



**UN/CEFACT**

United Nations Centre for Trade Facilitation and Electronic Business

Techniques and Methodologies  
Group (TMG)

***UN/CEFACT's Modeling Methodology (UMM)***  
***UMM Meta Model – Foundation Module***  
***Candidate for 2.0***  
***Draft for IMPLEMENTATION VERIFICATION***  
***2009-01-30***

1	<b>Table of Contents</b>	
2	Table of Contents .....	2
3	1 About this Document .....	4
4	1.1 Status of this Document .....	4
5	1.2 Revision History .....	4
6	1.3 Document Context .....	4
7	1.4 Conventions.....	5
8	2 Project Team.....	6
9	2.1 Disclaimer .....	6
10	2.2 Contact .....	6
11	2.3 Project Team Participants .....	6
12	3 Introduction.....	7
13	3.1 Audience .....	7
14	3.2 Related Documents .....	7
15	3.3 UN/CEFACT’s Modeling Methodology (UMM): Overview.....	8
16	3.4 Objectives .....	9
17	3.4.1 Goals of the Technical Specification.....	9
18	3.4.2 Requirements .....	10
19	3.4.3 Caveats and Assumptions.....	10
20	3.5 Structure of the UMM Foundation Module .....	11
21	4 Dependencies on other UMM Modules (normative).....	12
22	4.1 Abbreviations of Stereotypes .....	12
23	4.2 Dependency between Base Module and Foundation Module.....	12
24	5 UMM Foundation Module.....	13
25	5.0 Foundation Module Management .....	13
26	5.0.1 Abbreviations of Stereotypes .....	13
27	5.0.2 Conceptual Description (informative) .....	14
28	5.0.3 Stereotypes and Tag Definitions (normative).....	15
29	5.0.4 Constraints (normative).....	17
30	5.1 Business Requirements View.....	17
31	5.1.0 Sub-Views in the Business Requirements View.....	17
32	5.1.1 Business Domain View.....	20
33	5.1.2 Business Partner View .....	37
34	5.1.3 Business Entity View.....	40

35	5.2	Business Choreography View .....	47
36	5.2.1	Sub-Views in the Business Choreography View .....	47
37	5.2.2	Business Transaction View .....	51
38	5.2.3	Business Collaboration View .....	71
39	5.2.4	Business Realization View .....	89
40	5.3	Business Information View .....	94
41	5.3.1	Abbreviations of Stereotypes .....	94
42	5.3.2	Conceptual Description (informative) .....	94
43	5.3.3	Stereotypes and Tag Definitions (normative).....	95
44	5.3.4	Constraints (normative).....	95
45	5.3.5	Example using UPCC (UML Profile for Core Components) and CCMA (Core Components	
46		Message Assembly) (informative) .....	96
47	6	OCL Constraints .....	98
48		Copyright Statement .....	126
49			

## 50 1 About this Document

### 51 1.1 Status of this Document

52

53 This document is currently being developed, and is at Open Development Process Step 6: “Implementation  
54 Verification”

#### 55 (i) Activities

56 The UNECE secretariat provides links on the UNECE website to the implementation. The FMG  
57 notifies Heads of Delegation and various e-mail distribution list subscribers that the  
58 Implementation Draft is available for implementation verification and provides them with details  
59 regarding the process for submitting comments. The project team processes comments and  
60 posts updated Implementation Drafts and comment logs to the PG website or UNECE website  
61 (through the secretariat). The comment/update/posting cycle continues until at least two  
62 independent implementations have been confirmed and the PG approves a project team  
63 recommendation to conclude ODP6. While the criteria, evaluation, and ultimate decision to  
64 conclude ODP6 is left to the PG, the PG must ensure that the project team has met all comment  
65 processing requirements

#### 66 (ii) Artifacts

67 Typical artifacts produced by ODP6 include:

- 68 a. Final Draft
- 69 b. Two implementation verifications
- 70 c. Updated Comment Log

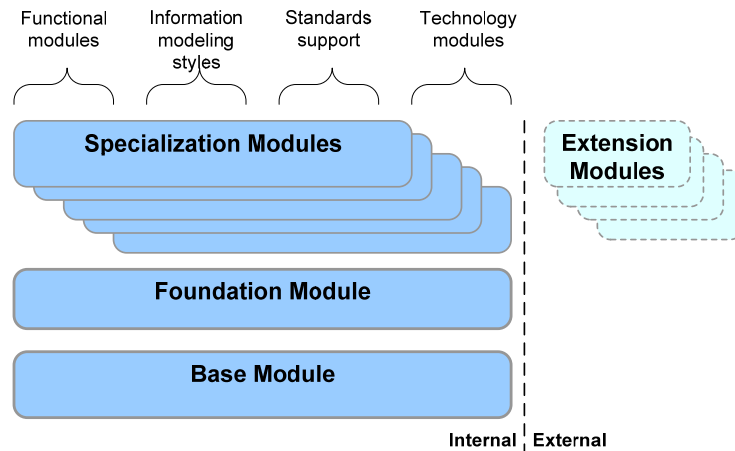
71 The first review cycle of ODP 6 is open until 17<sup>h</sup> April 2009. Please send your comments to the Project Lead  
72 Christian Huemer ([huemer@big.tuwien.ac.at](mailto:huemer@big.tuwien.ac.at))

### 73 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	Changes are marked in green
Candidate for 2.0	Implementation Verification	2009-01-30	Changes from both previous drafts are marked in green

### 74 1.3 Document Context

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



78  
79  
80  
81 **Figure 1 Module structure of the UMM meta model**

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

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

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

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

92 This specification defines the foundation module of UMM 2.0.

### 93 1.4 Conventions

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

- 97 • MUST: This word, or the terms "REQUIRED" or "SHALL", means that the definition is an absolute  
98 requirement of the specification.
- 99 • MUST NOT: This phrase, or the phrase "SHALL NOT", means that the definition is an absolute  
100 prohibition of the specification.
- 101 • SHOULD: This word, or the adjective "RECOMMENDED", means that there may exist valid reasons in  
102 particular circumstances to ignore a particular item, but the full implications MUST be understood  
103 and carefully weighed before choosing a different course.
- 104 • SHOULD NOT: This phrase, or the phrase "NOT RECOMMENDED", means that there may exist valid  
105 reasons in particular circumstances when the particular behavior is acceptable or even useful, but  
106 the full implications should be understood and the case carefully weighed before implementing any  
107 behavior described with this label.

- 108 • MAY: This word, or the adjective "OPTIONAL", means that an item is truly optional. One vendor may  
 109 choose to include the item because a particular marketplace requires it or because the vendor feels  
 110 that it enhances the product while another vendor may omit the same item. An implementation that  
 111 does not include a particular option MUST be prepared to interoperate with another  
 112 implementation which does include the option, though perhaps with reduced functionality. In the  
 113 same vein an implementation that does include a particular option MUST be prepared to  
 114 interoperate with another implementation which does not include the option (except, of course, for  
 115 the feature the option provides).

## 116 2 Project Team

### 117 2.1 Disclaimer

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

### 121 2.2 Contact

122 Name: Christian Huemer  
 123 Company: Vienna University of Technology  
 124 Street: Favoritenstrasse 9-11/188  
 125 City, state, zip/other: 1040 Vienna  
 126 Nation: Austria  
 127 Phone: +43 1 58801 18882  
 128 Email: huemer@big.tuwien.ac.at

### 129 2.3 Project Team Participants

130

131	Project Team Lead:	Christian Huemer	Austria
132	Editing Team:	Jens Dietrich	Germany
133		Birgit Hofreiter	Austria
134		Christian Huemer	Austria
135		Philipp Liegl	Austria
136		Glenn Miller	Canada
137		Harry Moyer	Australia
138	Rainer Schuster	Austria	
139	Marco Zapletal	Austria	

140

141

142	Contributors:	Steve Capell	Australia
143		Sylvie Colas	France
144		Barbara Flügge	Switzerland
145		William McCarthy	USA
146		Thomas Motal	Austria

147	Dennis Müller	Germany
148	Christian Senf	Germany
149	Nita Sharma	USA
150	Gunther Stuhec	Germany
151		

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

## 155 3 Introduction

### 156 3.1 Audience

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

162 The information described in this manual is aimed at

- 163 • advanced business process modelers who check a UML model for UMM compliance (if not  
164 supported by a tool)
- 165 • advanced business process modelers who train other business process modelers and business  
166 process analysts
- 167 • software designers who want to produce UML tools providing support for this UMM foundation  
168 module
- 169 • software designers who want to produce tools to transform UMM compliant business collaboration  
170 models into specifications within an IT-layer (ebXML, Web Services, UN/EDIFACT, etc.).
- 171 • software designers who want to produce repositories to register UMM compliant business  
172 collaboration models

### 173 3.2 Related Documents

- 174 • **UN/CEFACT**
  - 175 ○ UN/CEFACT Open Development Process (TRADE/R.650/Rev.4/Add.1/Rev.1 - 19 April 2007)  
176 [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)
  - 177 ○ UPCC: UML Profile for Core Components  
178 <http://unstandards.org:8080/display/public/UPCC+-+UML+Profile+for+Core+Components>
  - 179 ○ Core Component Technical Specification  
180 [http://www.unece.org/cefact/ebxml/CCTS\\_V2-01\\_Final.pdf](http://www.unece.org/cefact/ebxml/CCTS_V2-01_Final.pdf)
- 181 • **International Organization for Standardization (ISO)**
  - 182 ○ Open-edi Reference Model. ISO/IEC 14662  
183 [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)
- 184 • **Object Management Group (OMG)**
  - 185 ○ Unified Modeling Language Specification (UML), Version 2.1.2  
186 <http://www.omg.org/docs/formal/07-02-05.pdf>

### 187 3.3 UN/CEFACT's Modeling Methodology (UMM): Overview

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

198 The UMM, as described in this document, is the formal description technique for describing any Open-edi  
199 scenario as defined in ISO/IEC 14662 "Open-edi reference model". An Open-edi scenario is a formal means  
200 to specify a class of business transactions having the same business goal, such as, purchasing or inventory  
201 management. The primary scope of UMM is the Business Operations View (BOV) and not the Functional  
202 Service View (FSV) as defined in ISO/IEC IS 14662. The BOV is defined as "a perspective of business  
203 transactions limited to those aspects regarding the making of business decisions and commitments among  
204 organizations", while the FSV is focused on implementation specific, technological aspects of Open-edi. The  
205 commitments of the BOV layer are reflected in the choreography of the inter-organizational business  
206 processes and their information exchanges. At the FSV layer, this choreography must be implemented by a  
207 set of composite services. Therefore it follows, that UMM, which targets the BOV layer, defines what the  
208 business is about; and the technologies on the FSV layer define how to implement the business by a service-  
209 oriented architecture.

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

- 211 • Business Requirements View (bRequirementsV)
- 212 • Business Choreography View (bChoreographyV)
- 213 • Business Interaction View (bInteractionV)

214

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



229 Entities and their lifecycles of state changes are modeled in the Business Entity View. Furthermore, the  
230 Business Entity View also contains one or more packages which represent the conceptual data structures of  
231 the Business Entities.

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

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

## 258 3.4 Objectives

### 259 3.4.1 Goals of the Technical Specification

260 The goals of this specification are:

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

## 272 3.4.2 Requirements

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

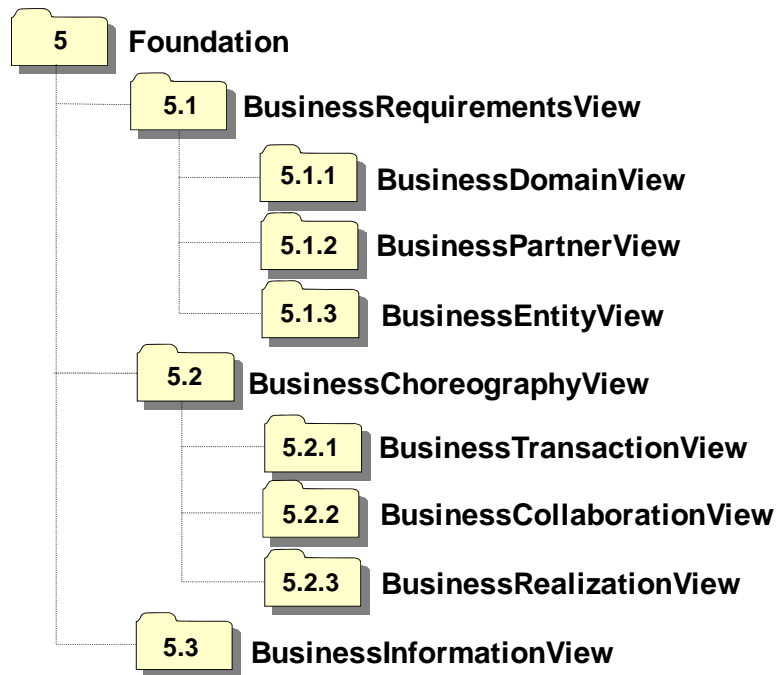
- 274 • The UMM foundation module defines only those modeling concepts that are considered as  
275 fundamental to deliver a UMM compliant model. Additional advanced modeling concepts shall be  
276 covered in specialization and extension modules.
- 277 • The UMM foundation module is directed towards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

## 289 3.4.3 Caveats and Assumptions

290 This specification makes the following assumptions:

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

300 **3.5 Structure of the UMM Foundation Module**



301

302

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

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

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

314 Each section describing a package is structured in the same way. The first subsection is informative. It  
315 describes the conceptual model of the artefact that is addressed by the package. The second subsection is  
316 normative and defines all the stereotypes and associated tag definitions that are defined in the package. The  
317 third subsection is normative and includes all the constraints in plain text that apply to the respective  
318 package. **The constraints are also expressed using the Object Constraint Language (OCL) in section 6. The**  
319 **two remaining informative subsections cover on the one hand side worksheets used to gather information**  
320 **from business people in order to create the UMM models and on the other hand side examples to depict**  
321 **instances of the artefact type addressed by the package.**

322 The

## 323 4 Dependencies on other UMM Modules (normative)

### 324 4.1 Abbreviations of Stereotypes

Stereotype Abbreviation	Full Stereotype Name
bInformation	BusinessInformation
bLibrary	BusinessLibrary
InfEnvelope	InformationEnvelope

325

### 326 4.2 Dependency between Base Module and Foundation Module

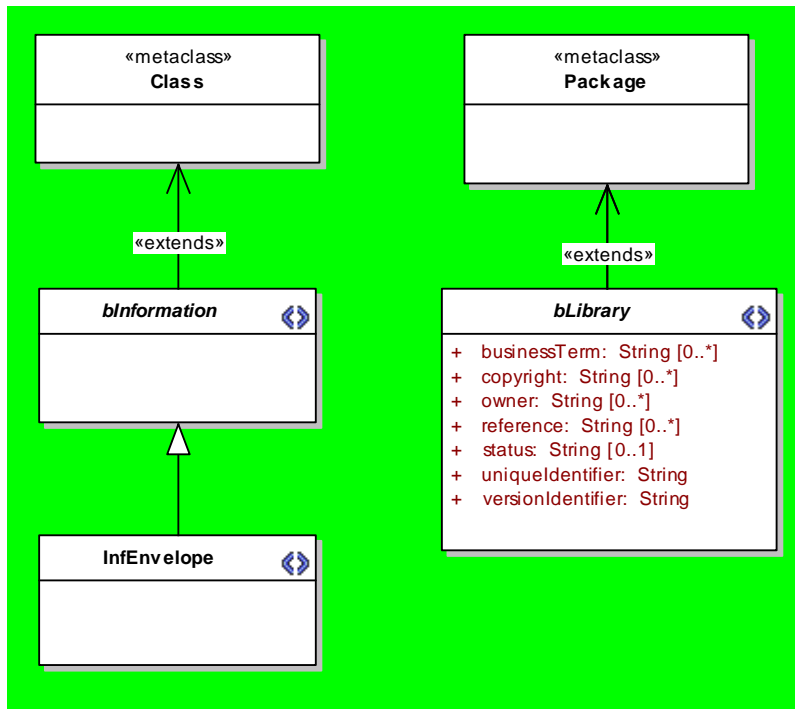


327

328 **Figure 3 UMM Foundation Dependencies**

329 The UMM foundation module 2.0 is built on top of the UMM base module 2.0. This means that all  
330 stereotypes and tag definitions defined in the UMM base module 2.0 are imported into the UMM  
331 foundation module 2.0. The figure below shows the stereotypes defined in the UMM base module also used  
332 in the foundation module. Note that the stereotypes of the base module are identified with notes in all  
333 figures of this specification. The formal definition of the stereotypes *bInformation*, *InfEnvelope* and *bLibrary*  
334 is given in the UMM base module 2.0 specification. In the foundation module, packages - that are containers  
335 of stereotypes realizing main UMM artefacts - are defined as specializations of the base stereotype *bLibrary*.  
336 This means that such packages and their contents are candidates for registration in a registry. In the UMM  
337 foundation module 2.0 we do not define any stereotype that directly inherits from *bLibrary*. As a  
338 consequence, only packages are candidates for registration.

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



341

342

343

Figure 4 UMM Base Abstract Syntax

## 344 5 UMM Foundation Module

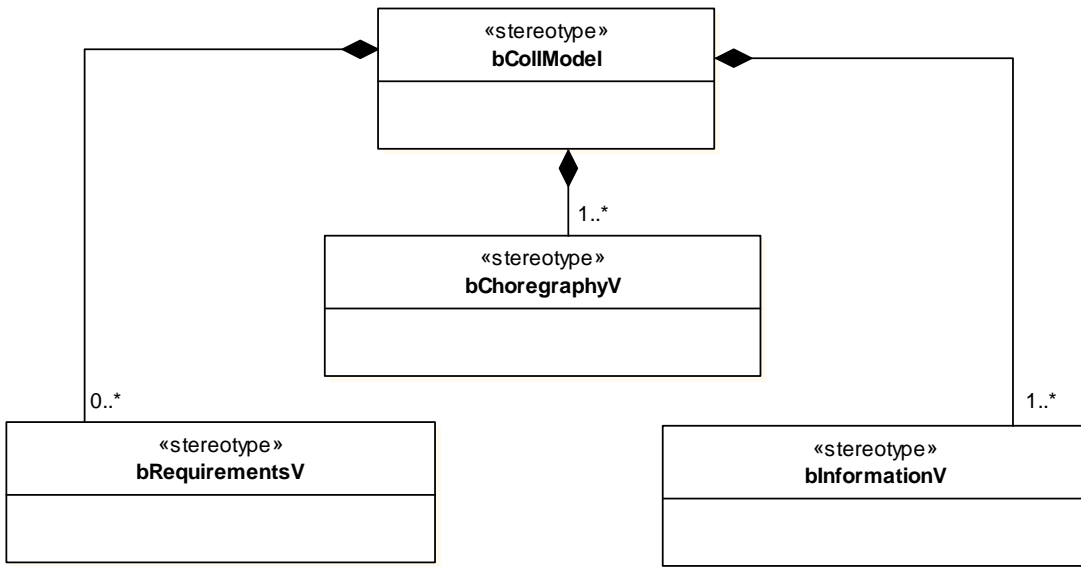
### 345 5.0 Foundation Module Management

#### 346 5.0.1 Abbreviations of Stereotypes

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

347

348 **5.0.2 Conceptual Description (informative)**



349

350

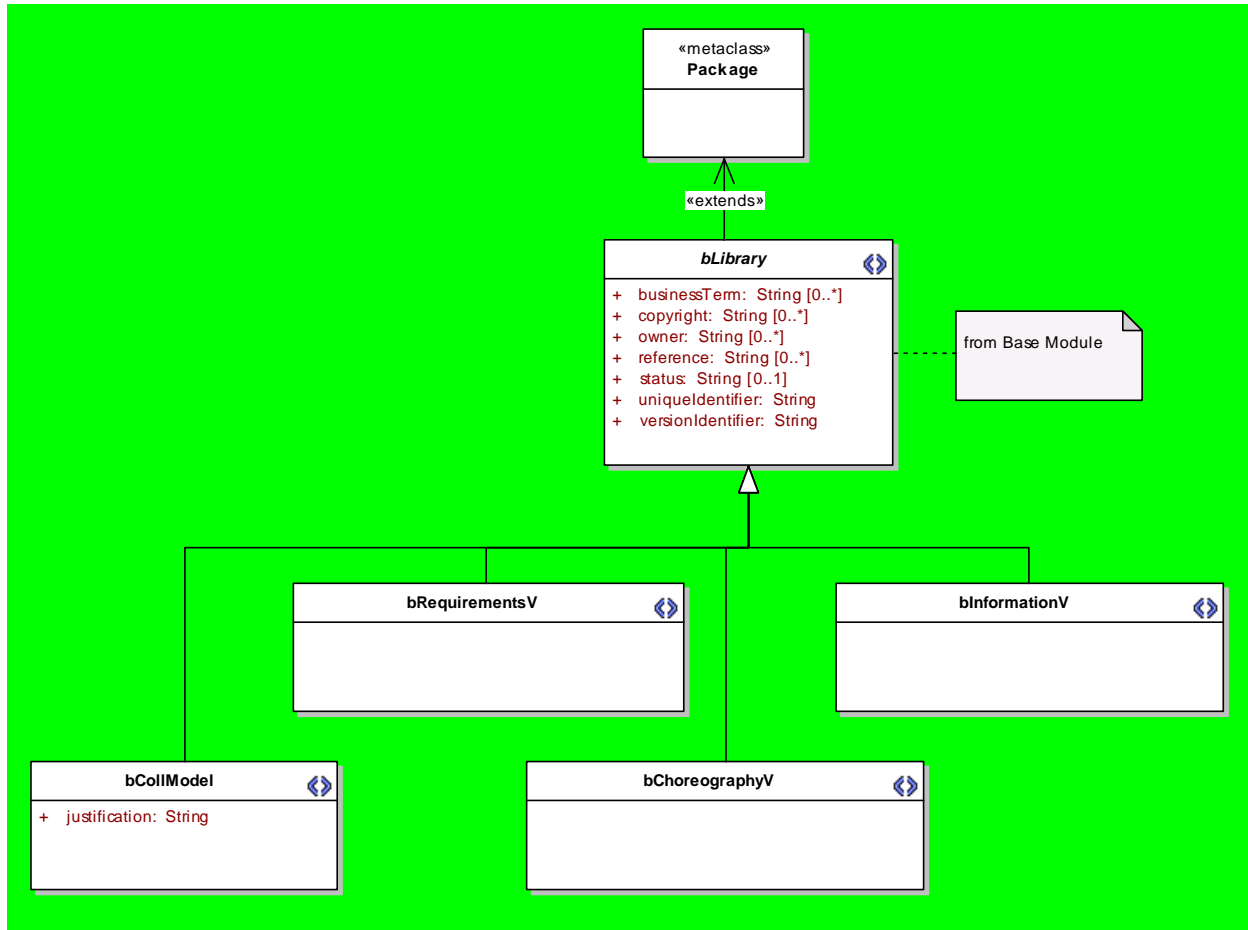
**Figure 5 UMM Foundation Module Management - Conceptual Overview**

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

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

362

363 5.0.3 Stereotypes and Tag Definitions (normative)



364  
365

366  
368

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 MUST be compliant to the base and foundation module, and it MAY 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 MAY 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.

	<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>– <b>uniqueIdentifier</b></li> <li>– <b>versionIdentifier</b></li> </ul>
--	---

369

<b>Stereotype</b>	<b>bRequirementsV (BusinessRequirementsView)</b>
<b>Base Class</b>	Package
<b>Parent</b>	<b>BusinessLibrary (from Base Module)</b>
<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><b>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.</b></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>– <b>uniqueIdentifier</b></li> <li>– <b>versionIdentifier</b></li> </ul>

370

<b>Stereotype</b>	<b>bChoreographyV (BusinessChoreographyView)</b>
<b>Base Class</b>	Package
<b>Parent</b>	<b>BusinessLibrary (from Base Module)</b>
<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>– <b>uniqueIdentifier</b></li> <li>– <b>versionIdentifier</b></li> </ul>

371

<b>Stereotype</b>	<b>bInforationV (BusinessInformationView)</b>
-------------------	---



<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>

372

### 373 5.0.4 Constraints (normative)

374

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

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

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

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

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

## 381 5.1 Business Requirements View

### 382 5.1.0 Sub-Views in the Business Requirements View

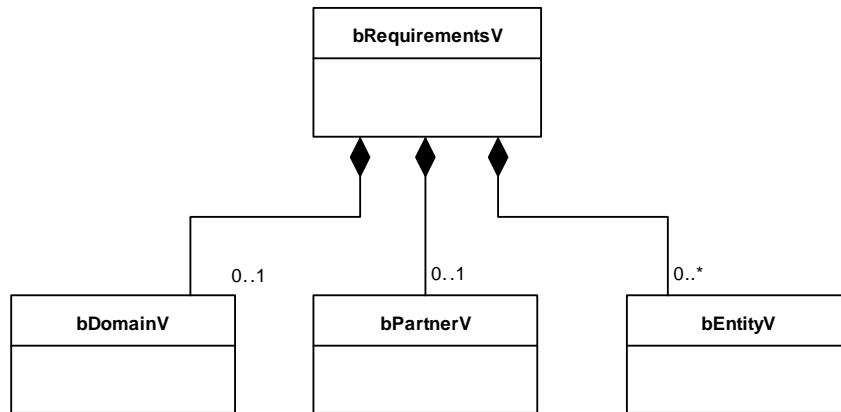
#### 383 5.1.0.1 Abbreviations and Stereotypes

384

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

385

386 5.1.0.2 Conceptual Description (informative)



387

388 **Figure 7 BusinessRequirementsView Conceptual Overview**

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

402 5.1.0.3 Stereotypes and Tag Definitions (normative)

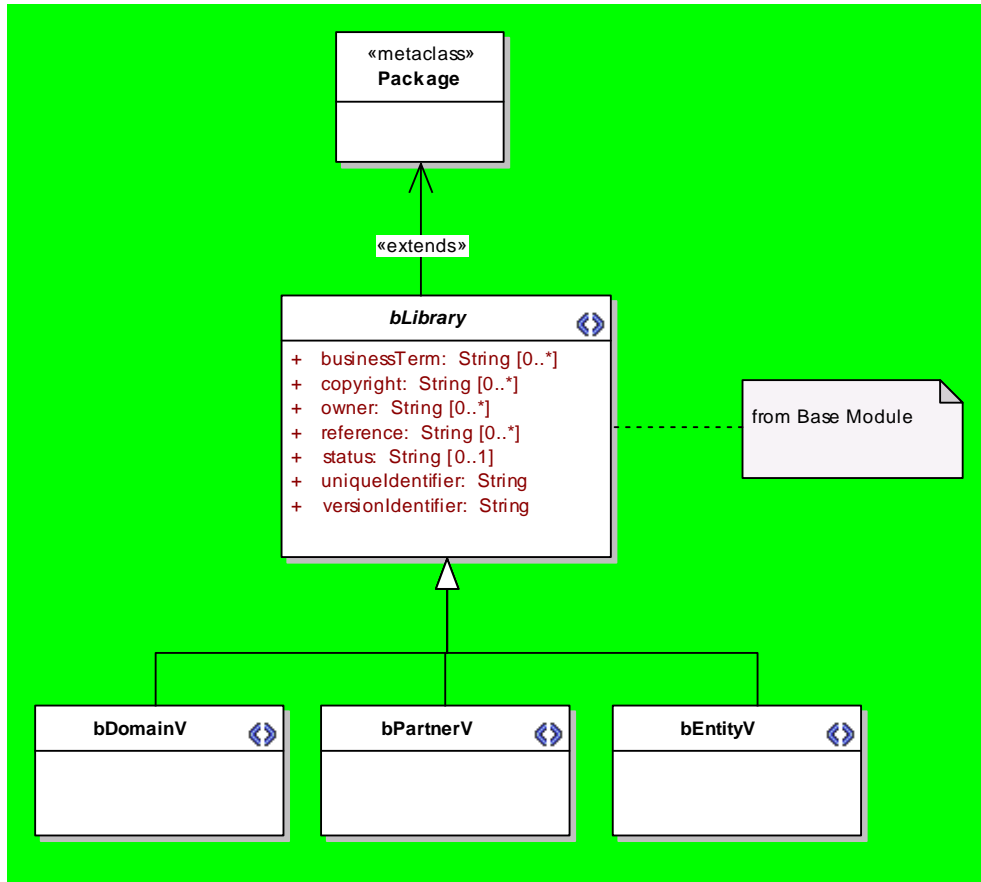


Figure 8 BusinessRequirementsView Abstract Syntax

403  
404  
406

Stereotype <b>bDomainV (BusinessDomainView)</b>	
Base Class	Package
Parent	BusinessLibrary (from Base Module)
Description	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.
Tag Definition	<b>Inherited tagged values:</b> <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>

