Business Architecture Core Metamodel

Version 1.0—beta-8

OMG Document Number: dtc/2023-05-02-06-04

Normative reference: https://www.omg.org/spec/BACM/1.0

This OMG document replaces the beta-submission document (dtc/22-06-01bmi/22-03-15). It is an OMG Adopted Beta Specification and is currently in the finalization phase. Comments on the content of this document are welcome and should be directed to issues@omg.org by May 31, 2023.

You may view the pending issues for this specification from the OMG revision issues web page https://issues.omg.org/issues/lists.

The FTF Recommendation and Report for this specification will be published in March 2024. If you are reading this after that date, please download the available specification from the OMG Specifications Catalog.
USE OF SPECIFICATION - TERMS, CONDITIONS & NOTICES

The material in this document details an Object Management Group specification in accordance with the terms, conditions and notices set forth below. This document does not represent a commitment to implement any portion of this specification in any company's products. The information contained in this document is subject to change without notice.

LICENSES

The companies listed above have granted to the Object Management Group, Inc. (OMG) a nonexclusive, royalty-free, paid up, worldwide license to copy and distribute this document and to modify this document and distribute copies of the modified version. Each of the copyright holders listed above has agreed that no person shall be deemed to have infringed the copyright in the included material of any such copyright holder by reason of having used the specification set forth herein or having conformed any computer software to the specification.

Subject to all of the terms and conditions below, the owners of the copyright in this specification hereby grant you a fully-paid up, non-exclusive, nontransferable, perpetual, worldwide license (without the right to sublicense), to use this specification to create and distribute software and special purpose specifications that are based upon this specification, and to use, copy, and distribute this specification as provided under the Copyright Act; provided that: (1) both the copyright notice identified above and this permission notice appear on any copies of this specification; (2) the use of the specifications is for informational purposes and will not be copied or posted on any network computer or broadcast in any media and will not be otherwise resold or transferred for commercial purposes; and (3) no modifications are made to this specification. This limited permission automatically terminates without notice if you breach any of these terms or conditions. Upon termination, you will destroy immediately any copies of the specifications in your possession or control.

PATENTS

The attention of adopters is directed to the possibility that compliance with or adoption of OMG specifications may require use of an invention covered by patent rights. OMG shall not be responsible for identifying patents for which a license may be required by any OMG specification, or for conducting legal inquiries into the legal validity or scope of those patents that are brought to its attention. OMG specifications are prospective and advisory only. Prospective users are responsible for protecting themselves against liability for infringement of patents.
GENERAL USE RESTRICTIONS

Any unauthorized use of this specification may violate copyright laws, trademark laws, and communications regulations and statutes. This document contains information which is protected by copyright. All Rights Reserved. No part of this work covered by copyright herein may be reproduced or used in any form or by any means--graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems--without permission of the copyright owner.

DISCLAIMER OF WARRANTY

WHILE THIS PUBLICATION IS BELIEVED TO BE ACCURATE, IT IS PROVIDED "AS IS" AND MAY CONTAIN ERRORS OR MISPRINTS. THE OBJECT MANAGEMENT GROUP AND THE COMPANIES LISTED ABOVE MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS PUBLICATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF TITLE OR OWNERSHIP, IMPLIED WARRANTY OF MERCHANTABILITY OR WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE OR USE. IN NO EVENT SHALL THE OBJECT MANAGEMENT GROUP OR ANY OF THE COMPANIES LISTED ABOVE BE LIABLE FOR ERRORS CONTAINED HEREIN OR FOR DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL, RELIANCE OR COVER DAMAGES, INCLUDING LOSS OF PROFITS, REVENUE, DATA OR USE, INCURRED BY ANY USER OR ANY THIRD PARTY IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS MATERIAL, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

The entire risk as to the quality and performance of software developed using this specification is borne by you. This disclaimer of warranty constitutes an essential part of the license granted to you to use this specification.

RESTRICTED RIGHTS LEGEND

Use, duplication or disclosure by the U.S. Government is subject to the restrictions set forth in subparagraph (c) (1)(ii) of The Rights in Technical Data and Computer Software Clause at DFARS 252.227-7013 or in subparagraph (c)(1) and (2) of the Commercial Computer Software - Restricted Rights clauses at 48 C.F.R. 52.227-19 or as specified in 48 C.F.R. 227-7202-2 of the DoD F.A.R. Supplement and its successors, or as specified in 48 C.F.R. 12.212 of the Federal Acquisition Regulations and its successors, as applicable. The specification copyright owners are as indicated above and may be contacted through the Object Management Group, 9C Medway Road, PMB 274, Milford, MA 01757, U.S.A.

TRADEMARKS

CORBA®, CORBA logos®, FIBO®, Financial Industry Business Ontology®, FINANCIAL INSTRUMENT GLOBAL IDENTIFIER®, IIOP®, IMM®, Model Driven Architecture®, MDA®, Object Management Group®, OMG®, OMG Logo®, SoaML®, SOAML®, SysML®, UAF®, Unified Modeling Language®, UML®, UML Cube Logo®, VIPL®, and XMI® are registered trademarks of the Object Management Group, Inc.

For a complete list of trademarks, see: https://www.omg.org/legal/tm_list.htm. All other products or company names mentioned are used for identification purposes only, and may be trademarks of their respective owners.

COMPLIANCE

The copyright holders listed above acknowledge that the Object Management Group (acting itself or through its designees) is and shall at all times be the sole entity that may authorize developers, suppliers and sellers of
computer software to use certification marks, trademarks or other special designations to indicate compliance with these materials.

Software developed under the terms of this license may claim compliance or conformance with this specification if and only if the software compliance is of a nature fully matching the applicable compliance points as stated in the specification. Software developed only partially matching the applicable compliance points may claim only that the software was based on this specification but may not claim compliance or conformance with this specification. In the event that testing suites are implemented or approved by Object Management Group, Inc., software developed using this specification may claim compliance or conformance with the specification only if the software satisfactorily completes the testing suites.
OMG’s Issue Reporting Procedure

All OMG specifications are subject to continuous review and improvement. As part of this process, we encourage readers to report any ambiguities, inconsistencies, or inaccuracies they may find by completing the Issue Reporting Form listed on the main web page https://www.omg.org, under Documents, Report a Bug/Issue.
Preface

OMG

Founded in 1989, the Object Management Group, Inc. (OMG) is an open membership, not-for-profit computer industry standards consortium that produces and maintains computer industry specifications for interoperable, portable, and reusable enterprise applications in distributed, heterogeneous environments. Membership includes Information Technology vendors, end users, government agencies, and academia.

OMG member companies write, adopt, and maintain its specifications following a mature, open process. OMG’s specifications implement the Model Driven Architecture® (MDA®), maximizing ROI through a full-lifecycle approach to enterprise integration that covers multiple operating systems, programming languages, middleware and networking infrastructures, and software development environments. OMG’s specifications include UML® (Unified Modeling Language®); CORBA® (Common Object Request Broker Architecture); CWM™ (Common Warehouse Metamodel™); and industry-specific standards for dozens of vertical markets.

More information on the OMG is available at http://www.omg.org/.

OMG Specifications

As noted, OMG specifications address middleware, modeling, and vertical domain frameworks. All OMG Specifications are available from the OMG website at:
https://www.omg.org/spec

All of OMG’s formal specifications may be downloaded without charge from our website. (Products implementing OMG specifications are available from individual suppliers.) Copies of specifications, available in PostScript and PDF format, may be obtained from the Specifications Catalog cited above or by contacting the Object Management Group, Inc. at:

OMG Headquarters
9C Medway Road, PMB 274
Milford, MA 01757
USA
Tel: +1-781-444-0404
Fax: +1-781-444-0320
Email: pubs@omg.org

Certain OMG specifications are also available as ISO standards. Please consult https://www.iso.org
Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Scope</td>
<td>1</td>
</tr>
<tr>
<td>2 Conformance</td>
<td>1</td>
</tr>
<tr>
<td>2.1 Content completeness</td>
<td>1</td>
</tr>
<tr>
<td>2.1.1 Basic level</td>
<td>1</td>
</tr>
<tr>
<td>2.1.2 Professional level</td>
<td>1</td>
</tr>
<tr>
<td>2.1.3 Expert level</td>
<td>2</td>
</tr>
<tr>
<td>2.2 SMOF Capability</td>
<td>2</td>
</tr>
<tr>
<td>2.2.1 Basic level</td>
<td>2</td>
</tr>
<tr>
<td>2.2.2 Professional level</td>
<td>2</td>
</tr>
<tr>
<td>2.2.3 Expert level</td>
<td>2</td>
</tr>
<tr>
<td>2.3 Shortcut capability</td>
<td>2</td>
</tr>
<tr>
<td>2.3.1 Basic level</td>
<td>2</td>
</tr>
<tr>
<td>2.3.2 Professional level</td>
<td>2</td>
</tr>
<tr>
<td>2.3.3 Expert level</td>
<td>3</td>
</tr>
<tr>
<td>2.4 MOF Extension (MEF) capability</td>
<td>3</td>
</tr>
<tr>
<td>2.4.1 Basic level</td>
<td>3</td>
</tr>
<tr>
<td>2.4.2 Professional level</td>
<td>3</td>
</tr>
<tr>
<td>2.4.3 Expert level</td>
<td>3</td>
</tr>
<tr>
<td>3 References</td>
<td>3</td>
</tr>
<tr>
<td>3.1 Normative References</td>
<td>3</td>
</tr>
<tr>
<td>3.2 Non-normative References</td>
<td>4</td>
</tr>
<tr>
<td>4 Terms and Definitions</td>
<td>4</td>
</tr>
<tr>
<td>5 Symbols and Abbreviations</td>
<td>5</td>
</tr>
<tr>
<td>6 Additional Information</td>
<td>5</td>
</tr>
<tr>
<td>6.1 Changes to Adopted OMG Specifications (optional)</td>
<td>5</td>
</tr>
<tr>
<td>6.2 Acknowledgements</td>
<td>5</td>
</tr>
<tr>
<td>6.3 IPR Mode</td>
<td>6</td>
</tr>
<tr>
<td>6.4 Document Style Conventions</td>
<td>6</td>
</tr>
<tr>
<td>7 Business Architecture Core Metamodel</td>
<td>7</td>
</tr>
<tr>
<td>7.1 Overview of the Metamodel (non-normative)</td>
<td>7</td>
</tr>
<tr>
<td>7.1.1 Capability Package</td>
<td>7</td>
</tr>
<tr>
<td>7.1.2 Customer Package</td>
<td>8</td>
</tr>
<tr>
<td>7.1.3 Organization Package</td>
<td>8</td>
</tr>
<tr>
<td>7.1.4 Process Package</td>
<td>9</td>
</tr>
<tr>
<td>7.1.5 Product Package</td>
<td>9</td>
</tr>
<tr>
<td>7.1.6 Strategy Package</td>
<td>9</td>
</tr>
<tr>
<td>7.1.7 The BACM Package</td>
<td>10</td>
</tr>
<tr>
<td>7.2 Interpreting and Implementing the Metamodel (normative)</td>
<td>10</td>
</tr>
<tr>
<td>7.2.1 Interpreting the UML metamodel and generated XMI</td>
<td>10</td>
</tr>
<tr>
<td>7.2.1.1 Un-stereotyped class translation</td>
<td>11</td>
</tr>
<tr>
<td>7.2.1.2 Un-stereotyped association translation</td>
<td>11</td>
</tr>
<tr>
<td>7.2.1.3 &lt;&lt;class&gt;&gt; stereotyped binary association translation</td>
<td>11</td>
</tr>
<tr>
<td>7.2.1.4 &lt;&lt;association&gt;&gt; stereotyped origin class translation</td>
<td>11</td>
</tr>
<tr>
<td>7.2.1.5 &lt;&lt;shortcut&gt;&gt; stereotyped class and association translation</td>
<td>12</td>
</tr>
<tr>
<td>7.2.1.6 &lt;&lt;individual&gt;&gt; stereotype translation</td>
<td>12</td>
</tr>
<tr>
<td>7.2.2 Unique name of associations</td>
<td>12</td>
</tr>
<tr>
<td>7.2.3 Model elements (instances) represent sets and individuals</td>
<td>12</td>
</tr>
<tr>
<td>7.2.4 Meta-model association instances as association classes</td>
<td>12</td>
</tr>
<tr>
<td>7.2.5 Distinguished association names</td>
<td>13</td>
</tr>
<tr>
<td>7.2.5.1 aggregates</td>
<td>13</td>
</tr>
<tr>
<td>7.2.5.2 generalizes</td>
<td>13</td>
</tr>
<tr>
<td>7.2.5.3 owns</td>
<td>13</td>
</tr>
</tbody>
</table>
7.3.4 related
7.3.5 N-ary Associations reified as Classes and Binary Associations
7.3.6 Application of business architecture frameworks with MEF
7.3 BACM Metamodel (Normative)

7.3.1 Package: BACM Model

7.3.1.1 Diagram: BACM Element
7.3.1.2 Diagram: BACM Entities
7.3.1.3 Diagram: BACM Model
7.3.1.4 Diagram: BACM Model
7.3.1.5 Diagram: BusinessElement
7.3.1.6 Diagram: DisjointElements
7.3.1.7 Diagram: DisjointEntities

7.3.1.8 Class Name: AbstractAction Class Type: Class Stereotype:
7.3.1.9 Class Name: AbstractResult Class Type: Class Stereotype:
7.3.1.10 Class Name: AbstractThing Class Type: Class Stereotype:
7.3.1.11 Class Name: Annotation Class Type: Class Stereotype:
7.3.1.12 Class Name: BACM Model Class Type: Class Stereotype:
7.3.1.13 Class Name: BACMBinDirRelation Class Type: Class Stereotype:
7.3.1.14 Class Name: BACMElement Class Type: Class Stereotype:
7.3.1.15 Class Name: BACMEntity Class Type: Class Stereotype:
7.3.1.16 Class Name: BACMPInent Entity Class Type: Class Stereotype:
7.3.1.17 Class Name: BACMRelation Class Type: Class Stereotype:
7.3.1.18 Class Name: BACMShortcut Class Type: Class Stereotype:
7.3.1.19 Class Name: BusinessElement Class Type: Class Stereotype:
7.3.1.20 Class Name: ExternalData Class Type: Class Stereotype:
7.3.1.21 Class Name: ExternalRelationship Class Type: Class Stereotype:
7.3.1.22 Class Name: IR Class Type: Class Stereotype:

7.3.2 Package: Capability

7.3.2.1 Diagram: Capability
7.3.2.2 Diagram: CapabilityValue
7.3.2.3 Diagram: Roles
7.3.2.4 Class Name: AbstractBusinessObject Class Type: Class Stereotype:
7.3.2.5 Class Name: AbstractCapability Class Type: Class Stereotype:
7.3.2.6 Class Name: BusinessObject Class Type: Class Stereotype:
7.3.2.7 Class Name: Capability Class Type: Class Stereotype:
7.3.2.8 Class Name: CapabilityBehavior Class Type: Class Stereotype:
7.3.2.9 Class Name: CapabilityImplementation Class Type: Class Stereotype:
7.3.2.10 Class Name: InformationItem Class Type: Class Stereotype:
7.3.2.11 Class Name: ObjectRelation Class Type: Class Stereotype: «association»
7.3.2.12 Class Name: Outcome Class Type: Class Stereotype:
7.3.2.13 Class Name: OutcomeRelation Class Type: Class Stereotype: «association»
7.3.2.14 Class Name: PerformerRole Class Type: Class Stereotype: «association»
7.3.2.15 Class Name: ResourceRole Class Type: Class Stereotype: «association»
7.3.2.16 Class Name: Role Class Type: Class Stereotype: «association»

