actor).isAuthorizedRole()))-> select(dep |
2362     dep.oclasType(Dependency).supplier->
2363     exists(t|t.oclasType(Actor)=self))-> size()=1 and
2364     self.oclasType(Actor).helpMapsToAuthRoleParticipates()
2365

```

```

2366 -- Constraint 98
2367 -- AuthorizedRoles in a BusinessRealizationView must have a unique
2368 -- name within the scope of the package, they are located in
2369 context Package inv:
2370     let authRoles : Set(Element) = self.ownedElement->select( roles |
2371     roles.oclasType(Actor).isAuthorizedRole()
2372     in self.isBRealizationV() implies authRoles->size() =
2373     authRoles->collect(p|p.oclasType(Actor).name)->asSet->size()
2374

```

```

2375 -- Constraint 99
2376 -- The number of AuthorizedRoles participating in a
2377 -- BusinessCollaborationUseCase MUST match the number of
2378 -- AuthorizedRoles participating in the BusinessRealization realizing
2379 -- this BusinessCollaborationUseCase
2380 context Package inv:
2381     let ass : Set(Element) = self.ownedElement->select( ass |
2382     ass.oclasType(Association).isParticipates()
2383     in self.isBRealizationV()
2384     implies ass->select( ends
2385     |ends.oclasType(Association).ownedEnd.type->exists( auth |
2386     auth.oclasType(Actor).isAuthorizedRole()
2387     and ends.oclasType(Association).ownedEnd.type->exists( uc |
2388     uc.oclasType(UseCase).isBRealizationUC())) ->
2389     size() = ass->select( ends
2390     |ends.oclasType(Association).ownedEnd.type->exists( auth |
2391     auth.oclasType(Actor).isAuthorizedRole()
2392     and ends.oclasType(Association).ownedEnd.type->exists( uc |
2393     uc.oclasType(UseCase).isBCollaborationUC())) -> size()
2394

```

```

2395 -- Constraint 100
2396 -- A BusinessInformationView MUST contain one to many
2397 -- InformationEnvelopes or subtypes thereof defined in any other
2398 -- extension/specialization module. Furthermore, it MAY contain any
2399 -- other document modeling artifacts.
2400 context Class inv: self.isBInformationV()
2401     implies self.ownedElement->exists( doc | doc.isInfEnvelope()
2402

```

```

2403 -- Convenience method:

```

```
2404 -- Evaluates if a package is stereotyped as bCollMode
2405 def:
2406     let: isBCollModel() : Boolean =
2407         self.oclAsType(Package).extension_bLibrary.
2408         oclIsKindOf(UMM2_Profile__bCollModel)
2409
```

```
2410 -- Convenience method:
2411 -- Evaluates if a package is stereotyped as bCollaborationV
2412 def:
2413     let: isBCollaborationV() : Boolean =
2414         self.oclAsType(Package).extension_bLibrary.
2415         oclIsKindOf(UMM2_Profile__bCollaborationV)
2416
```

```
2417 -- Convenience method:
2418 -- Evaluates if a package is stereotyped as bRealizationV
2419 def:
2420     let: isBRealizationV() : Boolean =
2421         self.oclAsType(Package).extension_bLibrary.
2422         oclIsKindOf(UMM2_Profile__bRealizationV)
2423
```

```
2424 -- Convenience method:
2425 -- Evaluates if a package is stereotyped as bTransactionV
2426 def:
2427     let: isBTransactionV() : Boolean =
2428         self.oclAsType(Package).extension_bLibrary.
2429         oclIsKindOf(UMM2_Profile__bTransactionV)
2430
```

```
2431 -- Convenience method:
2432 -- Evaluates if a package is stereotyped as bChoreographyV
2433 def:
2434     let: isBChoreographyV() : Boolean =
2435         self.oclAsType(Package).extension_bLibrary.
2436         oclIsKindOf(UMM2_Profile__bChoreographyV)
2437
```

```
2438 -- Convenience method:
2439 -- Evaluates if a package is stereotyped as bInformationV
2440 def:
2441     let: isBInformationV() : Boolean =
2442         self.oclAsType(Package).extension_bLibrary.
2443         oclIsKindOf(UMM2_Profile__bInformationV)
2444
```

```
2445 -- Convenience method:
2446 -- Evaluates if a package is stereotyped as bRequirementsV
2447 def:
2448     let: isBRequirementsV() : Boolean =
2449         self.oclAsType(Package).extension_bLibrary.
2450         oclIsKindOf(UMM2_Profile__bRequirementsV)
2451
```