407

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

408

<b>Stereotype</b>	<b>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>

409

410

#### 411 **5.1.0.4 Constraints (normative)**

412 Constraints with respect to a *BusinessRequirementsView*:

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

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

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

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

418

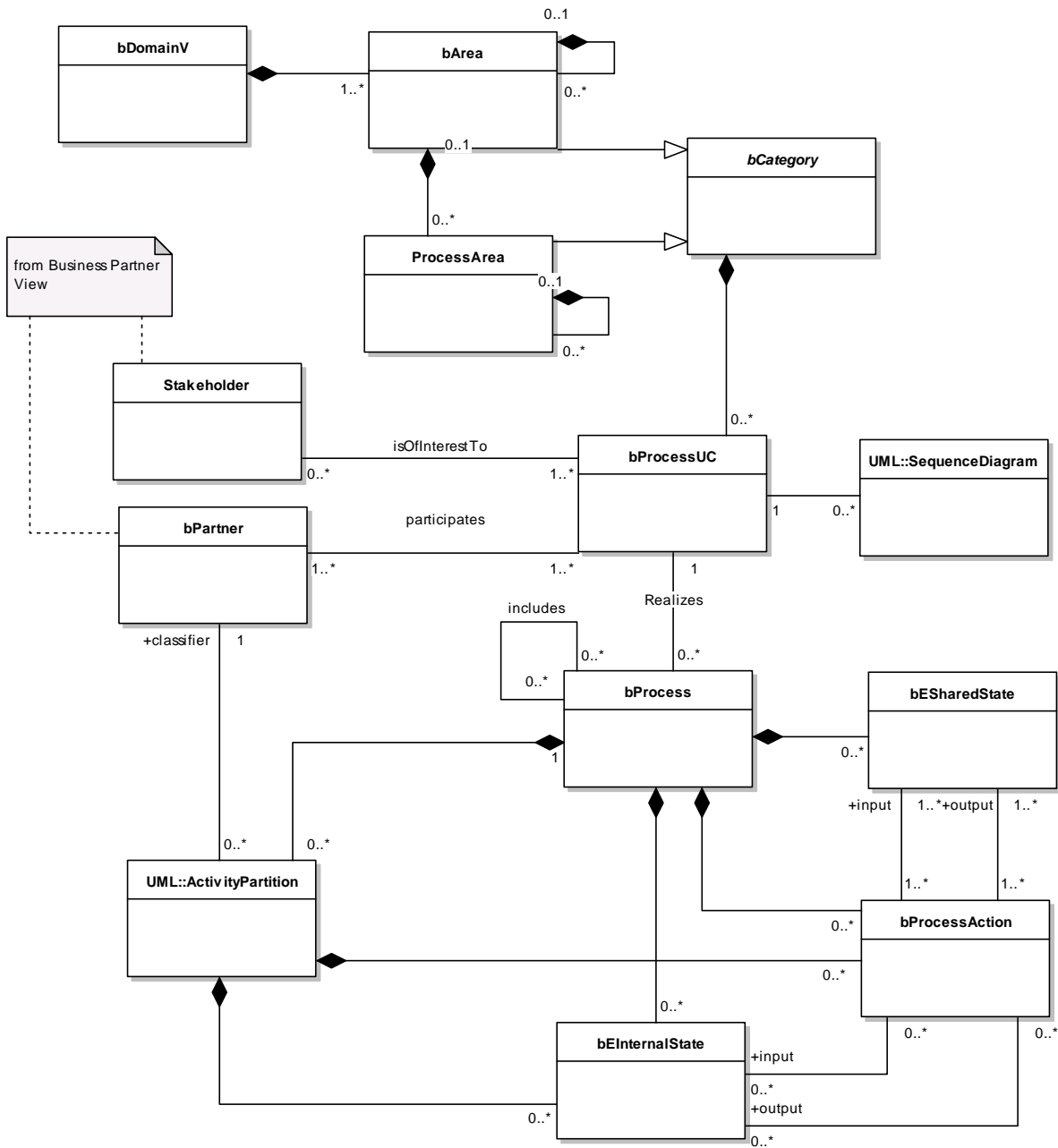
### 419 **5.1.1 Business Domain View**

#### 420 **5.1.1.1 Abbreviations of Stereotypes**

Stereotype Abbreviation	Full Stereotype Name
bDomainV	BusinessDomainView
bCategory	BusinessCategory
bArea	BusinessArea

ProcessArea	ProcessArea
bProcessUC	BusinessProcessUseCase
bProcess	BusinessProcess
bProcessAction	BusinessProcessAction
bESharedState	SharedBusinessEntityState
bEInternalState	InternalBusinessEntityState
participates	participates
isOfInterestTo	isOfInterestTo

421



423

424

Figure 9 BusinessDomainView – Conceptual Overview

425

The business domain view is used to discover business processes use cases that are of relevance in a project.

426

A business process use case is executed by at least one (but possibly more) business partners. A business

427

partner might execute multiple business process use cases. Thus, the *participates* association between

428

*BusinessPartner* and *BusinessProcessUseCase* is a (1..n) to (0..n) association. A stakeholder does not need to

429

participate in a business process use case. A stakeholder might have interest in multiple business process use

430

cases and a business process use case might be of interest to multiple stakeholders. The relationship

431

between a *BusinessProcessUseCase* and a *Stakeholder* is described by the *isOfInterestTo* dependency in

432 UMM. A business process can be decomposed into sub-processes using the «include» and «extends»  
433 association stereotypes.

434 To enable users to readily identify business process use cases, they should be classified into business  
435 categories. A business category is an abstract concept, which has two concrete specializations – business  
436 area and process area. A business area corresponds to a division of an organization and a process area  
437 corresponds to a set of common operations within the business area. A business area might be composed of  
438 other business areas. This means, a business area may form a hierarchy. Thus, a unary (0..1) to (0..n)  
439 composition is defined for a *BusinessArea*. The lowest level of a business area hierarchy includes process  
440 areas or business processes use cases. Therefore, we have a (0..1) to (0..n) composition between  
441 *BusinessArea* and *ProcessArea*. Furthermore, a *BusinessArea* may also include 0..n *BusinessProcessUseCases*  
442 if no further classification using process areas is required. Similar to a business area, a process area may form  
443 a hierarchy. This means, a unary (0..1) to (0..n) composition is defined for a *ProcessArea*. Similar to a  
444 business area, a *ProcessArea* may contain zero to many *BusinessProcessUseCases*. On the lowest level of a  
445 *ProcessArea* hierarchy, at least one *BusinessProcessUseCase* must be present.

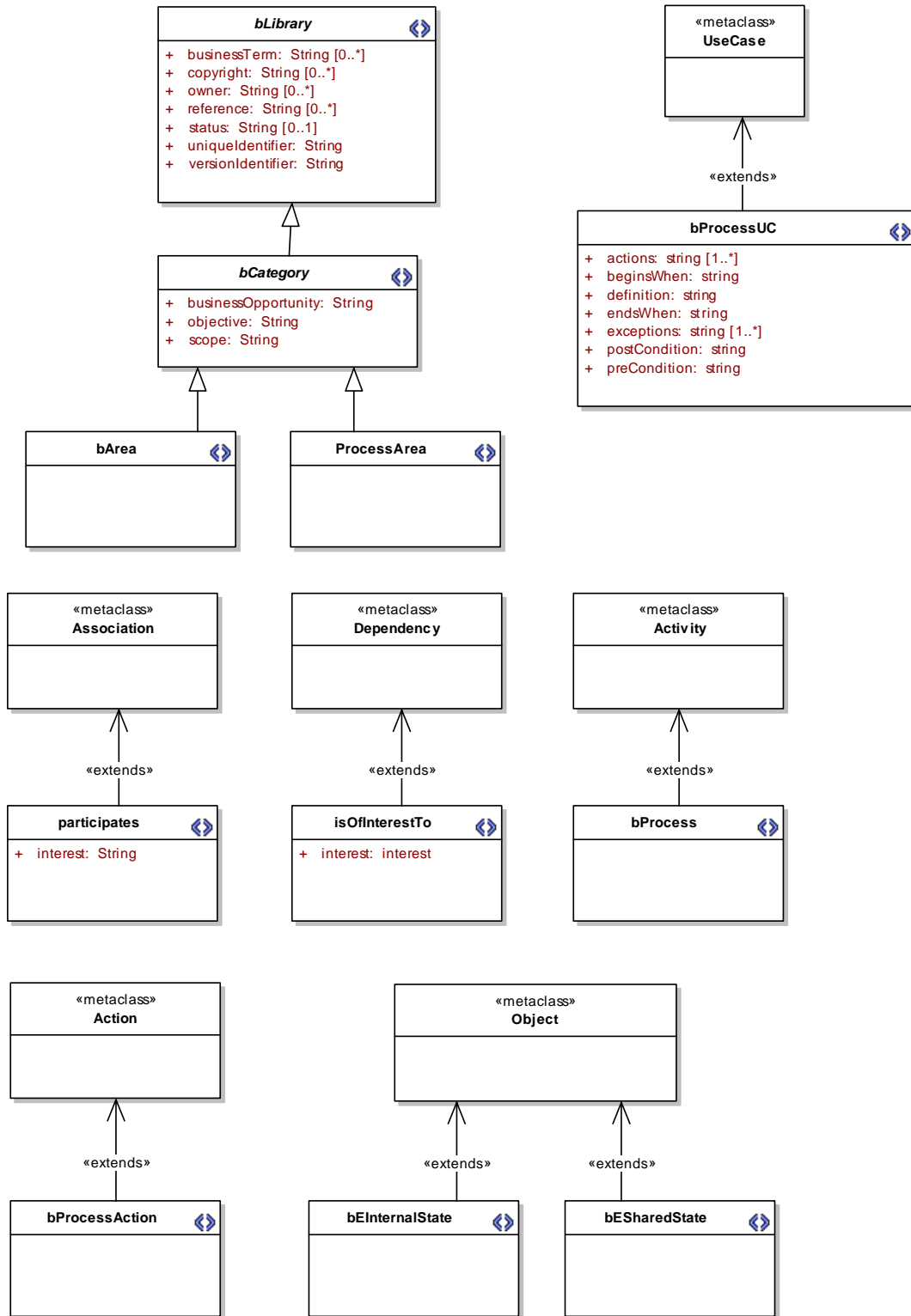
446 The flow of a business process use case may be described by business processes. Thus, a  
447 *BusinessProcessUseCase* is realized by zero to many *BusinessProcesses*. A business process represents the  
448 dynamic behavior of a business process use case. A business process corresponds to a flow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

456 A business process is described as a flow of business process actions. In the case where no activity partition  
457 is used, the business process actions are directly included in the Activity Diagram of the business process. In  
458 case of activity partitions, a business process action is assigned to the partition of the business partner  
459 executing the action. The need for a collaborative business process is identified whenever a transition  
460 connecting two business process actions crosses activity partitions. It follows, that either a *BusinessProcess* is  
461 composed of one or more *BusinessProcessAction* or an *ActivityPartition* (which is part of a business process)  
462 is composed of one or more *BusinessProcessActions*. A business process action might be refined by another  
463 business process. Thus a *BusinessProcessAction* is composed of zero or one *BusinessProcess* which in turn is a  
464 composite of zero or one *BusinessProcessActions*.

465 A business process may also denote important states of business entities that are manipulated during the  
466 execution of a business process. A business entity state is the output from one business action and input to  
467 another business action. There is a transition from a business process action to a business entity state  
468 signaling an output as well as a transition from a business entity state to a business process action signaling  
469 an input. Some business entity states are meaningful to one business partner only. These are internal  
470 business entity states. Business entity states that must be communicated to a business partner are shared  
471 business entity states. A business process may include both internal and shared business entity states.  
472 Hence, a *BusinessProcess* is composed of zero to many *InternalBusinessEntityStates* and of zero to many  
473 *SharedBusinessEntityStates*. If a business process uses activity partitions, the two business process actions

474 creating and consuming an internal business entity state are in the same activity partition. In contrast, the  
475 two business process actions creating and consuming a shared business entity state are in different activity  
476 partitions. A shared business entity state signals the need for a collaborative business process.





478

479

480

Figure 10 BusinessDomainView Abstract Syntax

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

481

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

	<p>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>

482

<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>

483

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

<b>Description</b>	<p>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.</p>	
<b>Tag Definition</b>	<b>definition</b>	
	<b>Type</b>	String
	<b>Multiplicity</b>	1
	<b>Description</b>	<p>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.</p>
	<b>beginsWhen</b>	
	<b>Type</b>	String
	<b>Multiplicity</b>	1
	<b>Description</b>	<p>Specifies a business event that triggers the initiation of the business process use case.</p>
	<b>preCondition</b>	
	<b>Type</b>	String
	<b>Multiplicity</b>	1
	<b>Description</b>	<p>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.</p>
	<b>endsWhen</b>	
	<b>Type</b>	String
	<b>Multiplicity</b>	1
	<b>Description</b>	<p>Specifies a business event that leads to the termination of the business process use case.</p>
	<b>postCondition</b>	
	<b>Type</b>	String
	<b>Multiplicity</b>	1
	<b>Description</b>	<p>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.</p>
	<b>exceptions</b>	

	<b>Type</b>	String
	<b>Multiplicity</b>	1..*
	<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>	1..*
	<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.

484

<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.

485

<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.

486

<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.

487

<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 MUST be used as base class for the business process action
<b>Tag Definition</b>	No tagged values.

488

<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.

489

<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.

490

Form for Business Domain View	
<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)

Form for Business Area	
<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)



<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)

494

495

Form for Business Process	
<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 <b>and their states</b>	

497

498 **5.1.1.5 Constraints (normative)**

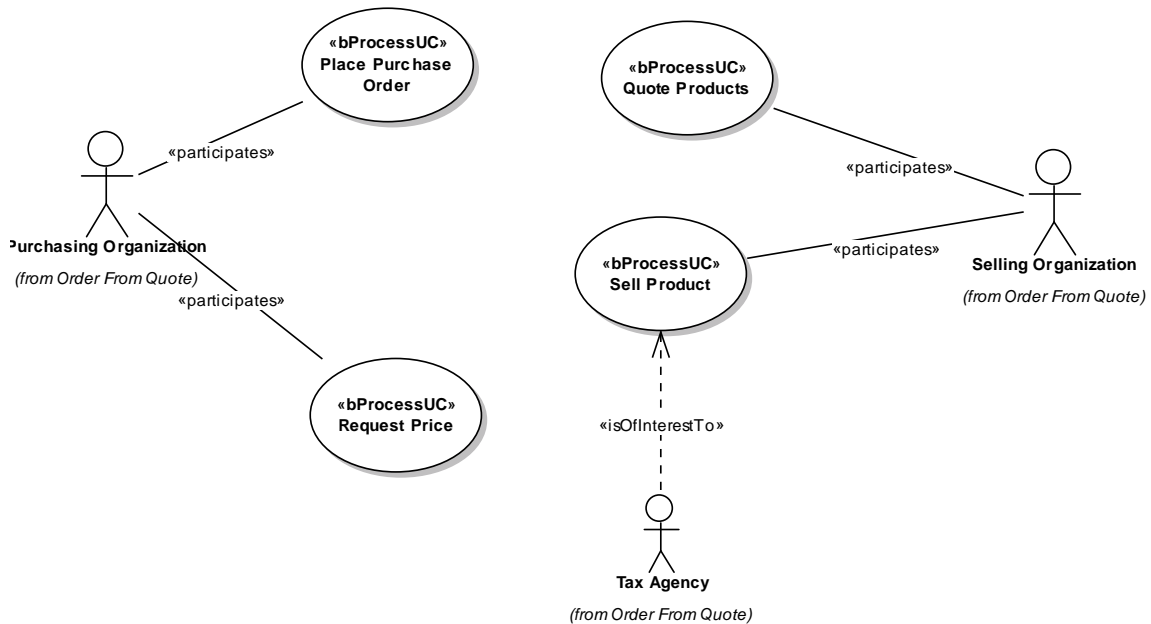
499 **C.9. A BusinessDomainView MUST include one to many BusinessAreas.**

500 **C.10. A BusinessArea MUST include one to many BusinessAreas or one to many ProcessAreas or**  
 501 **one to many BusinessProcessUseCases.**

502 **C.11. A ProcessArea MUST contain one to many other ProcessAreas or one to many**  
 503 **BusinessProcessUseCases**

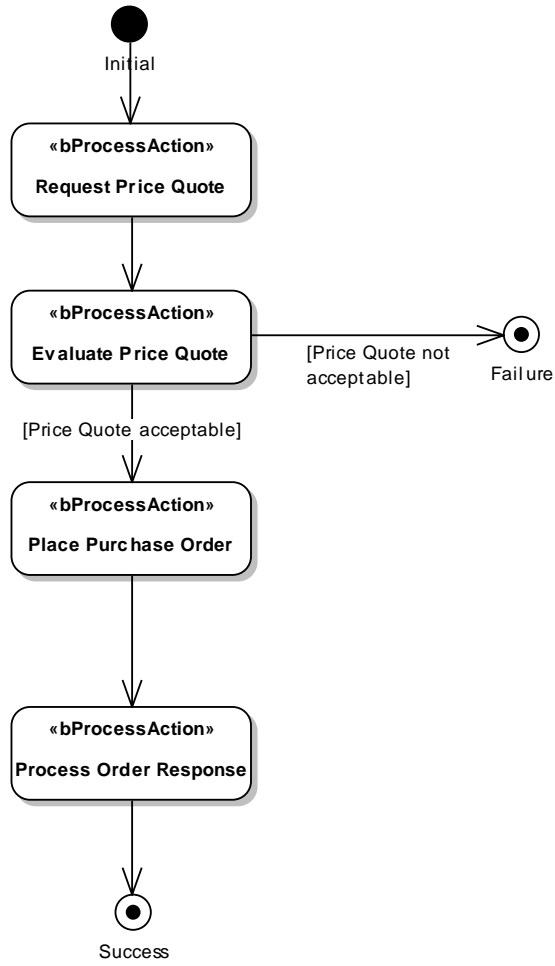
- 504 C.12. A *BusinessProcessUseCase* MUST be associated with one to many *BusinessPartners* using the  
 505 *participates* relationship
- 506 C.13. A *BusinessProcessUseCase* MAY be associated with zero to many *Stakeholders* using the  
 507 *isOfInterestTo* relationship
- 508 C.14. A *BusinessProcessUseCase* SHOULD be refined by zero to many *BusinessProcesses*. These  
 509 relationships MAY also be visualized by realize relationships from each of the owned  
 510 *BusinessProcesses* to the owning *BusinessProcessUseCase*
- 511 C.15. A *BusinessProcess* MUST be modeled as a child of a *BusinessProcessUseCase*
- 512 C.16. A *BusinessProcessUseCase* MAY be refined by zero to many UML Sequence Diagrams
- 513 C.17. A *BusinessProcess* MAY contain zero to many *ActivityPartitions*
- 514 C.18. A *BusinessProcess*, which has no *ActivityPartitions*, MUST contain one or more  
 515 *BusinessProcessActions* and MAY contain zero to many *InternalBusinessEntityStates* and zero to  
 516 many *SharedBusinessEntityStates*.
- 517 C.19. An *ActivityPartition* being part of a *BusinessProcess* MUST contain one to many  
 518 *BusinessProcessActions* and MAY contain zero to many *InternalBusinessEntityStates*.
- 519 C.20. A *SharedBusinessEntityState* MUST NOT be located in an *ActivityPartition*. (They must be  
 520 contained within the *BusinessProcess* even if this *BusinessProcess* contains *ActivityPartitions*.)

521 5.1.1.6 Example (informative)



522

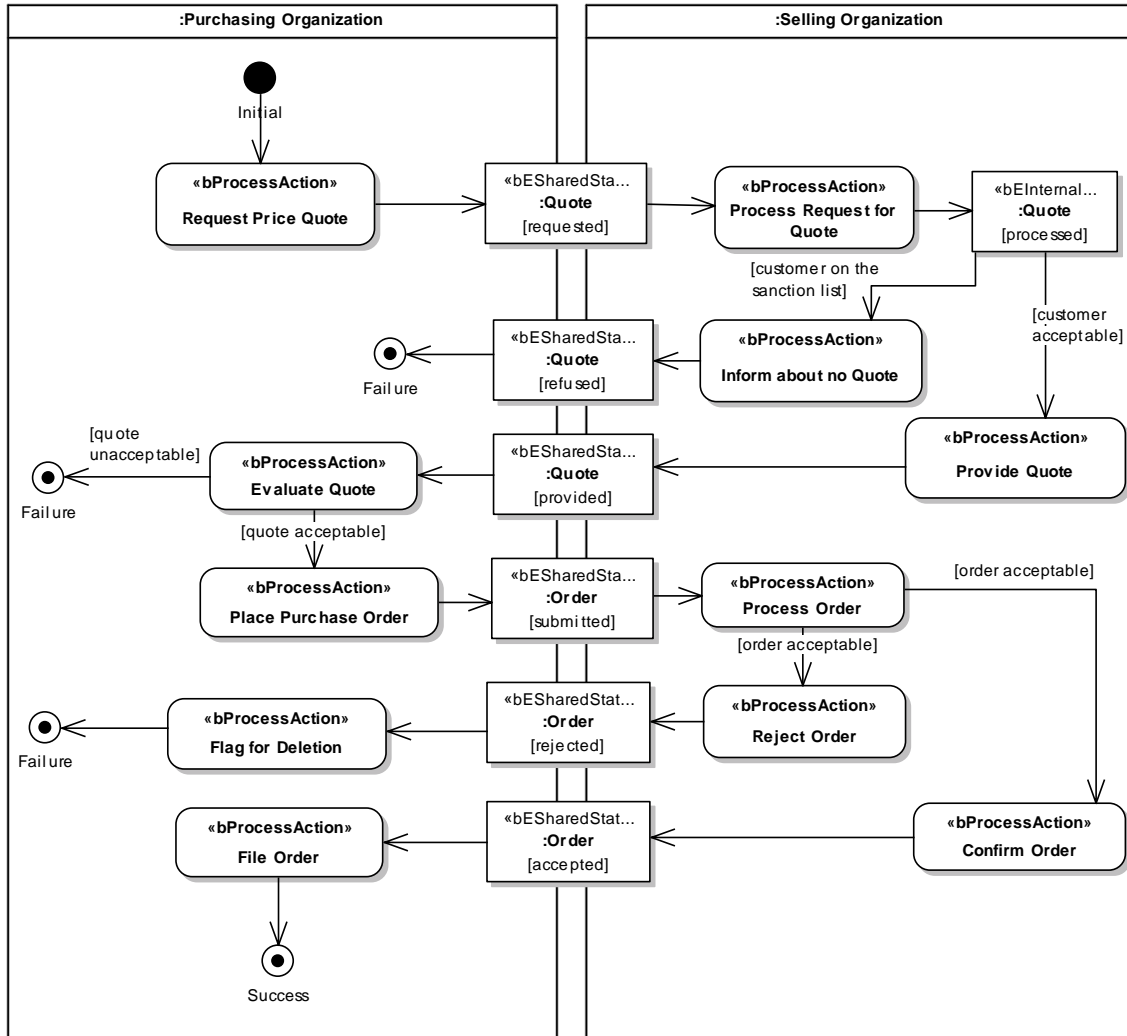
523 Figure 11 Business Domain View Example: Negotiation In the Order from Quote Example (Use Case Diagram showing  
 524 Business Process Use Cases)



525

526  
527

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



528

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

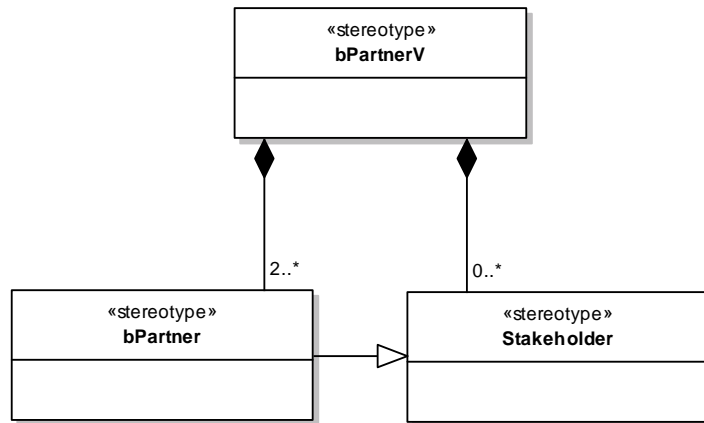
531

532 **5.1.2 Business Partner View**

533 **5.1.2.1 Abbreviations of Stereotypes**

Stereotype Abbreviation	Full Stereotype Name
bPartnerV	BusinessPartnerView
bPartner	BusinessPartner
Stakeholder	Stakeholder

534 **5.1.2.2 Conceptual Description (informative)**

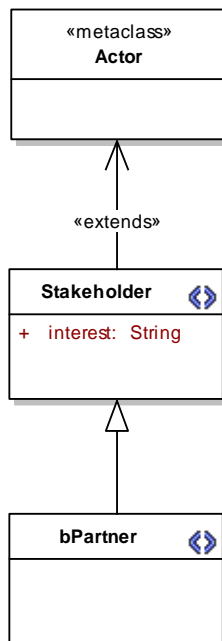


535

536 **Figure 14 BusinessPartnerView – Conceptual Overview**

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

545 **5.1.2.3 Stereotypes and Tag Definitions (normative)**



546

547 **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.

548

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

549 **5.1.2.4 Constraints (normative)**

550 C.21. A *BusinessPartnerView* MUST contain at least two to many *BusinessPartners*. If the  
551 *BusinessPartnerView* is hierarchically decomposed into sub packages these *BusinessPartners* MAY be  
552 contained in any of these sub packages.

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

554 **5.1.2.5 Example (informative)**



555

556

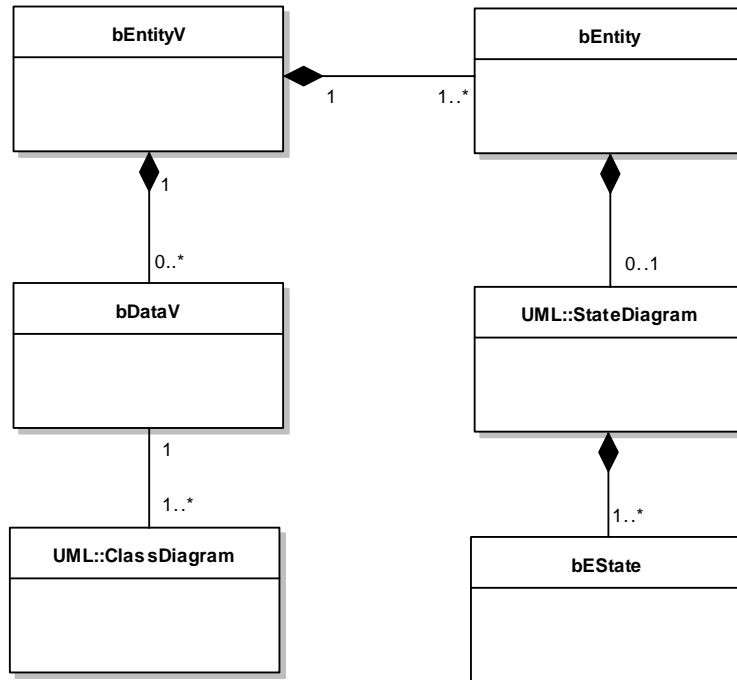
**Figure 16 Business Partner View Example**

557 **5.1.3 Business Entity View**

558 **5.1.3.1 Abbreviations of Stereotypes**

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

559 **5.1.3.2 Conceptual Description (informative)**



560

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

562

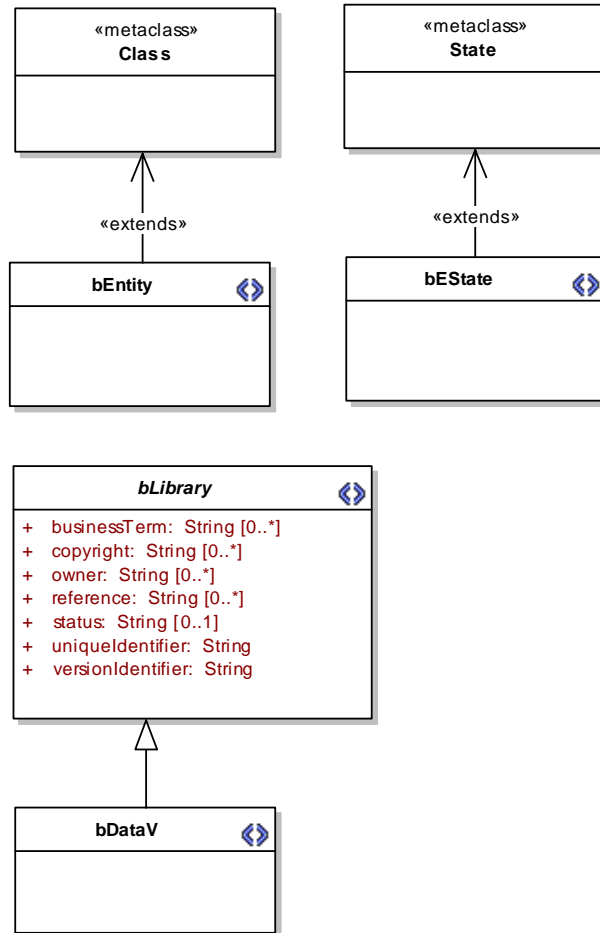
563 A business entity is a real-world thing having business significance that is shared between two or more  
 564 business partners in a collaborative business process (e.g. “order”, “account”, etc.). Within the business  
 565 entity view at least one, but possibly more business entities are described.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

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



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

577 **5.1.3.3 Stereotypes and Tag Definitions (normative)**



578

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

580

Stereotype	bEntity (BusinessEntity)
Base Class	Class
Parent	N/A
Description	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.).
Tag Definition	No tagged values.