7.3.3 Package: Customer

7.3.3.1 Diagram: Customer
7.3.3.2 Diagram: ValueFit
7.3.3.3 Diagram: ValueStream
7.3.3.4 Class Name: Customer Class Type: Class Stereotype:
7.3.3.5 Class Name: CustomerJourney Class Type: Class Stereotype:
7.3.3.6 Class Name: CustomerJourneyStage Class Type: Class Stereotype:
7.3.3.7 Class Name: CustomerSegment Class Type: Class Stereotype:
7.3.3.8 Class Name: JSTP Class Type: Class Stereotype:
7.3.3.9 Class Name: Touchpoint | Class Type: Class Stereotype: ................................................. 60
7.3.3.10 Class Name: ValueCharacteristic | Class Type: Class Stereotype: »association« 61
7.3.3.11 Class Name: ValueItem | Class Type: Class Stereotype: ............................................................... 62
7.3.3.12 Class Name: ValueProposition | Class Type: Class Stereotype: ............................................ 63
7.3.3.13 Class Name: ValueStream | Class Type: Class Stereotype: .................................................... 64
7.3.3.14 Class Name: ValueStreamStage | Class Type: Class Stereotype: ............................................... 65
7.3.3.15 Class Type: Class Stereotype: ................................................................. 66
7.3.4 Package: Organization ......................................................................................................................... 66
7.3.4.1 Diagram: Organization .................................................................................................................. 67
7.3.4.2 Diagram: Participant ...................................................................................................................... 69
7.3.4.3 Class Name: Jurisdiction | Class Type: Class Stereotype: ............................................................ 69
7.3.4.4 Class Name: LegalEntity | Class Type: Class Stereotype: ............................................................. 70
7.3.4.5 Class Name: OrgUnit | Class Type: Class Stereotype: ................................................................. 71
7.3.4.6 Class Name: Performer | Class Type: Class Stereotype: .............................................................. 71
7.3.4.7 Class Name: Resource | Class Type: Class Stereotype: ............................................................... 72
7.3.4.8 Class Name: Responsible | Class Type: Class Stereotype: »association« 73
7.3.4.9 Class Name: System | Class Type: Class Stereotype: ............................................................... 73
7.3.4.10 Class Name: theBusiness | Class Type: Class Stereotype: »individual« ................................. 74
7.3.5 Package: Process ................................................................................................................................. 75
7.3.5.1 Diagram: Process .......................................................................................................................... 75
7.3.5.2 Class Name: AbstractProcess | Class Type: Class Stereotype: ..................................................... 76
7.3.5.3 Class Name: Activity | Class Type: Class Stereotype: ................................................................. 78
7.3.5.4 Class Name: Process | Class Type: Class Stereotype: ................................................................. 78
7.3.5.5 Class Name: VSVSS | Class Type: Class Stereotype: ............................................................... 79
7.3.6 Package: Product ................................................................................................................................. 79
7.3.6.1 Diagram: MerchandiseOffering ................................................................................................. 80
7.3.6.2 Diagram: OutsourcedServiceOffering ..................................................................................... 82
7.3.6.3 Diagram: ProcurementOffering ............................................................................................... 83
7.3.6.4 Diagram: Product ......................................................................................................................... 85
7.3.6.5 Diagram: ServiceOffering ............................................................................................................ 87
7.3.6.6 Class Name: APCICB | Class Type: Class Stereotype: ................................................................. 89
7.3.6.7 Class Name: ContractRelation | Class Type: Class Stereotype: »association« ........................... 90
7.3.6.8 Class Name: MerchandiseOffering | Class Type: Class Stereotype: ........................................... 90
7.3.6.9 Class Name: MerchandiseOutcome | Class Type: Class Stereotype: .......................................... 91
7.3.6.10 Class Name: Offering | Class Type: Class Stereotype: ............................................................. 91
7.3.6.11 Class Name: OutsourcedServiceOffering | Class Type: Class Stereotype: ............................... 92
7.3.6.12 Class Name: OutsourcedServiceOutcome | Class Type: Class Stereotype: .............................. 93
7.3.6.13 Class Name: ProcurementOffering | Class Type: Class Stereotype: ........................................... 93
7.3.6.14 Class Name: ProcurementOutcome | Class Type: Class Stereotype: ........................................... 94
7.3.6.15 Class Name: ProductOffering | Class Type: Class Stereotype: .................................................... 94
7.3.6.16 Class Name: ServiceOffering | Class Type: Class Stereotype: .................................................... 95
7.3.6.17 Class Name: ServiceOutcome | Class Type: Class Stereotype: .................................................... 96
7.3.7 Package: Strategy ............................................................................................................................... 96
7.3.7.1 Diagram: OperatingModel ........................................................................................................ 97
7.3.7.2 Diagram: Strategy ...................................................................................................................... 100
7.3.7.3 Diagram: StrategyNeeds ............................................................................................................. 101
7.3.7.4 Diagram: ValueModel .................................................................................................................. 102
7.3.7.5 Class Name: AbstractOperatingModel | Class Type: Class Stereotype: ................................. 102
7.3.7.6 Class Name: AbstractValueModel | Class Type: Class Stereotype: ........................................... 104
7.3.7.7 Class Name: Change | Class Type: Class Stereotype: ................................................................. 105
7.3.7.8 Class Name: Ends | Class Type: Class Stereotype: ................................................................. 106
7.3.7.9 Class Name: Initiative | Class Type: Class Stereotype: ............................................................. 107
7.3.7.10 Class Name: Means | Class Type: Class Stereotype: ............................................................... 108
7.3.7.11 Class Name: StrategyChoices | Class Type: Class Stereotype: ................................................... 109
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview of the Metamodel (non-normative)</td>
<td>7</td>
</tr>
<tr>
<td>7.3.7.12 Class Name: StrategyModel Class Type: Class Stereotype:</td>
<td>110</td>
</tr>
<tr>
<td>8.1.1 Shortcuts and Touchpoints (normative)</td>
<td>110</td>
</tr>
<tr>
<td>9.1.1.5 Basic level</td>
<td>111</td>
</tr>
<tr>
<td>9.1.2 Professional level</td>
<td>111</td>
</tr>
<tr>
<td>9.1.3 Export level</td>
<td>111</td>
</tr>
<tr>
<td>8.2.1 Scope</td>
<td>1</td>
</tr>
<tr>
<td>2.1 Conformance</td>
<td>1</td>
</tr>
<tr>
<td>2.1.1 Content completeness</td>
<td>1</td>
</tr>
<tr>
<td>2.1.2 Professional level</td>
<td>2</td>
</tr>
<tr>
<td>2.1.3 Export level</td>
<td>2</td>
</tr>
<tr>
<td>2.2 MOF Capabilities</td>
<td>2</td>
</tr>
<tr>
<td>2.2.1 Basic level</td>
<td>2</td>
</tr>
<tr>
<td>2.2.2 Professional level</td>
<td>2</td>
</tr>
<tr>
<td>2.2.3 Export level</td>
<td>2</td>
</tr>
<tr>
<td>2.3 Shortcut capabilities</td>
<td>2</td>
</tr>
<tr>
<td>2.3.1 Basic level</td>
<td>2</td>
</tr>
<tr>
<td>2.3.2 Professional level</td>
<td>2</td>
</tr>
<tr>
<td>2.3.3 Export level</td>
<td>2</td>
</tr>
<tr>
<td>2.4 MOF Extension (MEF) capability</td>
<td>3</td>
</tr>
<tr>
<td>2.4.1 Basic level</td>
<td>3</td>
</tr>
<tr>
<td>2.4.2 Professional level</td>
<td>3</td>
</tr>
<tr>
<td>2.4.3 Export level</td>
<td>3</td>
</tr>
<tr>
<td>3 References</td>
<td>3</td>
</tr>
<tr>
<td>3.1 Non-normative References</td>
<td>3</td>
</tr>
<tr>
<td>3.2 Normative References</td>
<td>3</td>
</tr>
<tr>
<td>4 Terms and Definitions</td>
<td>4</td>
</tr>
<tr>
<td>5 Symbols and Abbreviations</td>
<td>4</td>
</tr>
<tr>
<td>6 Additional Information</td>
<td>5</td>
</tr>
<tr>
<td>6.1 Changes to Adopted OMG Specifications (optional)</td>
<td>5</td>
</tr>
<tr>
<td>6.2 Acknowledgements</td>
<td>5</td>
</tr>
<tr>
<td>6.3 IPR Mode</td>
<td>6</td>
</tr>
<tr>
<td>6.4 Document Style Conventions</td>
<td>6</td>
</tr>
<tr>
<td>7 Business Architecture Core Metamodel</td>
<td>7</td>
</tr>
<tr>
<td>7.1 Overview of the Metamodel (non-normative)</td>
<td>7</td>
</tr>
<tr>
<td>7.1.1 Capability Package</td>
<td>7</td>
</tr>
<tr>
<td>7.1.2 Customer Package</td>
<td>8</td>
</tr>
<tr>
<td>7.1.3 Organization Package</td>
<td>8</td>
</tr>
<tr>
<td>7.1.4 Process Package</td>
<td>11</td>
</tr>
<tr>
<td>7.1.5 Product Package</td>
<td>11</td>
</tr>
<tr>
<td>7.1.6 Strategy Package</td>
<td>11</td>
</tr>
<tr>
<td>7.1.7 BACM Package</td>
<td>11</td>
</tr>
<tr>
<td>7.1.7.1 The BACM_Model package</td>
<td>10</td>
</tr>
<tr>
<td>7.2 Interpreting and Implementing the Metamodel (normative)</td>
<td>10</td>
</tr>
<tr>
<td>7.2.1 Interpreting the UML metamodel and generated XMI</td>
<td>10</td>
</tr>
<tr>
<td>7.2.2 Meta-class instances as classes</td>
<td>14</td>
</tr>
<tr>
<td>7.2.3 Meta-model association instance as association classes</td>
<td>12</td>
</tr>
<tr>
<td>7.2.4 Distinguished association names</td>
<td>12</td>
</tr>
<tr>
<td>7.2.4.1 aggregation</td>
<td>12</td>
</tr>
<tr>
<td>7.2.4.2 generalization</td>
<td>12</td>
</tr>
<tr>
<td>7.2.4.3 owns</td>
<td>12</td>
</tr>
<tr>
<td>7.2.4.4 related</td>
<td>12</td>
</tr>
<tr>
<td>7.2.5 N-ary Associations reified as Classes and Binary Associations</td>
<td>12</td>
</tr>
<tr>
<td>7.2.6 Application of business architecture frameworks with MEF</td>
<td>13</td>
</tr>
<tr>
<td>7.2.7 BACM Metamodel (Normative)</td>
<td>13</td>
</tr>
<tr>
<td>2.3.1</td>
<td>Package: BACM_Model</td>
</tr>
<tr>
<td>2.3.1.1</td>
<td>Diagram: BACM_Element</td>
</tr>
<tr>
<td>2.3.1.2</td>
<td>Diagram: BACM_Entity</td>
</tr>
<tr>
<td>2.3.1.3</td>
<td>Diagram: BACM_Model</td>
</tr>
<tr>
<td>2.3.1.4</td>
<td>Diagram: BACM_Entity</td>
</tr>
<tr>
<td>2.3.1.5</td>
<td>Diagram: BusinessElement</td>
</tr>
<tr>
<td>2.3.1.6</td>
<td>Diagram: DisjointElementsByTag</td>
</tr>
<tr>
<td>2.3.1.7</td>
<td>Diagram: DisjointEntities</td>
</tr>
<tr>
<td>2.3.1.8</td>
<td>Class Name: AbstractAction</td>
</tr>
<tr>
<td>2.3.1.9</td>
<td>Class Name: AbstractResult</td>
</tr>
<tr>
<td>2.3.1.10</td>
<td>Class Name: ABSTRACTTHING</td>
</tr>
<tr>
<td>2.3.1.11</td>
<td>Class Name: Annotation</td>
</tr>
<tr>
<td>2.3.1.12</td>
<td>Class Name: BACM_Model</td>
</tr>
<tr>
<td>2.3.1.13</td>
<td>Class Name: BACMBinDRelation</td>
</tr>
<tr>
<td>2.3.1.14</td>
<td>Class Name: BACMElement</td>
</tr>
<tr>
<td>2.3.1.15</td>
<td>Class Name: BACMEMEntity</td>
</tr>
<tr>
<td>2.3.1.16</td>
<td>Class Name: BACMParentEntity</td>
</tr>
<tr>
<td>2.3.1.17</td>
<td>Class Name: BACMRelation</td>
</tr>
<tr>
<td>2.3.1.18</td>
<td>Class Name: BAMSmChange</td>
</tr>
<tr>
<td>2.3.1.19</td>
<td>Class Name: BusinessElement</td>
</tr>
<tr>
<td>2.3.1.20</td>
<td>Class Name: ExternalData</td>
</tr>
<tr>
<td>2.3.1.21</td>
<td>Class Name: ExternalRelationship</td>
</tr>
<tr>
<td>2.3.1.22</td>
<td>Class Name: IRI</td>
</tr>
<tr>
<td>2.3.2</td>
<td>Package: Capability</td>
</tr>
<tr>
<td>2.3.2.1</td>
<td>Diagram: Capability</td>
</tr>
<tr>
<td>2.3.2.2</td>
<td>Diagram: CapabilityValue</td>
</tr>
<tr>
<td>2.3.2.3</td>
<td>Diagram: Role</td>
</tr>
<tr>
<td>2.3.2.4</td>
<td>Class Name: AbstractBusinessObject</td>
</tr>
<tr>
<td>2.3.2.5</td>
<td>Class Name: AbstractCapability</td>
</tr>
<tr>
<td>2.3.2.6</td>
<td>Class Name: BusinessObject</td>
</tr>
<tr>
<td>2.3.2.7</td>
<td>Class Name: Capability</td>
</tr>
<tr>
<td>2.3.2.8</td>
<td>Class Name: CapabilityBehavior</td>
</tr>
<tr>
<td>2.3.2.9</td>
<td>Class Name: CapabilityImplementation</td>
</tr>
<tr>
<td>2.3.2.10</td>
<td>Class Name: InformationItem</td>
</tr>
<tr>
<td>2.3.2.11</td>
<td>Class Name: ObjectRelaiton</td>
</tr>
<tr>
<td>2.3.2.12</td>
<td>Class Name: Outcome</td>
</tr>
<tr>
<td>2.3.2.13</td>
<td>Class Name: OutcomeRelation</td>
</tr>
<tr>
<td>2.3.2.14</td>
<td>Class Name: PerformerRole</td>
</tr>
<tr>
<td>2.3.2.15</td>
<td>Class Name: ResourceRole</td>
</tr>
<tr>
<td>2.3.2.16</td>
<td>Class Name: Role</td>
</tr>
<tr>
<td>2.3.3</td>
<td>Package: Customer</td>
</tr>
<tr>
<td>2.3.3.1</td>
<td>Diagram: Customer</td>
</tr>
<tr>
<td>2.3.3.2</td>
<td>Diagram: ValueFit</td>
</tr>
<tr>
<td>2.3.3.3</td>
<td>Diagram: ValueStream</td>
</tr>
<tr>
<td>2.3.3.4</td>
<td>Class Name: Customer</td>
</tr>
<tr>
<td>2.3.3.5</td>
<td>Class Name: CustomerJourney</td>
</tr>
<tr>
<td>2.3.3.6</td>
<td>Class Name: CustomerJourneyStage</td>
</tr>
<tr>
<td>2.3.3.7</td>
<td>Class Name: CustomerSegment</td>
</tr>
<tr>
<td>2.3.3.8</td>
<td>Class Name: ISTD</td>
</tr>
<tr>
<td>2.3.3.9</td>
<td>Class Name: Touchpoint</td>
</tr>
<tr>
<td>2.3.3.10</td>
<td>Class Name: ValueCharacteristic</td>
</tr>
<tr>
<td>2.3.3.11</td>
<td>Class Name: ValueItem</td>
</tr>
<tr>
<td>2.3.3.12</td>
<td>Class Name: ValueProposition</td>
</tr>
<tr>
<td>2.3.3.13</td>
<td>Class Name: ValueStream</td>
</tr>
<tr>
<td>2.3.3.14</td>
<td>Class Name: ValueStreamStage</td>
</tr>
<tr>
<td>2.3.4</td>
<td>Package: Organization</td>
</tr>
</tbody>
</table>
1 Scope

The Business Architecture Core Metamodel defines concepts suitable for modeling business concepts found to be useful in business direction and strategy and not found in business operating models. These concepts include value and its delivery to stakeholders of a business, capability, abstract organization, process, product and strategy. The concepts are represented at a high level typical of executive management and staffs who are responsible for overall business management and direction. Business architecture models derived from this metamodel are not intended to represent all aspects of a business; they are intended to be used in conjunction with other models, with the ensemble of models being a sufficient basis for strategic and business analysis and planning. While the business architecture models are high level, they must be grounded in the reality and details of the business. For this reason, an ability to align or link elements or groups of elements of a business architecture model with elements and groups of elements of other models or even portions of prose documents or business data is a strong requirement. The OMG has produced or is working on specifications for other business models, but the business architect will need to include models not based on any OMG specification. This specification defines a general mechanism for linking a BACM-derived model to other models and data sources. These mechanisms respond to the RFP request for a “touchpoint” mechanism.

Business architects typically make use of conceptual frameworks to create models of a business or type of business. There are many such frameworks and they change with frequency, consequently it would be inappropriate to encode particular frameworks in the metamodel. A general mechanism, MEF [MEF] has been defined for MOF that allows the dynamic application of stereotypes to any MOF-based model. The specification requires MEF and recommends that business architects develop profiles of stereotypes for such frameworks. The concepts of the framework, represented as stereotypes, may then be applied to BACM model elements to characterize them and provide supplementary information according to the framework.

2 Conformance

2.1 Overview

Implementers of this specification must be able to create, edit and delete instances of each of the meta-classes and meta-associations in this specification. Implementers may perform these operations in any suitable manner, provided that the effect is as if each meta-class and meta-association were present in the implementation and associated with each instance. XMI exports of an instance model must include in each instance of a class or association, a reference to the definition of the meta-class or meta-association used to create the instance definition in the normative XMI of the relevant specification version.

The specification diagrams and prose define and use a “shortcut” mechanism for certain associations. Shortcuts are defined in the metamodel and these definitions are used to create constraints associated with instances of shortcuts. The representation of shortcuts is mandatory, but implementing the evaluation of the constraints is an optional point of compliance. Shortcuts are important tools for the evolution of BACM models, so full implementations are strongly encouraged.

The specification also defines an approach to implementing “touchpoints” (references to information contained in models outside the scope of the BACM). The approach employs a “resourceIdentifier”, typically an IRI, to identify a resource (which may be an external model, document or data set) and an alignment specification whose language is not controlled by this specification, except that if the language attribute is “Natural”, then the specification will be a prose description of the alignment mapping in a natural language.

Implementations must support all packages defined in this specification.

Implementations must implement the Metamodel Extension Facility (MEF) [MEF], which implies the implementation of [SMOF] and [MOF] or an equivalent facility. This conformance requirement supports the requirement that model elements be instances from one or more metalevels and the requirement that stereotypes be used to define frameworks for the additional interpretation of BACM models.
A conforming implementation may directly implement the MOF metamodel for BACM or implement a semantically equivalent metamodel using e.g. [RDF] or any other type of implementation that is semantically equivalent to the MOF metamodel for BACM and can import and export that model in [XMI]. Other import and export concrete syntaxes are allowed, but are not controlled by this specification and they may not allow exchange of models between implementations.

Conformance criteria for implementations of this specification are specified in four independent domains:
- completeness of implementation of the classes and associations in the metamodel
- implementation of SMOF
- implementation of shortcuts
- implementation of MEF

2.1 Content completeness

The rationale for these compliance points is to allow implementations to exhibit different levels of detail and complexity with respect to business architecture modeling and different implementation and maintenance costs. Note that the instances of classes and associations in the metamodel are also classes and associations. The BACM specification does not define “individuals”, but it may have external relationships to model elements or data that represent individuals or sets of individuals and their individual relationships.

2.1.1 Basic level

Completely implement the BACM Model package classes and associations including ExternalReferences with SMM integration optional.

- Implement the Capability package, replacing the variable arity associations OutcomeRelation and ObjectRelation with binary class associations.
- Implement the Customer package without the ValueCharacteristic 4-ary association. ProductOffering must be implemented, but the other classes and associations in the Product package need not be implemented.
- Implement the Organization package, omitting the ofProcess association and the AbstractProcess class. The Responsible association should be implemented as a binary, directed association between OrgUnits.
- Do not implement the Process package
- Do not implement the Strategy package

2.1.2 Professional level

Completely implement all packages except Strategy.

- Replace the variable-arity associations OutcomeRelation and ObjectRelation with binary class associations
- Replace the variable-arity association ContractRelation with a binary class association.

2.1.3 Expert level

Implement all packages and variable arity associations.

2.2 SMOF Capability

SMOF is a modification to MOF that allows instances to be members of multiple classes and to change the membership linkage between instances and their classes dynamically. SMOF also removes the assumption of disjointness of class extents and adds constraint types to declare when class extents are disjoint. This capability becomes important when aligning business architecture models with models of actual businesses as an instance such as a tool may be classified as a Resource, a Performer or a BusinessObject that is an asset.

2.2.1 Basic level

SMOF is not implemented. The extensions of classes are disjoint and may not be dynamically changed (however, the model may be edited to reflect changes). The disjointness constraints expressed in the MOF metamodel are ignored.
2.2.2 **Professional level**

Enough of SMOF is implemented to allow instances to be members of the extent of multiple classes and disjointness constraints as specified by [SMOF] to be expressed in the metamodel and by modelers in their models.

2.2.3 **Expert level**

SMOF is completely implemented.

2.3 **Shortcut capability**

Shortcuts allow the modeler to express an association that implies the existence of a chain of classes and associations that would justify the shortcut association. In plain language, the English term “uncle” means “the brother of a parent” in a concept graph, “uncle” would label a direct arc between a person and that person’s uncle. However, the concept graph might also contain a direct arc labeled “parent” between the person and the person’s parent and a direct arc labeled “brother” between the person’s parent and the person’s uncle. If a shortcut association like “uncle” was created in a model, one could infer that a parent also exists (but may not be represented in the model) and that the “parent” and “brother” associations may also exist.

Shortcuts are a way to abstract detail, but it is important to be able to determine if the abstraction and the details are consistent. For this reason, shortcut associations have constraint specifications that may be applied to the model to determine this consistency. The short constraint describes the structure of a chain of instance classes and associations that should exist in the model to justify the assertion of the shortcut association. It is not an invariant and it is not required to be true in a valid model.

2.3.1 **Basic level**

Shortcuts are not implemented. They are ignored on import and will not be exported. The constraint language associated with a shortcut is also ignored on import and may not be created in the implementation.

2.3.2 **Professional level**

Shortcuts and their constraint specifications may be imported and exported. The implementation does not need to provide the capability to evaluate constraint expressions. The implementation may allow the creation of constraint specifications and must support the [OCL] language for specification of shortcut constraints. An implementation may also support constraints specified in the [SPARQL] language where the implementation is based on [RDF] or [RDF*].

2.3.3 **Expert level**

Shortcuts and their constraint specifications are fully implemented, including all of the requirements for the Professional level. An implementation capable of evaluating the constraints must be provided.

2.4 **MOF Extension (MEF) capability**

This requirement is derived from the need to apply methodological frameworks to a BACM model. As an example of a framework, consider Value Proposition Design [Osterwalder, et al., Value Proposition Design, Wiley 2014]. This methodology defines a “Customer Profile” to characterize customers according to three aspects: “Customer Job”, “Customer Pain” and “Customer Gain”. It also defines a “Value Map” to characterize the offerings from a business with aspects “Goods and Services”, “Pain Relievers” and “Gain Creators”. The intent of this analytic framework is to allow the evaluation of the fit of a product offering to the characterized customer type.

There are a variety of such frameworks available, and they change over time; consequently, it is not appropriate to constrain the methodological framework by defining a metamodel for it. Fortunately, such frameworks have a lot in common with UML profiles. A profile, when applied to instances of UML classes and associations, can associate properties and default values with these instances. Considering the BACM Customer Package, a “Customer Profile” could be represented by an instance of BACM::Customer::Customer and the associated instances of BACM::Customer::CustomerSegment. Conceptually, a “Customer Profile” could be created in a BACM model by applying a “Customer Profile” UML stereotype to the BACM::Customer::Customer and a “Customer Job” stereotype to one or more of the BACM::Customer::CustomerSegment instances. The modeler determined properties of these...
stereotypes are able to hold analysis values derived from following the analysis processes outlined in the methodology reference previously cited.

### 2.4.1 Basic level
The implementation is not required to provide an extension capability as described above. The modeler may elect to use the annotation capability of the BACM metamodel to capture framework defined values.

### 2.4.2 Professional level
The implementation may implement the UML 2.5.1 profile capability or its equivalent. This capability would effectively import the BACM metamodel, excluding elements not derived from BACM::BACM_Model::BusinessElement and make them available to Stereotypes. If the implementation provides the BACM tailored implementation of SMM, it may choose to effectively import any or all elements of the SMM metamodel and make them available to Stereotypes.

### 2.4.3 Expert level
The OMG Metamodel Extension Framework [MEF] relies on the dynamic metamodeling provision of SMOF to allow Profiles and Stereotypes to be defined at the metamodel level, simplifying the process for extending metamodels by adding properties and constraints to an existing metamodel without changing it. Implementations at this level will include an implementation of the [MEF] specification.

### 3 References

#### 3.1 Normative References
The following normative documents contain provisions which, through reference in this text, constitute provisions of this specification. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply.

- [OCL] Object Constraint Language, [https://www.omg.org/spec/OCL/2.4](https://www.omg.org/spec/OCL/2.4)

#### 3.2 Non-normative References
- [DMN] Decision Model Notation [https://www.omg.org/spec/DMN](https://www.omg.org/spec/DMN)
4 Terms and Definitions

The terms used to label metaelements in this specification and their definitions are contained in Annex A:

[The term "M1" has been previously used in OMG specifications to designate a model that is properly derived from a metamodel (typically designated by the term "M2"). In this specification, the term "M1" designates a model that is properly derived from the metamodel defined in this specification. Elements of an M1 model are instances of the metaclasses and meta-associations of the metamodel, and these instances will have metaclasses and meta-associations as their metaclasses.]

5 Symbols and Abbreviations

The specification employs UML symbols and diagrams to present the metamodel.

Various abbreviations, acronyms and symbols are used in this document as a terse form of reference to references contained in section 3 of this document. They are not repeated here.

6 Additional Information

6.1 Changes to Adopted OMG Specifications [optional]

No changes are proposed to any adopted OMG specifications by this specification.

6.2 Acknowledgements
The following companies submitted this specification:
  • Business Architecture Guild
  • Mega Corporation
  • Trisotech
  • Model Driven Solutions
  • Tactical Strategy Group
  • Capsifi USA
The following companies supported this specification:
  • VDMbee
  • Thematix Partners
  • Airbus
  • Boeing
The following individuals contributed to this specification:
  • Jim Rhyne
  • Antoine Lonjon
  • Henk de Man
  • Fred Cummins
  • Lloyd Dugan
  • Hermann Schlamann
  • Michel Sauvage
  • Chalon Mullins

6.3 IPR Mode
The IPR mode of this specification is “non-assert”.

6.4 Document Style Conventions
The following stylistic conventions apply to text about the Clause 7 (Metamodel):
Italicized names in the descriptive text refer to the corresponding named elements in the diagrams and in the element syntax definitions. In general, such terms can be taken as referring to instances of the named metaclasses and meta-associations. Where necessary, an ambiguity will be resolved by using the metamodel element label followed by the “metaclass” or “meta-association” term.
7 Business Architecture Core Metamodel

7.1 Overview of the Metamodel (non-normative)

The metamodel is specified as a collection of packages containing metaclasses and meta-association that refer to metaclasses and meta-associations in other packages. The metamodel is intended to define an abstract syntax for BACM models that are instances of the overall metamodel. The packaging is a convenience and should not be construed by implementers as specifying a modeling palette structure or any other characteristic of model presentation except for the names of the metaclasses and meta-associations.

Many meta-associations in the metamodel are given a <<class>> stereotype. This stereotype should be interpreted as meaning that these associations can have properties, be specialized and participate in associations. Some associations are n-ary; these associations are sometimes represented as classes with an <<association>> stereotype and sometimes as a UML n-ary association. In either case, these associations should be able to have properties, be specialized and participate in associations.

The <<shortcut>> stereotype is sometimes applied to an association in the metamodel. This stereotype indicates that the association must be consistent with other details that may or may not be elaborated in the metamodel. For example, a business architect may wish to assert that a value stream is intended to satisfy the values of a set of customers (represented as a customer) without immediately specifying the details of value propositions, customer journeys and customer segments. This assertion would be modeled as an instance of a <<shortcut>> stereotyped meta-association. The assertions associated with shortcuts are represented in the MOF-compliant XMI as OCL constraints.

The model instances of the meta-classes and meta-associations represent types or sets of entities that would be found in a real or imagined business. Some of these entities will be tangible (occupying time and space) while others will be intangible (conventions of thought). The model instances have potentially intensional and extensional semantics and are not required to have disjoint extensions; for example, a BusinessObject instance, AssemblyRobot, may in a different context be an instance of Performer or an instance of Resource. This may create a problem for tool implementations that do not allow an instance to have multiple metaclasses. The OMG specification MOF Support for Semantic Structures – SMOF [SMOF] provides such a facility for MOF metamodels and the implementation of the MOF metamodel of this specification will also require implementation of SMOF.