```
2452 -- Convenience method:
```

```
2453 -- Evaluates if a package is stereotyped as bDomainV
2454 def:
2455     let: isBDomainV() : Boolean =
2456         self.oclAsType(Package).extension_bLibrary.
2457         oclIsKindOf(UMM2_Profile__bDomainV)
```

```
2458
2459 -- Convenience method:
2460 -- Evaluates if a package is stereotyped as bPartnerV
2461 def:
2462     let: isBPartnerV() : Boolean =
2463         self.oclAsType(Package).extension_bLibrary.
2464         oclIsKindOf(UMM2_Profile__bPartnerV)
```

```
2465
2466 -- Convenience method:
2467 -- Evaluates if a package is stereotyped as bEntityV
2468 def:
2469     let: isBEntityV() : Boolean =
2470         self.oclAsType(Package).extension_bLibrary.
2471         oclIsKindOf(UMM2_Profile__bEntityV)
```

```
2472
2473 -- Convenience method:
2474 -- Evaluates if a package is stereotyped as bArea
2475 def:
2476     let: isBArea() : Boolean =
2477         self.oclAsType(Package).extension_bLibrary.
2478         oclIsKindOf(UMM2_Profile__bArea)
```

```
2479
2480 -- Convenience method:
2481 -- Evaluates if a package is stereotyped as ProcessArea
2482 def:
2483     let: isProcessArea() : Boolean =
2484         self.oclAsType(Package).extension_bLibrary.
2485         oclIsKindOf(UMM2_Profile__ProcessArea)
```

```
2486
2487 -- Convenience method:
2488 -- Evaluates if a use case is stereotyped as bProcessUC
2489 def:
2490     let: isBProcessUseCase() : Boolean =
2491         self.oclAsType(UseCase).extension_bProcessUC.isDefined
```

```
2492
2493 -- Convenience method:
2494 -- Evaluates if an activity is stereotyped as bProcess
2495 def:
2496     let: isBProcess() : Boolean =
2497         self.oclAsType(Activity).extension_bProcess.isDefined
```

```
2498
2499 -- Convenience method:
2500 -- Evaluates if an actor is stereotyped as bPartner
2501 def:
```

```
2502     let: isBPartner() : Boolean =
2503         self.oclAsType(Actor).extension_Stakeholder.
2504         oclIsKindOf(UMM2_Profile__bPartner)
2505
```

```
2506 -- Convenience method:
2507 -- Evaluates if an association is stereotyped as participates
2508 def:
2509     let: isParticipates() : Boolean =
2510         self.oclAsType(Association).extension_participates.isDefined
2511
```

```
2512 -- Convenience method:
2513 -- Evaluates if an actor is stereotyped as stakeholder
2514 def:
2515     let: isStakeholder() : Boolean =
2516         self.oclAsType(Actor).extension_Stakeholder.isDefined
2517
```

```
2518 -- Convenience method:
2519 -- Evaluates if a dependency is stereotyped as isOfInterestTo
2520 def:
2521     let: isOfInterestTo() : Boolean =
2522         self.oclAsType(Dependency).extension_isOfInterestTo.isDefined
2523
```

```
2524 -- Convenience method:
2525 -- Evaluates if an actor is stereotyped as authorized role
2526 def:
2527     let: isAuthorizedRole() : Boolean =
2528         self.oclAsType(Actor).extension_AuthorizedRole.isDefined
2529
```

```
2530 -- Convenience method:
2531 -- Evaluates if an use case is stereotyped as BCollaborationUC
2532 def:
2533     let: isBCollaborationUC() : Boolean =
2534         self.oclAsType(UseCase).extension_bProcessUC.
2535         oclIsKindOf(UMM2_Profile__bCollaborationUC)
2536
```

```
2537 -- Convenience method:
2538 -- Evaluates if a class is stereotyped as bEntity
2539 def:
2540     let: isBEntity() : Boolean =
2541         self.oclAsType(Class).extension_bEntity.isDefined
2542
```

```
2543 -- Convenience method:
2544 -- Evaluates if a package is stereotyped as bDataV
2545 def:
2546     let: isBDataV() : Boolean =
2547         self.oclAsType(Package).extension_bLibrary.
2548         oclIsKindOf(UMM2_Profile__bDataV)
2549
```