581

582

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

583

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>

584 **5.1.3.4 Constraints (normative)**

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

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

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

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

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

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

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

595 **5.1.3.5 Worksheets**

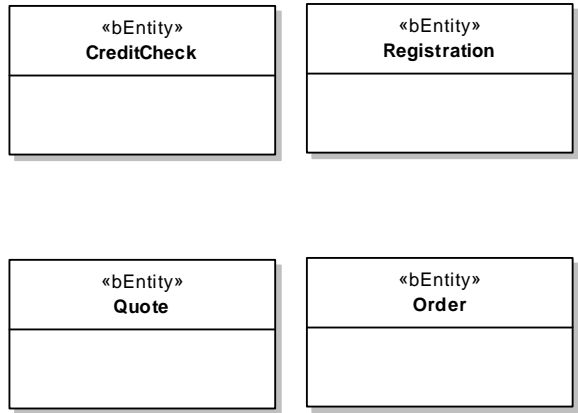
<b>Form for Business Entity</b>	
<b>General</b>	
Business Entity Name	
Description	

<b>Business Library Information</b>	
UniqueIdentifier	
BusinessTerm	
VersionIdentifier	
Status	
Owner	
Copyright	
Reference(s)	
<b>Lifecycle</b>	
Pre-Condition	
Post-Condition	
Begins When	
Ends When	
Exceptions	
<b>Lifecycle States (add more Business Entity States if needed)</b>	
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	

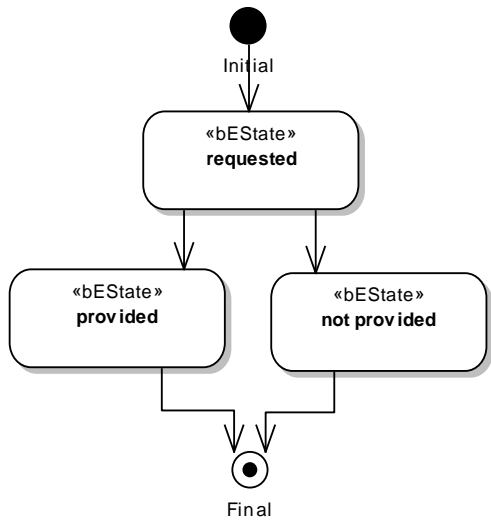
596

597 5.1.3.6 Example (informative)



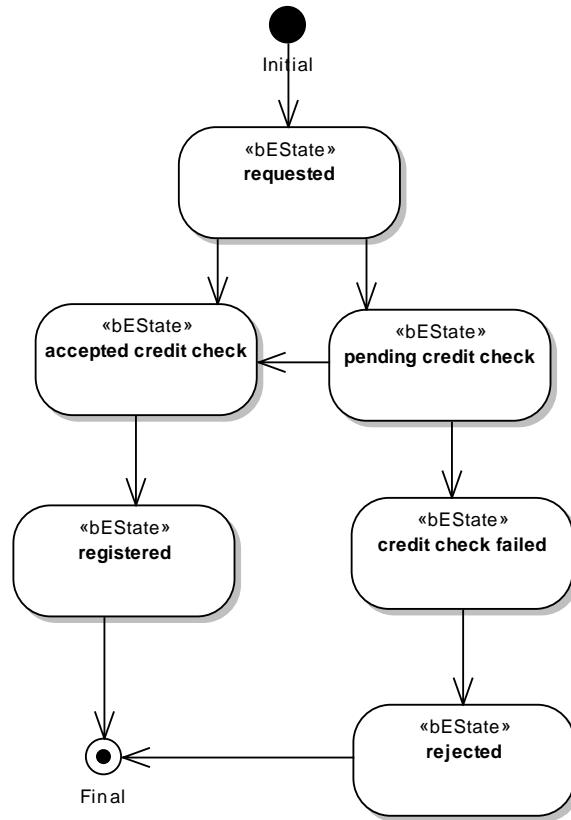
598

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



600

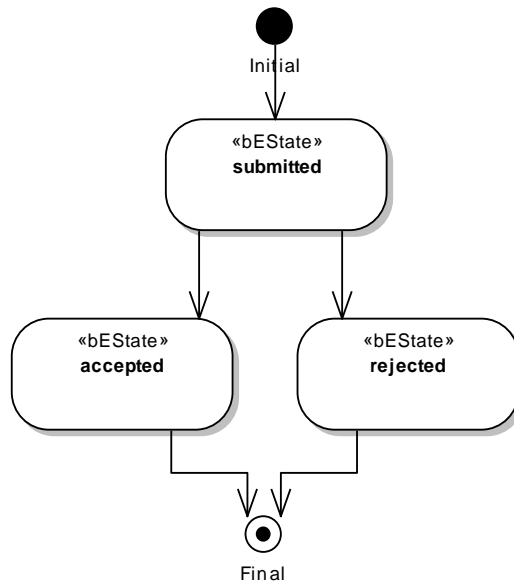
601 Figure 20 BusinessEntityView Example: CreditCeck Lifecycle (State Diagram)



602

603

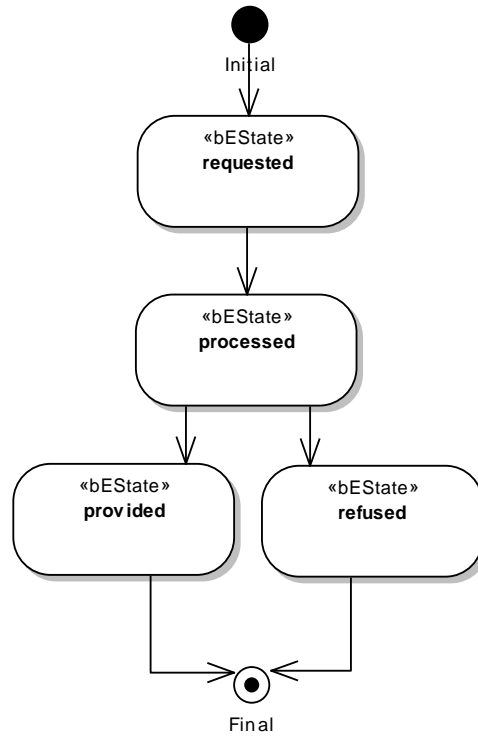
Figure 21 BusinessEntityView Example: Registration Lifecycle (State Diagram)



604

605

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



606

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

608

609 **5.2 Business Choreography View**

610 **5.2.1 Sub-Views in the Business Choreography View**

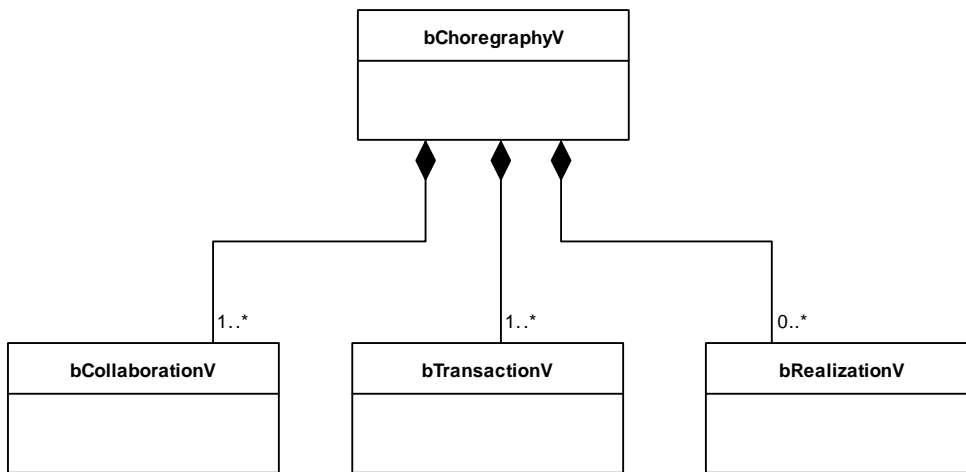
611 **5.2.1.1 Abbreviations and Stereotypes**

612

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

613

614 5.2.1.2 Conceptual Description (informative)



615

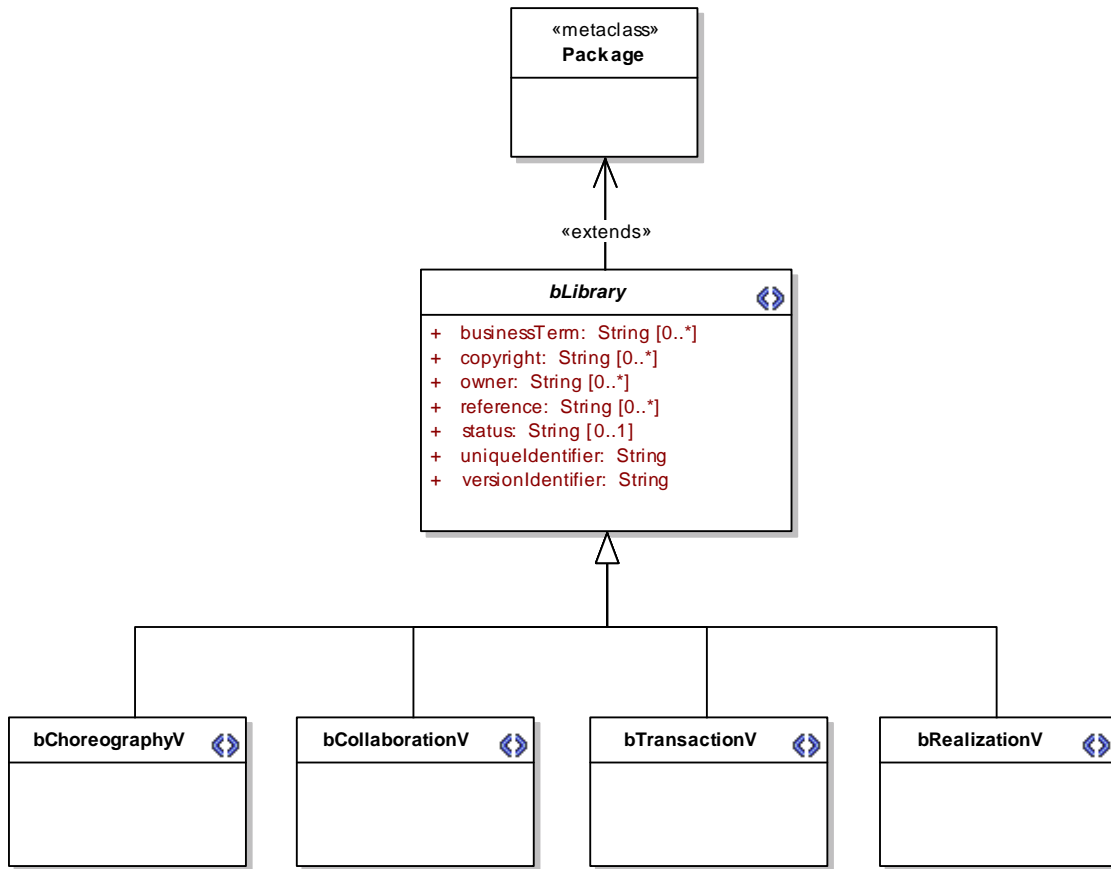
616

**Figure 24 BusinessChoreographyView Conceptual Overview**

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



634 5.2.1.3 Stereotypes and Tag Definitions (normative)



635  
636  
637

Figure 25 BusinessChoreographyView Abstract Syntax

<b>Stereotype</b>	<b>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>

638

<b>Stereotype</b>	<b>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>

639

<b>Stereotype 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>

640

<b>Stereotype 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>

641

642

643 **5.2.1.4 Constraints (normative)**

644 Constraints with respect to a *BusinessChoreographyView*:

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

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

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

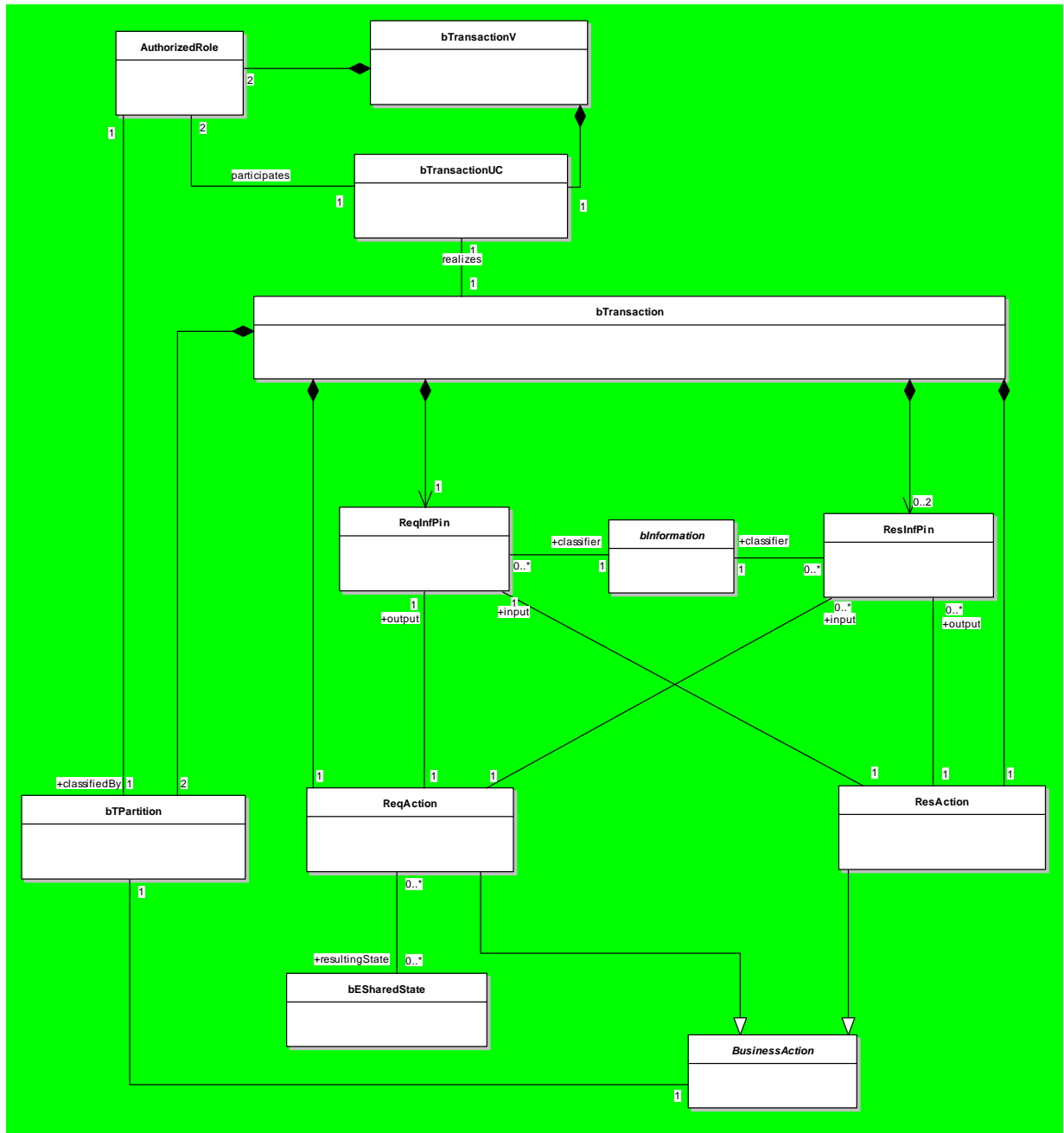
648 C.33. A *BusinessTransactionView*, a *BusinessCollaborationView*, and a *BusinessRealizationView* MUST be  
649 directly located under a *BusinessChoreographyView*

650 **5.2.2 Business Transaction View**

651 **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

652



655

656 **Figure 26 Business Transaction View – Conceptual Overview**

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

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

#### 664 5.2.2.2.1 Business Transaction Use Case Diagram

665 Two authorized roles participate in a business transaction use case. These authorized roles must be defined  
666 in the same business transaction view package as the corresponding business transaction use case.  
667 Accordingly, a *BusinessTransactionView* is composed of exactly two *AuthorizedRoles*. This means, if a certain  
668 role (e.g. buyer, seller, etc.) participates in multiple business transactions, it requires a different authorized  
669 role for each business transaction use case. Each authorized role of the same role (i.e., with the same name)  
670 is in a different namespace of a corresponding business transaction view. Therefore, an authorized role  
671 participates in only one business transaction use case – it is the one in the same business transaction view.  
672 Accordingly, *BusinessTransactionUseCase* and *AuthorizedRole* are related by a 1 to 2 association. It is  
673 important to note, that the same authorized role is not associated twice to the same business transaction  
674 use case.

#### 675 5.2.2.2.2 Business Transaction Diagram

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

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

689 Within a business transaction each authorized role performs exactly one business action – the requesting  
690 authorized role performs a requesting business action and the responding authorized role performs a  
691 responding business action. Each business action – no matter whether requesting or responding business  
692 action – is assigned to a swimlane, and each swimlane comprises exactly one business action. It follows that  
693 a *BusinessTransaction* is composed of exactly one *RequestingBusinessAction* and exactly one  
694 *RespondingBusinessAction*. Both, *RequestingBusinessAction* and *RespondingBusinessAction* are  
695 specializations of the abstract type *BusinessAction*. A *BusinessAction* is assigned to one  
696 *BusinessTransactionPartition*, and a *BusinessTransactionPartition* comprises one *BusinessAction*. Since a  
697 partition is dedicated to exactly one authorized role, it follows that the business action is executed by this  
698 authorized role. Furthermore an authorized role executes just one business action, because only one  
699 business action sits within a partition.

700 The requesting business action outputs the requesting information through the requesting information pin  
701 that is input to the responding business action’s requesting information pin. Business information created by  
702 the responding business action and returned to the requesting business action is optional. If business

703 information is returned by the responding business action zero to many responding information pins might  
704 be specified. Multiple responding information pins may be used to describe different business intentions  
705 (e.g., a positive and a negative response to a purchase order).

706 It follows, that a *BusinessTransaction* is composed of exactly two *RequestingInformationPins* and zero to  
707 many *RespondingInformationPins*. Both *RequestingInformationPin* and *RespondingInformationPin* are  
708 instances of the type *BusinessInformation*. A *RequestingBusinessAction* has exactly one  
709 *RequestingInformationPin* and zero to many *RespondingInformationPins*.

710 A *RespondingBusinessAction* has exactly one *RequestingInformationPin* and zero to many  
711 *RespondingInformationPins*.

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

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

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

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



738

739

Figure 27 Business Transaction View – Abstract Syntax

Stereotype	bTransactionUC (BusinessTransactionUseCase)
Base Class	UseCase
Parent	bProcessUC
Description	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.
Tag Definition	Inherited tagged values:

	<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>
--	--

740

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.

741

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>	
<b>Tag Definition</b>	<b>businessTransactionType</b>	
	<b>Type</b>	<p>Enumeration:</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> <p>Authenticate sender identity – Verify the identity of the sender (employee or</p>	

		<p>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>
--	--	--

742

Stereotype	bTPartition (BusinessTransactionPartition)
Base Class	Partition
Parent	N/A
Description	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.
Tag Definition	<b>No Tagged Values</b>

743

Stereotype	BusinessAction (BusinessAction, abstract)	
Base Class	Action	
Parent	N/A	
Description	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.	
Tag Definition	isAuthorizationRequired	
	Type	Boolean
	Multiplicity	1
	Description	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.
	isNonRepudiationRequired	
	Type	Boolean
Multiplicity	1	

<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>timeToAcknowledge Receipt</b>	
<b>Type</b>	<b>TimeExpression</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>TimeExpression</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, we use again the terms sender and affirmant in the explanation of the

		<p>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>

744

<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>	TimeExpression
	<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 send notification of failed business control (possibly revoking a contractual offer) if</p>

	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	<p><b>Inherited tagged values:</b></p> <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> <p>Default assignment of tagged values for the responding business action:</p> <table border="1"> <thead> <tr> <th></th> <th>Time to Acknowledge Receipt</th> <th>Time to Acknowledge Processing</th> <th>Is Authorization Required</th> <th>Is Non Repudiation Required</th> <th>Is Non Repudiation of Receipt Required</th> <th>Is Intelligible Check Required</th> </tr> </thead> <tbody> <tr> <td>Commercial Transaction</td> <td>2h</td> <td>6hr</td> <td>TRUE</td> <td>TRUE</td> <td>TRUE</td> <td>TRUE</td> </tr> <tr> <td>Request/Confirm</td> <td>2h</td> <td>NULL</td> <td>TRUE</td> <td>FALSE</td> <td>TRUE</td> <td>TRUE</td> </tr> <tr> <td>Request/Response</td> <td>NULL</td> <td>NULL</td> <td>FALSE</td> <td>FALSE</td> <td>FALSE</td> <td>TRUE</td> </tr> <tr> <td>Query/Response</td> <td>NULL</td> <td>NULL</td> <td>FALSE</td> <td>FALSE</td> <td>FALSE</td> <td>TRUE</td> </tr> <tr> <td>Notification</td> <td>NULL</td> <td>NULL</td> <td>FALSE</td> <td>FALSE</td> <td>FALSE</td> <td>TRUE</td> </tr> <tr> <td>Information Distribution</td> <td>NULL</td> <td>NULL</td> <td>FALSE</td> <td>FALSE</td> <td>FALSE</td> <td>TRUE</td> </tr> </tbody> </table>							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
	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.	
Tag Definition	<b>isConfidential</b>	
	Type	Boolean
	Multiplicity	1
	Description	If the flag is set, the exchanged information is encrypted so that unauthorized parties cannot view the information.
	<b>isTamperProof</b>	
	Type	Boolean
	Multiplicity	1
	Description	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>	
	Type	Boolean
	Multiplicity	1
	Description	If the flag is set, there is a digital certificate associated with the document entity. This provides proof of the signer's identity.

748

<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

749

<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

750

751 **5.2.2.4 Constraints (normative)**

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



- 784 C.50. A RequestingBusinessAction and a RespondingBusinessAction MUST embed same number of  
 785 RespondingInformationPins.
- 786 C.51. The RequestingInformationPin of the RequestingBusinessAction MUST be connected with the  
 787 RequestingInformationPin of the RespondingBusinessAction using an object flow relationship leading  
 788 from the RequestingBusinessAction to the RespondingBusinessAction.
- 789 C.52. Each RespondingInformationPin of the RespondingBusinessAction MUST be connected with  
 790 exactly one RespondingInformationPin of the RequestingBusinessAction using an object flow  
 791 relationship leading from the RespondingBusinessAction to the RequestingBusinessAction
- 792 C.53. If a BusinessTransactionPartition contains SharedBusinessEntityStates, each  
 793 SharedBusinessEntityState MUST be the target of exactly one control flow relationship starting from  
 794 the RequestingBusinessAction and MUST be the source of exactly one control flow relationship  
 795 targeting a FinalState.
- 796 C.54. Each FinalState MUST be the target of one to many control flow relationships starting from  
 797 the RequestingBusinessAction or from a SharedBusinessEntityState.
- 798 C.55. Each RequestingInformationPin and each RespondingInformationPin MUST have a classifier,  
 799 this classifier MUST be an InformationEnvelope or a subtype defined in an extension/specialization  
 800 module.
- 801 C.56. Two RequestingInformationPins which are connected using an object flow MUST have the  
 802 same classifier.
- 803 C.57. Two RespondingInformationPins which are connected using an object flow MUST have the  
 804 same classifier.

805 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	

806

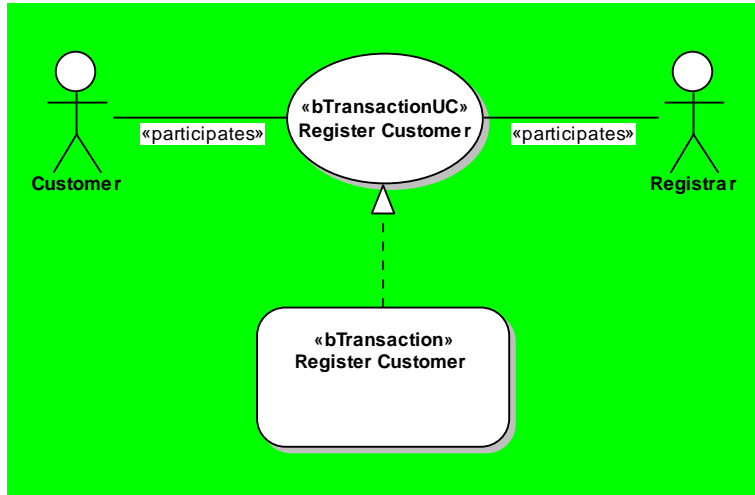
<b>Form for Business Transaction</b>	
<b>General</b>	
Name	
Description	
<b>Business Library Information</b>	
UniqueIdentifier	
BusinessTerm	
VersionIdentifier	
Status	
Owner	
Copyright	
Reference(s)	
<b>Details</b>	

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?	