7.1.1 Capability Package

The Capability package specifies abstract syntax for Capability, Outcome, BusinessObject and InformationItem and their related metaclasses. It also specifies associations that link these metaclasses together. Capability is an abstraction of a unit of work that does not specify how the work is done. In effect, a Capability is specified by its Outcomes that are states of BusinessObjects or InformationItems. A Capability produces Outcomes and needs Outcomes. An Outcome that is produced by a Capability and seen by an entity outside theBusiness corresponds to an external event or state; typically such Outcomes would be experienced by stakeholders such as customers and regulators. An Outcome that is needed by a Capability and not produced by any Capability is effectively a triggering event that occurs outside the business, such as receipt of an order.

Outcomes effectively externalize the modeled state of a BusinessObject or InformationItem. This allows the modeler to define a BusinessObject or InformationItem without having to define its state variables, properties or characteristic associations; these can be specified separately as Outcomes. The resulting abstract model is more complex but allows a BusinessObject or InformationItem to be represented in the model in multiple states in the structural model of the business. The alternative, internalized states, requires the separate specification of state machines that control the state behavior and tie it to Capabilities.

Capabilities are also associated with Role instances that are abstract specifications of a type of work that may be accomplished by the Capability while producing an Outcome. Roles are useful for defining Capabilities that can be used to manage behavioral variation in a business. There are two types of Roles that can be instantiated in a model:
PerformerRoles and ResourceRoles. PerformerRoles specify a kind of skill. ResourceRoles specify actions that may be performed with or on a Resource.

Capabilities must be tied to an operating model of a business to be useful for analysis of the business. The BACM Capability package provides two metaclasses, CapabilityBehavior and CapabilityImplementation as intermediaries that can be tied to a business operating model. CapabilityBehavior represents specific behaviors of a Capability (that might be described in a BPMN or VDM model). CapabilityImplementation instances represent a specification of Performers and Resources that can be assigned to Roles of a CapabilityBehavior. These instances can represent specifications of project resourcing for planning purposes, or they can represent actual elements of an organization for purposes of analysis.

InformationItems can be used to control decisions and other behaviors of Capabilities and CapabilityBehaviors. InformationItems can also represent metadata (is_about) about a BusinessObject. InformationItems are typically intangible but may also represent a tangible such as a report or a dataset. BusinessObjects are typically tangible but may also represent collections of tangible and intangible things.

Capabilities are also associated with the production of value by ValueStreams. The abstract syntax in the CapabilityValue diagram shows that Capabilities support ValueStreamStages and Outcomes are valued by ValueItems. In effect, the Capabilities supporting ValueStreamStages represent abilities that the business must have to produce values that are experienced by Customers (and other stakeholders). The ValueStreams, ValueStreamStages, ValueItems and Value Propositions create a value perspective on the underlying Capabilities and Outcomes. The Customer package provides additional details.

### 7.1.2 Customer Package

The Customer package defines abstract syntax for Customer, CustomerSegment, CustomerJourney, JourneyStage and Touchpoint. The Customer identifies a customer (or any value-receiving stakeholder) but does not describe the customer/stakeholder. Descriptions are held in CustomerSegments associated with a Customer. The CustomerSegments would describe a Customer in terms of needs and avoidances as well as information that would allow targeting the customer type (e.g., demographic information). A Customer owns all of the CustomerSegments associated with it; CustomerSegments cannot exist independently of a Customer. A Customer is also defined relative to a ProductOffering (see the Product package) targeting the Customer.

A Customer may be associated with a CustomerJourney, consisting of several JourneyStages that usually represent important decision and interaction points the Customer experiences in the course of finding, acquiring and using a product type. The CustomerJourney is a view of customer behavior that is relevant to the objectives of the business and CustomerJourneys are usually created by the business, not by customers. CustomerSegments are also associated with JourneyStages and Touchpoints; they describe the needs and avoidances of the Customer at the associated JourneyStage or Touchpoint.

Customers, CustomerSegments, Value Propositions, ValueItems and ValueCharacteristics can be categorized tagged by a common set of ValueCategories. These categories tags typically define a framework for analyzing types of delivered value, analyzing the satisfaction (“fit”) the customer has with the ValueProposition and its components. For example, ValueCategories can also framework tags with values such as “uses”, “pains” and “gains” could be used to categorize the aforementioned elements and support different kinds of value analysis based on the categories.

The CustomerPackage also defines ValueStream, ValueStreamStage, ValueProposition and ValueItem. These element types represent values the business believes it is offering the Customer and how those values are accumulated. The ValueProposition is “of” a ProductOffering. These believed values may match the needs and avoidances of the Customer or they may not. The degree to which the ValueProposition and its components match or fit the Customer needs and avoidances is captured by the ValueCharacteristic. This fit is typically a complex set of measures.

ValueStreamStages represent the accumulation of value leading to a ValueProposition. Consequently, the definition of the ValueStreamStages by the modeler determines the relationship between Capabilities and components of the ValueProposition by way of the ValueStreamStage.

### 7.1.3 Organization Package

The Organization package defines abstract syntax for Performer and Resource. A Performer is an OrgUnit (the humans) or a System (IT system or robot). A Performer is described by a set of abilities that match the skills required by a...
Because representing the creation of delivered value (in the form of a ValueProposition) is important, Processes and Activities can implement ValueStreams. However, Processes and Activities may also implement ValueStreamStages when it is useful to represent process detail for a ValueStreamStage. The Process metamodel provides the implements association between ValueStreamStages and Processes/Activities. This complex set of associations to a ValueStream define the Processes and Activities that produce the Outcomes that are valued as ValueItems and compose the ValueProposition.

Processes and Activities also share roles (PerformerRoles and ResourceRoles) with Capabilities, allowing the same assignments of Performers and Resources that Capabilities permit. Activities and Processes are scoped differently from Capabilities, so the roles will be associated differently as well. In addition, some roles that are associated with a Capability may not appear in a process model because they are not used in the process.

7.1.5 Product Package

The Product package defines abstract syntax for ProductOffering, representing the description of a product or product family, including terms and conditions pertaining to the acquisition and/or use of the product. ProductOffering has four subtypes:

- Merchandise Offering – a ProductOffering that includes one or more BusinessObjects for sale/lease to and use by the Customer;
- Service Offering – a ProductOffering that promises to deliver a result (Outcomes) to a Customer.
- OutsourcedServiceOffering – a ProductOffering that is a solicitation for a service to be performed for the business by another business.
- ProcurementOffering – a ProductOffering that is a solicitation by the business to acquire products from another business.

A ProductOffering is a BusinessObject or InformationItem and inherits the properties and associations of these model elements.

7.1.6 Strategy Package

The abstract syntax defined in the Strategy package is premised on the need by analysts to compare and evaluate strategy options. The package defines StrategyChoices, a container of StrategyModels, to satisfy this need. A StrategyModel represents a complete strategy, consisting of Means and Ends. Ends represent the desired results of the StrategyModel and are often changes to the value offered to the customer (ValuePropositions and ValueItems) or the fit of the offered value to the customer needs and avoidances (ValueCharacteristics). Sometimes Ends will represent an outreach to a new
customer type (Customer, CustomerSegment, CustomerJourney, CustomerJourneyStage, Touchpoint). The model element types noted are all abstractly represented by the AbstractValueModel metaclass.

The Means represent ways or approaches that are expected to produce the Means. The Means are associated with the Ends by the expects association. This association must be instanced as an association classifier to allow the modeler to express the influence of environmental factors, risks and to provide a rational for the expectation.

The Ends also represent expectations of change to results of business operations (Outcomes). The Outcomes associated with Ends represent a baseline operating state of the business and the Ends describe the hoped for operating state of the business (and are thus effectively future Outcomes).

The Means represent changes to the operating structure and behavior of the business (Capabilities, CapabilityBehaviors, CapabilityImplementations and Role assignments). These changes impact the corresponding BACM model elements. Recording the impacts helps strategists and planners deal with collaboration and conflict in the execution of business strategies.

Businesses need to track the implementation of strategies for several reasons: 1) to determine if strategy implementations (Initiatives) are on the expected trajectory; 2) to understand the impact of a change in strategy to ongoing or planned implementations; 3) to analyze and predict the impact of variances in execution on the delivered value of the business. The Initiatives represent in-process, planned, or recently completed strategy implementation efforts. These efforts should implement the general strategy Means of the adopted StrategyModel.

Initiatives are expected to produce Changes to elements of the types in the AbstractValueModel and the AbstractOperatingModel. These Changes should implement the Ends of the chosen StrategyModel. The expects association connecting Initiatives to Changes must be consistent with the expects association connecting the Means and Ends being implemented by the Initiatives and Changes.

The Initiatives and Changes elements are intended for use as gateways to actual planning documents such as project objectives, staffing, schedules and work breakdowns. These alignments allow the upward flow of information into the BACM model for analysis and management of the strategy execution. They also support change management of ongoing and planned strategy executions when strategy changes are made.

7.1.7 BACM Package

The BACM package includes two sub-packages that define abstract syntax for BACM models and foundational elements, along with importing the SMM metamodel and specializing some of its classes._1_.

7.1.7.1 The BACM_Model package

The BACM_Model package defines BACMElement as the base metaclass. It provides for a name and description of each element as well as providing multiple categorized tagged Annotation elements to be associated with any BACMElement concrete subclass instance.

BusinessElement is a specialization of BACMElement that is the base metaclass for all metaclasses representing business entities and relationships. BusinessElement can be associated with ExternalRelationship and ExternalData allowing the architect to record a relationship to an external model or document. This metamodel structure is adapted from the metamodel structure defined in the SysML V2 API and Services submission.

The BACM_Model package also defines BACM_Model as the root element in a BACM model. This element holds associations to SMM MeasureLibraries, StrategyChoices and all BusinessElements.
7.2.1.1 Un-stereotyped class translation
An un-stereotyped class in a diagram becomes a class in MOF. When such a class inherits from “BusinessElement”, a generalization association is added to specialize the “BACMPlainEntity” abstract class. These specializations are presented in diagram 7.3.1.3.

7.2.1.2 Un-stereotyped association translation
An un-stereotyped association between classes that specialize “BusinessElement” in a diagram becomes an association in MOF, except when such an association is a leg association of an <<association>> stereotyped class – see below. All other associations are translated into MOF associations.

7.2.1.3 <<class>> stereotyped binary association translation
A binary, directed association in a diagram with a <<class>> stereotype is translated into a MOF class and two binary MOF associations. Navigability is ignored and the implementation must provide bidirectional navigation for both the generated, binary, directed MOF associations. The MOF class represents a relationship and the two associations specify the types of elements that can participate in the relationship. By convention, the cardinality of the association end opposite the MOF class is 0..1, representing the notion that instances of the MOF class contain single valued properties (the owned ends) that reference a single instance of the defined type. The cardinality of the association end at the MOF class is the cardinality of the origin binary association. Since there are two associations, each one represents an end of the origin association. To preserve directionality, a naming convention is used; the name of the MOF association representing the starting association end is prefixed with “from_” and the name of the MOF association representing the ending association end is prefixed with “to_”. The MOF class is given the name of the origin association. The generated ownedEnds resulting from this translation are given names that are the names of the MOF association prefixed with “src_” and “dst_” respectively to preserve the directionality of these associations (from “src” to “dst”). Thus the origin association “produces” (see Diagram 7.3.2.1) is translated into a MOF class named “produces” and two associations:

from_produces: "src_from_produces" and “dst_from_produces”, and “to_produces” with ownedEnds “src_to_produces” and “dst_to_produces”.

The generated MOF class specializes the ”BACMBinDirRelation” abstract class and redefines the ownedEnds:

from_bacm_entity”, “from_bacm_relation”, “to_bacm_entity” and “to_bacm_relation”. This specialization and the redefinitions are created in the translation and are not shown or described in this document. The specialization permits MOF reflection to distinguish binary directed relationship instances from other types of instances.

The lifecycle semantics of the configuration of MOF class and MOF associations is equivalent to the lifecycle of the origin association. In particular, if an instance coupled to the “dst_” prefixed ownedEnd is deleted, then the corresponding instance of the class and the other association instance must also be deleted from the model.

7.2.1.4 <<association>> stereotyped origin class translation
A class in a diagram with an <<association>> stereotype is translated into a MOF class and each un-stereotyped association whose starting ownedEnd is at this class is effectively a component of an n-ary relationship that is represented by the class. The term “leg” is used in this document section to refer to such associations. Note that this allows an origin <<association>> class to participate in other associations where it is the “dst” of such an association (i.e. an <<association>> stereotyped class and some number of leg associations) is translated directly into MOF as a MOF class and MOF associations, but without the <<association>> stereotype on the class. In this case, the class and leg association names remain unchanged in the MOF metamodel as do the ownedEnd names. The ownedEnd cardinalities are also directly translated into MOF associations.

In the MOF translation, the MOF class specializes “BACMRelation”, allowing MOF reflection to distinguish that instances of the MOF class represent n-ary relations and to identify the associations that represent legs of the relationship.

The lifecycle semantics of the configuration of instances of such a MOF class and instances of its leg associations obey the same rule as for the translated binary directed associations with the <<class>> stereotype. If an participating instance
that is referenced by the “dst” slot of the link instance of a leg association is deleted from the model, then the class instance and all other leg association link instances must be deleted from the model.

7.2.1.5 <<shortcut>> stereotyped class and association translation
The basic translation is as if the class was stereotyped <<association>> or the association was stereotyped <<class>>. In addition, the MOF class representing the <<class>> stereotyped association or the <<association>> stereotyped class carries an ownedRule represented by an OpaqueExpression that is specified both in OCL and in a path language whose interpretation is equivalent to the OCL representation. This OCL expression may be invoked by the business architecture modeler to determine if additional model information is available that implies the existence of an instance of the <<shortcut>> element. The shortcut mechanism is described in section 8.1 of this specification. The language type of the alternative path expression is “BACMPathLanguage” and is intended for implementors who are based on RDF (the expression can be directly translated into a SPARQL query) or a graph language.

7.2.1.6 <<individual>> stereotype translation
The specification has a single metaclass, theBusiness with this stereotype. The concept represents the particular business being modeled and its purpose is to designate Performers that belong to this business (i.e. are employees or contract workers). There should be at most one instance of this metaclass in a model. In translation to MOF, the stereotype is removed and an OCL constraint is added that at most one instance has theBusiness as its metaclass.

The non-normative XMI that is provided with the specification must be interpreted according to a set of rules to create a conforming implementation that is not based on MOF.

In general, metamodel classes in the diagrams in this document will become meta-classes (class prototypes or templates) in an implementation. However, classes stereotyped as “association” will become association classes. These entities are binary or n-ary associations that can be specialized (from other meta-association classes), have features, and participate in other associations. UML binary associations with a <<class>> stereotype should be implemented as binary meta-association classes. UML does not require that associations be uniquely named, as noted above, some UML associations are translated into MOF as class and association pattern and in this case the class names should be unique. Consequently, the UML model adopts a suffixing convention to uniquely name associations. In this convention, the business meaning of the association is defined by a prefix term followed by an underscore and a distinguishing number. All associations with the same prefix should be understood as having the same general business meaning, even though the classes they associate may differ.

The implemented meta-association classes are presumed to have “leg” that represent roles in the association part (and argument positions in an equivalent predicate expression of the association). For convenience, each leg is assumed to have a distinct role name in the context of the association. For binary, directed meta-association classes, these names are assumed to be “src” and “tgt” and the direction of the association is “src” to “tgt”. Where an n-ary association is displayed in the UML model, the legs are represented as n-stereotyped binary associations whose names are the leg names. The leg names are defined in the model and in the meta-class-association template. Instances of the meta-class-association template must have the same set of legs and preserve the leg names of the template. These derived instance leg names may not be modified by the business architecture modeler.

A leg may have a quantification expression applied to the target end of the leg. Such expressions, as in UML, restrict the number of instances allowed as targets in an instance model. These expressions consist of an upper and lower bound that are non-negative integers. A “*” symbol in the visual representation (and a “*” in the XMI) indicate no upper bound. An instance of an association some of whose legs may have more than one targets should be taken to represent a set of tuples created by taking the cross-product of the sets of targets. For example, P(a:{x}, b:{p,q}, c:{r,s}) would generate the tuple set {(x,p,r),(x,p,s),(x,q,r),(x,q,s),(x,s,r),(x,s,s),(x,p,r),(x,p,s)}.

The specification does not provide a way to specify complex or co-occurrence constraints on the targets of a leg or on targets of two or more legs of a meta-class-association instance. However, many query/constraint languages specified for RDF or property graphs will be able to express such constraints either by signaling a constraint violation or by producing a non-empty result set of a query.

When producing an instance of an element defined in the specification metamodel, the implementation should note the meta-class template the instance is created from. Features and legs of the meta-element should be replicated in the instance and may not be changed by the business architecture modeler, except that the values of features and the targets of legs may be changed along with an instance label and description (defined as property features name and description in the metamodel—see the BACM element). The exception to this rule is any UML binary association or <<association>> stereotyped class, with a single leg labelled “related”. In this case, instances may be created with an arity specified by the
business architecture modeler and with instance leg names specified by the business architecture modeler. These meta-
associations are intended for use in representing relationships between instance elements of the same meta-element other-
than generalization and aggregation. The business architecture modeler is allowed to add property features to instances
and define their names and types, subject to type restrictions that may be provided by the implementation. The business
architecture modeler may also indicate that one instance generalizes another, but the implementation is not obligated to
determine that the instance model is consistent.

7.2.2 Unique naming of associations

UML does not require that associations be uniquely named; as noted above, some UML associations are translated into
MOF as a class and association pattern and in this case the class names should be unique. Consequently, the UML model
adopts a suffixing convention to uniquely name associations. In this convention, the business meaning of the association
is defined by a prefix term followed by an underscore and a distinguishing number. All associations with the same prefix
should be understood as having the same general business meaning, even though the classes they associate may differ.

7.2.3 Model elements (instances) represent sets and individuals

7.2.2.1 Meta-class Instances as classes

A business architecture model represents entity and relational concepts of the business. These concepts typically
represent sets of things in a business. For example, an instance of a BusinessObject labeled as “part bin” represents
several hundred actual part bins used by the business. All the part bins can be represented by a single model element
because they have identical or similar properties and are used in identical or similar ways. The “part bin” model element
needs to describe these similar properties and the similar behaviors the part bins participate in.

Sometimes a business concept represents an individual thing. A metamodeling specification that represents individuals,
will typically represent types (classes) to which the individuals belong. In the BACM metamodel, theBusiness is a
metaclass that should only ever have a single model element and that model element represents an identifiable, individual
business (i.e. the one being modeled). Normally, an instance of a BACM metaclass would be interpreted as a class (type).
However, in this case the instance of theBusiness is to be interpreted as an individual. Expression of this interpretation
requires two constraints:

1. The extent of the theBusiness metaclass is restricted to 0 or one model elements (instances);
2. The model element that is the sole instance of theBusiness represents an individual

Because the second of these constraints involves model extents in the real world, it cannot be enforced except in a model
that has classes and individuals as disjoint domains. The BACM model does not have a domain of individuals and the
constraint can neither be represented in the model nor enforced. Other modeling languages, such as UAF and SysML, do
have classes and individuals as disjoint domains and such a constraint can be specified.

The first constraint can be expressed in OCL as a constraint that no more than one model element may have theBusiness
as a metaclass. The <<individual>> stereotype is used in the specification document to indicate this constraint. The OCL
expression of this constraint is found in the normative MOF XMI that is part of this specification.

While it is theoretically possible to apply the <<individual>> stereotype to a binary or n-ary association, such a construct
is not used in this specification.

7.2.3.2.4 Meta-model association instances as association classes

The UML 2.5.1 specification allows N-ary associations to be class associations and distinguishes owned features as
pertaining to the class and owned ends as pertaining to the association (see [OMG UML] 11.5.3.2). Instances of meta-
model associations should be treated similarly, i.e. as a combination of a class and an association. Where applicable, the
semantics of class associations should be followed. The metamodel also makes use of metaclasses stereotyped as
<<association>>; instances of these metaclasses in M1 models should be implemented as class associations.
Simple associations in the metamodel with a stereotype of <<class>> or <<shortcut>> should also be implemented at the
M1 model as binary, directed class associations.

7.2.47.2.5 Distinguished association names

These associations are exempted from requirement previously stated to implement associations as class-
associations or a similar representation permitting associations to have properties and participate in other associations.
The meaning and usage of these associations is defined here and not in the generated content of section 7.3. The
distinguished association names may consist of a prefix and a suffix separated by an underscore. The prefix designates
the semantic interpretation, while the suffix designates a distinct association.

7.2.47.2.5.1 aggregates

This association name identifies an association type that creates hierarchies of same-typed instances of meta-classes. An
example is the use of this association name on a self-association with the ValueCategory meta-class. ValueCategory
eventually specializes Category, and Category is associated ("aggregates") with CategoryLibrary. The semantics of this
association in this case are that an instance of CategoryLibrary aggregates one or more instances of ValueCategory that
aggregate other instances of ValueCategory.

The association and cardinalities ensure that each ValueCategory instance has a single aggregator, which may be either
another instance of ValueCategory or an instance of CategoryLibrary. This interpretation shall be applied to any other
uses of the "aggregates" association in the meta-model. The common use of the "aggregates" label indicates a semantic
association type that is specialized by the meta-classes at the association ends, e.g., the "aggregate" self-association of
ValueCategory specializes the generic "aggregates" to an association between instances of ValueCategory. They
semantic between instances of the associated types. Duplicates may appear in the collection and the same instance may
appear in more than one collection. Associations with this name may appear multiple times in the diagrams and in the
MOF model. They all have the same semantic interpretation, but are distinguished by the meta-classes they associate. In
the BusinessElement diagram, BusinessElements may aggregate other BusinessElements as long as the elements being
aggregated have the same, concrete metaclass (this is enforced by an OCL constraint). In the ValueStream diagram, an
aggregates association allows ValuePropositions to aggregate ValueItems. This aggregates association has the same
semantic, but is restricted in the end types it allows.

7.2.47.2.5.2 generalizes

The instances of this association create a generalization semantic relationship between the meta-class instances at the
association ends. The association is restricted to 1) self-association of a concrete meta-class; 2) association between
concrete meta-classes such that one meta-class eventually specializes the other. In case 2), the instance of this association
may not contradict the generalization relationship between the meta-classes. The restriction to instances of the same
metaClass is defined in the BusinessElement diagram by an OCL constraint on BusinessElement,

7.2.47.2.5.3 owns

The instances of this association carry the semantic of exclusive ownership. The target of the association may not exist
separately from the source.

7.2.47.2.5.4 related

Some meta-classes stereotyped as associations should be realized in models as n-ary relations, whose arity is determined
by the architect. These meta-classes have a single association, related, to a target meta-class. When realized in a model,
multiple instances of the related association may be created by the architect and given distinct labels to distinguish them.
The category mechanism can be used to indicate that one or more instances of these n-ary associations are representatives
of a type identified by the category.