```
2550 -- Convenience method:
2551 -- Evaluates if an use case is stereotyped as bTransactionUC
2552 def:
2553     let: isBTransactionUC() : Boolean =
2554         self.oclAsType(UseCase).extension_bProcessUC.
2555         oclIsKindOf(UMM2_Profile__bTransactionUC)
```

```
2557 -- Convenience method:
2558 -- Evaluates if a modeling is an activity partition
2559 def:
2560     let: isActivityPartition() : Boolean =
2561         self.oclIsTypeOf(ActivityPartition)
```

```
2563 -- Convenience method:
2564 -- Evaluates if an action is stereotyped as bProcessAction
2565 def:
2566     let: isBProcessAction() : Boolean =
2567         self.oclAsType(Action).extension_bProcessAction.isDefined
```

```
2569 -- Convenience method:
2570 -- Evaluates if an use case is not stereotyped as bTransactionUC
2571 def:let: isNotBTransactionUC() : Boolean =
2572     not(self.oclAsType(UseCase).isBTransactionUC())
```

```
2574 -- Convenience method:
2575 -- Evaluates if an activity is stereotyped as bCollaborationProtocol
2576 def:let: isBCollaborationProtocol() : Boolean =
2577
2578     self.oclAsType(Activity).extension_bCollaborationProtocol.isDefined
```

```
2580 -- Convenience method:
2581 -- Evaluates if a call behavior action is stereotyped as
2582     bTransactionAction
2583 def:
2584     let: isBTransactionAction() : Boolean =
2585
2586         self.oclAsType(CallBehaviorAction).extension_bTransactionAction.isD
2587         efined
```

```
2589 -- Convenience method:
2590 -- Evaluates if a call behavior action is stereotyped as
2591     bCollaborationAction
2592 def:
2593     let: isBCollaborationAction() : Boolean =
2594         self.oclAsType(CallBehaviorAction).
2595         extension_bCollaborationAction.isDefined
```

```
2597 -- Convenience method:
2598 -- Evaluates if a class is stereotyped as InfEnvelope
```

```
2599 def:
2600     let: isInfEnvelope() : Boolean =
2601         self.oclAsType(Class).extension_InfEnvelope.isDefined
2602
```

```
2603 -- Convenience method:
2604 -- Evaluates if an object node is stereotyped as bESharedState
2605 def:
2606     let: isBSharedState() : Boolean =
2607         self.oclAsType(ObjectNode).extension_bESharedState.isDefined
2608
```

```
2609 -- Convenience method:
2610 -- Evaluates if a state is stereotyped as bEState
2611 def:
2612     let: isBEState() : Boolean =
2613         self.oclAsType(State).extension_bEState.isDefined
2614
```

```
2615 -- Convenience method:
2616 -- Evaluates if an activity is stereotyped as bTransaction
2617 def:
2618     let: isBTransaction() : Boolean =
2619         self.oclAsType(Activity).extension_bTransaction.isDefined
2620
```

```
2621 -- Convenience method:
2622 -- Evaluates if an activity partition is stereotyped as bTPartition
2623 def:
2624     let: isBTPartition() : Boolean =
2625         self.oclAsType(ActivityPartition).extension_bTPartition.isDefined
2626
```

```
2627 -- Convenience method:
2628 -- Evaluates if an action is stereotyped as ReqAction
2629 def:
2630     let: isReqAction() : Boolean =
2631         self.oclAsType(Action).extension_BusinessAction.
2632         oclIsKindOf(UMM2_Profile__ReqAction)
2633
```

```
2634 -- Convenience method:
2635 -- Evaluates if an action is stereotyped as ResAction
2636 def:
2637     let: isResAction() : Boolean =
2638         self.oclAsType(Action).extension_BusinessAction.
2639         oclIsKindOf(UMM2_Profile__ResAction)
2640
```

```
2641 -- Convenience method:
2642 -- Evaluates if an use case is stereotyped as bRealizationUC
2643 def:
2644     let: isBRealizationUC() : Boolean =
2645         self.oclAsType(UseCase).extension_bRealizationUC.isDefined
2646
```

```

2647 -- Convenience method:
2648 -- Evaluates if a dependency is stereotyped as realizes
2649 def:
2650     let: isRealizes() : Boolean =
2651         self.oclasType(Realization).extension_realizes.isDefined
2652

```

```

2653 -- Convenience method:
2654 -- Evaluates if a dependency is stereotyped as mapsTo.
2655 def:
2656     let: isMapsTo() : Boolean =
2657         self.oclasType(Dependency).extension_mapsTo.isDefined
2658

```