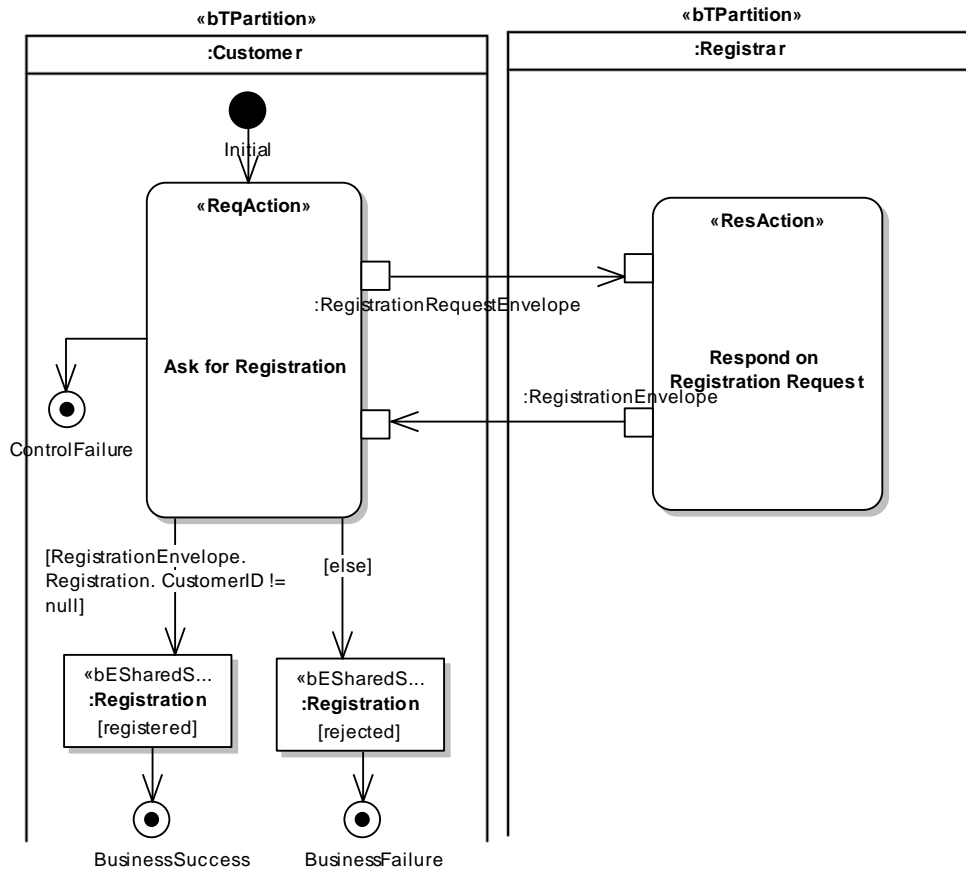
807

808 5.2.2.6 Example (informative)



809

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



812

813

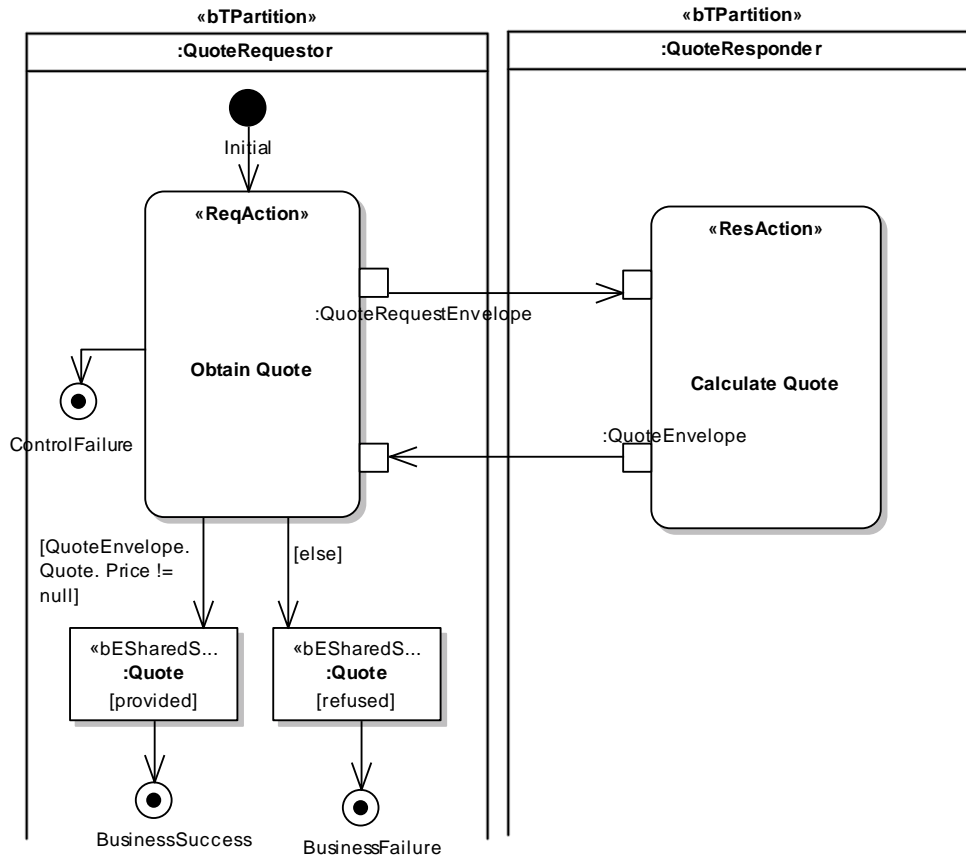
Figure 29 Business Transaction Example: Register Customer

814



815

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



817

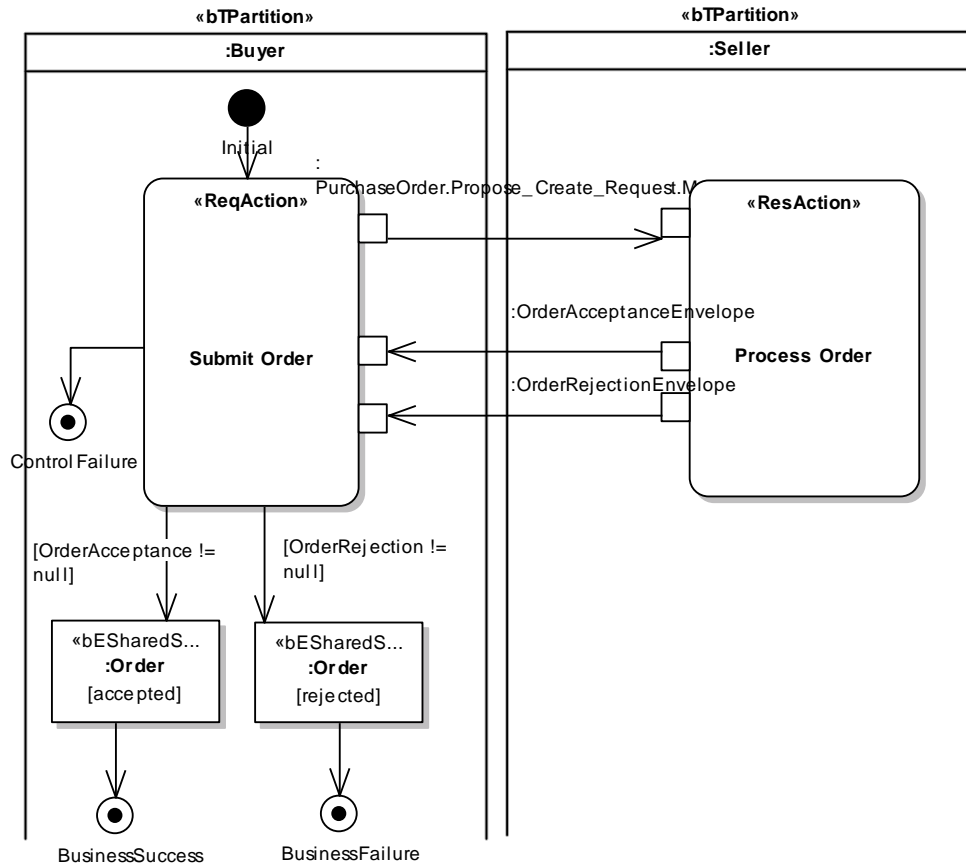
818 Figure 31 Business Transaction Example: Request For Quote



819

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

821



822

823

Figure 33 Business Transaction Example: Place Order

824

825 **5.2.3 Business Collaboration View**

826 **5.2.3.1 Abbreviations of Stereotypes**

Stereotype Abbreviation	Full Stereotype Name
bCollaborationV	BusinessCollaborationView
AuthorizedRole	AuthorizedRole
bCollaborationUC	BusinessCollaborationUseCase
bCollaborationProtocol	BusinessCollaborationProtocol
bCPartition	BusinessCollaborationPartition
bTransactionAction	BusinessTransactionAction
bCollaborationAction	BusinessCollaborationAction
bNestedCollaboration	NestedBusinessCollaboration

827





849 A business collaboration use case may include additional business collaboration use cases. A business  
850 collaboration use case may optionally have multiple parent business collaboration use cases. Hence,  
851 *BusinessCollaborationUseCase* has a unary (0..n) to (0..n) include-composition. A business collaboration use  
852 case may include multiple business transaction use cases. A business transaction use case must be included  
853 in at least one business collaboration use case. Consequently, an (1..n) to (0..n) aggregation between  
854 *BusinessCollaborationUseCase* and *BusinessTransactionUseCase* exists. It is important that a business  
855 collaboration use case includes at minimum one use case – no matter whether this is a business  
856 collaboration use case or a business transaction use case. A hierarchy of business collaboration use cases  
857 built by include-compositions must not include any cycles. A business transaction uses case cannot be  
858 further decomposed by an include-association.

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

#### 862 5.2.3.2.2 Business Collaboration Protocol Diagram

863 A business choreography view is used to define the business choreography of exactly one business  
864 collaboration. The *BusinessCollaborationProtocol* follows exactly the requirements defined by the  
865 corresponding *BusinessCollaborationUseCase*. The business collaboration protocol that describes the  
866 business collaboration use case is defined as a child beneath. Accordingly, each  
867 *BusinessCollaborationUseCase* has exactly one *BusinessCollaborationProtocol* beneath. A business  
868 collaboration protocol is a “composite” UML *Activity*. The diagram of a business collaboration protocol is  
869 described by a flow of UML *Actions*.

870 A business collaboration protocol defines the collaborative business process between two or more  
871 authorized roles. The business collaboration protocol must have a *BusinessCollaborationPartiton* for each of  
872 the authorized roles defined in the *BusinessCollaborationView*. Hence, a *BusinessCollaborationProtocol* is  
873 composed of two or more *BusinessCollaborationPartitions*. Each *BusinessCollaborationPartition* relates to  
874 one and only one *AuthorizedRole* defined in the *BusinessCollaborationView*. Each *AuthorizedRole* in the  
875 *BusinessCollaborationView* is assigned to exactly one *BusinessCollaborationPartition*. It follows, that the  
876 every *BusinessCollaborationPartiton* of a *BusinessCollaborationProtocol* must be assigned to different  
877 authorized roles.

878 The collaborative actions of a business collaboration protocol are business collaboration actions and/or  
879 business transaction actions. Hence, a *BusinessCollaborationProtocol* is composed of zero to many  
880 *BusinessCollaborationActions* and of zero to many *BusinessTransactionActions*. However, at least one  
881 business collaboration action or at least one business transaction action must be present in a business  
882 collaboration protocol. Transitions defining the flow among the business collaboration activities and/or  
883 business transaction activities may be guarded by the states of business entities.

884 A business collaboration action is characterized by the fact that it is executed by calling a business  
885 collaboration protocol. This calling behavior is accomplished by classifying the behavior of the business  
886 collaboration action by the desired business collaboration protocol. Not every business collaboration  
887 protocol is a called by a business collaboration action. A business collaboration protocol may be called by  
888 multiple business collaboration actions. Thus, the behavioral classifying relationship between  
889 *BusinessCollabortionAction* and *BusinessCollaborationProtocol* is (0..n) to 1.

890 A business transaction action is characterized by the fact that it is executed by calling a business transaction.  
891 This calling behavior is accomplished by classifying the behavior of the business transaction action by the  
892 desired business transaction. Since the business transaction is a concept of the business transaction view it is  
893 described in more detail above. Each business transaction must be at least once used to refine a business  
894 transaction action. A business transaction may be called by many business transaction actions. Hence, the  
895 behavioral classifying relationship between *BusinessTransactionAction* and *BusinessTransaction* is (1..n) to 1.

896 In many scenarios, there is a requirement for a nested business collaboration within the scope of execution  
897 of a given business transaction action. In other words, before a responding authorized role can send a  
898 response as required by the business transaction action that calls a two-way business transaction, the  
899 responding authorized role has to first participate in a business collaboration with other business partners.  
900 UMM supports this scenario by introducing the concept of a *NestedBusinessCollaboration*. Like the business  
901 collaboration action, the *NestedBusinessCollaboration* is characterized by the fact that it is executed by  
902 calling another business collaboration protocol. This calling behavior is accomplished by classifying the  
903 behavior of the *NestedBusinessCollaboration* by the desired business collaboration protocol. Not every  
904 business collaboration protocol is called by a *NestedBusinessCollaboration*. A business collaboration  
905 protocol may be called by multiple *NestedBusinessCollaborations*. Thus, the behavioral classifying  
906 relationship between *NestedBusinessCollaboration* and *BusinessCollaborationProtocol* is (0..n) to 1.  
907 However, unlike the business collaboration action, the *NestedBusinessCollaboration* must reside within a  
908 business collaboration partition representing the responding authorized role of a given business transaction  
909 action. Accordingly, not every business collaboration partition representing a business transaction action  
910 responding role includes a *NestedBusinessCollaboration*. Thus, there is a 1 to (0..n) composition between a  
911 *BusinessCollaborationPartiton* and a *NestedBusinessCollaboration*.

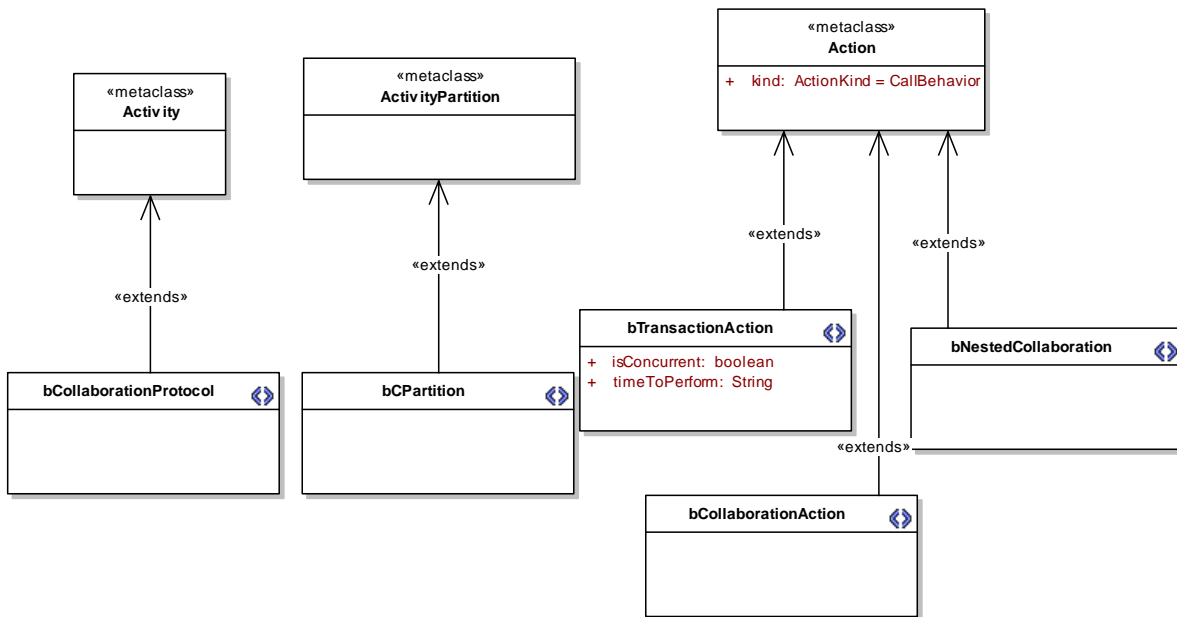
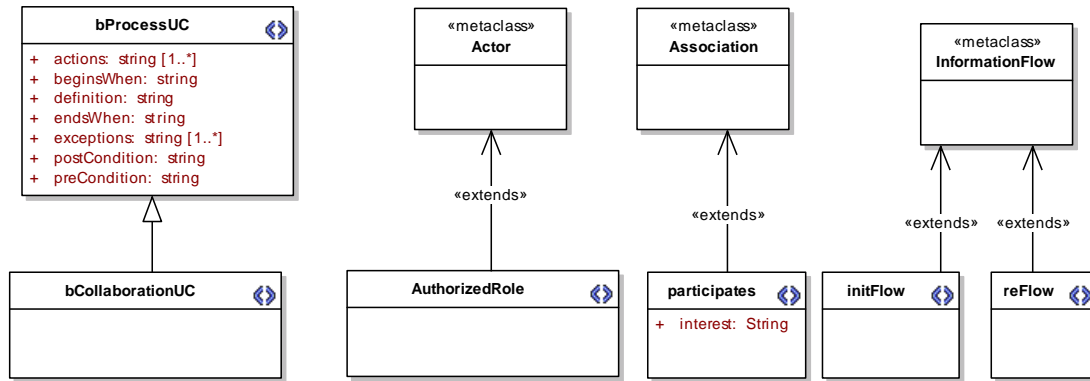
912 In UMM 2.0, role mapping between business collaboration authorized roles and either called business  
913 transaction authorized roles or business collaboration protocol authorized roles is defined in the business  
914 collaboration protocol and no longer by the business collaboration use case. This role mapping is  
915 accomplished by information flows and specializations of information flows, i.e. *InitiatingFlow* and  
916 *RespondingFlow*, between either business collaboration partitions or nested collaborations and either  
917 business collaboration actions or business transaction actions. Using the approach also enhances the  
918 business collaboration protocol by graphically illustrating the relationships between authorized roles and the  
919 choreography of actions within a business collaboration protocol. This mapping approach is defined by the  
920 following cases:

921 1. *From business collaboration partitions to business collaboration actions*. The UML 2.0 Information  
922 flow is used. The source of Information flow must be the business collaboration partition and the  
923 target must be the business collaboration action. This means that the authorized role in this business  
924 collaboration protocol is participating in the called business collaboration protocol. Therefore the  
925 calling business collaboration protocol must have at least the same number of business collaboration  
926 partitions as the number of business collaboration partitions in the called business collaboration  
927 protocol. There are two cases for defining a role mapping:

928 • The authorized role name of the calling business collaboration protocol maps to an  
929 authorized role in the called business collaboration protocol which has exactly the same  
930 name. In this case, the information flow association explicitly defines the role mapping  
931 between these two roles.

- 932                   • The authorized role name of the calling business collaboration protocol maps to an  
933                   authorized role in the called business collaboration protocol which has a different name. In  
934                   this case, the information flow association must be classified by the authorized role of the  
935                   called business collaboration protocol.
- 936           2. *From business collaboration partitions to business transaction actions.* In this case two  
937           specializations of the Information flow are used, *InitiatingFlow* and *RespondingFlow*. In all cases an  
938           authorized role of a business collaboration protocol initiates a business transaction action, which  
939           calls a business transaction, which also has an initiating authorized role. To provide this role mapping  
940           the business collaboration partition classified by an authorized role is the source of the  
941           *InitiatingFlow* and the business transaction action is the target. Likewise for two-way business  
942           transactions, a business collaboration partition is the source of the *RespondingFlow* and the business  
943           transaction action is the target.
- 944           3. *From business transaction actions to business collaboration partitions.* In this case two  
945           specializations of the Information Flow are used, *InitiatingFlow* and *RespondingFlow*. In all cases an  
946           authorized role of a business collaboration protocol responds to a business transaction action which  
947           calls a business transaction which also has a responding authorized role. To provide this role  
948           mapping the business transaction action is the source of an *InitiatingFlow* and a business  
949           collaboration partition is the target. Therefore the authorized role that classifies the business  
950           collaboration partition maps to the responding business transaction authorized role. Likewise, for  
951           two-way business transactions, a business transaction action is the source of a *RespondingFlow* and  
952           a business collaboration partition is the target.
- 953           4. *From business transaction actions to nested business collaborations.* In this case only one  
954           specialization of the Information Flow is used, *InitiatingFlow*. The source of the *InitiatingFlow* is a  
955           business transaction action and the target is a nested business collaboration. This means that the  
956           responding authorized role of the business collaboration initiates the business collaboration protocol  
957           called by the nested business collaboration and therefore maps to initiating authorized role of the  
958           called business collaboration protocol.
- 959           5. *From nested business collaborations to business transaction actions.* In this case only one  
960           specialization of the Information Flow is used, *RespondingFlow* and only applies in the case of two-  
961           way business transactions. The *RespondingFlow* indicates that the called business collaboration has  
962           completed and the responding authorized role can now return the response envelope to the  
963           initiating authorized role. The *NestedBusinessCollaboration* is the source of the *RespondingFlow* and  
964           a business transaction action is the target.

965 5.2.3.3 Stereotypes and Tag Definitions (normative)



966

967

Figure 35 Business Collaboration View - Abstract Syntax

968

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>Inherited tagged values:</b>

	<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>
--	--

969

<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.

970

<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 actions and/or business collaboration actions. 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>

971

<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 action and which will initiate the nested collaboration.</p>
<b>Tag Definition</b>	No tagged values.

972

973

<b>Stereotype</b>	<b>bTransactionAction (BusinessTransactionAction)</b>
-------------------	---

<b>Base Class</b>	Action with call behaviour action kind (CallBehaviorAction)	
<b>Parent</b>	N/A	
<b>Description</b>	A business transaction action 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 action by one and only one business transaction. The business transaction action executes the called business transaction. The business transaction action 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>	TimeExpression
	<b>Multiplicity</b>	1
	<b>Description</b>	A business transaction action 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 action, 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 action is concurrent then more than one business transaction action 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 action is not concurrent then only one business transaction action of the same underlying business transaction can be open at one time.

974

<b>Stereotype</b>	<b>bCollaborationAction (BusinessCollaborationAction)</b>
<b>Base Class</b>	Action with call behaviour action kind (CallBehaviorAction)
<b>Parent</b>	N/A
<b>Description</b>	A business collaboration action is an action within a business collaboration protocol. This business collaboration action is refined by using the call behaviour to classify the behaviour of this business collaboration action 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>

975

<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 action after receiving a requesting information envelope which is represented as an <code>initFlow</code> (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 action indicated by a <code>reFlow</code> (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.

976

<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 action. The source of the initiating flow is the business collaboration partition that is classified by the authorized role initiating the business transaction action and the target is the business transaction action. In this case the initiating flow provides the authorized role mapping between the initiating role of the business transaction action and the initiating authorized role of the business transaction that is called by this business transaction action.</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 action and the target is the business collaboration partition which is classified by the authorized role responding in the business transaction action. In this case the initiating flow provides the authorized role mapping between the responding role of the business transaction action and the responding authorized role of the business transaction that is called by this business transaction action. 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.

977

<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 action. The source of the responding flow is the business collaboration partition that is classified by the authorized role responding in the business transaction action and the target is the business transaction action. 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</li> </ul>

	<p>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 action on the initiator's side. The source of this initiating flow is the business transaction action and the target is the business collaboration partition which is classified by the authorized role initiating the business transaction action.</li> </ul>
<b>Tag Definition</b>	No tagged values.

978 **5.2.3.4 Constraints (normative)**

- 979 C.58. A *BusinessCollaborationView* MUST contain exactly one *BusinessCollaborationUseCase*.
- 980 C.59. A *BusinessCollaborationView* MUST contain two to many *AuthorizedRoles*.
- 981 C.60. A *BusinessCollaborationUseCase* MUST have two to many *participates* associations to
- 982 *AuthorizedRoles* contained in the same *BusinessCollaborationView*.
- 983 C.61. Each *AuthorizedRole* contained in the *BusinessCollaborationView* MUST have exactly one
- 984 *participates* association to the *BusinessCollaborationUseCase* included in the same
- 985 *BusinessCollaborationView*.
- 986 C.62. A *BusinessCollaborationUseCase* MUST have one to many *include* relationships to another
- 987 *BusinessCollaborationUseCase* or to a *BusinessTransactionUseCase*.
- 988 C.63. Exactly one *BusinessCollaborationProtocol* MUST be placed beneath each
- 989 *BusinessCollaborationUseCase*. This relationship MAY also be visualized by a *realize* relationship
- 990 from the *BusinessCollaborationProtocol* to the *BusinessCollaborationUseCase*.
- 991 C.64. A *BusinessCollaborationProtocol* MUST contain one to many *BusinessTransactionActions*
- 992 and/or *BusinessCollaborationAction*.
- 993 C.65. Each *BusinessTransactionAction* MUST call exactly one *BusinessTransaction*
- 994 C.66. Each *BusinessTransaction* called by a *BusinessTransactionAction* MUST be placed beneath a
- 995 *BusinessTransactionUseCase* which is included in the *BusinessCollaborationUseCase* that covers the
- 996 corresponding *BusinessCollaborationProtocol*.
- 997 C.67. Each *BusinessCollaborationProtocol* called by a *BusinessCollaborationAction* MUST be placed
- 998 beneath a *BusinessCollaborationProtocolUseCase* which is included in the
- 999 *BusinessCollaborationUseCase* that covers the corresponding *BusinessCollaborationProtocol*.
- 1000 C.68. A *BusinessCollaborationProtocol* MUST contain two to many *BusinessCollaborationPartitions*.
- 1001 C.69. The number of *AuthorizedRoles* in the *BusinessCollaborationView* MUST match the number
- 1002 of *BusinessCollaborationPartitions* in the *BusinessCollaborationProtocol* which is placed beneath the
- 1003 *BusinessCollaborationUseCase* of the same *BusinessCollaborationView*.
- 1004 C.70. Each *AuthorizedRole* in the *BusinessCollaborationView* MUST be assigned to a
- 1005 *BusinessCollaborationPartition* in the *BusinessCollaborationProtocol* which is placed beneath the
- 1006 *BusinessCollaborationUseCase* of the same *BusinessCollaborationView*.
- 1007 C.71. Each *BusinessCollaborationPartition* MUST be classified by exactly one *AuthorizedRole*
- 1008 included in the same *BusinessCollaborationView* as the *BusinessCollaborationUseCase* covering the
- 1009 *BusinessCollaborationProtocol* containing this *BusinessCollaborationPartition*.
- 1010 C.72. A *BusinessCollaborationPartition* MUST be either empty or contain one to many
- 1011 *NestedBusinessCollaborations*.
- 1012 C.73. Each *BusinessTransactionAction* MUST be the target of exactly one *InitialFlow* which source
- 1013 MUST be a *BusinessCollaborationPartition*.



- 1014 C.74. Each *BusinessTransactionAction* MUST be the source of exactly one *InitialFlow* which target  
1015 MUST be either a *BusinessCollaborationPartition* or a *NestedBusinessCollaboration*.
- 1016 C.75. The *InitialFlow* sourcing from a *BusinessTransactionAction* and the *InitialFlow* targeting a  
1017 *BusinessTransactionAction* MUST NOT be targeting to / sourcing from the same  
1018 *BusinessCollaborationPartition*, nor targeting to a *NestedBusinessCollaboration* within the same  
1019 *BusinessCollaborationPartition*.
- 1020 C.76. If a *BusinessTransactionAction* calls a two-way *BusinessTransaction*, this  
1021 *BusinessTransactionAction* MUST be the source of exactly one *RespondingFlow* which target MUST  
1022 be a *BusinessCollaborationPartition*.
- 1023 C.77. If a *BusinessTransactionAction* calls a two-way *BusinessTransaction*, this  
1024 *BusinessTransactionAction* MUST be the target of exactly one *RespondingFlow* which source MUST  
1025 be either a *BusinessCollaborationPartition* or a *NestedBusinessCollaboration*.
- 1026 C.78. The *RespondingFlow* sourcing from a *BusinessTransactionAction* and the *RespondingFlow*  
1027 targeting a *BusinessTransactionAction* MUST NOT be targeting to /sourcing from the same  
1028 *BusinessCollaborationPartition*, nor targeting to a *NestedBusinessCollaboration* within the same  
1029 *BusinessCollaborationPartition*.
- 1030 C.79. If a *BusinessTransactionAction* calls a one-way *BusinessTransaction*, this  
1031 *BusinessTransactionAction* MUST NOT be the source of a *RespondingFlow* and MUST NOT be the  
1032 target of a *RespondingFlow*.
- 1033 C.80. The *RespondingFlow* targeting a *BusinessTransactionAction* must start from the  
1034 *BusinessCollaborationPartition* / *NestedBusinessCollaboration* which is the target of the *InitialFlow*  
1035 starting from the same *BusinessTransactionAction*.
- 1036 C.81. The *RespondingFlow* starting from a *BusinessTransactionAction* must target the  
1037 *BusinessCollaborationPartition* which is the source of the *InitialFlow* targeting to the same  
1038 *BusinessTransactionAction*.
- 1039 C.82. A *NestedBusinessCollaboration* MUST be the target of exactly one *InitialFlow*.
- 1040 C.83. A *NestedBusinessCollaboration* MAY be the source of a *RespondingFlow*, but MUST NOT be  
1041 the source of more than one *RespondingFlow*.
- 1042 C.84. A *BusinessCollaborationAction* MUST be the target of two to many *InformationFlows* (UML  
1043 standard: <<flow>>).
- 1044 C.85. A *BusinessCollaborationAction* MUST not be the source of an *InformationFlow*.
- 1045 C.86. A *BusinessCollaborationAction* MUST not be the source and MUST not be the target of an  
1046 *InitialFlow*.
- 1047 C.87. A *BusinessCollaborationAction* MUST not be the source and MUST not be the target of a  
1048 *RespondingFlow*.
- 1049 C.88. A *BusinessTransactionAction* MUST not be the source and MUST not be the target of an  
1050 *InformationFlow* (<<flow>>) that is neither stereotyped as *InitialFlow* nor as *RespondingFlow* nor is  
1051 of type <<flow>>.
- 1052 C.89. A *NestedBusinessCollaboration* MUST not be the source and MUST not be the target of an  
1053 *InformationFlow* that targets to / sources from a *BusinessCollaborationAction*.
- 1054 C.90. The number of *InformationFlows* targeting a *BusinessCollaborationAction* MUST match the  
1055 number of *BusinessCollaborationPartitions* contained in the *BusinessCollaborationProtocol* that is  
1056 called by this *BusinessCollaborationAction*.
- 1057 C.91. Either an *AuthorizedRole* classifying a *BusinessCollaborationPartition* that is the source of an  
1058 *InformationFlow* (UML standard: <<flow>>) targeting a *BusinessCollaborationAction* MUST match an

1059 AuthorizedRole classifying a *BusinessCollaborationPartition* in the *BusinessCollaborationProtocol*  
 1060 that is called by this *BusinessCollaborationAction* or the *InformationFlow* must be classified by an  
 1061 AuthorizedRole classifying a *BusinessCollaborationPartition* in the *BusinessCollaborationProtocol*  
 1062 that is called by this *BusinessCollaborationAction*.