7.2.57.2.6 N-ary Associations reified as Classes and Binary Associations

N-ary associations in this metamodel are represented as classes with an <<association>> stereotype. In the diagrams, the
n-ary association class may be represented either by a box or a diamond. The roles of the n-ary association are modeled
as binary associations between the n-ary association class and the classes allowed to participate in these roles (i.e. the
participants). However, the UML interpretation of this configuration is deficient in some important ways: 1) the UML
specification states that the cardinality specification of a role assumes that the other n-1 role entities are held constant; 2) the specification is unclear about how to interpret optional role participants.

An n-ary association specified in this way in this specification should be interpreted in its extension as a set of n-tuples, possibly with constraints between elements in each tuple and among the tuples, in addition to the requirement that the entries in the kth position of the tuple are instances of the class participating in the kth role. This specification does not determine an implementation, and implementors are free to number the roles of each association as they choose.

Likewise, the specification does not determine a technical language for the specification of constraints. In the specification, prose is used to define constraints.

**7.2.6 Like-named associations with same meta-class source**

This is a UML notational pattern used to represent that the instance association should have as its target instance classes of one target or the other, but target instances may not a mixture of instances of the targets. For example, in the Process diagram, the implements association has as targets \textit{Customer::ValueStream} and \textit{Customer::ValueStreamStage}. At the model level, an implements instance can have as targets some instances of \textit{Customer::ValueStream} or some instances of \textit{Customer::ValueStreamStage} but not a mixture of such instances.

**7.2.7 Application of business architecture frameworks with MEF**

The Metamodel Extension Facility (MEF) provides for the definition and application of profiles and stereotypes that can be applied to any MOF-based model elements. The implementation of MEF or its equivalent is a requirement.

It is recommended that architects encode conceptual frameworks, such as the Value Proposition Canvas (VPC) in a MEF profile and use the stereotypes to characterize model elements, such as ValueItems and CustomerSegments according to the principles of the Value Proposition Canvas by applying stereotypes, such as “pains”, “gains” and “uses” to the model elements.

**7.3 BACM Metamodel (Normative)**

The following material describes the classes and associations that comprise the BACM metamodel

**7.3.1 Package: BACM\_Model**
7.3.1.1 **Diagram: BACM_Element**

The BACM_Element diagram defines abstract syntax for the BACMElement and for Annotation. BACMElement is an abstract class that provides annotation, description and name to all classes used to represent concepts of the business being modeled.

The BusinessElement abstract class provides a boolean feature allowing modelers to specify whether an instance is abstract or concrete. BusinessElement also provides an ExternalRelationship that allows the modeler to specify ExternalData that is associated with the BusinessElement.
The BACM_Entities diagram displays the specializations of BACMPlainEntity as model classes. These specializations are not transformed in the production of the MOF-compliant XMI model.

Commented [JR17]: Issue BACM-1 resolved by BACM-4
The BACM_Model diagram defines abstract syntax for BACM_Model, whose instance is the root element for a BACM model.
The BACM_Model element is a container for all BusinessElements in the model. The BACM is associated with a single StrategyChoices (which may contain several alternative StrategyModels). Finally, the BACM_Model contains an SMMModel element and a selected set of SMM MeasureLibraries. The integration with SMM allows any instance of BusinessElement to be the measurand of a SMM Measurement. The BACM specification effectively imports at least the SMM 1.2 specification.
7.3.1.4 Diagram: BACM_Model

Commented [JR18]: Issue BACM-2 resolved by BACM-10
The package diagram shows the inclusion relationships between the BACM package and the sub-packages that contain the metamodel classes and associations. The packaging is done for convenience in managing the metamodel and should not be construed as defining domains of business architecture concepts.
7.3.1.5 **Diagram: BusinessElement**

```plaintext
Inherit
generalizations

BACMElement
+ description: String
+ name: String

BusinessElement
+ abstract: Boolean

Constraints:
{self->oclType = self.owner->oclType()}
{self->oclType() = self.aggregator->oclType()}
{self->oclType() = self.more_general->oclType()}

BACMRelation
+ constr: String [1..*]
+ hasDetail(): Boolean

BACMEntity
+ aggregator 0..*
+ aggregated 0..* from_bacm_relation 0..*
+ to_bacm_relation 0..* to_bacm_entity 1..* from_bacm_entity 1..* {redefines bacm_entity}
+ owner 0..1 owns_0 0..*
+ owned 0..*
+ less_general 0..*
generalizes_0
generalizes_0
+ more_general 0..*
generalizes_0
+ less_general 0..*
generalizes_0
+ more_general 0..*
generalizes_0

BACMRefEntity
+ from_bacm_relation 0..*
+ to_bacm_relation 0..*
+ less_general 0..*
generalizes_0
+ more_general 0..*
generalizes_0

BACMShortcut
+ constr: String [1..*]
+ hasDetail(): Boolean
```

**Commented [JR20]:** Issue BACM-1 resolved by BACM-4
Issue BACM-83 resolved by BACM-90
Issue BACM-43 resolved by BACM-44
The BusinessElement diagram defines abstract syntax for BusinessElement, the abstract base class for all metaclasses whose instances represent business entities. Refer to the normative XMI file for details. This diagram also defines specializations of BusinessElement that are used to support the transformation of the XMI for this model to a MOF compliant XMI. These specializations are BACMEntity, BACMRelation, BACMPlainEntity, BACMBinDirRelation and BACMShortcut. BACMPlainEntity is the generalization of all BACM classes representing concepts of the business being modeled that are not stereotyped and not transformed in the production of MOF-compliant XMI (see the BACM_Entities diagram and the normative MOF XMI file for details).
The BACM metamodel is based on SMOF, not MOF. Consequently, the assumption of disjointness of concrete classifiers does not hold and explicit assertions of disjointness must be made. The DisjointElements diagram asserts that the set of elements shown are pairwise disjoint (i.e. no object can have two or more of these classes as its metaclass).
7.3.1.7 **Diagram: DisjointEntities**

The BACM metamodel is based on SMOF, not MOF. Consequently, the assumption of disjointness of concrete classifiers does not hold and explicit assertions of disjointness must be made. The DisjointEntities diagram asserts that the set of elements shown are pairwise disjoint (i.e. no object can have two or more of these classes as its metaclass).

7.3.1.8 **Class Name:** AbstractAction  **Class Type:** Class  **Class Stereotype:**

Base Classes:

AbstractAction is used to classify entities that should be disjoint from Capability, AbstractResult and AbstractThing. It is not used for any other purpose in the metamodel.

7.3.1.8.1 Attributes, Methods and Connectors:

- **Association Name:** Association Type: Generalization  **Stereotype:**  
  **Source Class:** ValueStreamStage  **Target Class:** AbstractAction

- **Association Name:** Association Type: Generalization  **Stereotype:**  
  **Source Class:** AbstractProcess  **Target Class:** AbstractAction
7.3.1.9  **Class Name:** AbstractResult  **Class Type:** Class  **Stereotype:**

Base Classes:  
AbstractResult is used to classify entities that should be disjoint from Capability, AbstractAction and AbstractThing. It is not used for any other purpose in the metamodel.

7.3.1.9.1  **Attributes, Methods and Connectors:**

Association Name:  Association Type: Generalization  **Stereotype:**
Source Class: ValueProposition  Target Class: AbstractResult

Association Name:  Association Type: Generalization  **Stereotype:**
Source Class: Ends  Target Class: AbstractResult

Association Name:  Association Type: Generalization  **Stereotype:**
Source Class: ValueItem  Target Class: AbstractResult

Association Name:  Association Type: Generalization  **Stereotype:**
Source Class: CustomerSegment  Target Class: AbstractResult
7.3.1.10 **Class Name:** AbstractThing  **Class Type:** Class Stereotype:
Base Classes: AbstractThing is used to classify entities that should be disjoint from Capability, AbstractResult and AbstractAction. It is not used for any other purpose in the metamodel.

7.3.1.10.1 Attributes, Methods and Connectors:

Association Name: Association Type: Generalization Stereotype:
Source Class: AbstractBusinessObject Target Class: AbstractThing
Association Name: Association Type: Generalization Stereotype:
Source Class: Customer Target Class: AbstractThing
Association Name: Association Type: Generalization Stereotype:
Source Class: Resource Target Class: AbstractThing
Association Name: Association Type: NoteLink Stereotype:
Source Class: Constraint Target Class: AbstractThing
Association Name: Association Type: Generalization Stereotype:
Source Class: Performer Target Class: AbstractThing

7.3.1.11 **Class Name:** Annotation  **Class Type:** Class Stereotype:
Base Classes: BACMElement
**Definition:** Annotation provides the modeler an ability to associate tag/value pairs to any BACMElement in a BACM model.
**Usage:** Annotations may be annotated. Annotations may also be specialized in a BACM model to add additional attributes.

7.3.1.11.1 Attributes, Methods and Connectors:

**Attribute Name:** tag  **Attribute Type:** String
**Definition:** The property identifies the intended meaning of the value property.
Attribute Name: value  Attribute Type: String
Definition: The value property holds the value of the annotation. The meaning of this value is provided by the tag property.

Association Name: annotates Association Type: Association Stereotype: Source Class: Annotation [0..*] Target Class: BACMElemnt [1]
Definition: The annotates association links an Annotation to the BACMElemnt being annotated.

Association Name: generalizes Association Type: Association Stereotype: Source Class: Annotation [0..1] Target Class: Annotation [0..*]

Association Name: generalizes Association Type: Association Stereotype: Source Class: Annotation [0..1] Target Class: Annotation [0..*]

Association Name: Association Type: NoteLink Stereotype: Source Class: Constraint [] Target Class: Annotation []

7.3.1.12  Class Name: BACM_Model  Class Type: Class Stereotype:  
Base Classes: BACMElemnt
Definition: The BACMModel represents the root element of a BACM model (i.e. the element from which a tool or person can navigate to every other element in the model)
Usage: A single instance of this class must exist in an instance model.

7.3.1.12.1 Attributes, Methods and Connectors:

Association Name: strategy_choices Association Type: Association Stereotype: Source Class: BACM_Model [1] Target Class: StrategyChoices [0..*]
Definition: strategy_choices links a set of StrategyChoices to a BACMModel.
Usage: To facilitate reuse of the BACM model in different strategy situations, multiple StrategyChoices may be associated with a BACMModel.

Association Name: measure_library Association Type: Association Stereotype: Source Class: BACM_Model [0..1] Target Class: MeasureLibrary [0..*]
Definition: The measure_library association links an SMM measure library to the BACM model.

Association Name: smm_model Association Type: Association Stereotype: Source Class: BACM_Model [0..1] Target Class: SmmModel [0..1]
Definition: The smm_model association links an SMM model to the BACM model.

Association Name: bacm_element Association Type: Association Stereotype: Source Class: BACM_Model [1] Target Class: BACMElemnt [0..*]
Definition: bacm_element links the BACM_Model to all of the BACMElemnts contained in a model.
Usage: This association should be interpreted to include all n-ary associations, associations stereotyped <class>> and classes stereotyped <<association>>. The translation of this model to MOF creates classes representing these types of associations that are not allowed in MOF and these classes specifically inherit from BusinessElement.
Usage: This association is exclusive; BACMElemnts are not allowed to be shared in different BACM_Models.
7.3.1.13 **Class Name:** BACMBinDirRelation  **Class Type:** Class  **Stereotype:**

**Base Classes:** BACMRelation

**Definition:** BACMBinDirRelation is an abstract class that generalizes the classes resulting from the transformation of model associations stereotyped as <<class>> or <<shortcut>>. It specializes BACMRelation to represent binary directed relations and redefines the association between BACMRelation and BACMEntity to designate the start (from_bacm_entity) and end (to_bacm_entity) of the relation direction.

### Attributes, Methods and Connectors:

- **Association Name:** BACMRelToEntity  **Association Type:** Association  **Stereotype:**
  **Source Class:** BACMBinDirRelation [0..*]  **Target Class:** BACMEntity [1..*]

- **Association Name:** BACMRelFromEntity  **Association Type:** Association  **Stereotype:**
  **Source Class:** BACMBinDirRelation [0..*]  **Target Class:** BACMEntity [1..*]

- **Association Name:** Generalization  **Association Type:** Generalization  **Stereotype:**
  **Source Class:** BACMBinDirRelation []  **Target Class:** BACMRelation []

7.3.1.14 **Class Name:** BACMElement  **Class Type:** Class  **Stereotype:**

**Base Classes:**

**Definition:** The BACMElement represents the class of all elements in a BACM model. It provides elements with a name and description and allows elements to be annotated.

**Usage:** BACMElement is an abstract class and cannot be instantiated in a model.

### Attributes, Methods and Connectors:

- **Attribute Name:** description  **Attribute Type:** String
  **Definition:** The description property provides a description of the BACMElement.
  **Usage:** Typically the description states what business concept or entity the BACMElement is intended to represent.

- **Attribute Name:** name  **Attribute Type:** String
  **Definition:** The name property provides a term that indicates what the BACMElement represents in the BACM model.
  **Usage:** The description property should provide a more detailed description of the represented business concept or entity.

- **Association Name:** annotates  **Association Type:** Association  **Stereotype:**
  **Source Class:** Annotation [0..*]  **Target Class:** BACMElement [1]
Definition: The *annotates* association links an *Annotation* to the *BACMElement* being annotated.

**Association Name:** *annotates*  
**Association Type:** Generalization  
**Stereotype:**  
**Source Class:** ExternalData  
**Target Class:** BACMElement

**Association Name:** *annotates*  
**Association Type:** Generalization  
**Stereotype:**  
**Source Class:** ExternalRelationship  
**Target Class:** BACMElement

**Association Name:** *annotates*  
**Association Type:** Generalization  
**Stereotype:**  
**Source Class:** Annotation  
**Target Class:** BACMElement

**Association Name:** *annotates*  
**Association Type:** Generalization  
**Stereotype:**  
**Source Class:** BusinessElement  
**Target Class:** BACMElement

**Association Name:** bacm_element  
**Association Type:** Association  
**Stereotype:**  
**Source Class:** BACM_Model  
**Target Class:** BACMElement

**Definition:** *bacm_element* links the *BACM_Model* to all of the *BACMElements* contained in a model.  
**Usage:** This association should be interpreted to include all n-ary associations, associations stereotyped `<class>>` and classes stereotyped `<<association>>`. The translation of this model to MOF creates classes representing these types of associations that are not allowed in MOF and these classes specifically inherit from *BusinessElement*.  
This association is exclusive; *BACMElements* are not allowed to be shared in different *BACM_Models*.

**Association Name:** *annotates*  
**Association Type:** Generalization  
**Stereotype:**  
**Source Class:** BusinessElement  
**Target Class:** BACMElement

---

### 7.3.1.15 *Class Name:* BACMEntity  
### Class Type: Class  
### Stereotype:

**Base Classes:** *BusinessElement*  
**Definition:** *BACMEntity* is an abstract class that is characterized by participating in relationships defined by *BACMRelation* and *BACMBinDirRelation*. *BACMEntity* is also a generalization of all classes intended to represent concepts of the modeled business. See the normative XMI file for details.  
**Usage:** Both *BACMRelation* and *BACMBinDirRelation* are specializations of *BACMEntity* allowing these relationships to participate in other relationships

### 7.3.1.15.1 Attributes, Methods and Connectors:

**Association Name:** *annotates*  
**Association Type:** Generalization  
**Stereotype:**  
**Source Class:** BACMEntity  
**Target Class:** BusinessElement

**Association Name:** BACMRelToEntity  
**Association Type:** Association  
**Stereotype:**  
**Source Class:** BACMBinDirRelation  
**Target Class:** BACMEntity

**Association Name:** BACMRelation  
**Association Type:** Generalization  
**Stereotype:**  
**Source Class:** BACMRelation  
**Target Class:** BACMEntity

**Association Name:** BACMRelEntity  
**Association Type:** Association  
**Stereotype:**  
**Source Class:** BACMRelation  
**Target Class:** BACMEntity
7.3.1.16 **Class Name:** BACMPlainEntity  **Class Type:** Class  **Stereotype:**

**Base Classes:** BACMEntity

**Definition:** BACMPlainEntity is an abstract class disjoint from BACMRelation that classifies all BACM classes representing concepts of the modeled business that are not specializations of BACMRelation.

**Usage:** BACMPlainEntity and BACMRelation distinguish classes intended to represent entities from those intended to represent associations.

### 7.3.1.16.1 Attributes, Methods and Connectors:

**Association Name:** Association Type: Generalization Stereotype:

- **Source Class:** BACMPlainEntity  **Target Class:** BACMPlainEntity

- **Source Class:** CustomerJourney  **Target Class:** BACMPlainEntity

- **Source Class:** AbstractBusinessObject  **Target Class:** BACMPlainEntity

- **Source Class:** AbstractCapability  **Target Class:** BACMPlainEntity

- **Source Class:** StrategyChoices  **Target Class:** BACMPlainEntity

- **Source Class:** AbstractProcess  **Target Class:** BACMPlainEntity

- **Source Class:** ValueProposition  **Target Class:** BACMPlainEntity

**Association Name:** Association Type: NoteLink Stereotype:

- **Source Class:** Constraint  **Target Class:** BACMPlainEntity

**Association Name:** Association Type: Generalization Stereotype:

- **Source Class:** ValueStreamStage  **Target Class:** BACMPlainEntity
Association Name:  Association Type: Generalization Stereotype: 
Source Class: Ends  []  Target Class: BACMPlainEntity []

Association Name:  Association Type: Generalization Stereotype: 
Source Class: Jurisdiction  []  Target Class: BACMPlainEntity []

Association Name:  Association Type: Generalization Stereotype: 
Source Class: Valueltm  []  Target Class: BACMPlainEntity []

Association Name:  Association Type: Generalization Stereotype: 
Source Class: Outcome  []  Target Class: BACMPlainEntity []

Association Name:  Association Type: Generalization Stereotype: 
Source Class: Touchpoint  []  Target Class: BACMPlainEntity []

Association Name:  Association Type: Generalization Stereotype: 
Source Class: ValueStream  []  Target Class: BACMPlainEntity []

Association Name:  Association Type: Generalization Stereotype: 
Source Class: Performer  []  Target Class: BACMPlainEntity []

Association Name:  Association Type: Generalization Stereotype: 
Source Class: Customer  []  Target Class: BACMPlainEntity []

Association Name:  Association Type: Generalization Stereotype: 
Source Class: CapabilityImplementation  []  Target Class: BACMPlainEntity []

Association Name:  Association Type: Generalization Stereotype: 
Source Class: CustomerJourneyStage  []  Target Class: BACMPlainEntity []

Association Name:  Association Type: Generalization Stereotype: 
Source Class: Resource  []  Target Class: BACMPlainEntity []

Association Name:  Association Type: Generalization Stereotype: 
Source Class: Means  []  Target Class: BACMPlainEntity []

Association Name:  Association Type: Generalization Stereotype: 
Source Class: Initiative  []  Target Class: BACMPlainEntity []

Association Name:  Association Type: Generalization Stereotype: 
Source Class: CustomerSegment  []  Target Class: BACMPlainEntity []

Association Name:  Association Type: Generalization Stereotype: 
Source Class: StrategyModel  []  Target Class: BACMPlainEntity []
7.3.1.17 **Class Name:** BACMRelation  **Class Type:** Class  **Stereotype:**

Base Classes: BACMEntity

**Definition:** BACMRelation is an abstract class that models n-ary relations with features and the ability to participate in other specializations and instances of this class as bacm_entity ends.

**Usage:** BACMRelation is the generalization of all classes resulting from the transformation of <<association>> stereotyped classes. The model associations determined to be legs of the <<association>> stereotyped classes are transformed to specialize the association with ends bacm_entity and bacm_relation.

7.3.1.17.1 Attributes, Methods and Connectors:

**Association Name:** Association Type: Generalization  **Stereotype:**

Source Class: BACMRelation  Target Class: BACMEntity

**Association Name:** BACMRelEntity  **Association Type:** Association  **Stereotype:**

Source Class: BACMRelation  Target Class: BACMEntity

**Association Name:** Association Type: NoteLink  **Stereotype:**

Source Class: Constraint  Target Class: BACMRelation

**Association Name:** Association Type: Generalization  **Stereotype:**

Source Class: BACMBinDiRelRelation  Target Class: BACMRelation

7.3.1.18 **Class Name:** BACMShortcut  **Class Type:** Class  **Stereotype:**

Base Classes:

**Definition:** BACMShortcut is an abstract class inherited by the transformation of all metamodel classes stereotyped as <<shortcut>> and all generated classes that result from the transformation of model classes stereotyped as <<shortcut>>. It declares a string (constr) that defines the shortcut constraint and a boolean valued function (hasDetail) that evaluates the constraint string and determines whether it is true or false.

**Usage:** In the normative XMI, the constraint string defined in the model is represented as an OCL function that determines if there is a specified path between the instances at the ends of the association. The modeler is allowed to use the constraint mechanism to define shortcut associations within the instance model. In this case, the constr attribute will contain the constraint string and the modeler must provide an implementation of the hasDetail function that evaluates the string and returns a boolean result.

7.3.1.18.1 Attributes, Methods and Connectors:

**Attribute Name:** constr  **Attribute Type:** String

**Method Name:** hasDetail
Class Name: BusinessElement  Class Type: Class Stereotype:
Base Classes: BACMElement
Definition: BusinessElement represents a concept or entity that existing or is planned to exist in the business.
Usage: BusinessElement is an abstract base class for all classes whose instances represent business entities.

7.3.1.19 Attributes, Methods and Connectors:

Attribute Name: abstract  Attribute Type: Boolean
Definition: The abstract property of a BusinessElement has a boolean value and the true value means that the represented business concept is not a tangible entity.
Usage: This property allows a business architect to create a framework through generalization at the M1 level that prevents instances marked as abstract from being included in the instance of the M1 model that is also at the M1 level.

Association Name: aggregates_0  Association Type: Association Stereotype:
Source Class: BusinessElement [0..*]  Target Class: BusinessElement [0..*]
Definition: The aggregates association instance defines an undifferentiated and non-exclusive relationship between BusinessElements (in contrast with owns which is an exclusive relationship).
Usage: The aggregates association instance defines an undifferentiated and non-exclusive relationship between instances of BusinessElement allowing a BusinessElement instance to be a collection of other BusinessElement instances.
Constraint: The aggregates association is restricted to BusinessElement concrete subtypes that are of the same type (see BusinessElement).

Association Name: generalizes_0  Association Type: Association Stereotype:
Source Class: BusinessElement [0..*]  Target Class: BusinessElement [0..*]
Definition: The generalizes association classifies links with the semantic that the less_general end of the link is less general than the more_general end of the link.
Usage: Instances of this association are used to create a generalization relationship between instances.
Constraint: The generalizes association is restricted to instances of the same type (see BusinessElement).