```

2659 -- Convenience method for evaluating if there is a mapsTo dependency
2660 -- from an authorized role
2661 -- that leads to another authorized role participating in a business
2662 -- collaboration use case
2663 def:
2664     let: hlpMapsToAuthRoleParticipates() : Boolean =
2665         self.owner.ownedElement->
2666         select( dep | dep.oclasType(Dependency).isMapsTo()
2667         and dep.oclasType(Dependency).client->
2668         exists( t | t.oclasType(Actor)=self)
2669         and dep.oclasType(Dependency).supplier->
2670         exists(t | t.oclasType(Actor).isAuthorizedRole()
2671         and t.oclasType(Actor).hlpParticipatesBCollaborationUC() ))
2672         ->size()=1
2673

```

```

2674 -- Convenience method for evaluating if a certain authorized role
2675 -- participates in a business
2676 -- collaboration use case
2677 def:
2678     let: hlpParticipatesBCollaborationUC() : Boolean =
2679         self.owner.ownedElement->select( ass |
2680         ass.oclasType(Association).isParticipates()
2681         and ass.oclasType(Association).ownedEnd.type->exists( end |
2682         end.oclasType(Actor)=self)
2683         and ass.oclasType(Association).ownedEnd.type->exists( end |
2684         end.oclasType(UseCase).isBCollaborationUC() )->size()=1
2685

```

```

2686 -- Convenience method:
2687 -- Evaluates if an activity partition is stereotyped as
2688 -- bCollaborationPartition
2689 def:
2690     let: isBCollaborationPartition() : Boolean =
2691         self.oclasType(ActivityPartition).
2692         extension_bCollaborationPartition.isDefined
2693

```

```

2694 -- Convenience method:
2695 -- Evaluates if a business transaction MUST have an responding
2696 -- information pin

```

```
2697 -- (i.e., is a two-way transaction) according to its business
2698 transaction pattern
2699 def:
2700     let: hlpMustHaveResInfPin() : Boolean =
2701         self.oclasType(Activity).extension_bTransaction.
2702         businessTransactionType='RequestResponse'
2703     or self.oclasType(Activity).extension_bTransaction.
2704         businessTransactionType='QueryResponse'
2705     or self.oclasType(Activity).extension_bTransaction.
2706         businessTransactionType='RequestConfirm'
2707     or self.oclasType(Activity).extension_bTransaction.
2708         businessTransactionType='CommercialTransaction'
2709
```

```
2710 -- Convenience method:
2711 -- Evaluates if a business transaction MUST NOT have an responding
2712 information pin
2713 -- (i.e., is an one-way transaction) according to its business
2714 transaction pattern
2715 def:
2716     let: hlpMustNotHaveResInfPin() : Boolean =
2717         self.oclasType(Activity).extension_bTransaction.
2718         businessTransactionType='Notification'
2719     or self.oclasType(Activity).extension_bTransaction.
2720         businessTransactionType='InformationDistribution'
2721
```

```
2722 -- Convenience method:
2723 -- Evaluates if an action is stereotyped as bNestedCollaboration
2724 def:
2725     let: isBNestedCollaboration() : Boolean =
2726         self.oclasType(Action).extension_bNestedCollaboration.isDefined
2727
```

```
2728 -- Convenience method:
2729 -- Evaluates if a dependency is stereotyped as InitFlow
2730 def:
2731     let: isInitFlow() : Boolean =
2732         self.oclasType(Dependency).extension_initFlow.isDefined
2733
```

```
2734 -- Convenience method:
2735 -- Evaluates if a dependency is stereotyped as ReFlow
2736 def:
2737     let: isReFlow() : Boolean =
2738         self.oclasType(Dependency).extension_reFlow.isDefined
```

2739 **Copyright Statement**

2740

2741 Copyright © UN/CEFACT 2008. All Rights Reserved.

2742 This document and translations of it may be copied and furnished to others, and derivative works that  
2743 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published  
2744 and distributed, in whole or in part, without restriction of any kind, provided that the above copyright  
2745 notice and this paragraph are included on all such copies and derivative works. However, this document  
2746 itself may not be modified in any way, such as by removing the copyright notice or references to  
2747 UN/CEFACT except as required to translate it into languages other than English.

2748 The limited permissions granted above are perpetual and will not be revoked by UN/CEFACT or its  
2749 successors or assigns.

2750 This document and the information contained herein is provided on an "AS IS" basis and UN/CEFACT  
2751 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY  
2752 THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED  
2753 WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.