1063 **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	

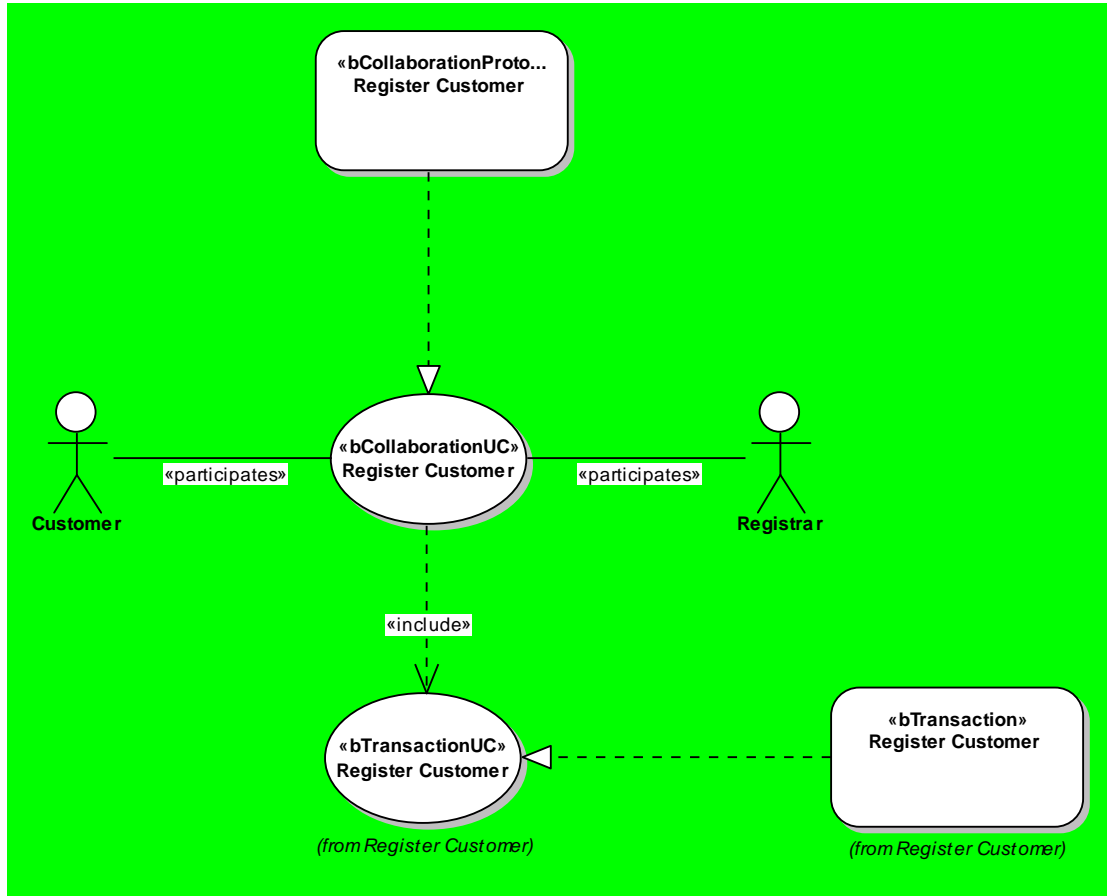
1064

<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 Actions / Business Collaboration Actions</b>	
<b>Business Transaction Action</b>	
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 Action		
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 Action [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 Action [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]		

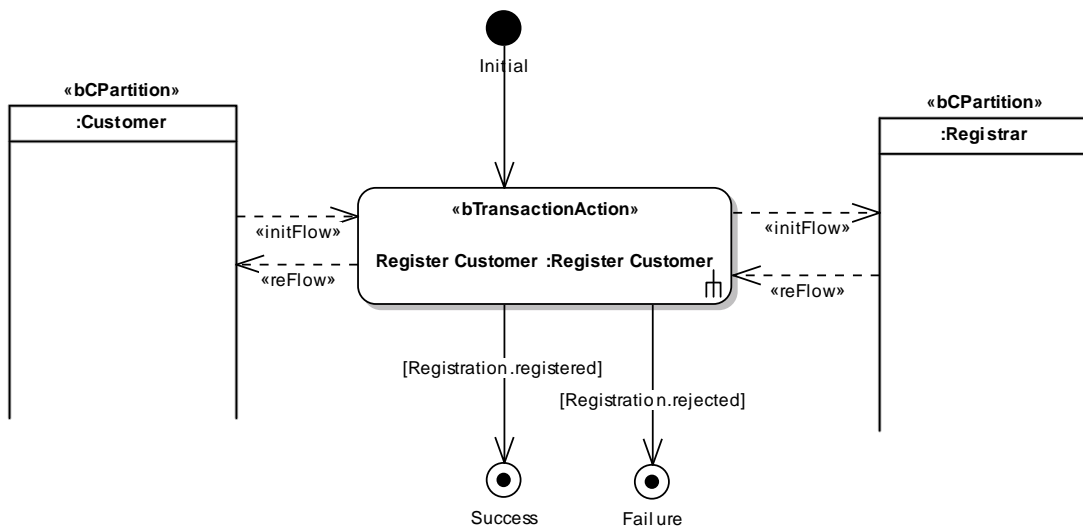
1065

1066 5.2.3.6 Normal Business Collaboration View Example (informative)



1067

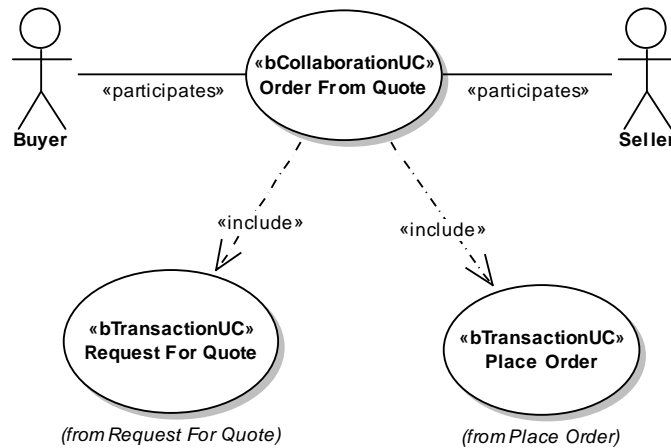
1068 Figure 36 Business Collaboration Use Case Example: Register Customer (including the optional realize relationships)



1069

1070 Figure 37 Business Collaboration Protocol Example: Register Customer

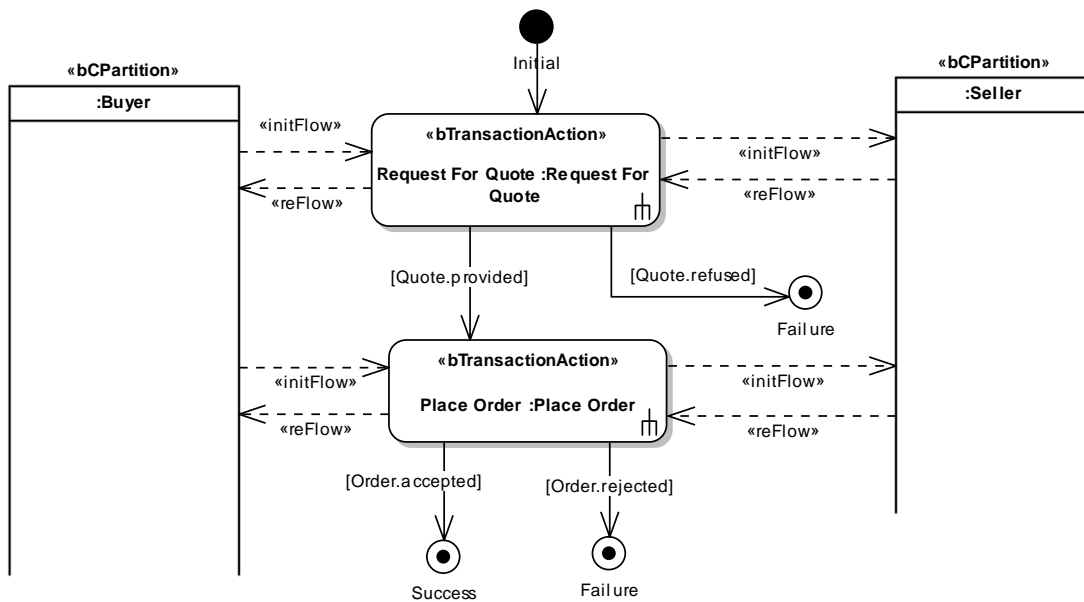
1071



1072

1073 Figure 38 Business Collaboration Use Case Example: Order From Quote (the optional realize relationships are not shown)

1074

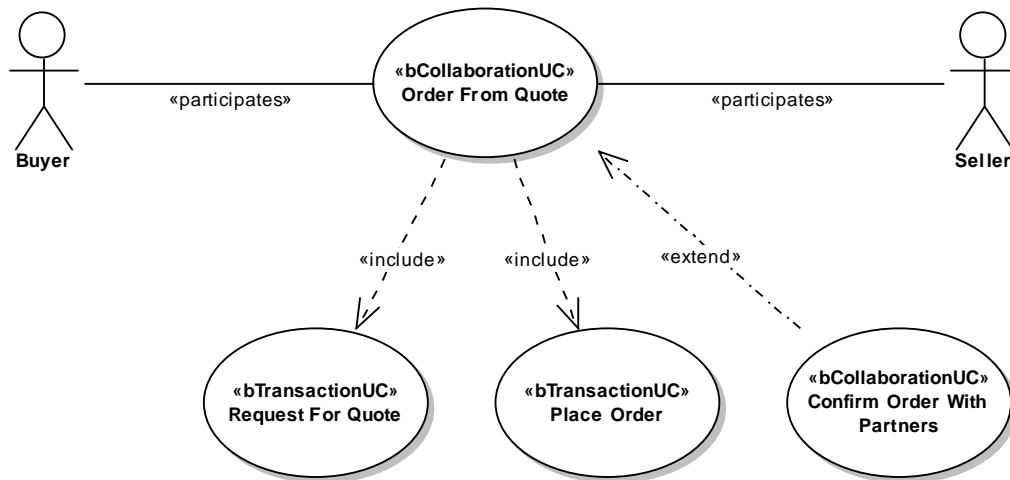


1075

1076 Figure 39 Business Collaboration Protocol Example: Request From Quote

1077 **5.2.3.7 Nested Business Collaboration View Example (informative)**

1078 In this example the Order from Quote example is modified to show an example of a Nested Collaboration.  
 1079 Before the Seller can either accept or reject the Buyer's order, the seller must confirm the order with his/her  
 1080 business partners. If this confirmation is unsuccessful, it follows that the Seller will respond by sending a  
 1081 negative response message (e.g. *OrderRejectEnvelope*) to the Buyer.

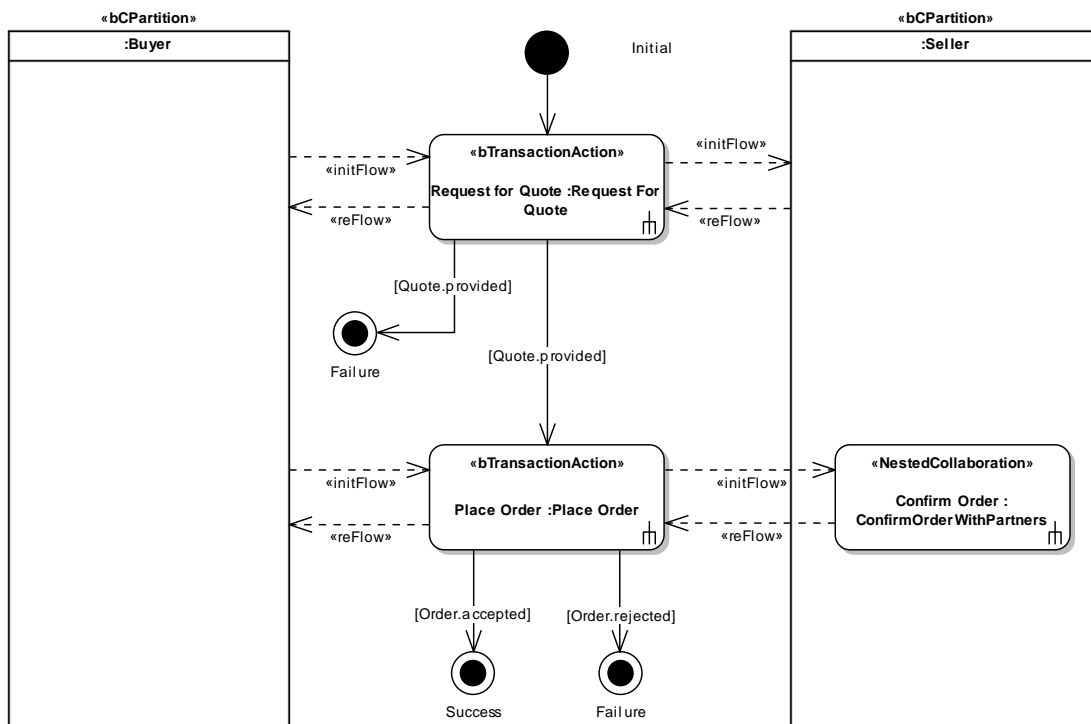


1082

1083

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

1084

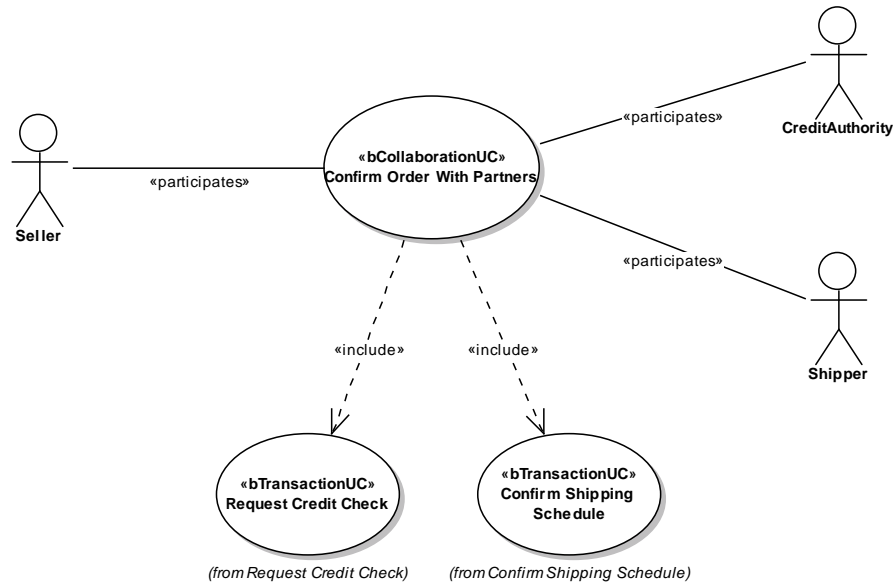


1085

1086

Figure 41 Business Collaboration Protocol Example: Request For Quote

1087

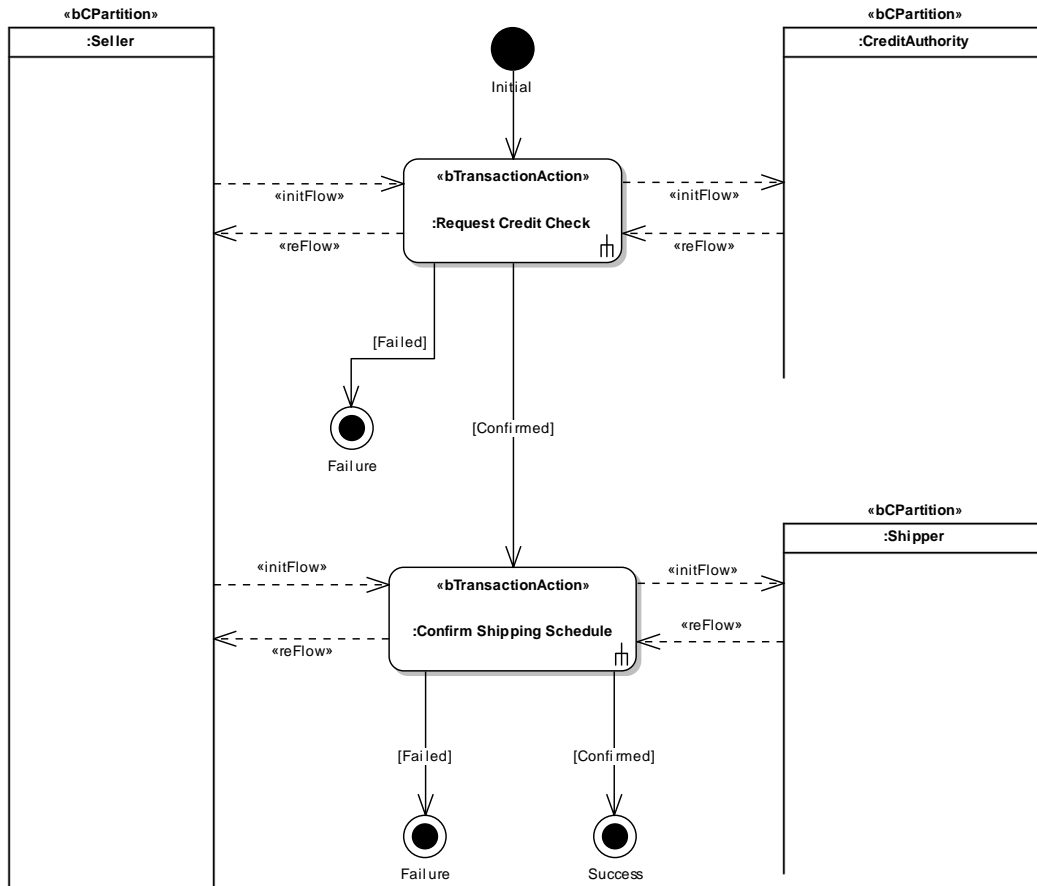


1088

1089  
1090

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

1091



1092



1093

Figure 43 Business Collaboration Protocol Example: Confirm Order with Partners

1094 5.2.4 Business Realization View

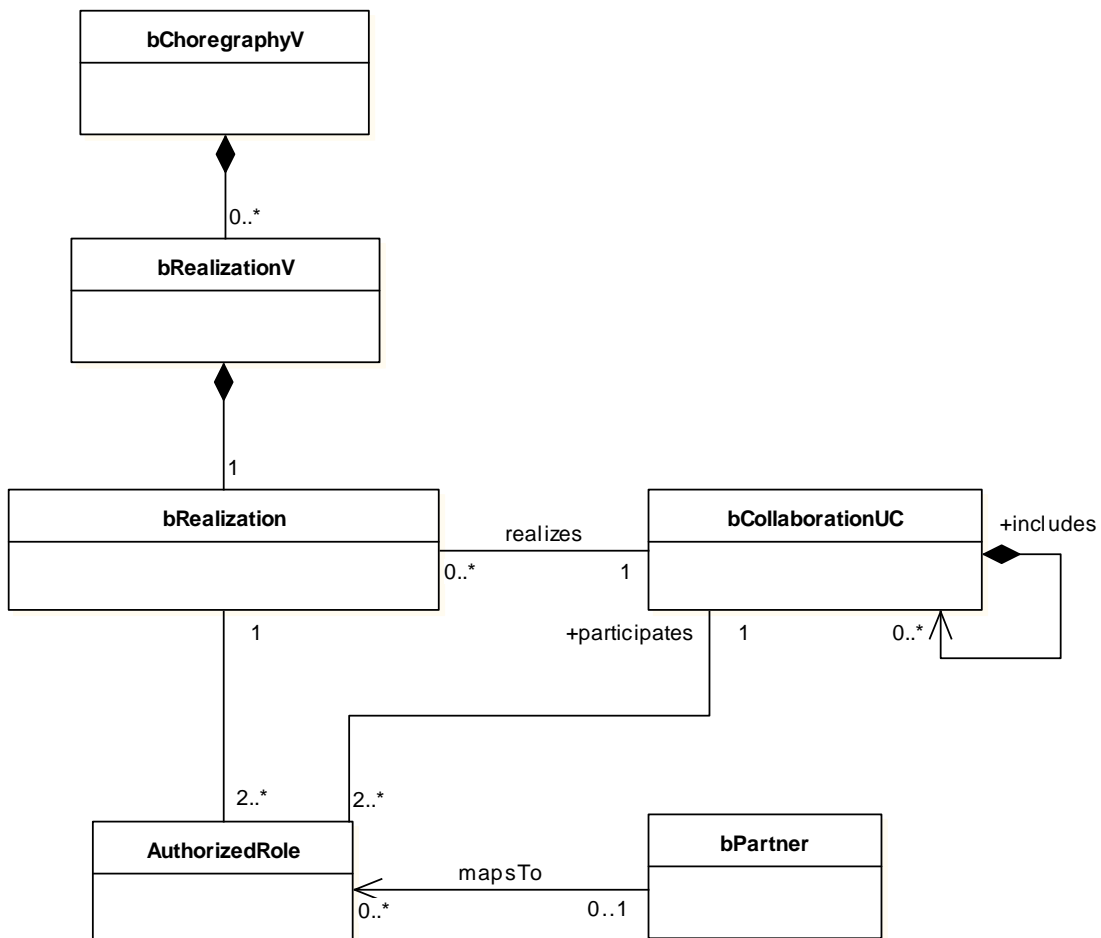
1095 5.2.4.1 Abbreviations and Stereotypes

1096

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

1097

1098 5.2.4.2 Conceptual Description (informative)



1099

1100 Figure 44 BusinessRealizationView - Conceptual Overview

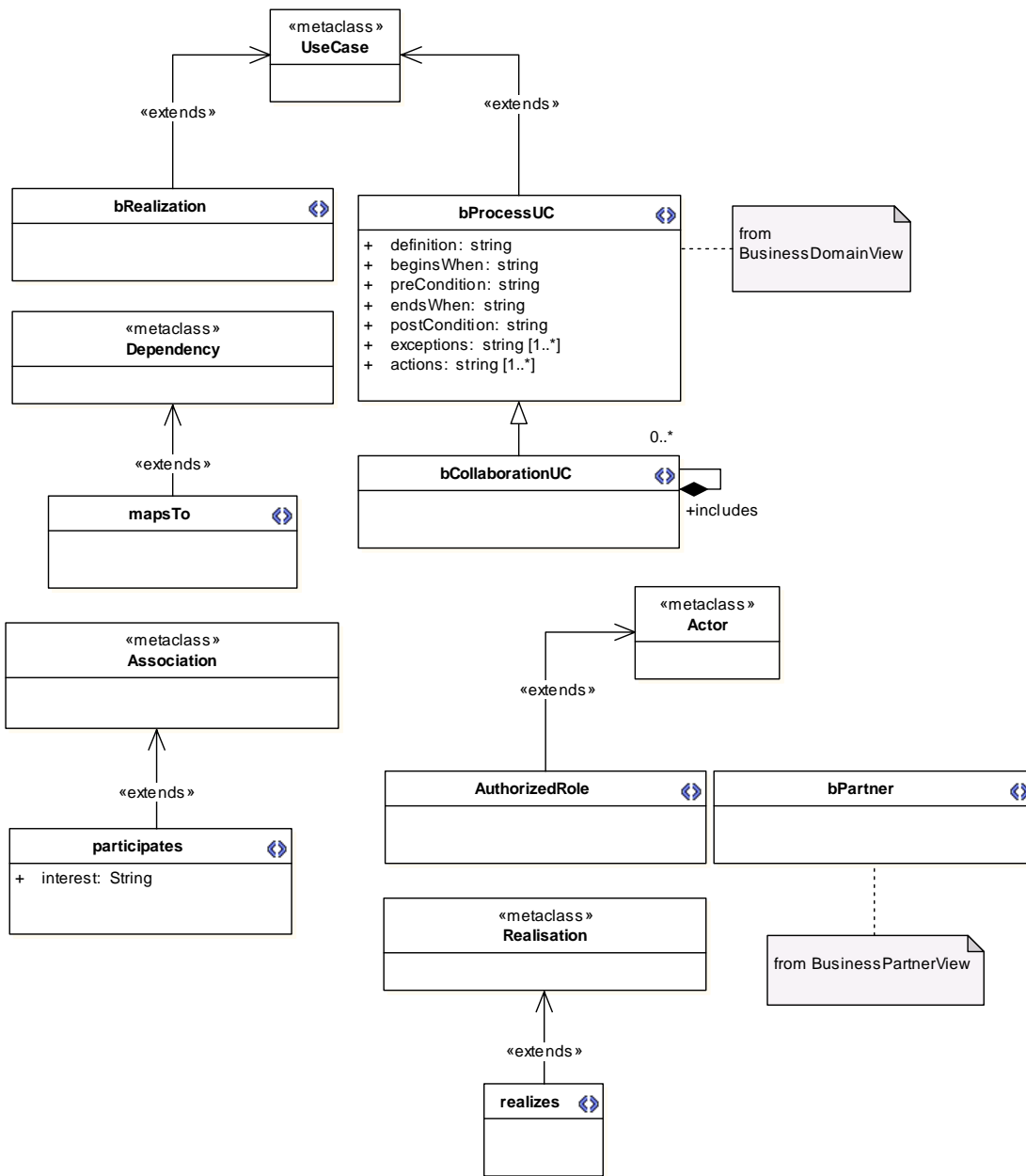
1101 Business partners identified in the previous business requirements view must not directly be associated with  
1102 business collaboration use cases and business transaction use cases.

1103 In order to specify that a specific set of business partners collaborate, we use the concept of a business  
1104 realization. Each business realization is defined in its own business realization view. Accordingly, the  
1105 *BusinessRealizationView* is composed of exactly one *BusinessRealization*. A business realization realizes  
1106 exactly one business collaboration use case. Each business collaboration use case may be realized by  
1107 multiple business realizations. Not each business collaboration use case (e.g. one that is nested within  
1108 another one) needs to have a corresponding business realization. As a consequence, the *realizes*-association  
1109 between a *BusinessCollaborationUseCase* and *BusinessRealization* is a 1 to (0..n).

1110 Two or more authorized roles participate in a business realization. These authorized roles (e.g. seller, payee)  
1111 must be defined in the same business realization view package as the corresponding business realization.  
1112 Accordingly a *BusinessRealizationView* is composed of two or more *AuthorizedRoles*. Usually, the names of  
1113 the authorized roles participating in the business collaboration use case (e.g. payer and payee) will be the  
1114 names of the authorized roles in the business realization (e.g. payer and payee) realizing it. However, the  
1115 authorized roles participating in the business collaboration use case and the business realization will be  
1116 defined in different namespaces – each in the package of the corresponding view. In **Figure 46** the authorized  
1117 role *Buyer* on the lower left hand side participates in the business collaboration use case. It is defined in a  
1118 different namespace than the *Buyer* participating in the business realization.

1119 Similar to the business collaboration use case, the *BusinessRealization* and *AuthorizedRole* are related by an  
1120 1 to (2..n) association. Furthermore, the number of actors participating in a business collaboration use case  
1121 must be the same as the number of actors participating in the business realization realizing it.

1122 In order to bind a business realization to the business partners executing it, business partners are mapped to  
1123 the authorized roles participating in the business realization. It is required that each authorized role of a  
1124 business realization (but not an authorized role in general) is target of exactly one *mapsTo*-association from  
1125 a business partner. A business partner may play multiple authorized roles of a business realization.  
1126 Consequently, there is a (0..1) to (0..n) *mapsTo*-association between *BusinessPartner* and *AuthorizedRole*.



1128  
1129  
1130

Figure 45 BusinessRealizationView - Abstract Syntax

Stereotype	bRealization (BusinessRealization)
Base Class	UseCase
Parent	N/A
Description	A business realization realizes a business collaboration use case between a specific set of business partners. The requirements of the business realization are the ones defined in the tags of the corresponding business collaboration use case. Thus, the business realization does not include any tag

	definitions for capturing requirements.
<b>Tag Definition</b>	No tagged values

1131

<b>Stereotype</b>	<b>AuthorizedRole (AuthorizedRole)</b>
<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 “broker”) 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 “broker” or a “custodian”) in different scenarios of the same domain or even in different domains.
<b>Tag Definition</b>	No tagged values.

1132

<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 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.

1133

1134 **5.2.4.4 Constraints (normative)**

1135 C.92. A *BusinessRealizationView* MUST contain exactly one *BusinessRealization*, two to many  
1136 *AuthorizedRoles*, and two to many *participates* associations.

1137 C.93. A *BusinessRealization* MUST be associated with two to many *AuthorizedRoles* via  
1138 stereotyped binary *participates* associations.

1139 C.94. A *BusinessRealization* MUST be the source of exactly one realization dependency to a  
1140 *BusinessCollaborationUseCase*.