Association Name: owns_0  Association Type: Generalization Stereotype:
Source Class: BusinessElement [0..1]  Target Class: BACMElement []

Association Name: aggregates_0  Association Type: Association Stereotype:
Source Class: BusinessElement [0..*]  Target Class: BusinessElement [0..*]
Definition: The aggregates association instance defines an undifferentiated and non-exclusive relationship between BusinessElements (in contrast with owns which is an exclusive relationship).
Usage: The aggregates association instance defines an undifferentiated and non-exclusive relationship between instances of BusinessElement allowing a BusinessElement instance to be a collection of other BusinessElement instances.
Constraint: The aggregates association is restricted to BusinessElement concrete subtypes that are of the same type (see BusinessElement).

Association Name: owns_0  Association Type: Association Stereotype:
Source Class: BusinessElement [0..1]  Target Class: BusinessElement [0..*]
Definition: The owns association instance defines an undifferentiated and exclusive relationship between BusinessElements (in contrast with aggregates which is a non-exclusive relationship).
Usage: The owns association instance defines an undifferentiated and exclusive relationship between instances of BusinessElement allowing a BusinessElement instance to be a container of other BusinessElement instances.
Constraint: The owns association is restricted to BusinessElement concrete subtypes that are of the same type (see BusinessElement).
Association Name: nature Association Type: Association Stereotype:
Source Class: Responsible [0..*] Target Class: BusinessElement [0..*]
Definition: The nature leg of the Responsible designates a BusinessElement that helps define the scope and/or nature of the Responsible association.

Association Name: class Association Type: Association Stereotype:
Source Class: Scope [0..*] Target Class: BusinessElement [0..1]
Definition: The class association provides the SMM::Scope element with a scoping reference to one or more BusinessElements.

Association Name: BusEleExtRel Association Type: Association Stereotype:
Source Class: ExternalRelationship [1..*] Target Class: BusinessElement [1..*]

Association Name: generalizes_0 Association Type: Association Stereotype:
Source Class: BusinessElement [0..*] Target Class: BusinessElement [0..*]
Definition: The generalizes association classifies links with the semantic that the less_general end of the link is less general than the more_general end of the link.
Usage: Instances of this association are used to create a generalization relationship between instances.
Constraint: The generalizes association is restricted to instances of the same type (see BusinessElement).

Association Name: element Association Type: Association Stereotype:
Source Class: Measurement [0..*] Target Class: BusinessElement [1..*]
Definition: The measurand association specializes the SMM::measurand association to associate a SMM::Measurement with a BusinessElement.
Usage: Any n-ary association, class stereotyped as <<association>> or association stereotyped as <<class>> should be treated as a BusinessElement target of this association.

Association Name: Association Type: Generalization Stereotype:
Source Class: BACMEntity [] Target Class: BusinessElement []

Association Name: owns_0 Association Type: Association Stereotype:
Source Class: BusinessElement [0..1] Target Class: BusinessElement [0..*]
Definition: The owns association instance defines an undifferentiated and exclusive relationship between BusinessElements (in contrast with aggregates which is a non-exclusive relationship).
Usage: The owns association instance defines an undifferentiated and exclusive relationship between instances of BusinessElement allowing a BusinessElement instance to be a container of other BusinessElement instances.
Constraint: The owns association is restricted to BusinessElement concrete subtypes that are of the same type (see BusinessElement).

7.3.20 Class Name: ExternalData Class Type: Class Stereotype:
Base Classes: BACMElement
Definition: ExternalData is a class that wraps an IRI. An ExternalRelationship instance may be associated with multiple ExternalData instances.

7.3.20.1 Attributes, Methods and Connectors:

Attribute Name: resourceIdentifier Attribute Type: IRI

Association Name: Association Type: Generalization Stereotype:
Source Class: ExternalData [] Target Class: BACMElement []
7.3.1.21 **Class Name:** ExternalRelationship  **Class Type:** Class  **Stereotype:**

**Base Classes:** BACMElement  

**Definition:** ExternalRelationship represents a relationship between a BusinessElement in a provider tool or repository to ExternalData in another tool or Repository. The external data may be a BusinessElement (or a linked collection of BusinessElements) or some other element (or linked collection of elements) from a model that is not a BACM model. The IRI must identify a resource to which the specification String can be applied to identify the element (or linked set of elements) in that resource. The language attribute of the ExternalRelationship identifies the language of the specification String.

Note that BusinessElement classifies all BACM classes and associations that are intended to represent business concepts (as opposed to model concepts or analysis concepts).

**Usage:** The tool provider may elect to provide services to dereference the ExternalData and apply the specification to allow the architect to view and interact with the results. However, a compliant implementation may just implement, import and export the ExternalRelationship, the ExternalData and the links connecting them and connecting the ExternalRelationship to the BusinessElement.

If the language string is the string "Natural" or a string that identifies a natural language, then the specification String will be a natural language description of the alignment mapping.

### Attributes, Methods, and Connectors:

- **Attribute Name:** language  **Attribute Type:** String
- **Attribute Name:** specification  **Attribute Type:** String

7.3.1.22 **Class Name:** IRI  **Class Type:** DataType  **Stereotype:**

**Base Classes:**

BACM 1.0
7.3.1.22.1 Attributes, Methods and Connectors:

7.3.2 Package: Capability

7.3.2.1 Diagram: Capability

Commented [JR25]: Issue BACM-83 resolved by BACM-9 removed owns, aggregates and specializes and added OCL2 constraints to Capability.
The Capability diagram defines abstract syntax for Capability and CapabilityBehavior classes. Both metaclasses inherit from AbstractCapability which allows their instances to produce and need FlowOutcomes and to be informed by InformationItems (e.g. a decision or action associated with the Capability is influenced by the InformationItems).

While Capability represents an ability to produce a FlowOutcome, CapabilityBehavior represents a particular way, process or manner of producing that FlowOutcome. A CapabilityBehavior that delivers a Capability must produce and/or need FlowOutcomes that are equivalent to, specialize, or contribute parts to the Outcomes produced by the Capability. A CapabilityBehavior may produce and need FlowOutcomes not needed or produced by the Capability it delivers.

The Capability diagram also specifies abstract syntax for BusinessObjects and InformationItems. FlowOutcomes represent the state of BusinessObjects and/or InformationItems. The AbstractBusinessObject defines associations and properties common to BusinessObject and InformationItem.

Capabilities may be decomposed by the owns association creating a strict hierarchy (i.e. a subCapability may not have multiple owners)
Where a FlowOutcome is obvious, the scopes shortcut association may be used to omit it from the model. The modeler may later elect to define FlowOutcomes that are consistent with the possess shortcut association constraint.

A CapabilityImplementation represents actual or planned role occupants (see Roles diagram) implementing a Capability and/or CapabilityBehavior. The modeler may also specify Roles and assignments to these Roles that are consistent with the constraint defined in the implements shortcut association.
See also the Process diagram in the Process package.

**Diagram: CapabilityValue**

[Diagram of CapabilityValue]

Commented [JR26]: Issue BACM-83 resolved by BACM-9
The CapabilityValue diagram extends the abstract syntax defined by the Capability diagram to show the relationship between Capability/Outcome and ValueStreamStage/ValueItem. The business architecture core metamodel recognizes that different stakeholders may place different values on the same Outcome. The metamodel provides the ValueItem as a model element to represent value beliefs. Refer to the document section on the Customer package for a full description of the model elements for representing customers (and other stakeholders) and value beliefs.

The supports association between Capability and ValueStreamStage represents the business capabilities needed to produce the Outcomes that are valued by the Customer to whom the ValueProposition is directed.

Not all Outcomes that are produced by Capabilities are experienced by the Customer. The Outcomes that are experienced by the Customer are associated with a Touchpoint representing an interaction between the business and the customer. The Outcomes experienced by the Customer at a Touchpoint should have ValueItems.

Most of the Outcomes in a business architecture model will be internal to the business, not experienced by the Customer and should not have ValueItems, unless they represent intermediate Outcomes that contribute to a ValueProposition. For example, a design capability following an "ease of use" design principle, and a test capability for testing "ease of use" with a prototype produce an "ease of use" outcome that is realized in the final product. The "ease of use" ValueItem can value the design Outcome, the test Outcome and the customer use Outcome.
7.3.2.3 **Diagram: Roles**

Commented [JR27]: Issue BACM-83 resolved by BACM-9
The Roles diagram defines the abstract syntax for roles and role assignments. Roles define how Performers and Resources participate in Capabilities and CapabilityBehaviors.

The Role acts as a ternary association that represents assignments of Performers and Resources to PerformerRoles and ResourceRoles that are role associated to Capabilities and CapabilityBehaviors.

A Role may not exist except in a role association to a Capability and/or CapabilityBehavior. Assignment of Resource or Performer to their respective Roles is optional.

A CapabilityImplementation represents a bundling of (aggregates) Resources and Performers to implement a Capability and/or CapabilityBehavior. A CapabilityImplementation may be empty and contain an annotation suggesting future contents.

The Implements shortcut association allows a CapabilityImplementation to be associated with a Capability and/or a CapabilityBehavior without having to specify the Roles or the Performers and/or Resources or the associations that link them together. The modeler may subsequently add these details consistent with the implements shortcut association constraints.

The assignment shortcut association allows a CapabilityImplementation to be assigned to selected Roles without having to specify the details about Performers and Resources in the CapabilityImplementation and how they have assignments to these Roles.
7.3.2.4 **Class Name:** AbstractBusinessObject  **Class Type:** Class Stereotype:

**Base Classes:** AbstractOperatingModel, AbstractThing, BACMPlainEntity

**Definition:** AbstractBusinessObject represents BusinessObjects or InformationItems.

**Usage:** AbstractBusinessObject cannot be instanced or specialized in a business architecture model. The AbstractBusinessObject metaclass has two concrete subclasses:

- BusinessObject - instances represent tangible things of importance to the business.
- InformationItem - instances represent intangible (mental) concepts important to the business.

The AbstractBusinessObject metaclass provides its concrete specializations with the state_of association to Outcomes and the scope association to Capability and CapabilityBehavior. AbstractBusinessObject also provides for ObjectRelations that may relate any collection of BusinessObjects and InformationItems.

7.3.2.4.1 Attributes, Methods and Connectors:

**Association Name:** scopes  **Association Type:** Association Stereotype: <shortcut>

**Source Class:** AbstractBusinessObject [0..1]  **Target Class:** AbstractCapability [0..1]

**Definition:** The scopes shortcut association allows a Capability and/or CapabilityBehavior to be associated with some BusinessObjects and/or InformationItems without defining Outcomes produced or needed by the Capability and/or CapabilityBehavior.

**Usage:** The modeler may elect to subsequently define such Outcomes, which must be consistent with the constraint specified by the scopes shortcut association.

**Constraint:** Let BO1 be a BusinessObject and C1 be a Capability that are associated by scopes s1. Then there should exist in the model an Outcome O1 such that C1 produces O1 and O1 is a stateOf BO1.

**Association Name:** object_1  **Association Type:** Association Stereotype: <shortcut>

**Source Class:** ProcurementOffering [0..*]  **Target Class:** AbstractBusinessObject [0..*]

**Definition:** The object shortcut association asserts that the ProcurementOffering incorporates unspecified Outcomes describing the states of AbstractBusinessObjects.

**Usage:** This association allows the business architect to omit the Outcome in the procurement of some AbstractBusinessObjects for use by the Business when those Outcomes are obvious or irrelevant to the purposes of the analysis that is using the business architecture model.

**Constraint:** Let PO1 be a ProcurementOffering and BO1 be a BusinessObject associated by o1 an "object" association. Then PO1 should incorporate ProcurementOutcomes {POj} that represent either the change of ownership of BO1 or the establishment of a limited right to use BO1.
Source Class: Offering  |  Target Class: AbstractBusinessObject  |

Association Name: recordedAs  Association Type: Association  Stereotype: «class»  
Source Class: Outcome  [0..*]  Target Class: AbstractBusinessObject  [0..*]

Association Name: stateOf  Association Type: Association  Stereotype: «class»  
Source Class: Outcome  [0..*]  Target Class: AbstractBusinessObject  [0..*]

The "state_of" meta-association applies a state to an AbstractBusinessObject instance. For example, a passenger may be transported from one location to another by a Capability, and the Outcome resulting from the Capability execution represents the fact that the passenger is now in the destination location.

Association Name:  Association Type: Generalization  Stereotype:  
Source Class: InformationItem  []  Target Class: AbstractBusinessObject  []

Association Name:  Association Type: Generalization  Stereotype:  
Source Class: BusinessObject  []  Target Class: AbstractBusinessObject  []

Association Name: object_0  Association Type: Association  Stereotype: «shortcut»  
Source Class: MerchandiseOffering  [0..*]  Target Class: AbstractBusinessObject  [0..*]

Definition: The object association represents a shortcut relationship between a MerchandiseOffering and a BusinessObject or InformationItem offered for sale or lease to the Customer.

Usage: This shortcut implies that there is an unspecified MerchandiseOutcome of the AbstractBusinessObject that would describe the terms of ownership/use incorporated in the MerchandiseOffering.

Constraint: Let MOFi be a MerchandiseOffering and BOi be a BusinessObject associated by o1 an "object" association. Then MOFi should incorporate MerchandiseOutcomes {MOj} that represent either the change of ownership of BOi or the establishment of a limited right to use BOi.

7.3.2.5  Class Name: AbstractCapability  Class Type: Class  Stereotype:  
Base Classes: AbstractOperatingModel, BACMPlainEntity
Definition: AbstractCapability is not intended to represent a business concept. It is a metamodeling device to provide relationships to Capability and CapabilityBehavior that would otherwise be duplicated.

Usage: The AbstractCapability metaclass has two concrete specializations: Capability and CapabilityBehavior. Only the specializations can be instanced in models.

AbstractCapability provides the following to its concrete specializations:
1. to represent the production of an Outcome;
2. to represent the need for an Outcome;
3. to represent the ability of an InformationItem to inform the behavior of a Capability and/or CapabilityBehavior;
4. to represent the ability of a CapabilityImplementation to implement a Capability and/or a CapabilityBehavior;
5. to represent the notion that a BusinessObject and/or an InformationItem scopes a Capability and/or a CapabilityBehavior.

7.3.2.5.1 Attributes, Methods and Connectors:

Association Name:  Association Type: Generalization  Stereotype:  
Source Class: AbstractCapability  []  Target Class: AbstractOperatingModel  []

Association Name: needs_0  Association Type: Association  Stereotype: «class»  
Source Class: AbstractCapability  [0..*]  Target Class: Outcome  [0..*]
**Definition:** The needs association represents the assertion that a Capability and/or CapabilityBehavior needs, desires or requires a particular Outcome representing a state of an BusinessObject or InformationItem.

**Association Name:** produces_0  **Association Type:** Association  **Stereotype:** «class»  
**Source Class:** AbstractCapability [0..*]  **Target Class:** Outcome [0..*]  
**Definition:** The produces association represents that a Capability and/or CapabilityBehavior may produce the Outcome.

**Association Name:** Association Type: Generalization  **Stereotype:**  
**Source Class:** AbstractCapability []  **Target Class:** BACMPlainEntity []

**Association Name:** ofCapability Association Type: Association  **Stereotype:**  
**Source Class:** Role [0..*]  **Target Class:** AbstractCapability [0..1]  
**Definition:** The ofCapability leg of the Role association links the Role to the AbstractCapability.

**Association Name:** Association Type: Generalization  **Stereotype:**  
**Source Class:** Capability []  **Target Class:** AbstractCapability []

**Association Name:** object_2 Association Type: Association  **Stereotype:** «shortcut»  
**Source Class:** ServiceOffering [0..*]  **Target Class:** AbstractCapability [0..*]  
**Definition:** the object shortcut association designates an AbstractCapability possessed by theBusiness that is intended to produce the ServiceOutcome incorporated into the ServiceOffering.  
**Constraint:** Let SOI be a ServiceOffering and C1 be a Capability that is associated by o1 an object association. Then there should exist a ServiceOutcome SO1 such that SO1 is incorporated in SOI and SO1 is produced by C1.

**Association Name:** scopes Association Type: Association  **Stereotype:** «shortcut»  
**Source Class:** AbstractBusinessObject [0..1]  **Target Class:** AbstractCapability [0..1]  
**Definition:** The scopes shortcut association allows a Capability and/or CapabilityBehavior to be associated with some BusinessObjects and/or an InformationItems without defining Outcomes produced or needed by the Capability and/or CapabilityBehavior.  
**Usage:** The modeler may elect to subsequently define such Outcomes, which must be consistent with the constraint specified by the scopes shortcut association.  
**Constraint:** Let BO1 be a BusinessObject and C1 be a Capability that are associated by scopes s1. Then there should exist the model an Outcome O1 such that C1 produces O1 and O1 is a stateOf BO1.

**Association Name:** implements_5 Association Type: Association  **Stereotype:** «shortcut»  
**Source Class:** CapabilityImplementation [0..*]  **Target Class:** AbstractCapability [0..*]  
**Definition:** The implements association represents a relationship meaning that the CapabilityImplementation provides Performers and Resources to implement a Capability or CapabilityBehavior.  
**Usage:** The implements association is a shortcut linking a CapabilityImplementation to a Capability or CapabilityBehavior. It carries a constraints that Performers and Resources in the CapabilityImplementation should be assigned to Roles of the Capability or CapabilityBehavior.  
**Constraint:** Given a CapabilityImplementation C1 and a Capability C1, if an implements association I1 exists between C1 and C1, then paths should exist between C1 and C1 such that for some subset of the Performers {Pi} aggregated by C1 and some subset of the Resources {Ri} aggregated by C1, the Pi are assignTo PerformerRoles PRj and assignTo ResourceRoles RRj ofCapability C1

**Association Name:** require_1 Association Type: Association  **Stereotype:** «class»  
**Source Class:** Initiative [0..*]  **Target Class:** AbstractCapability [0..*]  
**Definition:** The require association represents that a Capability and/or CapabilityBehavior is required for performance of the Initiative.  
**Usage:** Definition of this association in an M1 level model allows the business architect to record and analyze such requirements.

**Association Name:** Association Type: Generalization
Source Class: CapabilityBehavior  Target Class: AbstractCapability

Association Name: implements_0  Association Type: Association  Stereotype: «shortcut»
Source Class: AbstractProcess  Target Class: AbstractCapability

Definition: The implements shortcut represents that a CapabilityBehavior and an AbstractProcess have related Outcomes.
Usage: It could also be justified by a common Performer playing a role in the CapabilityBehavior and the AbstractProcess.
Constraint: Let P1 be a Process and C1 be a capability associated by an implements association. Then there should exist Outcomes O1 and O2 such that O1 is produced by (needed by) C1 and O2 is output (input) by P1 and O1 and O2 are related such that they are the same Outcome or one is in the extended aggregation of the other or one is the extended specialization of the other or any chain of relationships connecting the two where the chain consists exclusively of being aggregated by or being a specialization of the predecessor Outcome.

Association Name: informs_0  Association Type: Association  Stereotype: «class»
Source Class: InformationItem  Target Class: AbstractCapability

Definition: The informs association represents the influence of information (represented by InformationItem) on a Capability or a CapabilityBehavior.
Usage: Information, such as weather, production targets, and results of a business analysis project will change how a business behaves and how a Capability or CapabilityBehavior performs.

Association Name: require_0  Association Type: Association  Stereotype: «class»
Source Class: Means  Target Class: AbstractCapability

Definition: The require association represents that a Capability and/or CapabilityBehavior is required for performance of the Means.
Usage: Definition of this association in an M1 level model allows the business architect to record and analyze such requirements.

7.3.2.6  Class Name: BusinessObject  Class Type: Class  Stereotype: 
Base Classes: AbstractBusinessObject

Definition: BusinessObject represents a tangible thing that is of significance to a business.
Usage: BusinessObjects may also overlap with other classes in the model; for example a BusinessObject may also be a Resource used by a Capability.
Typically, the BusinessObject represents tangible things that are acted on by the Capabilities of a business to create a new Outcome that defines a new state of the BusinessObject. An assembly robot may be a Performer associated with an assembly Capability. The same assembly robot may be a BusinessObject when it is no longer needed and is sold.

7.3.2.6.1 Attributes, Methods and Connectors:

Association Name: contains Association Type: Association  Stereotype: «class»
Source Class: BusinessObject  Target Class: System

Definition: The contains association represents that BusinessObjects may contain System.
Usage: In some cases, a BusinessObject and a System may represent different aspects of the same entity; since meta-classes in this meta-model are not assumed disjoint, an instance may have both BusinessObject and System as metaclasses. However, a BusinessObject may contain several Systems and other BusinessObjects as well. In this case, the Systems are not aspects of the primary BusinessObject, and the contains association allows the architect to represent this. An example of this latter case is a primary BusinessObject that is a computer and the System is a software package hosted on that computer (along with other software packages). The software package may be an instance of a System and also an instance of a BusinessObject (i.e. the code).

Association Name: realize Association Type: Association  Stereotype: «class»
Source Class: BusinessObject  Target Class: InformationItem

Definition: The realizes association represents that a BusinessObject realizes an InformationItem.
The is about association represents the coupling of information to a business object where the information represents some aspects or characterizations of the business object. For example, a business object that is a car may have a "digital twin", i.e. a collection of information that describes the state of being of the car. Such a "digital twin" is more accessible for controlling and auditing the business than the actual car.

**7.3.2.7 Class Name: Capability Class Type: Class Stereotype:**

**Base Classes:** AbstractCapability

**Definition:** Capability represents generalization over variations in behavior and variations in structure applied to the behavior where the same general Outcome is produced by the behavior. A Capability represents the ability a business has to produce an Outcome without specifying how that Outcome is produced.

**Usage:** Capability is defined in this way to allow executives to analyze variation in business behaviors and structures that all produce the same or similar outcomes.

In addition, observing problems or successes that recur in most or all of the variations of a Capability is a clue that the business has a systemic problem with respect to the capability. For example, if all behavior variants and implementations of a Capability are underperforming, then one might wish to understand why.

Capabilities may be decomposed in a strict hierarchy, but are not allowed to be specialized. The CapabilityBehavior that delivers a Capability is used to represent behavioral variants of a Capability.

A Capability may be implemented by a CapabilityImplementation, a collection of Resources and Performers that are assigned Roles in the Capability.

The modeler may use any of the following patterns:

1. Capability is defined without CapabilityBehaviors or CapabilityImplementations;
2. Capability is defined with CapabilityImplementations annotated with proposed resources and performers but without Roles, Resources and Performers;
3. Capability is defined with Roles, CapabilityImplementations, Performers, Resources where the Performers and Resources are aggregated to the CapabilityImplementation and are assigned to Roles of the Capability;
4. Capability is defined as in 3. and CapabilityBehaviors are defined delivering the Capability with Role assignments to CapabilityBehavior compatible with the assignments to Capability Roles;
5. Capability is defined with delivering CapabilityBehaviors but no CapabilityImplementation;
6. Capability is defined with Roles and delivering CapabilityBehaviors are defined with consistent Roles;
7. All other configurations are disallowed.