1141 C.95. A *BusinessRealization* MUST NOT be the source or target of an *include* or *extends*  
1142 association.

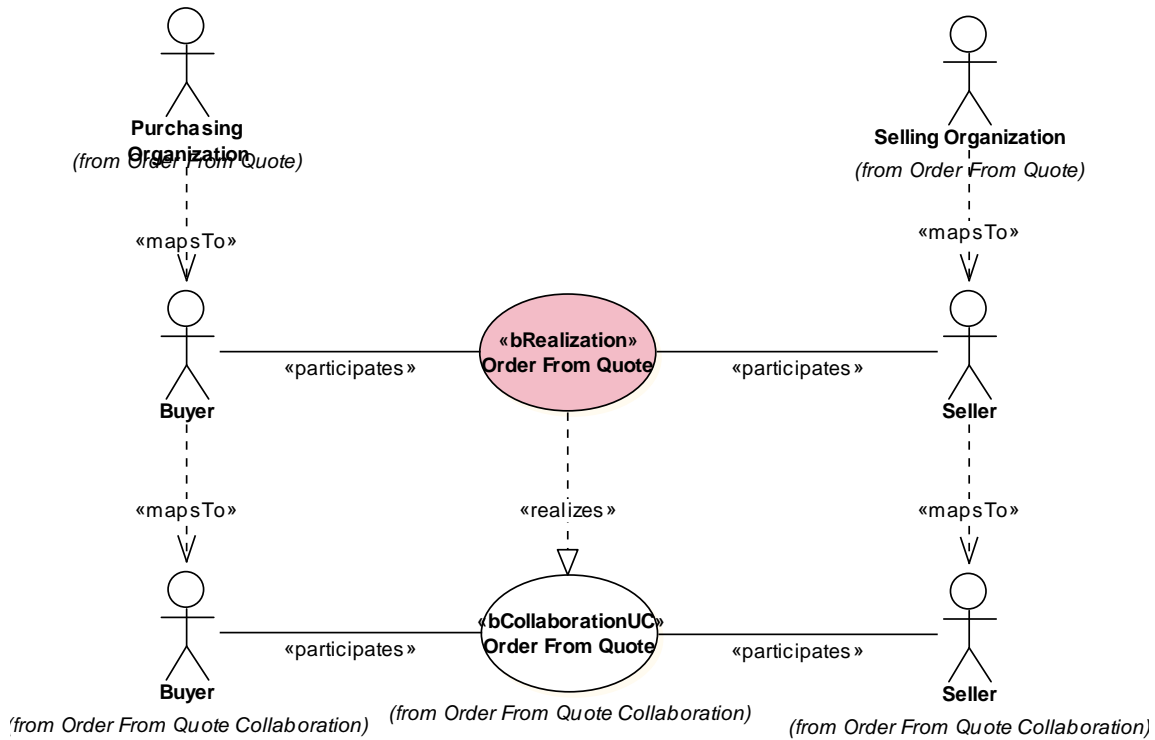
1143 C.96. All dependencies from/to an *AuthorizedRole* must be stereotyped as *mapsTo*.

1144 C.97. An *AuthorizedRole*, which participates in a *BusinessRealization*, must be the target of exactly  
1145 one *mapsTo* dependency starting from a *BusinessPartner*. Furthermore the *AuthorizedRole*, which  
1146 participates in the *BusinessRealization* must be the source of exactly one *mapsTo* dependency  
1147 targeting an *AuthorizedRole* participating in a *BusinessCollaborationUseCase*.

1148 C.98. *AuthorizedRoles* in a *BusinessRealizationView* must have a unique name within the scope of  
1149 the package, they are located in

1150 C.99. The number of *AuthorizedRoles* participating in a *BusinessCollaborationUseCase* MUST  
 1151 match the number of *AuthorizedRoles* participating in the *BusinessRealization* realizing this  
 1152 *BusinessCollaborationUseCase*  
 1153

1154 **5.2.4.5 Example (informative)**



1155  
 1156 **Figure 46 BusinessRealizationView - Example: Realization of the OrderFromQuote Collaboration between Purchasing**  
 1157 **Organization and SellingOrganization**

1158

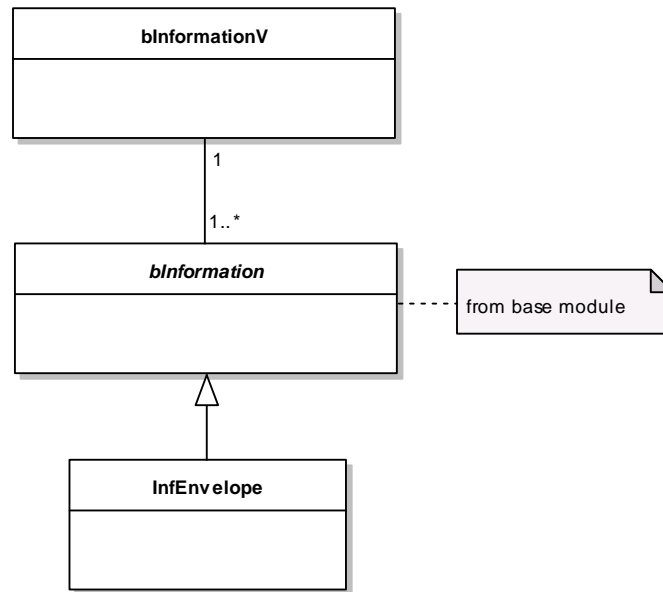
1159 **5.3 Business Information View**

1160 **5.3.1 Abbreviations of Stereotypes**

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

1161

1162 **5.3.2 Conceptual Description (informative)**



1163

1164 **Figure 47 BusinessInformationView - Conceptual Overview**

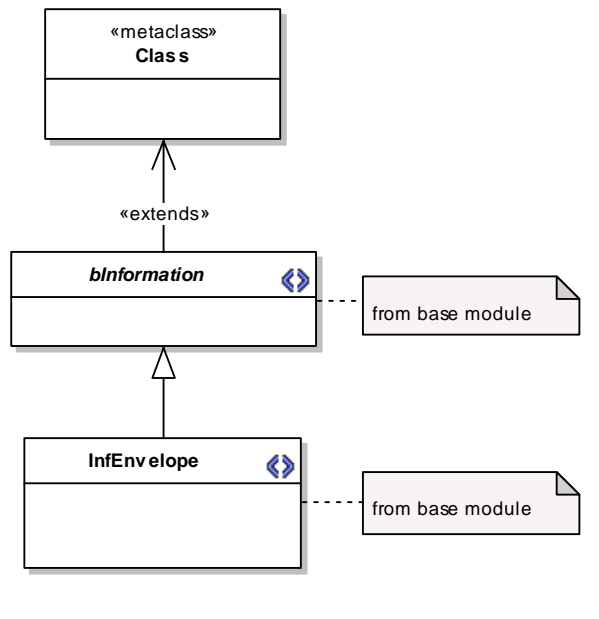
1165 A *BusinessInformationView* is a container of artifacts that describe the information exchanged in a  
1166 *BusinessTransaction*. As previously mentioned; *RequestingInformationPin* and  
1167 *RespondingInformationPin* are classified by an *InformationEnvelope* which is a subclass of a  
1168 *BusinessInformation*. A *BusinessInformation* serves as an abstract container for all of the information  
1169 exchanged between the *RequestingAction* and the *RespondingAction* or vice versa, respectively. The  
1170 stereotypes *BusinessInformation* and *InformationEnvelope* is part of the UMM base module and  
1171 imported into the UMM foundation module.

1172 The current UMM foundation module does not mandate a specific business information modeling  
1173 approach. All methodologies and rules to build quality class diagrams can be used in order to model the  
1174 exchanged information, as long as the root element of the data structure generalizes the  
1175 *InformationEnvelope* class being part of the UMM base module.

1176 However, UMM strongly suggests using UN/CEFACT's Core Components and Core Components Message  
1177 Assembly artifacts to model the business information. Because Core Components are syntax  
1178 independent and stereotyped, the usage of the UML Profile for Core Components is suggested within  
1179 the *BusinessInformationView*.

1180

1181 **5.3.3 Stereotypes and Tag Definitions (normative)**



1182

1183

1184

**Figure 48 BusinessInformationView - Abstract Syntax**

1185

Stereotype <b>bInformation (BusinessInformation)</b>	
Base Class	Class
Parent	N/A
Description	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.

1186

Stereotype <b>InfEnvelope (InformationEnvelope)</b>	
Base Class	Class
Parent	BusinessInformation
Description	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.

1187

1188

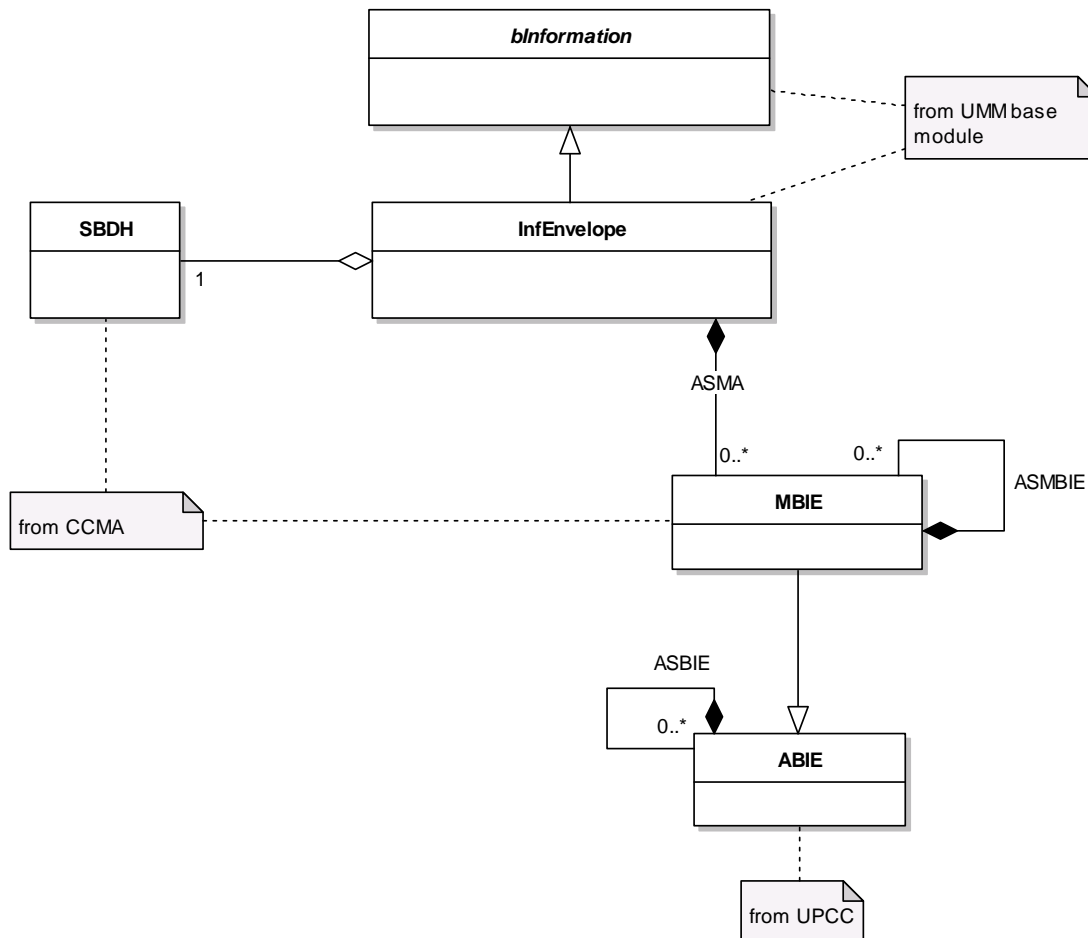
1189 **5.3.4 Constraints (normative)**

1190 **C.100.** A *BusinessInformationView* MUST contain one to many *InformationEnvelopes* or  
 1191 subtypes thereof defined in any other extension/specialization module. Furthermore, it MAY  
 1192 contain any other document modeling artifacts.

1193 **5.3.5 Example using UPCC (UML Profile for Core Components) and CCMA (Core**  
 1194 **Components Message Assembly) (informative)**

1195 The following example shows how to model the information exchanged in a *BusinessTransaction* using  
 1196 the UML Profile for Core Components (UPCC) and the Core Component Message Assembly (CCMA).

1197 Figure 49 shows how UMM, UPCC and CCMA are related to each other.



1198 **Figure 49 Abstract example for using UPCC artifacts to model business information**  
 1199

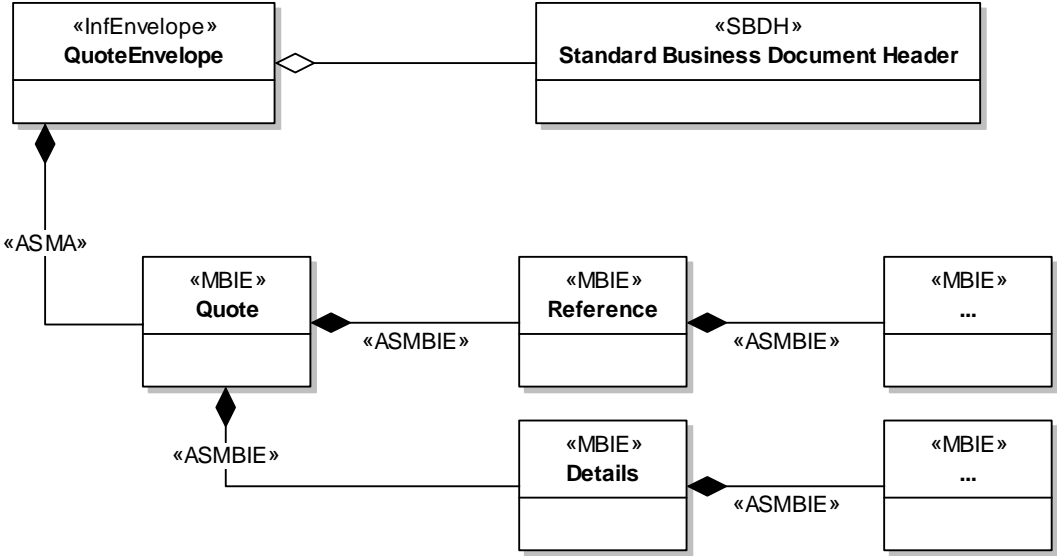
1200 An *information envelope* has a *standard business document header* (SBDH) which serves for  
 1201 identification purposes of technical sender and receiver, document type etc.

1202 The body of an *information envelope* consists of zero or more elements – so called *messaging business*  
 1203 *information entities* (MBIE). MBIEs are connected to an *information envelope* using so called *association*  
 1204 *message assemblies* (ASMA). If different *messaging business information entities* are connected to each  
 1205 other the concept of *association messageing business information entities* (ASMBIE) is used. A  
 1206 *messaging business information entity* inherits from an *aggregate business information entity* (ABIE).  
 1207 ABIEs are part of the UML Profile for Core Components (UPCC) standard. In order to relate different  
 1208 *aggregate business information entities* to each other so called *association business information entities*  
 1209 (ASBIE) are used.



1210 *Standard business document header and messaging business information entities are part of the core*  
 1211 *components message assembly (CCMA). Aggregate business information entities are part of the UML*  
 1212 *Profile for Core Components (UPCC).*

1213 Figure 50 shows an example for a QuoteEnvelope modelled using concepts from the Core Components  
 1214 Message Assembly standard. Not shown in Figure 50 are the dependencies between ABIEs and MBIEs.



1215  
 1216  
 1217

Figure 50 CCMA example

## 1218 6 OCL Constraints

1219

```
1220 -- Constraint 1
1221 -- A BusinessCollaborationModel MUST contain one to many
1222     BusinessChoreographyViews.
1223 context Package inv: self.isBCollModel()
1224     implies self.nestedPackage->exists(a|a.isBChoreographyV())
1225
```

```
1226 -- Constraint 2
1227 -- A BusinessCollaborationModel MUST contain one to many
1228     BusinessInformationViews.
1229 context Package inv: self.isBCollModel()
1230     implies self.nestedPackage->exists(a|a.isBInformationV())
1231
```

```
1232 -- Constraint 3
1233 -- A BusinessCollaborationModel MAY contain zero to many
1234     BusinessRequirementsViews
1235 context Package inv: self.isBCollModel()
1236     implies self.nestedPackage->select(a|a.isBRequirementsV())->
1237     size>=0
1238
```

```
1239 -- Constraint 4
1240 -- A BusinessRequirementsView, a BusinessChoreographyView and a
1241     BusinessInformationView
1242 -- MUST be directly located under a BusinessCollaborationModel
1243 context Package inv: (self.isBChoreographyV() or
1244     self.isBInformationV() or self.isBRequirementsV())
1245     implies self.hlpOwningPackage().isBCollModel()
1246
```

```
1247 -- Constraint 5
1248 --A BusinessRequirementsView MAY contain zero or one
1249     BusinessDomainViews.
1250 context Package inv: self.isBRequirementsV()
1251     implies self.nestedPackage->select(a|a.isBDomainV())->size <=1
1252
```

```
1253 -- Constraint 6
1254 -- A BusinessChoreographyView MAY contain zero or one
1255     BusinessPartnerViews.
1256 context Package inv: self.isBRequirementsV()
1257     implies self.nestedPackage->select(a | a.isBPartnerV())->size <=1
1258
```

```
1259 -- Constraint 7
1260 -- A BusinessRequirementsView MAY contain zero to many
1261     BusinessEntityViews.
1262 context Package inv: self.isBRequirementsV()
1263     implies self.nestedPackage->select( a | a.isBEntityV())->size >= 0
1264
```

```
1265 -- Constraint 8
```

```

1266 -- A BusinessDomainView, a BusinessPartnerView, and a
1267 BusinessEntityView
1268 -- MUST be located directly under a BusinessRequirementsView
1269 context Package inv:
1270     self.isBDomainV() or self.isBPartnerV() or self.isBEntityV()
1271     implies self.hlpOwningPackage().isBRequirementsV()
1272

```

```

1273 -- Constraint 9
1274 -- A BusinessDomainView MUST include one to many BusinessAreas.
1275 context Package inv: self.isBDomainV()
1276     implies self.nestedPackage->exists( a | a.isBArea() )
1277

```

```

1278 -- Constraint 10
1279 -- A BusinessArea MUST include one to many BusinessAreas or one to
1280 -- many ProcessAreas or one to many BusinessProcessUseCases.
1281 context Package inv: self.isBArea()
1282     implies ( self.nestedPackage->exists( a | a.isBArea()
1283     or a.isProcessArea() ) or ( self.ownedElement ->
1284     exists( b | b.oclAsType(UseCase).isBProcessUseCase() ) ) )
1285

```

```

1286 -- Constraint 11
1287 -- A ProcessArea MUST contain one to many other ProcessAreas or one to
1288 -- many BusinessProcessUseCases
1289 context Package inv: self.isProcessArea()
1290     implies (self.nestedPackage -> exists( a | a.isProcessArea() ) ) or
1291     (self.ownedElement ->
1292     exists( b | b.oclAsType(UseCase).isBProcessUseCase() ) )
1293

```

```

1294 -- Constraint 12
1295 --A BusinessProcessUseCase MUST be associated with one to many
1296 BusinessPartners using the participates relationship
1297 context UseCase inv: self.isBProcessUseCase() and
1298     not(self.isBCollaborationUC())
1299     and not(self.isBTransactionUC())
1300     implies self.owner.ownedElement->exists(a|
1301     a.oclAsType(Association).isParticipates()
1302     and a.oclAsType(Association).ownedEnd.type->
1303     exists(t|t.oclAsType(Actor).isBPartner()
1304     and a.oclAsType(Association).ownedEnd.type->
1305     exists(t|t.oclAsType(UseCase)=self))
1306

```

```

1307 -- Constraint 13
1308 -- A BusinessProcessUseCase may be associated with zero to many
1309 Stakeholders using the isOfInterestedTo relationship
1310 context UseCase inv: self.isBProcessUseCase()
1311     implies if self.owner.ownedElement->
1312     exists(a|a.oclAsType(Actor).isStakeholder() )
1313     then self.oclAsType(UseCase).clientDependency->
1314     exists(p|p.oclAsType(Dependency).isOfInterestTo()
1315     and p.client->exists(t|t.oclAsType(Actor).isStakeholder() ) )
1316     else true endif

```

1317

```
1318 -- Constraint 14
1319 -- A BusinessProcessUseCase SHOULD be refined by zero to many
1320 BusinessProcesses. These relationships MAY also be visualized by
1321 realize relationships from each of the owned BusinessProcesses to
1322 the owning BusinessProcessUseCase
1323 context UseCase inv: self.isBProcessUseCase()
1324     implies self.ownedElement->select( process |
1325     process.oclAsType(Activity).isBProcess()->size())>=0
1326
```

```
1327 -- Constraint 15
1328 -- A BusinessProcess MUST be modeled as a child of a
1329 BusinessProcessUseCase
1330 context Activity inv: self.isBProcess()
1331     implies self.owner.oclAsType(UseCase).isBProcessUseCase()
1332
```

```
1333 Constraint 16 refers to a Diagram and is therefore not represented in
1334 OCL
1335
```

```
1336 -- Constraint 17
1337 -- A BusinessProcess MAY contain zero to many ActivityPartitions
1338 context Activity inv: self.isBProcess()
1339     implies self.ownedElement->select( partitions |
1340     partitions.oclIsTypeOf(ActivityPartition))->size())>=0
1341
```

```
1342 -- Constraint 18
1343 -- A BusinessProcess, which has no ActivityPartitions, MUST contain
1344 one or more BusinessProcessActions and
1345 -- MAY contain zero to many InternalBusinessEntityStates and zero to
1346 many SharedBusinessEntityStates.
1347 context Activity inv: self.isBProcess()
1348     implies if self.ownedElement->
1349     select(b|b.oclAsType(ActivityPartition).isActivityPartition())->
1350     size=0
1351     then self.ownedElement-
1352     exists(bp|bp.oclAsType(Action).isBProcessAction()) else true endif
1353
```

```
1354 -- Constraint 19
1355 -- An ActivityPartition being part of a BusinessProcess MUST contain
1356 one to many BusinessProcessActions and
1357 -- MAY contain zero to many InternalBusinessEntityStates.
1358 context ActivityPartition inv: self.isActivityPartition() and
1359     self.owner.oclAsType(Activity).isBProcess()
1360     implies self.ownedElement->exists( act |
1361     act.oclAsType(Action).isBProcessAction())
1362
```

```
1363 -- Constraint 20
1364 -- A SharedBusinessEntityState MUST NOT be located in an
1365 ActivityPartition. (They must be contained within
```

```
1366 -- the BusinessProcess even if this BusinessProcess contains
1367 ActivityPartitions.)
1368 context ObjectNode inv: self.oclasType(ObjectNode).isBESharedState()
1369 and self.owner.oclasType(Activity).isBProcess()
1370 implies
1371 not(self.owner.oclasType(ActivityPartition).isActivityPartition())
1372
```

```
1373 -- Constraint 21
1374 -- A BusinessPartnerView MUST contain at least two to many
1375 BusinessPartners. If the BusinessPartnerView is hierarchically
1376 decomposed into subpackages these BusinessPartners MAY be contained
1377 in any of these subpackages.
1378 Missing on purpose
1379
```

```
1380 -- Constraint 22
1381 -- A BusinessPartnerView MAY contain zero to many Stakeholders
1382 Missing on purpose
1383
```

```
1384 -- Constraint 23
1385 -- A BusinessEntityView must contain one to many BusinessEntities
1386 context Package inv: self.isBEntityV()
1387 implies self.ownedElement->
1388 exists(a|a.oclasType(Class).isBEntity())
1389
```

```
1390 Constraint 24 refers to a diagram and is therefore not represented in
1391 OCL
1392
```

```
1393 -- Constraint 25 (Since the first part of the constraint refers to a
1394 diagram, only the second part of the constraint is represented in
1395 OCL)
1396 -- A UML State Diagram describing the lifecycle of a BusinessEntity
1397 MUST contain one to many BusinessEntityStates. The parent of a
1398 BusinessEntityState MUST be a BusinessEntity
1399 context Class inv: self.isBEntityState()
1400 and self.owner.isBEntity()
1401
```

```
1402 -- Constraint 26
1403 -- A BusinessEntity MAY contain zero to many BusinessDataViews that
1404 describes its conceptual design
1405 context Package inv: self.isBEntityV()
1406 implies self.nestedPackage->select( package |
1407 package.oclasType(Package).isBDataV()->size>=0)
1408
```

```
1409 -- Constraint 27
1410 -- The parent of a BusinessDataView MUST be a BusinessEntityView
1411 context Package inv: self.isBDataV()
1412 implies self.owner.isBEntityV()
1413
```

```

1414 -- Constraint 29
1415 -- A BusinessDataView SHOULD contain one to many classes.
1416 context Package inv: self.isBDataV()
1417     implies self.ownedElement->select( elem |
1418     elem.oclIsTypeOf(Class))->size(>=1
1419

```

```

1420 -- Constraint 30
1421 -- A BusinessChoreographyView MUST contain one to many
1422     BusinessCollaborationViews
1423 context Package inv: self.isBChoreographyV()
1424     implies self.nestedPackage->exists(c|c.isBCollaborationV())
1425

```

```

1426 -- Constraint 31
1427 --A BusinessChoreographyView MUST contain one to many
1428     BusinessTransactionViews
1429 context Package inv: self.isBChoreographyV()
1430     implies self.nestedPackage->exists(c|c.isBTransactionV())
1431

```

```

1432 -- Constraint 32
1433 -- A BusinessChoreographyView MAY contain zero to many
1434     BusinessRealizationViews
1435 context Package inv: self.isBChoreographyV()
1436     implies self.nestedPackage->select( package |
1437     package.oclAsType(Package).isBRealizationV()->size(>=0
1438

```

```

1439 -- Constraint 33
1440 -- A BusinessTransactionView, a BusinessCollaborationView, and a
1441     BusinessRealizationView
1442 -- MUST be directly located under a BusinessChoreographyView
1443 context Package inv: self.isBTransactionV() or
1444     self.isBCollaborationV() or self.isBRealizationV()
1445     implies self.owner.isBChoreographyV()
1446

```

```

1447 -- Constraint 34
1448 -- A BusinessTransactionView MUST contain exactly one
1449     BusinessTransactionUseCase, exactly two AuthorizedRoles,
1450 -- and exactly two participates associations.
1451 context Package inv: self.isBTransactionV()
1452     implies self.ownedElement->
1453     select(a|a.oclAsType(UseCase).isBTransactionUC()->size=1
1454     and self.ownedElement->
1455     select(b|b.oclAsType(Actor).isAuthorizedRole()->size=2
1456     and self.ownedElement->select(c|c.oclIsTypeOf(Association)
1457     and c.oclAsType(Association).isParticipates()->size=2
1458     and self.ownedElement->size=5
1459

```

```

1460 -- Constraint 35
1461 -- A BusinessTransactionUseCase MUST be associated with exactly two
1462     AuthorizedRoles via stereotyped binary participate associations.
1463 context UseCase inv: self.isBTransactionUC()

```

```
1464 implies self.owner.ownedElement->
1465 select(q|q.ocIsKindOf(Association))->
1466 forAll(a|a.ocAsType(Association).isParticipates()
1467 and a.ocAsType(Association).ownedEnd.type->
1468 forAll(t|t.ocAsType(Actor).isAuthorizedRole()
1469 or t.ocAsType(UseCase)=self))
1470
```

```
1471 -- Constraint 36
1472 -- A BusinessTransactionUseCase MUST NOT include further UseCases
1473 context UseCase inv: self.isBTransactionUC()
1474 implies self.include->size=0
1475
```

```
1476 -- Constraint 37
1477 -- A BusinessTransactionUseCase MUST be included in at least one
1478 BusinessCollaborationUseCase.
1479 context UseCase inv: self.isBTransactionUC()
1480 implies self.owner.owner.ocAsType(Package).nestedPackage
1481 ->select(collV|collV.isBCollaborationV())->
1482 exists(c|c.ownedElement
1483 ->exists(k|k.ocAsType(UseCase).isBCollaborationUC()
1484 and k.ocAsType(UseCase).include.addition
1485 ->exists(s|s.ocAsType(UseCase)=self)))
1486
```

```
1487 -- Constraint 38
1488 --A BusinessTransactionUseCase MUST NOT be source or target of an
1489 extend association.
1490 context UseCase inv: self.isBTransactionUC()
1491 implies self.extend->size=0 and UseCase.allInstances->
1492 forAll(u|u.ocAsType(UseCase).extend.extendedCase->
1493 forAll(t|t.ocAsType(UseCase).isNotBTransactionUC()))
1494
```

```
1495 -- Constraint 39
1496 -- The two AuthorizedRoles within a BusinessTransactionView MUST NOT
1497 be named identically
1498 context Actor inv: self.isAuthorizedRole() and
1499 self.owner.ocAsType(Package).isBTransactionV()
1500 implies self.owner.ownedElement->
1501 select(k|k.ocAsType(Actor).isAuthorizedRole())->
1502 collect(p|p.ocAsType(Actor).name)->asSet->size()=2
1503
```

```
1504 -- Constraint 40
1505 -- A BusinessTransactionUseCase MUST be described by exactly one
1506 BusinessTransaction defined as a child element of this
1507 BusinessTransactionUseCase.
1508 context UseCase inv: self.isBTransactionUC()
1509 implies self.ownedElement->select( act |
1510 act.ocAsType(Activity).isBTransaction()) -> size = 1
1511
```

```
1512 -- Constraint 41
1513 -- A BusinessTransaction MUST have exactly two partitions.
```

```

1514 -- Each of them MUST be stereotyped as BusinessTransactionPartition.
1515 context Activity inv: self.isBTransaction()
1516     implies self.ownedElement->select( part |
1517     part.oclassType(ActivityPartition).isBTPartition()->size=2
1518

```

```

1519 -- Constraint 42
1520 -- One of the two BusinessTransactionPartitions MUST contain one
1521 RequestingBusinessAction and
1522 -- the other one MUST contain one RespondingBusinessAction.
1523 context Activity inv:
1524     let ReqAct : Action = self.ownedElement->
1525     select( act | act.oclassType(Action).isReqAction()->
1526     asSequence->first().oclassType(Action)
1527     in let ResAct : Action = self.ownedElement->
1528     select( act | act.oclassType(Action).isResAction()->
1529     asSequence->first().oclassType(Action)
1530     in let Part1 : ActivityPartition = self.ownedElement->
1531     select( part | part.oclassType(ActivityPartition).isBTPartition()->
1532     asSequence->first().oclassType(ActivityPartition)
1533     in let Part2 : ActivityPartition = self.ownedElement->
1534     select( part | part.oclassType(ActivityPartition).isBTPartition()->
1535     asSequence->last().oclassType(ActivityPartition)
1536     in self.isBTransaction()
1537     implies self.ownedElement->
1538     select( act | act.oclassType(Action).isReqAction()->size=1
1539     and self.ownedElement->
1540     select( act | act.oclassType(Action).isResAction()->size=1
1541     and ReqAct.inPartition->size=1 and ResAct.inPartition->size=1
1542     and ( (ReqAct.inPartition->asSequence
1543     ->first().oclassType(ActivityPartition)=Part1
1544     and ResAct.inPartition->asSequence
1545     ->first().oclassType(ActivityPartition)=Part2)
1546     or (ResAct.inPartition->asSequence
1547     ->first().oclassType(ActivityPartition)=Part1
1548     and ReqAct.inPartition->asSequence
1549     ->first().oclassType(ActivityPartition)=Part2 ) )
1550

```

```

1551 -- Constraint 43
1552 -- A BusinessTransactionPartition MUST have a classifier, which MUST
1553 be one of the associated
1554 -- AuthorizedRoles of the corresponding BusinessTransactionUseCase.
1555 context ActivityPartition inv:
1556     let authRoles : Sequence(Element) =
1557     self.owner.owner.owner.ownedElement->
1558     select(roles | roles.oclassType(Actor).isAuthorizedRole())
1559     ->asSequence in self.isBTPartition()
1560     implies self.name=authRoles->first().oclassType(Actor).name
1561     or self.name=authRoles->last().oclassType(Actor).name
1562

```

```

1563 -- Constraint 44
1564 -- The two BusinessTransactionPartitions MUST have different
1565 classifiers.
1566 context Activity inv: self.isBTransaction()
1567     implies not(self.ownedElement->

```



```
1568     select( part | part.oclAsType(ActivityPartition).isBTPartition()  
1569     -> asSequence->first().oclAsType(ActivityPartition).name =  
1570     self.ownedElement->  
1571     select( part | part.oclAsType(ActivityPartition).isBTPartition()  
1572     -> asSequence->last().oclAsType(ActivityPartition).name)  
1573
```

```
1574 -- Constraint 45  
1575 -- The BusinessTransactionPartition containing the  
1576 -- RequestingBusinessAction MUST contain two or more FinalStates.  
1577 -- Each of the FinalStates MAY have a SharedBusinessEntityState as  
1578 -- predecessor.  
1579 -- One of the FinalStates SHOULD reflect a ControlFailure - this  
1580 -- FinalState SHOULD NOT have a  
1581 -- predecesing SharedBusinessEntityState.  
1582 context ActivityPartition inv: self.isBTPartition() and  
1583     self.containedNode->exists( action |  
1584     action.oclAsType(Action).isReqAction()  
1585     implies self.containedNode->select( finNode |  
1586     finNode.oclIsTypeOf(ActivityFinalNode))->size())>=2  
1587
```

```
1588 -- Constraint 46  
1589 -- A RequestingBusinessAction MUST embed exactly one  
1590 -- RequestingInformationPin  
1591 context Action inv: self.isReqAction()  
1592     implies  
1593     self.ownedElement->select( pin | pin.oclAsType(Pin).isReqInfPin()  
1594     ->size=1  
1595
```

```
1596 -- Constraint 47  
1597 -- A RespondingBusinessAction MUST embed exactly one  
1598 -- RequestingInformationPin  
1599 context Action inv: self.isResAction()  
1600     implies  
1601     self.ownedElement->select( pin | pin.oclAsType(Pin).isReqInfPin()  
1602     ->size=1  
1603
```

```
1604 -- Constraint 48  
1605 -- If the tagged value businessTransactionType of the  
1606 -- BusinessTransaction is either Request/Response, Query/Response,  
1607 -- Request/Confirm, or CommercialTransaction, then the  
1608 -- RequestingBusinessAction must embed one to many  
1609 -- RespondingInformationPins and the RespondingBusinessAction must  
1610 -- embed one to many RespondingInformationPins.  
1611 context Action inv: self.isBTransaction() and  
1612     self.hlpMustHaveResInfPin() implies  
1613     self.ownedElement-> select( action |  
1614     action.oclAsType(Action).isReqAction()  
1615     ->forall( actions | actions.ownedElement->  
1616     exists( pin | pin.oclAsType(Pin).isResInfPin()))  
1617
```

```
1618 -- Constraint 49
```

```

1619 -- If the tagged value businessTransactionType of the
1620 BusinessTransaction is either Notification or
1621 InformationDistribution, then both, the RequestingBusinessAction
1622 and the RespondingBusinessAction, MUST NOT embed a
1623 RespondingInformationPin
1624 context Action inv: self.isBTransaction() and
1625 self.hlpMustNotHaveResInfPin()
1626 implies
1627 not(self.ownedElement->select( action |
1628 ction.oclAsType(Action).isReqAction()
1629 or action.oclAsType(Action).isResAction())->
1630 forAll( actions | actions.ownedElement->
1631 exists( pin | pin.oclAsType(Pin).isResInfPin()))
1632

```

```

1633 -- Constraint 50
1634 -- A RequestingBusinessAction and a RespondingBusinessAction MUST
1635 embed same
1636 -- number of RespondingInformationPins.
1637 let ReqAct : Action = self.ownedElement->select( act |
1638 act.oclAsType(Action).isReqAction())->
1639 asSequence->first().oclAsType(Action) in
1640 let ResAct : Action = self.ownedElement->select( act |
1641 act.oclAsType(Action).isResAction())->
1642 asSequence->first().oclAsType(Action) in
1643 self.isBTransaction() implies
1644 ReqAct.ownedElement->select( resPin |
1645 resPin.oclAsType(Pin).isResInfPin())->size
1646 =
1647 ResAct.ownedElement->select( resPin |
1648 resPin.oclAsType(Pin).isResInfPin())->size
1649

```

```

1650 -- Constraint 51
1651 --
1652 The RequestingInformationPin of the RequestingBusinessAction MUST b
1653 e connected
1654 -- with the
1655 RequestingInformationPin of the RespondingBusinessAction using an o
1656 bject
1657 -- flow relationship leading from the RequestingBusinessAction
1658 -- to the RespondingBusinessAction.
1659 Missing on purpose
1660

```

```

1661 -- Constraint 52
1662 -- Each RespondingInformationPin of the RespondingBusinessAction MUST
1663 be
1664 -- connected with exactly one RespondingInformationPin of the
1665 RequestingBusinessAction
1666 -- using an object flow relationship leading from the
1667 RespondingBusinessAction
1668 -- to the RequestingBusinessAction
1669 Missing on purpose
1670

```

```

1671 -- Constraint 53

```

```

1672 -- If a BusinessTransactionPartition contains
1673 SharedBusinessEntityStates, each SharedBusinessEntityState
1674 -- MUST be the target of exactly one control flow relationship
1675 starting from the RequestingBusinessAction and
1676 -- MUST be the source of exactly one control flow relationship
1677 targeting a FinalState.
1678 context ActivityPartition inv: self.isBTPartition()
1679 implies self.owner.ownedElement->
1680 select( nodes | nodes.oclIsTypeOf(CentralBufferNode)
1681 and nodes.oclAsType(CentralBufferNode).isBESharedState() ) ->
1682 forAll( states | states.oclAsType(CentralBufferNode).incoming
1683 ->size()=1 and states.oclAsType(CentralBufferNode).incoming->
1684 forAll( income | income.oclAsType(ObjectFlow).source.
1685 oclAsType(Action).isReqAction() )
1686 and states.oclAsType(CentralBufferNode).outgoing->
1687 forAll( outgo | outgo.oclAsType(ObjectFlow).target.
1688 oclIsTypeOf(ActivityFinalNode)
1689 and states.oclAsType(CentralBufferNode).outgoing->size()=1 ) )
1690

```

```

1691 -- Constraint 54
1692 -- Each FinalState MUST be the target of one to many control flow
1693 relationships starting
1694 -- from the RequestingBusinessAction or from a
1695 SharedBusinessEntityState.
1696 context ActivityFinalNode inv: self.oclIsTypeOf(ActivityFinalNode)
1697 and self.owner.oclAsType(Activity).isBTransaction()
1698 implies self.incoming->forAll( income |
1699 income.oclAsType(ControlFlow).source.
1700 oclAsType(Action).isReqAction() or
1701 income.oclAsType(ObjectFlow).source.oclAsType(CentralBufferNode).is
1702 BESharedState() )
1703

```

```

1704 -- Constraint 55
1705 -- Each RequestingInformationPin and each RespondingInformationPin
1706 MUST have a classifier, this classifier MUST be an
1707 InformationEnvelope or a subtype defined in an
1708 extension/specialization module.
1709

```

1710 *Missing on purpose*

```

1712 -- Constraint 56
1713 -- Two RequestingInformationPins which are connected using an
1714 -- object flow MUST have the same classifier.
1715

```

1716 *Missing on purpose*

```

1718 -- Constraint 57
1719 -- Two RespondingInformationPins which are connected using an
1720 -- object flow MUST have the same classifier.
1721

```

1722 *Missing on purpose*

1723

```

1724 -- Constraint 58
1725 -- A BusinessCollaborationView MUST contain exactly one
1726 BusinessCollaborationUseCase.
1727 context Package inv: self.isBCollaborationV()
1728 implies self.ownedElement
1729 ->select(k|k.oclasType(UseCase).isBCollaborationUC())->size=1
1730

```

```

1731 -- Constraint 59
1732 -- A BusinessCollaborationView MUST contain two to many
1733 AuthorizedRoles.
1734 context Package inv: self.isBCollaborationV()
1735 implies self.ownedElement
1736 ->select(k|k.oclasType(Actor).isAuthorizedRole())->size>=2
1737

```

```

1738 -- Constraint 60
1739 -- A BusinessCollaborationUseCase MUST have two to many participates
1740 associations
1741 -- to AuthorizedRoles contained in the same BusinessCollaborationView.
1742 context UseCase inv: self.isBCollaborationUC()
1743 implies self.owner.ownedElement
1744 ->select(c|c.oclasTypeOf(Association)
1745 and c.oclasType(Association).isParticipates()
1746 and c.oclasType(Association).ownedEnd.type
1747 ->exists(t|t.oclasType(Actor).isAuthorizedRole()))->size>=2
1748

```

```

1749 -- Constraint 61
1750 -- Each AuthorizedRole contained in the BusinessCollaborationView MUST
1751 have exactly
1752 -- one participates association to the BusinessCollaborationUseCase
1753 included in the same BusinessCollaborationView.
1754 context Actor inv: self.isAuthorizedRole() and
1755 self.owner.oclasType(Package).isBCollaborationV()
1756 implies self.owner.ownedElement->
1757 select(as|as.oclasTypeOf(Association))->
1758 select(ass| ass.oclasType(Association).ownedEnd.type->
1759 exists(end|end.oclasType(Actor)=self)
1760 and ass.oclasType(Association).ownedEnd.type->
1761 exists(oend|oend.oclasType(UseCase).isBCollaborationUC())) ->size=1
1762

```

```

1763 -- Constraint 62
1764 -- A BusinessCollaborationUseCase MUST have one to many include
1765 relationships to another BusinessCollaborationUseCase
1766 -- or to a BusinessTransactionUseCase.
1767 context UseCase inv: self.isBCollaborationUC()
1768 implies self.include.addition->size>0
1769 and self.include.addition->
1770 forAll(Uc |Uc.oclasType(UseCase).isBCollaborationUC()
1771 or Uc.oclasType(UseCase).isBTransactionUC())
1772

```

```

1773 -- Constraint 63

```

```

1774 -- Exactly one BusinessCollaborationProtocol MUST be placed beneath
1775     each BusinessCollaborationUseCase.
1776 context Activity inv: self.isBCollaborationProtocol()
1777     implies self.owner.oclasType(UseCase).isBCollaborationUC()
1778

```

```

1779 -- Constraint 64
1780 -- A BusinessCollaborationProtocol MUST contain one to many
1781     BusinessTransactionActions and/or BusinessCollaborationAction.
1782 context Activity inv: self.isBCollaborationProtocol()
1783     implies self.ownedElement->exists( actions |
1784     actions.oclasType(CallBehaviorAction).isBTransactionAction()
1785     or actions.oclasType(CallBehaviorAction).isBCollaborationAction() )
1786

```

```

1787 -- Constraint 65
1788 -- Each BusinessTransactionAction MUST call exactly one
1789     BusinessTransaction
1790 context Action inv:
1791     self.oclasType(CallBehaviorAction).isBTransactionAction()
1792     implies
1793     self.oclasType(CallBehaviorAction).behavior.oclasType(Activity).isB
1794     Transaction()
1795

```

```

1796 -- Constraint 66
1797 -- Each BusinessTransaction called by a BusinessTransactionAction MUST
1798     be placed beneath a
1799     BusinessTransactionUseCase which is included in the
1800     BusinessCollaborationUseCase that covers
1801     the corresponding BusinessCollaborationProtocol.
1802 context Activity inv: self.oclasType(Activity).isBTransaction()
1803     and CallBehaviorAction.allInstances->
1804     exists( node | node.oclasType(CallBehaviorAction).behavior.
1805     oclasType(Activity)=self )
1806     implies self.owner.oclasType(UseCase).isBTransactionUC()
1807     and UseCase.allInstances->
1808     select( uc | uc.oclasType(UseCase).isBCollaborationUC() )->
1809     exists( anInclude | anInclude.oclasType(UseCase).include.addition
1810     -> exists( elem |
1811     elem.oclasType(UseCase)=self.owner.oclasType(UseCase) )
1812     and anInclude.oclasType(UseCase).ownedElement->exists( protocol |
1813     protocol.oclasType(Activity).isBCollaborationProtocol() ) )
1814

```

```

1815 -- Constraint 67
1816 -- Each BusinessCollaborationProtocol called by a
1817     BusinessCollaborationAction MUST be placed beneath a
1818     BusinessCollaborationProtocolUseCase which is included in the
1819     BusinessCollaborationUseCase that covers the corresponding
1820     BusinessCollaborationProtocol.
1821 context CallBehaviorAction inv: let protocol : Activity =
1822     self.behavior.oclasType(Activity) in self.isBCollaborationAction()
1823     implies protocol.owner.oclasType(UseCase).isBCollaborationUC()
1824     and self.owner.owner.oclasType(UseCase).isBCollaborationUC()
1825     and self.owner.owner.oclasType(UseCase).include.addition->

```

```
1826     exists( inc |
1827         inc.oclasType(UseCase)=protocol.owner.oclasType(UseCase))
1828
```

```
1829 -- Constraint 68
1830 -- A BusinessCollaborationProtocol MUST contain two to many
1831     BusinessCollaborationPartitions.
1832 context Activity inv: self.isBCollaborationProtocol()
1833     implies self.ownedElement->select( partitions |
1834         partitions.oclasType(ActivityPartition).
1835         isBCollaborationPartition()->size())>=2
1836
```

```
1837 -- Constraint 69
1838 -- The number of AuthorizedRoles in the BusinessCollaborationView MUST
1839     match the number of BusinessCollaborationPartitions
1840 -- in the BusinessCollaborationProtocol which is placed beneath the
1841     BusinessCollaborationUseCase of the same BusinessCollaborationView.
1842 context Package inv:
1843     let authRoles : Set(Element) = self.ownedElement->
1844         select( role | role.oclasType(Actor).isAuthorizedRole())
1845     in let collUC : Set(Element) = self.ownedElement->
1846         select( bCollUC | bCollUC.oclasType(UseCase).isBCollaborationUC())
1847     in let protocol : Set(Element) = collUC.ownedElement
1848         ->select(prot|prot.oclasType(Activity).isBCollaborationProtocol())
1849         ->asSet in let partitions : Set(Element) = protocol.ownedElement
1850         ->select(part|part.oclasType(ActivityPartition).
1851             isBCollaborationPartition()->
1852             asSet in self.isBCollaborationV()
1853             implies authRoles->size() = partitions->size())
1854
```

```
1855 -- Constraint 70
1856 -- Each AuthorizedRole in the BusinessCollaborationView MUST be
1857     assigned to a BusinessCollaborationPartition
1858 -- in the BusinessCollaborationProtocol which is placed beneath the
1859     BusinessCollaborationUseCase
1860 -- of the same BusinessCollaborationView.
1861 context Actor inv:
1862     let bCollUC : Set(Element) = self.owner.ownedElement->
1863         select ( uc | uc.oclasType(UseCase).isBCollaborationUC())
1864     in let bCollProtocol : Set(Element) = bCollUC.ownedElement->
1865         select ( protocol |
1866             protocol.oclasType(Activity).isBCollaborationProtocol()->asSet
1867         in let bCollPartition : Set(Element) =
1868             bCollProtocol.ownedElement -> select( partition |
1869                 partition.oclasType(ActivityPartition).isBCollaborationPartition())
1870         ->asSet in self.isAuthorizedRole() and
1871             self.owner.oclasType(Package).isBCollaborationV()
1872         implies bCollPartition.oclasType(ActivityPartition).represents->
1873             select( part | part.oclasType(Actor)=self.oclasType(Actor))
1874         ->size()=1
1875
```

```
1876 -- Constraint 71
1877 -- Each BusinessCollaborationPartition MUST be classified by exactly
1878     one AuthorizedRole included in the same
```

```

1879 -- BusinessCollaborationView as the BusinessCollaborationUseCase
1880 covering the BusinessCollaborationProtocol
1881 -- containing this BusinessCollaborationPartition.
1882 context ActivityPartition inv: self.isBCollaborationPartition()
1883 implies self.hlpOwningPackage().ownedElement->
1884 exists( actor | actor.oclAsType(Actor).isAuthorizedRole()
1885 and actor.oclAsType(Actor)=self.represents.oclAsType(Actor))
1886

```

```

1887 -- Constraint 72
1888 -- A BusinessCollaborationPartition MUST be either empty or contain
1889 one to many NestedBusinessCollaborations.
1890 context ActivityPartition inv: self.isBCollaborationPartition()
1891 implies self.owner.ownedElement->select( action |
1892 action.oclAsType(Action).inPartition->
1893 exists(element | element.oclAsType(ActivityPartition)=self))
1894 ->forall( elem | elem.oclAsType(Action).isBNestedCollaboration())
1895 and (self.ownedElement->size()==0 or self.ownedElement->forall(
1896 elem | elem.oclAsType(Action).isBNestedCollaboration()) )
1897

```

```

1898 -- Constraint 73
1899 -- Each BusinessTransactionAction MUST be the target of exactly one
1900 InitialFlow which source MUST be a BusinessCollaborationPartition.
1901 context CallBehaviorAction inv:
1902 self.oclAsType(CallBehaviorAction).isBTransactionAction()
1903 implies Dependency.allInstances->select( dependency |
1904 dependency.oclAsType(Dependency).supplier->
1905 exists( ends | ends.oclAsType(CallBehaviorAction)=self) and
1906 dependency.oclAsType(Dependency).isInitFlow()
1907 and dependency.oclAsType(Dependency).client->exists( end |
1908 end.oclAsType(ActivityPartition).isBCollaborationPartition()))->
1909 size()==1
1910

```

```

1911 -- Constraint 74
1912 -- Each BusinessTransactionAction MUST be the source of exactly one
1913 InitialFlow which target MUST be either a
1914 -- BusinessCollaborationPartition or a NestedBusinessCollaboration.
1915 context CallBehaviorAction inv:
1916 self.oclAsType(CallBehaviorAction).isBTransactionAction()
1917 and self.owner.oclAsType(Activity).isBCollaborationProtocol()
1918 implies Dependency.allInstances->select( dependency |
1919 dependency.oclAsType(Dependency).client->
1920 exists( ends | ends.oclAsType(CallBehaviorAction)=self)
1921 and dependency.oclAsType(Dependency).isInitFlow()
1922 and dependency.oclAsType(Dependency).supplier->exists(end |
1923 end.oclAsType(ActivityPartition).isBCollaborationPartition()
1924 or end.oclAsType(Action).isBNestedCollaboration()))->size()==1
1925

```

```

1926 -- Constraint 75
1927 -- The InitialFlow sourcing from a BusinessTransactionAction and the
1928 InitialFlow targeting a BusinessTransactionAction
1929 -- MUST NOT be targeting to / sourcing from the same
1930 BusinessCollaborationPartition, nor targeting to a
1931 NestedBusinessCollaboration

```

```

1932 -- within the same BusinessCollaborationPartition.
1933 context Dependency inv: self.isInitFlow()
1934     implies not(Dependency.allInstances->exists( dep |
1935 (dep.oclAsType(Dependency).supplier=self.client
1936 or ( dep.oclAsType(Dependency).supplier->forall(end |
1937 end.oclAsType(Action).isBNestedCollaboration())
1938 and
1939 dep.oclAsType(Dependency).supplier.oclAsType(Action).inPartition
1940 ->includesAll(self.client.oclAsType(ActivityPartition)) ) )
1941 and dep.oclAsType(Dependency).client=self.supplier and
1942 dep.oclAsType(Dependency).isInitFlow() ) )
1943

```

```

1944 -- Constraint 76
1945 -- If a BusinessTransactionAction calls a two-way BusinessTransaction,
1946 this BusinessTransactionAction MUST be the source of exactly one
1947 RespondingFlow which target MUST be a
1948 BusinessCollaborationPartition.
1949 context CallBehaviorAction inv: self.isBTransactionAction() and
1950 self.behavior.oclAsType(Activity).isTwoWayBTransaction()
1951 implies self.owner.ownedElement->
1952 select( dep | dep.oclAsType(Dependency).isReFlow() and
1953 dep.oclAsType(Dependency).client->
1954 exists( end | end.oclAsType(CallBehaviorAction)=self)
1955 and dep.oclAsType(Dependency).supplier->
1956 exists( end |
1957 end.oclAsType(ActivityPartition).isBCollaborationPartition() ) )
1958 ->size()=1
1959

```

```

1960 -- Constraint 77
1961 -- If a BusinessTransactionAction calls a two-way BusinessTransaction,
1962 this BusinessTransactionAction MUST be the target of exactly one
1963 RespondingFlow which source MUST be either a
1964 BusinessCollaborationPartition or a NestedBusinessCollaboration.
1965 context CallBehaviorAction inv: self.isBTransactionAction() and
1966 self.behavior.oclAsType(Activity).isTwoWayBTransaction()
1967 implies self.owner.ownedElement->
1968 select( dep | dep.oclAsType(Dependency).isReFlow()
1969 and dep.oclAsType(Dependency).supplier->
1970 exists( end | end.oclAsType(CallBehaviorAction)=self)
1971 and dep.oclAsType(Dependency).client->
1972 exists( end |
1973 end.oclAsType(ActivityPartition).isBCollaborationPartition()
1974 or end.oclAsType(Action).isBNestedCollaboration() ) )->size()=1
1975

```

```

1976 -- Constraint 78
1977 -- The RespondingFlow sourcing from a BusinessTransactionAction and
1978 the RespondingFlow targeting a BusinessTransactionAction
1979 -- MUST NOT be targeting to /sourcing from the same
1980 BusinessCollaborationPartition, nor targeting to a
1981 NestedBusinessCollaboration
1982 -- within the same BusinessCollaborationPartition.
1983 context Dependency inv: self.isReFlow()
1984     implies not(Dependency.allInstances->exists( dep |
1985 (dep.oclAsType(Dependency).supplier=self.client

```



```

1986     or dep.oclAsType(Dependency).supplier.owner.
1987     oclAsType(ActivityPartition)=self.client.
1988     oclAsType(ActivityPartition) )
1989     and ( dep.oclAsType(Dependency).client=self.supplier
1990     or dep.oclAsType(Dependency).supplier.owner.
1991     oclAsType(ActivityPartition)=self.supplier.
1992     oclAsType(ActivityPartition) )
1993     and dep.oclAsType(Dependency).isReFlow() ))
1994

```

```

1995 -- Constraint 79
1996 -- If a BusinessTransactionAction calls a one-way BusinessTransaction,
1997 -- this BusinessTransactionAction MUST NOT be the source of a
1998 -- RespondingFlow and MUST NOT be the target of a RespondingFlow.
1999 context CallBehaviorAction inv: self.isBTransactionAction() and
2000 self.behavior.oclAsType(Activity).isOneWayBTransaction()
2001 implies not(self.owner.ownedElement->
2002 exists( dep | dep.oclAsType(Dependency).isReFlow() and (
2003 dep.oclAsType(Dependency).client->
2004 exists( end | end.oclAsType(CallBehaviorAction)=self or
2005 dep.oclAsType(Dependency).supplier->
2006 exists( end | end.oclAsType(CallBehaviorAction)=self )))
2007

```

```

2008 -- Constraint 80
2009 -- The RespondingFlow targeting a BusinessTransactionAction must start
2010 -- from the
2011 -- BusinessCollaborationPartition / NestedBusinessCollaboration which
2012 -- is the target of the InitialFlow starting
2013 -- from the same BusinessTransactionAction.
2014 context Dependency inv: self.isReFlow()
2015     and self.supplier->forall( end |
2016     end.oclAsType(CallBehaviorAction).isBTransactionAction())
2017     implies Dependency.allInstances->
2018     exists( flow | flow.oclAsType(Dependency).isInitFlow() and (
2019     flow.oclAsType(Dependency).supplier.oclAsType(ActivityPartition)=
2020     self.oclAsType(Dependency).client.oclAsType(ActivityPartition) or
2021     flow.oclAsType(Dependency).supplier.oclAsType(Action)=
2022     self.oclAsType(Dependency).client.oclAsType(Action) ) and
2023     flow.oclAsType(Dependency).client.oclAsType(CallBehaviorAction)=
2024     self.supplier.oclAsType(CallBehaviorAction) )
2025

```

```

2026 -- Constraint 81
2027 -- The RespondingFlow starting from a BusinessTransactionAction must
2028 -- target the BusinessCollaborationPartition which is
2029 -- the source of the InitialFlow targeting to the same
2030 -- BusinessTransactionAction.
2031 context Dependency inv: self.isReFlow()
2032     and self.oclAsType(Dependency).client->forall( end |
2033     end.oclAsType(CallBehaviorAction).isBTransactionAction())
2034     implies Dependency.allInstances->exists( flow |
2035     flow.oclAsType(Dependency).isInitFlow() and
2036     flow.oclAsType(Dependency).client.oclAsType(ActivityPartition)=
2037     self.oclAsType(Dependency).supplier.oclAsType(ActivityPartition)
2038     and flow.oclAsType(Dependency).supplier.
2039     oclAsType(CallBehaviorAction)=