**Constraint:** Capability instances may own other Capability instances but may not aggregate or generalize them.

**7.3.2.7.1 Attributes, Methods and Connectors:**

**Association Name:** supports **Association Type:** Association **Stereotype:** «class**

**Source Class:** Capability [0..*] **Target Class:** ValueStreamStage [0..*]

**Definition:** The supports association represents the relationship between a Capability and a ValueStreamStage that means that the Capability is needed in the ValueStreamStage.

**Usage:** For example, an important stage in the creation of value for a manipulation puzzle such as Rubik's Cube is the production of a manufacturable design of the puzzle. A failure here can result in a puzzle that cannot be manufactured or is not attractive to purchasers.

Outcomes providing value are:
- a positive manufacturability review;
- a positive customer reaction in a focus group.
The Capabilities needed to produce these Outcomes are: product design, manufacturability analysis, focus group management. For this example, the previous three Capability instances would be associated with the "Design Ready" ValueStreamStage.

**Association Name**: Association **Type**: Generalization **Stereotype**:
**Source Class**: Capability [[]] **Target Class**: AbstractCapability [[]]

**Association Name**: delivers **Association Type**: Association **Stereotype**: «class»
**Source Class**: CapabilityBehavior [0..*] **Target Class**: Capability [0..1]
**Definition**: The delivers association represents a CapabilityBehavior that produces or is intended to produce Outcomes that satisfy the Outcomes produced by the Capability.
**Usage**: A CapabilityBehavior that delivers a Capability must provide at least the set of Roles provided by the Capability.

**Association Name**: Association **Type**: NoteLink **Stereotype**:
**Source Class**: Constraint [[]] **Target Class**: Capability [[]]

---

7.3.2.8 **Class Name**: CapabilityBehavior  **Class Type**: Class  **Stereotype**:
**Base Classes**: AbstractAction, AbstractCapability, APCICB
**Definition**: CapabilityBehavior represents a behavior description or specification, such as process diagrams, procedures manuals and other means of recording and publishing expected business practices.
**Usage**: CapabilityBehaviors also represent rules, regulations and policies that constrain behavior, whether imposed by statute, regulators or business executives.

CapabilityBehaviors deliver a Capability, indicating that the set CapabilityBehaviors associated to a Capability are variant ways of producing the same or similar Outcomes. CapabilityBehaviors may have associated Roles. These Roles define how Performers and Resources may participate in the described or specified behavior. CapabilityBehavior is a subtype of AbstractCapability and inherits associations with the Outcomes of Capabilities. These associations represent the ability of a behavior to produce an outcome. The Outcomes produced by a CapabilityBehavior are usually more specific than Outcomes produced by the Capability. Often the Outcome of a CapabilityBehavior will include side-effects that result from the particular behavior, such as resources consumed in executing the behavior or time taken by the execution. CapabilityBehaviors are not decomposable, but may be associated with Processes, which are decomposable.

7.3.2.8.1 **Attributes, Methods and Connectors**:

**Association Name**: delivers **Association Type**: Association **Stereotype**: «class»
**Source Class**: CapabilityBehavior [0..*] **Target Class**: Capability [0..1]
**Definition**: The delivers association represents a CapabilityBehavior that produces or is intended to produce Outcomes that satisfy the Outcomes produced by the Capability.
**Usage**: A CapabilityBehavior that delivers a Capability must provide at least the set of Roles provided by the Capability.

**Association Name**: Association **Type**: Generalization **Stereotype**:
**Source Class**: CapabilityBehavior [[]] **Target Class**: AbstractCapability [[]]

**Association Name**: Association **Type**: Generalization **Stereotype**:
**Source Class**: CapabilityBehavior [[]] **Target Class**: APCICB [[]]

**Association Name**: Association **Type**: Generalization **Stereotype**:
**Source Class**: CapabilityBehavior [[]] **Target Class**: AbstractAction [[]]
7.3.2.9 **Class Name:** CapabilityImplementation  **Class Type:** Class  **Stereotype:**

*Base Classes:* AbstractOperatingModel, APCICB, BACMPPlainEntity

*Definition:* The CapabilityImplementation represents a collection of Resources and Performers that may be used to implement a Capability or CapabilityBehavior (see the Roles diagram).

*Usage:* The Resources and Performers are optional; the modeler may create instances of CapabilityImplementation annotated with a description of proposed or planned resources and performers and subsequently add the Performers and Resources.

### 7.3.2.9.1 Attributes, Methods and Connectors:

**Association Name:** Association Type: Generalization Stereotype:

*Source Class:* CapabilityImplementation  []  *Target Class:* BACMPPlainEntity  []

**Association Name:** implements_3 Association Type: Association Stereotype: «class»

*Source Class:* CapabilityImplementation  [0..*]  *Target Class:* Initiative  [0..*]

*Definition:* The implements association represents an assertion that one or more CapabilityImplementations are required to perform the initiative.

*Usage:* Definition of this association in an M1 level model allows the business architect to record that specific CapabilityImplementations are needed to perform the Initiative.

**Association Name:** aggregates_1 Association Type: Association Stereotype:

*Source Class:* CapabilityImplementation  [0..*]  *Target Class:* Performer  [0..*]

**Association Name:** implements_5 Association Type: Association Stereotype: «shortcut»

*Source Class:* CapabilityImplementation  [0..*]  *Target Class:* AbstractCapability  [0..*]

*Definition:* The implements association represents a relationship meaning that the CapabilityImplementation provides Performers and Resources to implement a Capability or CapabilityBehavior.

*Usage:* The implements association is a shortcut linking a CapabilityImplementation to a Capability or CapabilityBehavior. It carries a constraints that Performers and Resources in the CapabilityImplementation should be assigned to Roles of the Capability or CapabilityBehavior.

*Constraint:* Given a CapabilityImplementation CI1 and a Capability C1, if an implements association I1 exists between CI1 and C1, then paths should exist between CI1 and C1 such that for some subset of the Performers {Pi} aggregated by CI1 and some subset of the Resources {Ri} aggregated by C1, the Pi are assignTo PerformerRoles PRj and assignTo ResourceRoles RRj ofCapability C1.

**Association Name:** Generalization Stereotype:

*Source Class:* CapabilityImplementation  []  *Target Class:* APCICB  []

**Association Name:** Generalization Stereotype:

*Source Class:* CapabilityImplementation  []  *Target Class:* AbstractOperatingModel  []

**Association Name:** aggregates_2 Association Type: Association Stereotype:

*Source Class:* CapabilityImplementation  [0..*]  *Target Class:* Resource  [0..*]

**Association Name:** assignTo_3 Association Type: Association Stereotype: «shortcut»

*Source Class:* Role  [0..*]  *Target Class:* CapabilityImplementation  [0..*]
**Definition:** The assignment shortcut association represents that a CapabilityImplementation provides Performers and Resources that can be assigned to ResourceRoles and PerformerRoles respectively of a Capability, CapabilityBehavior, Process or Activity.

**Constraint:** Let CI1 be a CapabilityImplementation and PR1 be a PerformerRole, then some subset of the Performers \{pj\} aggregated by CI1 are assignTo PR1. Let CI1 be a CapabilityImplementation and RR1 be a ResourceRole, then some subset of the Performers \{pj\} aggregated by CI1 are assignTo RR1.

### 7.3.2.10 Class Name: InformationItem  Class Type: Class  Stereotype:

**Base Classes:** AbstractBusinessObject  
**Definition:** The InformationItem represents a kind of information.  
**Usage:** The same InformationItem may represent a thought or piece of knowledge and a physical manifestation of that thought or knowledge as a document or a dataset.

#### 7.3.2.10.1 Attributes, Methods and Connectors:

- **Association Name:** informs_1  
  **Association Type:** Association  
  **Stereotype:** "class"  
  **Source Class:** InformationItem \[0..*\]  
  **Target Class:** AbstractProcess \[0..*\]  
  **Definition:** The informs association represents the influence of information (represented by InformationItem) on a Process or Activity.  
  **Usage:** Information, such as weather, production targets, and results of a business analysis project will change how a business behaves and how a Process or Activity performs.

- **Association Name:** isAbout  
  **Association Type:** Association  
  **Stereotype:** "class"  
  **Source Class:** InformationItem \[0..*\]  
  **Target Class:** BusinessObject \[0..*\]  
  **Definition:** The is about association represents the coupling of information to a business object where the information represents some aspects or characterizations of the business object. For example, a business object that is a car may have a "digital twin", i.e. a collection of information that describes the state of being of the car. Such a "digital twin" is more accessible for controlling and auditing the business than the actual car.

- **Association Name:** Association  
  **Association Type:** Generalization  
  **Stereotype:**  
  **Source Class:** InformationItem []  
  **Target Class:** AbstractBusinessObject []

- **Association Name:** informs_0  
  **Association Type:** Association  
  **Stereotype:** "class"  
  **Source Class:** InformationItem \[0..*\]  
  **Target Class:** AbstractCapability \[0..*\]  
  **Definition:** The informs association represents the influence of information (represented by InformationItem) on a Capability or a CapabilityBehavior.  
  **Usage:** Information, such as weather, production targets, and results of a business analysis project will change how a business behaves and how a Capability or CapabilityBehavior performs.

- **Association Name:** realize  
  **Association Type:** Association  
  **Stereotype:** "class"  
  **Source Class:** BusinessObject \[0..*\]  
  **Target Class:** InformationItem \[0..*\]

### 7.3.2.11 Class Name: ObjectRelation  Class Type: Class  Stereotype: "association"

**Base Classes:** AbstractOperatingModel  
**Definition:** ObjectRelation represents any relationship of any arity among BusinessObjects and InformationItems.  
**Usage:** The architect may use ObjectRelation to indicate that two BusinessObjects are joined together or that one BusinessObject is part of another.
7.3.2.11.1 Attributes, Methods and Connectors:

 Association Name: related_1  Association Type: Association  Stereotype:  
 Source Class: ObjectRelation [0..*]  Target Class: AbstractBusinessObject [0..*]  
 Definition: The related leg of the ObjectRelation association links an ObjectAssociation to 
 BusinessObjects and/or InformationItems that participate in the ObjectAssociation.  
 Usage: The ObjectRelation association does not have a fixed number of legs. The related leg may be 
 instanced multiple times as long as the name given to each instance is distinct. The related leg may be given 
 a label that defines a role the BusinessObject or InformationItem plays in the ObjectAssociation. 

 Association Name:  Association Type: Generalization  Stereotype:  
 Source Class: ObjectRelation [ ]  Target Class: AbstractOperatingModel [ ]

7.3.2.12 Class Name: Outcome  Class Type: Class  Stereotype:  
 Base Classes: AbstractOperatingModel, AbstractResult, BACMPlainEntity 
 Definition: An Outcome represents a fact or collection of facts about an experienced state of affairs pertaining to one or 
 more BusinessObjects and/or InformationItems. Outcomes are produced/needed by land outputs/inputs of 
 AbstractProcesses.  
 Usage: For example, a Capability to attach wheels to a vehicle being manufactured would require that a vehicle without 
 wheels be available and that wheels be available. This requirements would be modeled as two Outcomes: 
 1. A vehicle without wheels is available to the Capability, and  
 2. A set of wheels is available to the Capability.  
 The result of the Capability is another Outcome in which the wheels are no longer separate but are attached to the 
 vehicle.  
 Separating the state of a BusinessObject or InformationItem from the BusinessObject or InformationItem allows the 
 model to represent many possible states of the BusinessObject or InformationItem and associate each state with the 
 Capabilities and/or CapabilityBehaviors that produce the states.  
 Outcome and its AbstractBusinessObjects must represent a single, consistent set of facts whether viewed from the 
 capability perspective or the process perspective. However, the facts represented by a Outcome may not be at the same 
 level of detail when viewed in a capability perspective as when viewed in a process perspective. For example, a process 
 perspective may represent the wheel assembly activities in greater detail, specifying the additional tools and parts needed 
 to attach the wheels to the vehicle with intermediate Outcomes representing the stages of mounting the wheels to the 
 hubs, attaching the nuts to the hub bolts, and tightening them to the required torque specification. The beginning and end 
 of this sequence of Outcomes are the same in the process perspective and in the capability perspective. Other semantic 
 relationships provided for Outcome are generalization and aggregation.

7.3.2.12.1 Attributes, Methods and Connectors:

 Association Name:  Association Type: Generalization  Stereotype:  
 Source Class: Outcome [ ]  Target Class: AbstractResult [ ]

 Association Name:  Association Type: Generalization  Stereotype:  
 Source Class: Outcome [ ]  Target Class: AbstractOperatingModel [ ]

 Association Name: recordedAs  Association Type: Association  Stereotype: «class»  
 Source Class: Outcome [0..*]  Target Class: AbstractBusinessObject [0..*] 

 Association Name: stateOf  Association Type: Association  Stereotype: «class»  
 Source Class: Outcome [0..*]  Target Class: AbstractBusinessObject [0..*]
The "state_of" meta-association applies a state to an AbstractBusinessObject instance. For example, a passenger may be transported from one location to another by a Capability, and the Outcome resulting from the Capability execution represents the fact that the passenger is now in the destination location.

**Association Name:** Generalization **Stereotype:**
**Source Class:** Outcome [ ]  **Target Class:** BACMPlainEntity [ ]

**Association Name:** triggers **Association Type:** Association **Stereotype:** «shortcut»
**Source Class:** Outcome [0..*]  **Target Class:** ValueStreamStage [0..*]
**Definition:** The triggers association represents that the Outcome allows or initiates the ValueStreamStage.  
**Usage:** It is often useful in analysis to record the Outcomes that constitute the important beginning events of a ValueStreamStage. The triggers association allows the architect to record these relationships.

**Constraint:** Let O1 be an Outcome experienced at a Touchpoint T1 and VSS1 be a ValueStreamStage where O1 triggers VSS1, then there should exist a ValueItem VI1 that values O1 and is produced by VSS1.

**Association Name:** output **Association Type:** Association **Stereotype:** «class»
**Source Class:** AbstractProcess [0..*]  **Target Class:** Outcome [0..*]
**Definition:** The output association represents that the AbstractProcess outputs the Outcome.  
**Usage:** The output association in the process perspective corresponds to the produces association in the capability perspective. While it is possible that the same Outcome is output from a process and produced by a capability, it will usually be the case that a process outputs an Outcome that is related by generalization or aggregation (or another relation between Outcomes) to an Outcome produced by a capability. The process and capability in this case are semantically related by the relationship between their Outcomes. For example, a CustomerInformationManagement Capability may produce CustomerInformation_is_correct Outcomes. A process that updates the CustomerAddress (a component of CustomerInformation) may produce CustomerAddress_is_current and CustomerAddress_is_correct Outcomes, that are related to the other Outcomes by aggregation.

**Association Name:** Generalization **Stereotype:**
**Source Class:** OutsourcedServiceOutcome [ ]  **Target Class:** Outcome [ ]

**Association Name:** values **Association Type:** Association **Stereotype:** «class»
**Source Class:** ValueItem [0..*]  **Target Class:** Outcome [0..*]
**Definition:** The values association links a ValueItem to an Outcome and provides a valuation of that Outcome. An Outcome may have several ValueItems, reflecting the ways in which different stakeholders perceive the Outcome. Likewise, a ValueItem may value multiple Outcomes that must be valued as a group.

**Usage:** The Outcome may be present in the business architecture model without an associated ValueItem, but ValueItems may not exist without being associated to an Outcome.

**Association Name:** experiences **Association Type:** Association **Stereotype:** «class»
**Source Class:** Touchpoint [0..*]  **Target Class:** Outcome [0..*]
**Definition:** The experiences relation represents a relationship between an Outcome and a Touchpoint, meaning that the Customer will experience the Outcome at the Touchpoint.  
**Usage:** A Touchpoint experiences an Outcome:
1. when that Outcome is provided as a service or
2. when the Outcome is associated with acceptance of the ProductOffering (e.g. the customer is happy with the contract of sale), or
3. when the customer receives information that resolves a question, or
4. when the customer makes use of a business object that is provided as an Outcome of an exchange transaction

**Association Name:** needs **Association Type:** Association **Stereotype:** «class»
**Source Class:** AbstractCapability [0..*]  **Target Class:** Outcome [0..*]
**Definition:** The needs association represents the assertion that a Capability and/or CapabilityBehavior needs, desires or requires a particular Outcome representing a state of an BusinessObject or InformationItem.
Association Name: produces_0  Association Type: Association  Stereotype: «class»
Source Class: AbstractCapability [0..*]  Target Class: Outcome [0..*]
Definition: The produces association represents that a Capability and/or CapabilityBehavior may produce the Outcome.

Association Name: related_0  Association Type: Association  Stereotype:
Source Class: OutcomeRelation [0..*]  Target Class: Outcome [0..*]
Definition: The relatedOutcome leg of the OutcomeRelation association identifies an Outcome that is related to one or more other Outcomes.
Usage: The OutcomeRelation association does not have a fixed number of legs when instanced. The architect may define any number of instances of the relatedOutcome leg when instancing the OutcomeRelation as long as each leg is given a unique name.

Association Name: input  Association Type: Association  Stereotype: «class»
Source Class: AbstractProcess [0..*]  Target Class: Outcome [0..*]
Definition: The input association represents that the AbstractProcess inputs (requires or can use) the Outcome.
Usage: The input association in the process perspective corresponds to the needs association in the capability perspective. While it is possible that the same Outcome is input to a process and needed by a capability, it will usually be the case that a process inputs an Outcome that is related by generalization or aggregation (or another relation between Outcomes) to an Outcome needed by a capability. The process and capability in this case are semantically related by the relationship between their Outcomes. For example, a CustomerInformationManagement Capability may need CustomerInformation_change_pending Outcome. A process that updates the CustomerAddress (a component of CustomerInformation) may input CustomerAddress_change_pending Outcome, that is related to the other Outcome by aggregation.

Association Name: incorporates_0  Association Type: Association  Stereotype: «class»
Source Class: ProductOffering [0..*]  Target Class: Outcome [0..*]
Definition: The incorporates association represents that an Outcome is included in a ProductOffering.
Usage: It may be implied that the BusinessObject whose state is represented by the Outcome is also included in the ProductOffering. In the case of a service offering, the Outcome instance represents the intended result of performing the capability as a service for a customer (as opposed to performing the capability for the immediate benefit of the business).

Association Name:  Association Type: Generalization  Stereotype:
Source Class: ProcurementOutcome []  Target Class: Outcome []

Association Name:  Association Type: Generalization  Stereotype:
Source Class: ServiceOutcome []  Target Class: Outcome []

7.3.2.13 Class Name: OutcomeRelation  Class Type: Class  Stereotype: «association»
Base Classes: AbstractOperatingModel
Definition: OutcomeRelation represents any kind of semantic relationship between Outcomes.
Usage: The architect may create instances of any arity to define semantic relationships between Outcomes. For example, two Outcomes may be specified as alternative that cannot both be produced by a Capability or Process in a single execution.

7.3.2.13.1 Attributes, Methods and Connectors:
**Association Name**: OutcomeRelation  **Association Type**: Generalization  **Stereotype**: 
**Source Class**: OutcomeRelation  **Target Class**: AbstractOperatingModel

**Association Name**: related_Outcome  **Association Type**: Association  **Stereotype**: 
**Source Class**: OutcomeRelation  **Target Class**: Outcome

**Definition**: The relatedOutcome leg of the OutcomeRelation association identifies an Outcome that is related to one or more other Outcomes.

**Usage**: The OutcomeRelation association does not have a fixed number of legs when instanced. The architect may define any number of instances of the relatedOutcome leg when instancing the OutcomeRelation as long as each leg is given a unique name.

### 7.3.2.14 Class Name: PerformerRole  **Class Type**: Class  **Stereotype**: «association»

**Base Classes**: Role

**Definition**: PerformerRole represents skills, knowledge and willingness to use these in the production of the Outcomes of a Capability.

**Usage**: PerformerRole represents roles that must be fulfilled by human or automation actors.

#### 7.3.2.14.1 Attributes, Methods and Connectors:

**Association Name**: assignTo_PerformerRole  **Association Type**: Association  **Stereotype**: 
**Source Class**: PerformerRole  **Target Class**: Performer

**Definition**: The assignment leg of the PerformerRole association represents that a Performer is assigned to the PerformerRole.

**Association Name**: Generalization  **Stereotype**: 
**Source Class**: PerformerRole  **Target Class**: Role

### 7.3.2.15 Class Name: ResourceRole  **Class Type**: Class  **Stereotype**: «association»

**Base Classes**: Role

**Definition**: ResourceRole represents the set of roles that must be fulfilled by business entities that are passive participants in the Capability, CapabilityBehavior, Process or Activity. This includes tools, locations and materials that are used in the behavior but do not become incorporated into the Outcome of the behavior. Any materials or entities that are incorporated into a BusinessObject or InformationItem whose Outcomes are produced by the Capability or CapabilityBehavior should be represented as BusinessObjects or InformationItems associated with Outcomes needed by the Capability and not represented as Resources in this context.

**Usage**:

#### 7.3.2.15.1 Attributes, Methods and Connectors:

**Association Name**: assignTo_ResourceRole  **Association Type**: Association  **Stereotype**: 
**Source Class**: ResourceRole  **Target Class**: Resource

**Definition**: The assignTo leg of the ResourceRole association represents that a Resource has been assigned to a ResourceRole.
7.3.2.16 **Class Name:** Role  **Class Type:** Class **Stereotype:** «association»

**Base Classes:** AbstractOperatingModel

**Definition:** Role represents a specified way for an entity to participate in producing the Outcome of a Capability or a Process. However, only the concrete subclasses of Role may be used in a model.

**Usage:** Role is an abstract association meta-class used to model relationships between Performers and Resources and Capabilities and Processes. It represents how Performers and Resources participate in behavior descriptions as represented by CapabilityBehaviors and/or in Capabilities. The Role meta-class is stereotyped as an association and its concrete instances are effectively class associations.

Specifically, the Role meta-class acts as an n-ary association with three predominant patterns:

1. A Capability is associated with a Performer;
2. A CapabilityBehavior is associated with a Performer, or a choice of an OrgUnit or a System;
3. A CapabilityImplementation is associated with a CapabilityBehavior and a choice of an OrgUnit or a System.

These three patterns represent:

1. An abstract view of the business capability with detail added by the Role instance indicating the type of activity to be performed. Since a Capability may have multiple associated Roles, this implies that the Capability incorporates multiple activities.
2. An intermediate view of the business used in planning where details about the specific behaviors of a capability and the type of performer entity (OrgUnit or System) are specified, but the actual or planned assignment of real OrgUnits or Systems has not occurred.
3. A more detailed planning/implementation view of the business in which specific performers and resources have been or are planned to be allocated to a Capability and its CapabilityBehaviors by way of a set of CapabilityImplementations. Neither ResourceRoles nor PerformerRoles may exist without being linked to a Capability or a CapabilityBehavior or a Process or an Activity with the role link.