```

```
2040 self.client.oclAsType(CallBehaviorAction) )
2041
```

```
2042 -- Constraint 82
2043 -- A NestedBusinessCollaboration MUST be the target of exactly one
2044 InitialFlow.
2045 context Action inv: self.isBNestedCollaboration()
2046 implies Dependency.allInstances->
2047 select( dep | dep.oclAsType(Dependency).supplier->
2048 exists(end | end.oclAsType(Action)=self.oclAsType(Action)))
2049 ->size()=1
2050
```

```
2051 -- Constraint 83
2052 -- A NestedBusinessCollaboration MAY be the source of a
2053 RespondingFlow,
2054 -- but MUST NOT be the source of more than one RespondingFlow.
2055 context Action inv:
2056 self.isBNestedCollaboration()
2057 implies Dependency.allInstances->
2058 select( dep | dep.oclAsType(Dependency).isReFlow()
2059 and dep.oclAsType(Dependency).client->exists(end |
2060 end.oclAsType(Action)=self.oclAsType(Action)))->size()<=1
2061
```

```
2062 -- Constraint 84
2063 -- A BusinessCollaborationAction MUST be the target of two to many
2064 InformationFlows
2065 context CallBehaviorAction inv: self.isBCollaborationAction()
2066 implies Dependency.allInstances->select( dep |
2067 dep.oclAsType(Dependency).supplier->
2068 exists( end | end.oclAsType(CallBehaviorAction)=self))->size()>=2
2069
```

```
2070 -- Constraint 85
2071 -- A BusinessCollaborationAction MUST not be the source of an
2072 InformationFlow.
2073 context CallBehaviorAction inv: self.isBCollaborationAction()
2074 implies not(Dependency.allInstances->exists( dep |
2075 dep.oclAsType(Dependency).client->
2076 exists( end | end.oclAsType(CallBehaviorAction)=self)))
2077
2078
```

```
2079 -- Constraint 86
2080 -- A BusinessCollaborationAction MUST not be the source and MUST not
2081 be the target of an InitialFlow.
2082 context CallBehaviorAction inv: self.isBCollaborationAction()
2083 implies not(Dependency.allInstances->exists( dep |
2084 dep.oclAsType(Dependency).isInitFlow()
2085 and (dep.oclAsType(Dependency).client->exists( end |
2086 end.oclAsType(CallBehaviorAction).isBCollaborationAction())
2087 or dep.oclAsType(Dependency).supplier->exists( end |
2088 end.oclAsType(CallBehaviorAction).isBCollaborationAction()))))
2089
```

```

2090 -- Constraint 87
2091 -- A BusinessCollaborationAction MUST not be the source and MUST not
2092 -- be the target of a RespondingFlow.
2093 context CallBehaviorAction inv: self.isBCollaborationAction()
2094     implies not(Dependency.allInstances->exists( dep |
2095         dep.oclAsType(Dependency).isReFlow()
2096         and (dep.oclAsType(Dependency).client->exists( end |
2097             end.oclAsType(CallBehaviorAction).isBCollaborationAction()
2098             or dep.oclAsType(Dependency).supplier->exists( end |
2099                 end.oclAsType(CallBehaviorAction).isBCollaborationAction()))))
2100

```

```

2101 -- Constraint 88
2102 -- A BusinessTransactionAction MUST not be the source and MUST not be
2103 -- the target of an InformationFlow that is neither
2104 -- stereotyped as InitialFlow nor as RespondingFlow.
2105 context CallBehaviorAction inv: self.isBTransactionAction()
2106     implies not( Dependency.allInstances-> exists( dep |
2107         dep.oclAsType(Dependency).client->
2108         exists(end | end.oclAsType(CallBehaviorAction)=self)
2109         and not(dep.oclAsType(Dependency).isInitFlow()
2110             or dep.oclAsType(Dependency).isReFlow() ) )
2111         and not( Dependency.allInstances-> exists( dep |
2112             dep.oclAsType(Dependency).supplier->
2113             exists(end | end.oclAsType(CallBehaviorAction)=self)
2114             and not(dep.oclAsType(Dependency).isInitFlow()
2115                 or dep.oclAsType(Dependency).isReFlow() ) ) )
2116

```

```

2117 -- Constraint 89
2118 -- A NestedBusinessCollaboration MUST not be the source and MUST not
2119 -- be the target of an InformationFlow that
2120 -- targets to / sources from a BusinessCollaborationAction.
2121 context Action inv: self.oclAsType(Action).isBNestedCollaboration()
2122     implies not(Dependency.allInstances->exists( dep |
2123         dep.oclAsType(Dependency).supplier->exists(end |
2124             end.oclAsType(Action)=self)
2125             and dep.oclAsType(Dependency).client->exists(end |
2126                 end.oclAsType(CallBehaviorAction).isBCollaborationAction()))))
2127         and not(Dependency.allInstances->exists( dep |
2128             dep.oclAsType(Dependency).client->exists(end |
2129                 end.oclAsType(Action)=self)
2130                 and dep.oclAsType(Dependency).supplier->exists(end|
2131                     end.oclAsType(CallBehaviorAction).isBCollaborationAction()))))
2132

```

```

2133 -- Constraint 90
2134 -- The number of InformationFlows targeting a
2135 -- BusinessCollaborationAction MUST match the number of
2136 -- BusinessCollaborationPartitions
2137 -- contained in the BusinessCollaborationProtocol that is called by
2138 -- this BusinessCollaborationAction.
2139 context CallBehaviorAction inv: self.isBCollaborationAction()
2140     implies self.owner.ownedElement->select( dep |
2141         dep.oclAsType(Dependency).supplier->
2142         exists( end | end.oclAsType(CallBehaviorAction)=self ))->size() =
2143         self.behavior.oclAsType(Activity).ownedElement->

```

```
2144 select( partitions |
2145 partitions.oclAsType(ActivityPartition).
2146 isBCollaborationPartition()->size()
2147
```

```
2148 -- Constraint 91
2149 -- Either an AuthorizedRole classifying a
2150 BusinessCollaborationPartition that is the source of an
2151 InformationFlow
2152 -- targeting a BusinessCollaborationAction MUST match an
2153 AuthorizedRole classifying a BusinessCollaborationPartition in the
2154 BusinessCollaborationProtocol that is called by this
2155 BusinessCollaborationAction or the InformationFlow must be
2156 classified
2157 -- by an AuthorizedRole classifying a BusinessCollaborationPartition
2158 in the BusinessCollaborationProtocol that is called
2159 -- by this BusinessCollaborationAction.
2160
2161 context ActivityPartition inv: self.isBCollaborationPartition()
2162 and self.represents.oclAsType(Actor).isAuthorizedRole()
2163 and Dependency.allInstances->exists( dep |
2164 dep.oclAsType(Dependency).client->exists( end |
2165 end.oclAsType(ActivityPartition)=self)
2166 and dep.oclAsType(Dependency).supplier-> exists( end |
2167 end.oclAsType(CallBehaviorAction).isBCollaborationAction()))
2168 implies Dependency.allInstances->select( dep |
2169 dep.oclAsType(Dependency).client->
2170 exists( end | end.oclAsType(ActivityPartition)=self)
2171 and dep.oclAsType(Dependency).supplier-> exists( end |
2172 end.oclAsType(CallBehaviorAction).isBCollaborationAction())) ->
2173 forAll(dependency | dependency.oclAsType(Dependency).supplier->
2174 exists(end|end.oclAsType(CallBehaviorAction).behavior.
2175 oclAsType(Activity).ownedElement->
2176 exists( partition | partition.oclAsType(ActivityPartition).
2177 isBCollaborationPartition()
2178 and partition.oclAsType(ActivityPartition).represents.
2179 oclAsType(Actor)=self.represents.oclAsType(Actor))))
2180
```

```
2181 -- Constraint 92
2182 -- A BusinessRealizationView MUST contain exactly one
2183 BusinessRealizationUC, two to many AuthorizedRoles,
2184 -- and two to many participates associations.
2185 context Package inv: let ownEl : Set(Element) = self.ownedElement in
2186 self.isBRealizationV()
2187 implies ownEl -> select ( ass |
2188 ass.oclAsType(Association).isParticipates()->size>=2
2189 and ownEl -> select ( bRealUC |
2190 bRealUC.oclAsType(UseCase).isBRealizationUC()
2191 and not(bRealUC.oclAsType(UseCase).isBCollaborationUC()))
2192 ->size()=1 and ownEl
2193 -> select ( roles |roles.oclAsType(Actor).isAuthorizedRole())
2194 ->size()>=2
2195
```

```
2196 -- Constraint 93
```

```

2197 -- A BusinessRealization MUST be associated with two to many
2198 AuthorizedRoles via stereotyped binary participates associations.
2199 context UseCase inv: self.isBRealizationUC() and
2200 not(self.isBCollaborationUC())
2201 implies self.owner.ownedElement->
2202 select(a | a.oclasType(Association).isParticipates()
2203 and a.oclasType(Association).ownedEnd.type
2204 ->exists(t | t.oclasType(Actor).isAuthorizedRole())
2205 and a.oclasType(Association).ownedEnd.type
2206 ->exists(t | t.oclasType(UseCase)=self) )->size()>=2
2207

```

```

2208 -- Constraint 94
2209 -- A BusinessRealization MUST be the source of exactly one realization
2210 dependency to a BusinessCollaborationUseCase.
2211 context UseCase inv: self.isBRealizationUC()
2212 implies self.owner.ownedElement->exists( ass |
2213 ass.oclasType(Realization).isRealizes()
2214 and ass.oclasType(Realization).client->exists(uc |
2215 uc.oclasType(UseCase)=self)
2216 and ass.oclasType(Realization).supplier->exists(uc |
2217 uc.oclasType(UseCase).isBCollaborationUC())
2218

```

```

2219 -- Constraint 95
2220 -- A BusinessRealization MUST NOT be the source or target of an
2221 include or extends association.
2222 context UseCase inv: self.isBRealizationUC()
2223 implies self.include->size()=0
2224 and UseCase.allInstances
2225 ->forall(k | k.oclasType(UseCase).include.addition
2226 ->forall(s | not(s.oclasType(UseCase)=self)))
2227 and UseCase.allInstances
2228 ->forall(u | u.oclasType(UseCase).extend.extendedCase
2229 ->forall(t | not(t.oclasType(UseCase)=self)))
2230 and self.extend->size=0
2231

```

```

2232 -- Constraint 96
2233 -- All dependencies from/to an AuthorizedRole must be stereotyped as
2234 mapsTo.
2235 context Package inv: self.isBRealizationV()
2236 implies self.ownedElement->select( dep |
2237 dep.oclasType(Dependency).client->
2238 exists( k | k.oclasType(Actor).isAuthorizedRole() )
2239 and dep.oclasType(Dependency).supplier->
2240 exists( k | k.oclasType(Actor).isAuthorizedRole()))->
2241 forall(dep | dep.oclasType(Dependency).isMapsTo())
2242

```

```

2243 -- Constraint 97
2244 -- An AuthorizedRole, which participates in a BusinessRealization,
2245 must be the target of exactly one mapsTo dependency
2246 -- starting from a BusinessPartner. Furthermore the AuthorizedRole,
2247 which participates in the BusinessRealization must
2248 -- be the source of exactly one mapsTo dependency targeting an
2249 AuthorizedRole participating in a BusinessCollaborationUseCase.

```

```

2250 context Actor inv: self.isAuthorizedRole()
2251     and self.owner.ownedElement ->
2252     exists( ass | ass.oclasType(Association).isParticipates()
2253     and ass.oclasType(Association).ownedEnd.type->exists( end |
2254     end.oclasType(Actor)=self)
2255     and ass.oclasType(Association).ownedEnd.type->exists( end |
2256     end.oclasType(UseCase).isBRealizationUC()
2257     and not(end.oclasType(UseCase).isBCollaborationUC()))
2258     implies self.owner.ownedElement->select( dep |
2259     dep.oclasType(Dependency).isMapsTo()
2260     and dep.oclasType(Dependency).client
2261     ->exists(t|t.oclasType(Actor).isBPartner()
2262     and not(t.oclasType(Actor).isAuthorizedRole()))-> select(dep |
2263     dep.oclasType(Dependency).supplier->
2264     exists(t|t.oclasType(Actor)=self))-> size()=1 and
2265     self.oclasType(Actor).hlpMapsToAuthRoleParticipates()
2266

```

```

2267 -- Constraint 98
2268 -- AuthorizedRoles in a BusinessRealizationView must have a unique
2269     name within the scope of the package, they are located in
2270 context Package inv:
2271     let authRoles : Set(Element) = self.ownedElement->select( roles |
2272     roles.oclasType(Actor).isAuthorizedRole()
2273     in self.isBRealizationV() implies authRoles->size() =
2274     authRoles->collect(p|p.oclasType(Actor).name)->asSet->size()
2275

```

```

2276 -- Constraint 99
2277 -- The number of AuthorizedRoles participating in a
2278     BusinessCollaborationUseCase MUST match the number of
2279     AuthorizedRoles participating in the BusinessRealization realizing
2280     this BusinessCollaborationUseCase
2281 context Package inv:
2282     let ass : Set(Element) = self.ownedElement->select( ass |
2283     ass.oclasType(Association).isParticipates()
2284     in self.isBRealizationV()
2285     implies ass->select( ends
2286     |ends.oclasType(Association).ownedEnd.type->exists( auth |
2287     auth.oclasType(Actor).isAuthorizedRole()
2288     and ends.oclasType(Association).ownedEnd.type->exists( uc |
2289     uc.oclasType(UseCase).isBRealizationUC())) ->
2290     size() = ass->select( ends
2291     |ends.oclasType(Association).ownedEnd.type->exists( auth |
2292     auth.oclasType(Actor).isAuthorizedRole()
2293     and ends.oclasType(Association).ownedEnd.type->exists( uc |
2294     uc.oclasType(UseCase).isBCollaborationUC())) -> size()
2295

```

```

2296 -- Constraint 100
2297 -- A BusinessInformationView MUST contain one to many
2298     InformationEnvelopes or subtypes thereof defined in any other
2299     extension/specialization module. Furthermore, it MAY contain any
2300     other document modeling artifacts.
2301 context Class inv: self.isBInformationV()
2302     implies self.ownedElement->exists( doc | doc.isInfEnvelope()

```

2303

```
2304 -- Convenience method:
2305 -- Evaluates if a package is stereotyped as bCollMode
2306 def:
2307     let: isBCollModel() : Boolean =
2308         self.oclAsType(Package).extension_bLibrary.
2309         oclIsKindOf(UMM2_Profile__bCollModel)
2310
```

```
2311 -- Convenience method:
2312 -- Evaluates if a package is stereotyped as bCollaborationV
2313 def:
2314     let: isBCollaborationV() : Boolean =
2315         self.oclAsType(Package).extension_bLibrary.
2316         oclIsKindOf(UMM2_Profile__bCollaborationV)
2317
```

```
2318 -- Convenience method:
2319 -- Evaluates if a package is stereotyped as bRealizationV
2320 def:
2321     let: isBRealizationV() : Boolean =
2322         self.oclAsType(Package).extension_bLibrary.
2323         oclIsKindOf(UMM2_Profile__bRealizationV)
2324
```

```
2325 -- Convenience method:
2326 -- Evaluates if a package is stereotyped as bTransactionV
2327 def:
2328     let: isBTransactionV() : Boolean =
2329         self.oclAsType(Package).extension_bLibrary.
2330         oclIsKindOf(UMM2_Profile__bTransactionV)
2331
```

```
2332 -- Convenience method:
2333 -- Evaluates if a package is stereotyped as bChoreographyV
2334 def:
2335     let: isBChoreographyV() : Boolean =
2336         self.oclAsType(Package).extension_bLibrary.
2337         oclIsKindOf(UMM2_Profile__bChoreographyV)
2338
```

```
2339 -- Convenience method:
2340 -- Evaluates if a package is stereotyped as bInformationV
2341 def:
2342     let: isBInformationV() : Boolean =
2343         self.oclAsType(Package).extension_bLibrary.
2344         oclIsKindOf(UMM2_Profile__bInformationV)
2345
```

```
2346 -- Convenience method:
2347 -- Evaluates if a package is stereotyped as bRequirementsV
2348 def:
2349     let: isBRequirementsV() : Boolean =
2350         self.oclAsType(Package).extension_bLibrary.
2351         oclIsKindOf(UMM2_Profile__bRequirementsV)
2352
```

```
2353 -- Convenience method:
2354 -- Evaluates if a package is stereotyped as bDomainV
2355 def:
2356     let: isBDomainV() : Boolean =
2357         self.oclAsType(Package).extension_bLibrary.
2358         oclIsKindOf(UMM2_Profile__bDomainV)
```

```
2360 -- Convenience method:
2361 -- Evaluates if a package is stereotyped as bPartnerV
2362 def:
2363     let: isBPartnerV() : Boolean =
2364         self.oclAsType(Package).extension_bLibrary.
2365         oclIsKindOf(UMM2_Profile__bPartnerV)
```

```
2367 -- Convenience method:
2368 -- Evaluates if a package is stereotyped as bEntityV
2369 def:
2370     let: isBEntityV() : Boolean =
2371         self.oclAsType(Package).extension_bLibrary.
2372         oclIsKindOf(UMM2_Profile__bEntityV)
```

```
2374 -- Convenience method:
2375 -- Evaluates if a package is stereotyped as bArea
2376 def:
2377     let: isBArea() : Boolean =
2378         self.oclAsType(Package).extension_bLibrary.
2379         oclIsKindOf(UMM2_Profile__bArea)
```

```
2381 -- Convenience method:
2382 -- Evaluates if a package is stereotyped as ProcessArea
2383 def:
2384     let: isProcessArea() : Boolean =
2385         self.oclAsType(Package).extension_bLibrary.
2386         oclIsKindOf(UMM2_Profile__ProcessArea)
```

```
2388 -- Convenience method:
2389 -- Evaluates if a use case is stereotyped as bProcessUC
2390 def:
2391     let: isBProcessUseCase() : Boolean =
2392         self.oclAsType(UseCase).extension_bProcessUC.isDefined
```

```
2394 -- Convenience method:
2395 -- Evaluates if an activity is stereotyped as bProcess
2396 def:
2397     let: isBProcess() : Boolean =
2398         self.oclAsType(Activity).extension_bProcess.isDefined
```

```
2400 -- Convenience method:
2401 -- Evaluates if an actor is stereotyped as bPartner
```



```
2402 def:
2403     let: isBPartner() : Boolean =
2404         self.oclasType(Actor).extension_Stakeholder.
2405         oclIsKindOf(UMM2_Profile__bPartner)
2406
```

```
2407 -- Convenience method:
2408 -- Evaluates if an association is stereotyped as participates
2409 def:
2410     let: isParticipates() : Boolean =
2411         self.oclasType(Association).extension_participates.isDefined
2412
```

```
2413 -- Convenience method:
2414 -- Evaluates if an actor is stereotyped as stakeholder
2415 def:
2416     let: isStakeholder() : Boolean =
2417         self.oclasType(Actor).extension_Stakeholder.isDefined
2418
```

```
2419 -- Convenience method:
2420 -- Evaluates if a dependency is stereotyped as isOfInterestTo
2421 def:
2422     let: isOfInterestTo() : Boolean =
2423         self.oclasType(Dependency).extension_isOfInterestTo.isDefined
2424
```

```
2425 -- Convenience method:
2426 -- Evaluates if an actor is stereotyped as authorized role
2427 def:
2428     let: isAuthorizedRole() : Boolean =
2429         self.oclasType(Actor).extension_AuthorizedRole.isDefined
2430
```

```
2431 -- Convenience method:
2432 -- Evaluates if an use case is stereotyped as BCollaborationUC
2433 def:
2434     let: isBCollaborationUC() : Boolean =
2435         self.oclasType(UseCase).extension_bProcessUC.
2436         oclIsKindOf(UMM2_Profile__bCollaborationUC)
2437
```

```
2438 -- Convenience method:
2439 -- Evaluates if a class is stereotyped as bEntity
2440 def:
2441     let: isBEntity() : Boolean =
2442         self.oclasType(Class).extension_bEntity.isDefined
2443
```

```
2444 -- Convenience method:
2445 -- Evaluates if a package is stereotyped as bDataV
2446 def:
2447     let: isBDataV() : Boolean =
2448         self.oclasType(Package).extension_bLibrary.
2449         oclIsKindOf(UMM2_Profile__bDataV)
2450
```

```
2451 -- Convenience method:
2452 -- Evaluates if an use case is stereotyped as bTransactionUC
2453 def:
2454     let: isBTransactionUC() : Boolean =
2455         self.oclAsType(UseCase).extension_bProcessUC.
2456         oclIsKindOf(UMM2_Profile__bTransactionUC)
2457
```

```
2458 -- Convenience method:
2459 -- Evaluates if a modeling is an activity partition
2460 def:
2461     let: isActivityPartition() : Boolean =
2462         self.oclIsTypeOf(ActivityPartition)
2463
```

```
2464 -- Convenience method:
2465 -- Evaluates if an action is stereotyped as bProcessAction
2466 def:
2467     let: isBProcessAction() : Boolean =
2468         self.oclAsType(Action).extension_bProcessAction.isDefined
2469
```

```
2470 -- Convenience method:
2471 -- Evaluates if an use case is not stereotyped as bTransactionUC
2472 def:let: isNotBTransactionUC() : Boolean =
2473     not(self.oclAsType(UseCase).isBTransactionUC())
2474
```

```
2475 -- Convenience method:
2476 -- Evaluates if an activity is stereotyped as bCollaborationProtocol
2477 def:let: isBCollaborationProtocol() : Boolean =
2478
2479     self.oclAsType(Activity).extension_bCollaborationProtocol.isDefined
2480
```

```
2481 -- Convenience method:
2482 -- Evaluates if a call behavior action is stereotyped as
2483     bTransactionAction
2484 def:
2485     let: isBTransactionAction() : Boolean =
2486
2487         self.oclAsType(CallBehaviorAction).extension_bTransactionAction.isD
2488         efined
2489
```

```
2490 -- Convenience method:
2491 -- Evaluates if a call behavior action is stereotyped as
2492     bCollaborationAction
2493 def:
2494     let: isBCollaborationAction() : Boolean =
2495         self.oclAsType(CallBehaviorAction).
2496         extension_bCollaborationAction.isDefined
2497
```

```
2498 -- Convenience method:
2499 -- Evaluates if a class is stereotyped as InfEnvelope
```

```
2500 def:
2501     let: isInfEnvelope() : Boolean =
2502     self.oclasType(Class).extension_InfEnvelope.isDefined
2503
```

```
2504 -- Convenience method:
2505 -- Evaluates if an object node is stereotyped as bESharedState
2506 def:
2507     let: isBSharedState() : Boolean =
2508     self.oclasType(ObjectNode).extension_bESharedState.isDefined
2509
```

```
2510 -- Convenience method:
2511 -- Evaluates if a state is stereotyped as bEState
2512 def:
2513     let: isBEState() : Boolean =
2514     self.oclasType(State).extension_bEState.isDefined
2515
```

```
2516 -- Convenience method:
2517 -- Evaluates if an activity is stereotyped as bTransaction
2518 def:
2519     let: isBTransaction() : Boolean =
2520     self.oclasType(Activity).extension_bTransaction.isDefined
2521
```

```
2522 -- Convenience method:
2523 -- Evaluates if an activity partition is stereotyped as bTPartition
2524 def:
2525     let: isBTPartition() : Boolean =
2526     self.oclasType(ActivityPartition).extension_bTPartition.isDefined
2527
```

```
2528 -- Convenience method:
2529 -- Evaluates if an action is stereotyped as ReqAction
2530 def:
2531     let: isReqAction() : Boolean =
2532     self.oclasType(Action).extension_BusinessAction.
2533     oclIsKindOf(UMM2_Profile__ReqAction)
2534
```

```
2535 -- Convenience method:
2536 -- Evaluates if an action is stereotyped as ResAction
2537 def:
2538     let: isResAction() : Boolean =
2539     self.oclasType(Action).extension_BusinessAction.
2540     oclIsKindOf(UMM2_Profile__ResAction)
2541
```

```
2542 -- Convenience method:
2543 -- Evaluates if an use case is stereotyped as bRealizationUC
2544 def:
2545     let: isBRealizationUC() : Boolean =
2546     self.oclasType(UseCase).extension_bRealizationUC.isDefined
2547
```

```

2548 -- Convenience method:
2549 -- Evaluates if a dependency is stereotyped as realizes
2550 def:
2551     let: isRealizes() : Boolean =
2552         self.oclasType(Realization).extension_realizes.isDefined
2553

```

```

2554 -- Convenience method:
2555 -- Evaluates if a dependency is stereotyped as mapsTo.
2556 def:
2557     let: isMapsTo() : Boolean =
2558         self.oclasType(Dependency).extension_mapsTo.isDefined
2559

```

```

2560 -- Convenience method for evaluating if there is a mapsTo dependency
2561 -- from an authorized role
2562 -- that leads to another authorized role participating in a business
2563 -- collaboration use case
2564 def:
2565     let: hlpMapsToAuthRoleParticipates() : Boolean =
2566         self.owner.ownedElement->
2567         select( dep | dep.oclasType(Dependency).isMapsTo()
2568         and dep.oclasType(Dependency).client->
2569         exists( t | t.oclasType(Actor)=self)
2570         and dep.oclasType(Dependency).supplier->
2571         exists(t | t.oclasType(Actor).isAuthorizedRole()
2572         and t.oclasType(Actor).hlpParticipatesBCollaborationUC() ))
2573         ->size()-1
2574

```

```

2575 -- Convenience method for evaluating if a certain authorized role
2576 -- participates in a business
2577 -- collaboration use case
2578 def:
2579     let: hlpParticipatesBCollaborationUC() : Boolean =
2580         self.owner.ownedElement->select( ass |
2581         ass.oclasType(Association).isParticipates()
2582         and ass.oclasType(Association).ownedEnd.type->exists( end |
2583         end.oclasType(Actor)=self)
2584         and ass.oclasType(Association).ownedEnd.type->exists( end |
2585         end.oclasType(UseCase).isBCollaborationUC() )->size()-1
2586

```

```

2587 -- Convenience method:
2588 -- Evaluates if an activity partition is stereotyped as
2589 -- bCollaborationPartition
2590 def:
2591     let: isBCollaborationPartition() : Boolean =
2592         self.oclasType(ActivityPartition).
2593         extension_bCollaborationPartition.isDefined
2594

```

```

2595 -- Convenience method:
2596 -- Evaluates if a business transaction MUST have an responding
2597 -- information pin

```

```
2598 -- (i.e., is a two-way transaction) according to its business
2599 transaction pattern
2600 def:
2601     let: hlpMustHaveResInfPin() : Boolean =
2602         self.oclasType(Activity).extension_bTransaction.
2603         businessTransactionType='RequestResponse'
2604         or self.oclasType(Activity).extension_bTransaction.
2605         businessTransactionType='QueryResponse'
2606         or self.oclasType(Activity).extension_bTransaction.
2607         businessTransactionType='RequestConfirm'
2608         or self.oclasType(Activity).extension_bTransaction.
2609         businessTransactionType='CommercialTransaction'
2610
```

```
2611 -- Convenience method:
2612 -- Evaluates if a business transaction MUST NOT have an responding
2613 information pin
2614 -- (i.e., is an one-way transaction) according to its business
2615 transaction pattern
2616 def:
2617     let: hlpMustNotHaveResInfPin() : Boolean =
2618         self.oclasType(Activity).extension_bTransaction.
2619         businessTransactionType='Notification'
2620         or self.oclasType(Activity).extension_bTransaction.
2621         businessTransactionType='InformationDistribution'
2622
```

```
2623 -- Convenience method:
2624 -- Evaluates if an action is stereotyped as bNestedCollaboration
2625 def:
2626     let: isBNestedCollaboration() : Boolean =
2627         self.oclasType(Action).extension_bNestedCollaboration.isDefined
2628
```

```
2629 -- Convenience method:
2630 -- Evaluates if a dependency is stereotyped as InitFlow
2631 def:
2632     let: isInitFlow() : Boolean =
2633         self.oclasType(Dependency).extension_initFlow.isDefined
2634
```

```
2635 -- Convenience method:
2636 -- Evaluates if a dependency is stereotyped as ReFlow
2637 def:
2638     let: isReFlow() : Boolean =
2639         self.oclasType(Dependency).extension_reFlow.isDefined
2640
```

2641 **Copyright Statement**

2642

2643 Copyright © UN/CEFACT 2008. All Rights Reserved.

2644 This document and translations of it may be copied and furnished to others, and derivative works that  
2645 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published  
2646 and distributed, in whole or in part, without restriction of any kind, provided that the above copyright  
2647 notice and this paragraph are included on all such copies and derivative works. However, this document  
2648 itself may not be modified in any way, such as by removing the copyright notice or references to  
2649 UN/CEFACT except as required to translate it into languages other than English.

2650 The limited permissions granted above are perpetual and will not be revoked by UN/CEFACT or its  
2651 successors or assigns.

2652 This document and the information contained herein is provided on an "AS IS" basis and UN/CEFACT  
2653 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY  
2654 THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED  
2655 WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.