A Capability and a CapabilityBehavior may share a Role, but an assignment to that Role will be the same for both the Capability and the CapabilityBehavior. To indicate that a CapabilityBehavior and a Capability have related roles, the modeler should create a specialization of the CapabilityRole for each CapabilityBehavior that delivers the Capability and link the specialized Role to the CapabilityBehavior.

A Process and an Activity may not share a Role.

A Role may be shared between a Capability and/or a CapabilityBehavior, and either a Process or an Activity. In this case, any assignment to the Role is an assignment to both the Capability/CapabilityBehavior and the Process/Activity PerformerRoles and ResourceRoles may be linked to CapabilityImplementations with the assignment shortcut association. Performers and Resources aggregated in the CapabilityImplementation should be assigned to these roles.

### 7.3.2.16.1 Attributes, Methods and Connectors:

**Association Name:** assignTo_3  **Association Type:** Association  **Stereotype:** «shortcut»

**Source Class:** Role [0..*]  **Target Class:** CapabilityImplementation [0..*]

**Definition:** The assignment shortcut association represents that a CapabilityImplementation provides Performers and Resources that can be assigned to ResourceRoles and PerformerRoles respectively of a Capability, CapabilityBehavior, Process or Activity.

**Constraint:** Let C1 be a CapabilityImplementation and PR1 be a PerformerRole, then some subset of the Performers \( \{pj\} \) aggregated by C1 are assignTo PR1. Let C1 be a CapabilityImplementation and RR1 be a ResourceRole, then some subset of the Performers \( \{pj\} \) aggregated by C1 are assignTo RR1.

**Association Name:** ofCapability  **Association Type:** Association  **Stereotype:**

**Source Class:** Role [0..*]  **Target Class:** AbstractCapability [0..1]

**Definition:** The ofCapability leg of the Role association links the Role to the AbstractCapability.

**Association Name:** :GeneralizationStereotype:  **Association Type:** Generalization  **Stereotype:**

**Source Class:** Role []  **Target Class:** AbstractOperatingModel []

**Association Name:** ofProcess  **Association Type:** Association  **Stereotype:**

**Source Class:** Role [0..*]  **Target Class:** AbstractProcess [0..1]

**Definition:** The ofProcess leg of the Role association links a PerformerRole or ResourceRole to a Process or Activity.
7.3.3 Package: Customer

7.3.3.1 Diagram: Customer
The Customer diagram expresses the abstract syntax of elements that collectively describe the customer, the customer's interactions with the business or its products and services, and the customer's state of mind at these interactions. The CustomerJourney is a reusable tree that decomposes a journey into CustomerJourneyStages and CustomerJourneyStages into Touchpoints. This tree is reusable in that it allows multiple Customers to take the same CustomerJourney. It also allows a Customer to take different Journeys.

The CustomerSegment describes customer characteristics and/or state of mind at CustomerJourneyStages and Touchpoints of a CustomerJourney. The CustomerSegments are associated with CustomerJourneyStages and Touchpoints, but are effectively owned by the Customer taking the CustomerJourney. Separating the customer characteristics and state of mind from the CustomerJourneyStage and Touchpoint allows the CustomerJourneyStage and Touchpoint to be associated with different Customers.

Touchpoint differs from CustomerJourneyStage in experiencing Outcomes that may be incorporated in a ProductOffering. Such Outcomes might include the sale of a product item, the performance of a service or the customer's use of a product item after the sale. At each such Touchpoint, the customer characteristics and state of mind are described in a CustomerSegment.
The ValueFit diagram defines abstract syntax for an analysis of how well the ValuePropositions meet the Customer expectations. The ValueCharacteristic holds an assessment of the fit of the ValueProposition (and its constituent ValuePropositions and ValueItems) to the CustomerSegments associated with the targeted Customer.
7.3.3.3 **Diagram: ValueStream**

Commented [JR30]: Issue BACM-83 resolved by BACM-91
Issue BACM-91 resolved by BACM-92 - produces now shortcut

ValueStream
- constraints (self.aggregator->isEmpty()) (self.moreGeneral->isEmpty())
- owns 1 (ordered)

ValueStreamStage
- constraints (self.aggregator->isEmpty()) (self.moreGeneral->isEmpty())
- owns 1 (ordered)

ValueItem
- produces 0..* (value «class»)
- owns 0..1 (ordered)

ValueProposition
- produces 1 (target «class»)
- incorporates 0..1 (value «class»)

Customer
- produces 0..* (shortcut «class»)
- owns 0..* (ordered)

Product::ProductOffering
- produces 0..* (shortcut «class»)
- aggregates 0..*

Capability::Outcome
- target 0..*
- incorporates 0..*
The ValueStream diagram defines abstract syntax for models incorporating ValueStreams. The ValueStream owns ValueStreamStages representing business significant stages in the composition of value for a customer. The ValueStream itself produces ValuePropositions that aggregates other ValuePropositions and ValueItems. ValueStreams are abstractly realized by Capabilities that support ValueStreamStages. These Capabilities produce Outcomes that are Valued by ValueItems. Some of these Outcomes are incorporated into the ProductOffering (e.g. sale price, warranty). ValuePropositions target a Customer, who is also the target of the ProductOffering. The ValueProposition is of the ProductOffering.

7.3.3.4 **Class Name:** Customer  **Class Type:** Class  **Stereotype:**

**Base Classes:** AbstractThing, AbstractValueModel, BACMPlainEntity, LegalEntity  
**Definition:** Customer represents a customer type or a class of customers. Customer also represents partner businesses and other forms of contracted business relationships.  
**Usage:** Customer effectively owns a set of CustomerSegments, each of which contains a partial description of the Customer. The CustomerSegments of a Customer may characterize CustomerJourneyStages or Touchpoints (i.e. they describe the Customer characteristics and state of mind at the CustomerJourneyStage or Touchpoint. When this is the case, the Customer should take the CustomerJourney owning the CustomerJourneyStages and Touchpoints. The Customer is an acceptor of one or more ProductOfferings and target of the ValuePropositions of these ProductOfferings.

7.3.3.4.1 **Attributes, Methods and Connectors:**

<table>
<thead>
<tr>
<th>Association Name</th>
<th>Association Type</th>
<th>Generalization Stereotype</th>
<th>Source Class</th>
<th>Target Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer</td>
<td>Association</td>
<td>Generalization</td>
<td>Customer</td>
<td>BACMPlainEntity</td>
</tr>
<tr>
<td>Customer</td>
<td>Association</td>
<td>Generalization</td>
<td>Customer</td>
<td>AbstractThing</td>
</tr>
<tr>
<td>Customer</td>
<td>Association</td>
<td>Generalization</td>
<td>Customer</td>
<td>LegalEntity</td>
</tr>
<tr>
<td>Customer</td>
<td>Association</td>
<td>Generalization</td>
<td>Customer</td>
<td>AbstractValueModel</td>
</tr>
<tr>
<td>Customer</td>
<td>takes Association</td>
<td>Association</td>
<td>Customer</td>
<td>CustomerJourney</td>
</tr>
<tr>
<td>Customer</td>
<td>target Association</td>
<td>Association</td>
<td>ValueProposition</td>
<td>Customer</td>
</tr>
</tbody>
</table>

**Constraint:** Let VP1 be a ValueProposition and Cu1 be a Customer associated by t1, a target association. Then there should be a ValueCharacteristic VC1 with VP1 as its proposition and Cu1 as its customer. Also,
there should be ValueItems \{VI_j\} aggregated by VP1 that value Outcomes \{O_j\} incorporated in ProductOfferings \{PO_j\} accepted by LegalEntities \{LE_j\} that are also the Customer Cu_1. Note that it is commonly the case that the set of individuals represented intensionally by the Customer element are also LegalEntities.

**Association Name:** describes
**Association Type:** Association
**Stereotype:** «class»
**Source Class:** CustomerSegment \([0..*]\)
**Target Class:** Customer \([1..*]\)
**Definition:** The *describes* association represents the relationship between a CustomerSegment and a Customer asserting that the Customer is partially described by the CustomerSegment
**Usage:** If there is no CustomerJourney associated with a Customer, then the set of all CustomerSegments that describe the Customer represent the total customer description. If the Customer takes a CustomerJourney, then the CustomerSegments that describe the Customer may be qualified by the characterizes association to a CustomerJourneyStage or Touchpoint, indicating that the CustomerSegment partially describes the Customer at that CustomerJourneyStage or Touchpoint.

**Association Name:** customer
**Association Type:** Association
**Stereotype:**
**Source Class:** ValueCharacteristic \([0..*]\)
**Target Class:** Customer \([1]\)
**Definition:** The *customer* leg of the ValueCharacteristic association identifies the Customer participating in the value fit analysis represented by the ValueCharacteristic.
**Usage:** The ValueCharacteristic may define specific CustomerSegments, JourneyStages and Touchpoints as having weigts in the value fit analysis. The meta-model does not provide direct support for asserting such facts, but they can be recorded in Annotations associated with the ValueCharacteristic.

### 7.3.3.5 **Class Name:** CustomerJourney  **Class Type:** Class  **Stereotype:**
**Base Classes:** AbstractAction, BACMPlainEntity
**Definition:** A CustomerJourney represents a sequence of stages through which a Customer may pass with respect to a ProductOffering and its ValueProposition. The CustomerJourneyStages of the CustomerJourney capture the notion that the customer experience is cumulative.

#### 7.3.3.5.1 Attributes, Methods and Connectors:

**Association Name:**  Association Type: Generalization
**Stereotype:**
**Source Class:** CustomerJourney \([1]\)
**Target Class:** BACMPlainEntity \([1]\)

**Association Name:**  Association Type: Generalization
**Stereotype:**
**Source Class:** CustomerJourney \([1]\)
**Target Class:** AbstractAction \([1]\)

**Association Name:** owns_2
**Association Type:** Association
**Stereotype:**
**Source Class:** CustomerJourneyStage \([1]\)
**Target Class:** CustomerJourney \([0..*]\)

**Association Name:** takes
**Association Type:** Association
**Stereotype:** «class»
**Source Class:** Customer \([0..*]\)
**Target Class:** CustomerJourney \([0..*]\)
**Definition:** The *takes* association represents a relationship between a Customer and a CustomerJourney asserting that the Customer is likely to take the CustomerJourney.

### 7.3.3.6 **Class Name:** CustomerJourneyStage  **Class Type:** Class  **Stereotype:**
**Base Classes:** AbstractAction, BACMPlainEntity, JSTP
Definition: The CustomerJourneyStage represents a significant stage in the CustomerJourney. An example of the stages of a customer journey would be: awareness, seeking a solution, weighting alternatives, acquiring the solution, using the solution, disposing the solution. Usage: CustomerJourneyStages are often associated with decisions by the customer to proceed to the next stage or abandon the journey. However, the CustomerJourney is not a process and has no alternative sequences or paths.

7.3.3.6.1 Attributes, Methods and Connectors:

Association Name: owns_3 Association Type: Association Stereotype: Source Class: CustomerJourneyStage [1] Target Class: Touchpoint [0..*]

Association Name: owns_2 Association Type: Association Stereotype: Source Class: CustomerJourneyStage [1] Target Class: CustomerJourney [0..*]

Association Name: Association Type: Generalization Stereotype: Source Class: CustomerJourneyStage [] Target Class: BACMPlainEntity []

Association Name: Association Type: Generalization Stereotype: Source Class: CustomerJourneyStage [] Target Class: JSTP []

Association Name: Association Type: Generalization Stereotype: Source Class: CustomerJourneyStage [] Target Class: AbstractAction []

7.3.3.7 Class Name: CustomerSegment Class Type: Class Stereotype:
Base Classes: AbstractResult, AbstractValueModel, BACMPlainEntity
Definition: The CustomerSegment represents a characteristic of the Customer or a component of customer state of mind. CustomerSegments are owned by the Customer they describe.
Usage: When the owning Customer takes a CustomerJourney, CustomerSegments should be created for each CustomerJourneyStage and Touchpoint in the CustomerJourney. These CustomerSegments characterize the customer or the customer's state of mind at the CustomerJourneyStage or Touchpoint.

7.3.3.7.1 Attributes, Methods and Connectors:

Association Name: characterizes Association Type: Association Stereotype: «class» Source Class: CustomerSegment [0..*] Target Class: JSTP [0..*]
Definition: The characterizes association represents a relationship between a CustomerSegment and a Touchpoint meaning that the CustomerSegment partially describes the state of mind or capability of the Customer at the Touchpoint interaction.
Usage: This characterizes association represents the same kind of relationship as the characterizes association between the CustomerSegment and the CustomerJourneyStage. The range of the association is the union of CustomerJourneyStage and Touchpoint.

Association Name: Association Type: Generalization Stereotype: Source Class: CustomerSegment [] Target Class: AbstractResult []

Association Name: Association Type: Generalization Stereotype: Source Class: CustomerSegment [] Target Class: BACMPlainEntity []
Association Name: describes  
Association Type: Association  
Stereotype: «class»  
Source Class: CustomerSegment [0..*]  
Target Class: Customer [1..*]  

Definition: The describes association represents the relationship between a CustomerSegment and a Customer asserting that the Customer is partially described by the CustomerSegment.

Usage: If there is no CustomerJourney associated with a Customer, then the set of all CustomerSegments that describe the Customer represent the total customer description. If the Customer takes a CustomerJourney, then the CustomerSegments that describe the Customer may be qualified by the characterizes association to a CustomerJourneyStage or Touchpoint, indicating that the CustomerSegment partially describes the Customer at that CustomerJourneyStage or Touchpoint.

Association Name: Association Type: Generalization  
Stereotype:  
Source Class: CustomerSegment []  
Target Class: AbstractValueModel []

Association Name: segment  
Association Type: Association  
Stereotype:  
Source Class: ValueCharacteristic [0..*]  
Target Class: CustomerSegment [0..1]

7.3.3.8  
Class Name: JSTP  
Class Type: Class  
Stereotype:  

Base Classes:  

usage: This abstract class provides a union type for CustomerJourneyStage and Touchpoint, allowing the characterizes association to link instances of any concrete subclass of these classes.

7.3.3.8.1 Attributes, Methods and Connectors:

Association Name: characterizes  
Association Type: Association  
Stereotype: «class»  
Source Class: CustomerSegment [0..*]  
Target Class: JSTP [0..*]  

Definition: The characterizes association represents a relationship between a CustomerSegment and a Touchpoint meaning that the CustomerSegment partially describes the state of mind or capability of the Customer at the Touchpoint interaction.

Usage: This characterizes association represents the same kind of relationship as the characterizes association between the CustomerSegment and the CustomerJourneyStage. The range of the association is the union of CustomerJourneyStage and Touchpoint.

Association Name: Association Type: Generalization  
Stereotype:  
Source Class: Touchpoint []  
Target Class: JSTP []

Association Name: Association Type: Generalization  
Stereotype:  
Source Class: CustomerJourneyStage []  
Target Class: JSTP []

7.3.3.9  
Class Name: Touchpoint  
Class Type: Class  
Stereotype:  

Base Classes: AbstractAction, BACMPlainEntity, JSTP  
Definition: The Touchpoint represents an interaction between the business and the Customer.  
Usage: One or more Outcomes created by the business are experienced by the Customer at the Touchpoint (e.g. the customer finds the answer to a question in a brochure created by the business, or the customer receives the business object that was ordered in good condition and on time). Alternatively, one or more Outcomes created by customer uses of the business objects contained in the ProductOffering are experienced by the customer (e.g. the customer uses the purchased hammer to drive nails).
The analysis of value exchanged at the Touchpoint is represented by the ValueCharacteristic associated with the Touchpoint.

### 7.3.3.9.1 Attributes, Methods and Connectors:

**Association Name:** experiences  
**Association Type:** Association  
**Stereotype:** «class»  
**Source Class:** Touchpoint [0..*]  
**Target Class:** Outcome [0..*]  
**Definition:** The experiences relation represents a relationship between an Outcome and a Touchpoint meaning that the Customer will experience the Outcome at the Touchpoint.  
**Usage:** A Touchpoint experiences an Outcome:
1. when that Outcome is provided as a service or
2. when the Outcome is associated with acceptance of the ProductOffering (e.g. the customer is happy with the contract of sale), or
3. when the customer receives information that resolves a question, or
4. when the customer makes use of a business object that is provided as an Outcome of an exchange transaction

**Association Name:**  
**Association Type:** Generalization  
**Stereotype:**  
**Source Class:** Touchpoint []  
**Target Class:** AbstractAction []

**Association Name:**  
**Association Type:** Generalization  
**Stereotype:**  
**Source Class:** Touchpoint []  
**Target Class:** JSTP []

**Association Name:**  
**Association Type:** Generalization  
**Stereotype:**  
**Source Class:** Touchpoint []  
**Target Class:** BACMPlainEntity []

**Association Name:**  
**Association Type:** Association  
**Stereotype:**  
**Source Class:** CustomerJourneyStage [1]  
**Target Class:** Touchpoint [0..*]

### 7.3.3.10 Class Name: ValueCharacteristic  
**Class Type:** Class  
**Stereotype:** «association»

**Base Classes:** AbstractValueModel  
**Definition:** ValueCharacteristic represents the fit between the ValueProposition of a ProductOffering targeted at a Customer.  
**Usage:** ValueCharacteristic is intended to be used with a semantic tagging mechanism such as that provided by MEF or its equivalent. This allows the creation of tagging frameworks such as the Value Proposition Canvas categories of "use", "pain" and "gain". The ValuePropositions, CustomerSegments and ValueItems should be tagged by these categories. The ValueCharacteristic should be similarly tagged and should represent the fit of like tagged ValuePropositions and CustomerSegments. For example, a ValueProposition that relieves a "pain" should be fitted to a CustomerSegment that describes the "pain" by a ValueCharacteristic tagged as "pain".  
**Constraints:** A top level ValueCharacteristic defines a fit value for a relationship between a single Customer and a single ValueProposition. The ValueCharacteristic may be decomposed and account for fit between ValueItems and CustomerSegments that are part of the top level ValueProposition and Customer.

### 7.3.3.10.1 Attributes, Methods and Connectors:

**Association Name:** segment  
**Association Type:** Association  
**Stereotype:**  
**Source Class:** ValueCharacteristic [0..*]  
**Target Class:** CustomerSegment [0..1]
Association Name: item Association Type: Association Stereotype: 
Source Class: ValueCharacteristic [0..*] Target Class: ValueItem [0..1]

Association Name: Association Type: Generalization Stereotype: 
Source Class: ValueCharacteristic [ ] Target Class: AbstractValueModel [ ]

Association Name: customer Association Type: Association Stereotype: 
Source Class: ValueCharacteristic [0..*] Target Class: Customer [1]
Definition: The customer leg of the ValueCharacteristic association identifies the Customer participating in the value fit analysis represented by the ValueCharacteristic. 
Usage: The ValueCharacteristic may define specific CustomerSegments, JourneyStages and Touchpoints as having weights in the value fit analysis. The meta-model does not provide direct support for asserting such facts, but they can be recorded in Annotations associated with the ValueCharacteristic.

Association Name: proposition Association Type: Association Stereotype: 
Source Class: ValueCharacteristic [0..*] Target Class: ValueProposition [1]
Definition: The proposition leg of the ValueCharacteristic association identifies the ValueProposition that is a component of the value fit analysis represented by the ValueCharacteristic. 
Usage: A ValueCharacteristic may identify specific ValueItems of the ValueProposition and represent weights that these items may have in the analysis, but this information must be placed in an Annotation of the ValueCharacteristic as there is no direct support for such facts in the meta-model.

7.3.3.11 Class Name: ValueItem Class Type: Class Stereotype: 
Base Classes: AbstractResult, AbstractValueModel, BACMPlainEntity 
Definition: A ValueItem represents the business belief that a Customer will value one or more Outcomes that are experienced by the Customer. 
Usage: For example, the ability of a sales representative to answer customer questions about a product is deemed to be valuable to the customer. Another example Outcome is the exchange of a good for money; the associated ValueItem could represent the buyer's feeling of having gotten a good deal.

7.3.3.11.1 Attributes, Methods and Connectors:

Association Name: Association Type: Generalization Stereotype: 
Source Class: ValueItem [ ] Target Class: BACMPlainEntity [ ]

Association Name: Association Type: Generalization Stereotype: 
Source Class: ValueItem [ ] Target Class: AbstractValueModel [ ]

Association Name: Association Type: Generalization Stereotype: 
Source Class: ValueItem [ ] Target Class: AbstractResult [ ]

Association Name: values Association Type: Association Stereotype: «class» 
Source Class: ValueItem [0..*] Target Class: Outcome [0..*] 
Definition: The values association links a ValueItem to an Outcome and provides a valuation of that Outcome. An Outcome may have several ValueItems, reflecting the ways in which different stakeholders perceive the Outcome. Likewise, a ValueItem may value multiple Outcomes that must be valued as a group. 
Usage: The Outcome may be present in the business architecture model without an associated ValueItem, but ValueItems may not exist without being associated to an Outcome.

Association Name: item Association Type: Association Stereotype:
Source Class: ValueCharacteristic  [0..*]  Target Class: ValueItem  [0..1]

Association Name: aggregates_3  Association Type: Association  Stereotype: 
Source Class: ValueProposition  [0..*]  Target Class: ValueItem  [0..*]  
Definition: The aggregates association represents the aggregation of ValueItems into a ValueProposition. ValueItems may be shared with multiple ValuePropositions.

Association Name: produces_2  Association Type: Association  Stereotype: «shortcut» 
Source Class: ValueStreamStage  [0..*]  Target Class: ValueItem  [0..*]  
Definition: The produces association represents the fact of a ValueItem being produced by valuing one or more Outcomes produced by Capabilities that support the ValueStreamStage or Processes or Activities that implement the ValueStreamStage.
Usage: The ValueItems produced in a ValueStreamStage that is part of a ValueStream should contribute to the ValueProposition produced by the ValueStream. The meta-model does not enforce this restriction.
Constraint: The produces association is consistent with the ValueStreamStage being supported by some Capabilities that produce Outcomes that are valued by the ValueItem.

Class Name: ValueProposition  Class Type: Class  Stereotype: 
Base Classes: AbstractResult, AbstractValueModel, BACMPlainEntity  
Definition: The ValueProposition represents a collection of values the business believes it is offering to customers, partners and other stakeholders through a ProductOffering.

7.3.12.1 Attributes, Methods and Connectors:  
Association Name:  Association Type: Generalization  Stereotype: 
Source Class: ValueProposition  []  Target Class: BACMPlainEntity  []

Association Name: of  Association Type: Association  Stereotype: «shortcut» 
Source Class: ValueProposition  [0..*]  Target Class: ProductOffering  [0..*]  
Definition: The of association links a ValueProposition to a ProductOffering and represents that is the ValueProposition is about the ProductOffering.
Usage: The ValueProposition and PO1 be a ProductOffering associated by o1, an "of" association. Then for some subset of ValueItems {VIj} aggregated by VPI such that each VIj values an Outcome O1 that is incorporated in the ProductOffering PO1. Note that the ProductOfferings typically include Outcomes that are experienced by the Customer at a Touchpoint.
Constraint: Let VP1 be a ValueProposition and PO1 be a ProductOffering associated by o1, an "of" association. Then for some subset of ValueItems {VIj} aggregated by VPI such that each VIj values an Outcome O1 that is incorporated in the ProductOffering PO1. Note that the ProductOfferings typically include Outcomes that are experienced by the Customer at a Touchpoint.

Association Name:  Association Type: Generalization  Stereotype: 
Source Class: ValueProposition  []  Target Class: AbstractResult  []

Association Name:  Association Type: Generalization  Stereotype: 
Source Class: ValueProposition  []  Target Class: AbstractValueModel  []

Association Name: aggregates_3  Association Type: Association  Stereotype: 
Source Class: ValueProposition  [0..*]  Target Class: ValueItem  [0..*]  
Definition: The aggregates association represents the aggregation of ValueItems into a ValueProposition. ValueItems may be shared with multiple ValuePropositions.

Association Name: target  Association Type: Association  Stereotype: «shortcut» 
Source Class: ValueProposition  [0..*]  Target Class: Customer  [0..*]  
Definition: The target shortcut association asserts that the ValueProposition is intended to target the Customer.
Usage: This shortcut allows the architect to assert that a ValueProposition targets a Customer and imply that there is an unspecified ValueCharacteristic that represents the value fit analysis of the ValueProposition and Customer.

Constraint: Let VP1 be a ValueProposition and Cu1 be a Customer associated by t1, a target association. Then there should be a ValueCharacteristic VC1 with VP1 as its proposition and Cu1 as its customer. Also, there should be ValueItems {VIj} aggregated by VP1 that value Outcomes {Oj} incorporated in ProductOfferings {POj} accepted by LegalEntities {LEj} that are also the Customer Cu1. Note that it is commonly the case that the set of individuals represented intensionally by the Customer element are also LegalEntities.

Association Name: produces_1 Association Type: Association Stereotype: «shortcut»
Source Class: ValueStream [0..*] Target Class: ValueProposition [0..*]
Definition: The produces shortcut association represents the creation of a ValueProposition by a ValueStream.
Usage: The produces relation effectively aggregates the produces relations between the ValueStreamStages that are part of this ValueStream and the ValueItems that comprise the ValueProposition of this ValueStream.
Constraint: The produces association is implied by some owned ValueStreamStages that produce ValueItems that are aggregated into the ValueProposition.

7.3.3.13 Class Name: ValueStream Class Type: Class Stereotype:
Base Classes: AbstractAction, AbstractValueModel, BACMPlainEntity, VSVSS
Definition: A ValueStream represents a set of stages that accumulate value represented by the ValueProposition.
Usage: The notion that value accumulation can be broken into components has been central to strategic practices such as Michael Porter's value chains and high level, value oriented process architecture. The notion is well established in business architecture and analysis practice. In some cases, it may be desirable to order the stages in a ValueStream. For example, there is a natural order to the design, build, inventory, sell and service stages of a manufacturing business. However, in other cases, such as health care, it is difficult to order the stages of triage, diagnosis, treatment, prevention. Consequently, no strong semantic interpretation should be associated with the ordering of ValueStreamStages in a ValueStream.
Constraint: A ValueStream instance may not own, aggregate or generalize another ValueStream instance

7.3.3.13.1 Attributes, Methods and Connectors:

Association Name: owns_1 Association Type: Association Stereotype: Generalization Stereotype:
Source Class: ValueStream [] Target Class: BACMPlainEntity []
Association Name: Association Type: Generalization Stereotype:
Source Class: ValueStream [] Target Class: VSVSS []

Association Name: Association Type: Generalization Stereotype:
Source Class: ValueStream [] Target Class: AbstractValueModel []

Association Name: produces_1 Association Type: Association Stereotype: «shortcut»
Source Class: ValueStream [0..*] Target Class: ValueProposition [0..*]
Definition: The produces shortcut association represents the creation of a ValueProposition by a ValueStream.
Usage: The produces relation effectively aggregates the produces relations between the ValueStreamStages that are part of this ValueStream and the ValueItems that comprise the ValueProposition of this ValueStream.
Constraint: The produces association is implied by some owned ValueStreamStages that produce ValueItems that are aggregated into the ValueProposition.

Association Name: Association Type: Generalization Stereotype:
Source Class: ValueStream [] Target Class: AbstractAction []

7.3.3.14 Class Name: ValueStreamStage  Class Type: Class Stereotype:
Base Classes: AbstractAction, AbstractValueModel, BACMPlainEntity, VSVSS
Definition: ValueStreamStages represent significant points of value creation in a ValueStream.
Usage: ValueStreamStages are dependent on their containing ValueStream and are not shared with other ValueStreams. When the business architect intends to represent similar ValueStreamStages in different ValueStreams, the similarity should be represented by having the same set of relationships with the supporting Capabilities. ValueStreamStages are often defined by analysis and decomposition of the ValueProposition. They may also represent stages of completion of a "build to order" product that are of interest to the Customer (e.g. stages where the Customer may make changes in specifications of the ordered product).
Constraint: A ValueStreamStage may only own other ValueStreamStages and be owned by another ValueStreamStage or a ValueStream. ValueStreamStages may not participate in generalizes or aggregates associations.

7.3.3.14.1 Attributes, Methods and Connectors:

Association Name: Association Type: Generalization Stereotype:
Source Class: ValueStreamStage [] Target Class: AbstractAction []

Association Name: Association Type: Generalization Stereotype:
Source Class: ValueStreamStage [] Target Class: BACMPlainEntity []

Association Name: Association Type: Generalization Stereotype:
Source Class: ValueStreamStage [] Target Class: VSVSS []

Association Name: Association Type: Generalization Stereotype:
Source Class: ValueStreamStage [] Target Class: AbstractValueModel []

Association Name: produces_2 Association Type: Association Stereotype: «shortcut»
Source Class: ValueStreamStage [0..*] Target Class: ValueItem [0..*]
Definition: The produces association represents the fact of a ValueItem being produced by valuing one or more Outcomes produced by Capabilities that support the ValueStreamStage or Processes or Activities that implement the ValueStreamStage.

Usage: The ValueItems produced in a ValueStreamStage that is part of a ValueStream should contribute to the ValueProposition produced by the ValueStream. The meta-model does not enforce this restriction.

Constraint: The produces association is consistent with the ValueStreamStage being supported by some Capabilities that produce Outcomes that are valued by the ValueItem.

Association Name: owns_1 Association Type: Association Stereotype: 
Source Class: ValueStream [1] Target Class: ValueStreamStage [0..*]
Definition: The owns association represents that a ValueStream may be composed of ValueStreamStages. ValueStreamStages cannot be shared with other ValueStream instances.
This association may be ordered to facilitate the presentation of ValueStreamStages, but no business operating model implications should be assumed based on the ordering of ValueStreamStages.
This association effectively specializes the owns association in the BusinessElement diagram to add the ordered constraint.

Association Name: triggers Association Type: Association Stereotype: «shortcut»
Source Class: Outcome [0..*] Target Class: ValueStreamStage [0..*]
Definition: The triggers association represents that the Outcome allows or initiates the ValueStreamStage.
Usage: It is often useful in analysis to record the Outcomes that constitute the important beginning events of a ValueStreamStage. The triggers association allows the architect to record these relationships.
Constraint: Let O1 be an Outcome experienced at a Touchpoint T1 and VSS1 be a ValueStreamStage where O1 triggers VSS1, then there should exist a ValueItem VI1 that values O1 and is produced by VSS1.

Association Name: participate Association Type: Association Stereotype: «shortcut»
Source Class: Performer [0..*] Target Class: ValueStreamStage [0..*]
Definition: The participate shortcut asserts that a Performer is assigned to an unspecified PerformerRole of an unspecified Capability that supports the ValueStreamStage.
Constraint: Let P1 be a Performer participating in a ValueStreamStage VSS1. There should exist a PerformerRole PR1 that P1 is assignedTo and PR1 is a PerformerRole ofCapability C1 that produces some Outcome O1 valued by a ValueItem VI1 that is produced by ValueStream VSS1.

Association Name: supports Association Type: Association Stereotype: «class»
Source Class: Capability [0..*] Target Class: ValueStreamStage [0..*]
Definition: The supports association represents the relationship between a Capability and a ValueStreamStage that means that the Capability is needed in the ValueStreamStage.
Usage: For example, an important stage in the creation of value for a manipulation puzzle such as Rubik's Cube is the production of a manufacturable design of the puzzle. A failure here can result in a puzzle that cannot be manufactured or is not attractive to purchasers. Outcomes providing value are:
- a positive manufacturability review;
- a positive customer reaction in a focus group.
The Capabilities needed to produce these Outcomes are: product design, manufacturability analysis, focus group management. For this example, the previous three Capability instances would be associated with the "Design Ready" ValueStreamStage.

7.3.4 Package: Organization
7.3.4.1 **Diagram: Organization**

The Organization diagram defines abstract syntax representing relationships between structural components of the organization (OrgUnits, Systems and Resources) and Capabilities and/or Processes.
7.3.4.2 **Diagram: Participant**

![Diagram of Participant]

7.3.4.3 **Class Name:** Jurisdiction  **Class Type:** Class Stereotype:

**Base Classes:** AbstractOperatingModel, BACMPlainEntity

**Definition:** The Jurisdiction represents a legal jurisdiction with powers to charter and/or regulate businesses.

### 7.3.4.3.1 Attributes, Methods and Connectors:

- **Association Name:** legal_jurisdiction  **Association Type:** Association Stereotype: «class»
  **Source Class:** LegalEntity [0..*]  **Target Class:** Jurisdiction [0..*]

**Definition:** The "legal jurisdiction"association instances represent the jurisdiction to which an Enterprise belongs.

**Usage:** The meta-model allows Enterprise instances to be in multiple jurisdictions (e.g. a business that is subject to local, provincial and stage laws, regulations and processes).
7.3.4.4  **Class Name:** LegalEntity  **Class Type:** Class  **Stereotype:**

**Base Classes:** OrgUnit

**Definition:** LegalEntity represents a human organization that is subject to the laws and regulations of a Jurisdiction.

### Attributes, Methods and Connectors:

- **Association Name:** Association Type: Generalization  **Stereotype:**
  **Source Class:** LegalEntity  **Target Class:** OrgUnit

- **Association Name:** provides  **Association Type:** Association  **Stereotype:** «class»
  **Source Class:** LegalEntity [0..*]  **Target Class:** Offering [0..*]
  **Definition:** The provider relation represents a relationship between a LegalEntity and an Offering created by the LegalEntity that is intended to solicit the business of designated parties identified by the consumer relation.

- **Association Name:** legal_jurisdiction  **Association Type:** Association  **Stereotype:** «class»
  **Source Class:** LegalEntity [0..*]  **Target Class:** Jurisdiction [0..*]
  **Definition:** The "legal_jurisdiction" association instances represent the jurisdiction to which an Enterprise belongs.
  **Usage:** The meta-model allows Enterprise instances to be in multiple jurisdictions (e.g., a business that is subject to local, provincial, and stage laws, regulations, and processes).

- **Association Name:** accepts  **Association Type:** Association  **Stereotype:** «class»
  **Source Class:** LegalEntity [0..*]  **Target Class:** Offering [0..*]
  **Definition:** The acceptor relation represents a relationship between a party external to the business and an Offering intended to solicit business from the acceptor party represented by the Customer.
  **Usage:** Note that offering dies not represent a sale; in a sale, each party gives something of value and receives something of value.

- **Association Name:** recipient_1  **Association Type:** Association  **Stereotype:** «class»
  **Source Class:** ServiceOutcome [0..*]  **Target Class:** LegalEntity [0..*]
  **Definition:** The recipient association asserts that the LegalEntity is the recipient of ServiceOutcomes incorporated into a ServiceOffering.
  **Usage:** It is not necessarily the case that the recipient LegalEntity is the same as the accepting LegalEntity of the incorporating ServiceOffering.

- **Association Name:** buyer  **Association Type:** Association  **Stereotype:** «class»
  **Source Class:** MerchandiseOutcome [0..*]  **Target Class:** LegalEntity [0..*]
  **Definition:** The buyer association is related to the accept association and asserts that a LegalEntity (typically also a Customer) is the targeted buyer of the MerchandiseOutcome.
  **Usage:** The buyer of the MerchandiseOutcome is not necessarily the LegalEntity that accepts the MerchandiseOffering in the case when the acceptor is acting as an agent for the buyer.

- **Association Name:** supplier  **Association Type:** Association  **Stereotype:** «class»
  **Source Class:** ProcurementOutcome [0..*]  **Target Class:** LegalEntity [0..*]
Definition: The supplier association asserts that the LegalEntity is to be the supplier of the ProcurementOutcome.

Usage: The supplier LegalEntity is not necessarily the same as the provider LegalEntity for the ProcurementOffering incorporating the ProcurementOutcome.

Association Name: provider_0 Association Type: Association Stereotype: «class»
Source Class: OutsourcedServiceOutcome [0..*] Target Class: LegalEntity [0..*]
Definition: The provider association asserts that a LegalEntity is the provider of the OutsourcedServiceOutcome.

Usage: The provider LegalEntity is not necessarily the same LegalEntity as the provider of the OutsourcedServiceOffering.

7.3.4.5 Class Name: OrgUnit Class Type: Class Stereotype:
Base Classes: Performer
Definition: The OrgUnit meta-class represents the various types of human organizations and individuals capable of acting as performers.

7.3.4.5.1 Attributes, Methods and Connectors:

Association Name: Association Type: Generalization Stereotype:
Source Class: OrgUnit [] Target Class: Performer []

Association Name: Association Type: Generalization Stereotype:
Source Class: LegalEntity [] Target Class: OrgUnit []

Association Name: source Association Type: Association Stereotype:
Source Class: Responsible [0..*] Target Class: OrgUnit [0..1]
Definition: The source leg of the Responsible association asserts that the source OrgUnit is responsible in some way for the target OrgUnit.

Association Name: target Association Type: Association Stereotype:
Source Class: Responsible [0..*] Target Class: OrgUnit [0..1]
Definition: The target leg of the Responsible association asserts that the OrgUnit is responsible to another OrgUnit determined by the source leg of the Responsible association.

7.3.4.6 Class Name: Performer Class Type: Class Stereotype:
Base Classes: AbstractThing, BACMPlainEntity
Definition: The Performer represents entities that are capable of performing PerformerRoles. Performer has two specializations: OrgUnit and System, representing a human components of the business or a system.
Usage: The Performer is concrete to allow modeling the need for a Performer without committing to a human assignment, a system assignment, or a combination of both. Performers are generally described by skills or abilities. The Performer and Resource classes are not disjoint (i.e. an entity may be a Resource with respect to one Capability while being a Performer with respect to another Capability). An entity may not be both a Performer and a Resource of the same Capability.

7.3.4.6.1 Attributes, Methods and Connectors:

Association Name: participate Association Type: Association Stereotype: «shortcut»
Source Class: Performer [0..*] Target Class: ValueStreamStage [0..*]
Definition: The participate shortcut asserts that a Performer is assigned to an unspecified PerformerRole of an unspecified Capability that supports the ValueStreamStage.

Constraint: Let P1 be a Performer participating in a ValueStreamStage VSS1. There should exist a PerformerRole PR1 that P1 is assignedTo and PR1 is a PerformerRole ofCapability C1 that produces some Outcome O1 valued by a ValueItem VI1 that is produced by ValueStream VSS1.

Association Name: Generalization Stereotype:
Source Class: Performer [] Target Class: AbstractThing []

Association Name: Generalization Stereotype:
Source Class: Performer [] Target Class: BACMPlainEntity []

Definition: belongs_to represents that a Performer belongs to theBusiness.
Usage: In a model, there will typically be semantic overlap between belongs_to and Responsible. However, the metamodel syntax presently does not allow the specification of this overlap. The business architect may choose to use belongs_to in lieu of Responsible or vice versa. It would not be recommended to use both where there is the potential of semantic overlap.

Association Name: Generalization Stereotype:
Source Class: OrgUnit [] Target Class: Performer []

Definition: The assignment leg of the PerformerRole association represents that a Performer is assigned to the PerformerRole.

Association Name: Generalization Stereotype:
Source Class: System [] Target Class: Performer []

Association Name: Aggregates_1 Association Type: Association Stereotype:
Source Class: CapabilityImplementation [0..*] Target Class: Performer [0..*]

7.3.4.7 Class Name: Resource Class Type: Class Stereotype:
Base Classes: AbstractThing, BACMPlainEntity
Definition: Resource represents an entity that is required or needed by a ResourceRole but is not a Performer and does not become a part of a BusinessObject or InformationItem associated with any Outcome produced by the Capability or CapabilityBehavior.
Usage: These relationships are represented by the assignment of structural components of the business to roles of Capabilities/CapabilityBehaviors and Processes/Activities.

7.3.4.7.1 Attributes, Methods and Connectors:

Association Name: Generalization Stereotype:
Source Class: Resource [] Target Class: AbstractThing []

Association Name: Generalization Stereotype:
Source Class: Resource [] Target Class: BACMPlainEntity []
Association Name: assignTo
Association Type: Association
Stereotype: Source Class: ResourceRole [0..*] Target Class: Resource [0..1]
Definition: The assignTo leg of the ResourceRole association represents that a Resource has been assigned to a ResourceRole.

Association Name: aggregates
Association Type: Association
Stereotype: Source Class: CapabilityImplementation [0..*] Target Class: Resource [0..*]

### 7.3.4.8 Class Name: Responsible Class Type: Class Stereotype: «association»
Base Classes: AbstractOperatingModel
Definition: Responsible represents an unspecified kind of responsibility relationship between a source OrgUnit and a target OrgUnit. This relationship may also include a BusinessElement that defines the nature of the association.
Usage: Responsible instances may form generalization hierarchies. The business architect may create these hierarchies to represent particular types of responsibility relationships found in the business. When specializing Responsible instances, the source, target and nature association legs may be subsetted to restrict them to particular types of OrgUnit and BusinessElement.

#### 7.3.4.8.1 Attributes, Methods and Connectors:

Association Name: source
Association Type: Association
Stereotype: Source Class: Responsible [0..*] Target Class: OrgUnit [0..1]
Definition: The source leg of the Responsible association asserts that the source OrgUnit is responsible in some way for the target OrgUnit.

Association Name: target
Association Type: Association
Stereotype: Source Class: Responsible [0..*] Target Class: OrgUnit [0..1]
Definition: The target leg of the Responsible association asserts that the OrgUnit is responsible to another OrgUnit determined by the source leg of the Responsible association.

Association Name: nature
Association Type: Generalization
Stereotype: Source Class: Responsible [0..*] Target Class: AbstractOperatingModel [0..*]
Definition: The nature leg of the Responsible designates a BusinessElement that helps define the scope and/or nature of the Responsible association.

### 7.3.4.9 Class Name: System Class Type: Class Stereotype:
Base Classes: Performer
Definition: The System represents the concept of a non-human performer, such as an IT system or a robot. Tools such as jigs and drills are not considered Performers for the purpose of business architecture. They should be modeled as Resources.

#### 7.3.4.9.1 Attributes, Methods and Connectors:

Association Name: Generalization
Stereotype: Source Class: System [0..*] Target Class: Performer [0..*]
Association Name: contains  
Association Type: Association  
Stereotype: «class»  
Source Class: BusinessObject  [0..*]  
Target Class: System  [0..*]  
Definition: The contains association represents that BusinessObjects may contain System.  
Usage: In some cases, a BusinessObject and a System may represent different aspects of the same entity; since meta-classes in this meta-model are not assumed disjoint, an instance may have both BusinessObject and System as metaclasses. However, a BusinessObject may contain several Systems and other BusinessObjects as well. In this case, the Systems are not aspects of the primary BusinessObject, and the contains association allows the architect to represent this. An example of this latter case is a primary BusinessObject that is a computer and the System is a software package hosted on that computer (along with other software packages). The software package may be an instance of a System and also an instance of a BusinessObject (i.e. the code).

7.3.4.10  
Class Name: theBusiness  
Class Type: Class  
Stereotype: «individual»
Base Classes: LegalEntity  
Definition: theBusiness represents the particular business that is the subject of the business architecture model.  
Usage: Only one instance of this metaclass is allowed in a model. This instance should be the sole provider of the top level ProductOffering, representing that the business is responsible for the entire product offering, parts of which may be provided by business partners represented as LegalEntities.

7.3.4.10.1 Attributes, Methods and Connectors:

Association Name: Association Type: Generalization  
Stereotype:  
Source Class: theBusiness  []  
Target Class: LegalEntity  []  

Association Name: recipient 0  
Association Type: Association  
Stereotype: «class»  
Source Class: OutsourcedServiceOutcome  [0..*]  
Target Class: theBusiness  []  
Definition: The recipient association asserts that theBusiness is the intended recipient and beneficiary of the OutsourcedServiceOutcome.  
Usage: The recipient is not necessarily the same LegalEntity that accepts the OutsourcedServiceOffering.

Association Name: seller  
Association Type: Association  
Stereotype: «class»  
Source Class: MerchandiseOutcome  [0..*]  
Target Class: theBusiness  []  
Definition: The seller association asserts that theBusiness is the seller of the MerchandiseOutcome incorporated in the MerchandiseOffering.  
Usage: This association does not imply that the LegalEntity providing the MerchandiseOffering is the same as the seller of the MerchandiseOutcome.

Association Name: provider_1  
Association Type: Association  
Stereotype: «class»  
Source Class: ServiceOutcome  [0..*]  
Target Class: theBusiness  [0..1]  
Definition: The provider association asserts that theBusiness is the provider of the ServiceOutcome incorporated into a ServiceOffering.  
Usage: The provider LegalEntity is not necessarily the same as the LegalEntity that provides the ServiceOffering that incorporates the ServiceOutcome.

Association Name: belongs_to  
Association Type: Association  
Stereotype: «class»  
Source Class: Performer  [0..*]  
Target Class: theBusiness  [0..*]  
Definition: belongs_to represents that a Performer belongs to theBusiness.  
Usage: In a model, there will typically be semantic overlap between belongs_to and Responsible. However, the metamodel syntax presently does not allow the specification of this overlap. The business architect may choose to use belongs_to in lieu of Responsible or vice versa. It would not be recommended to use both where there is the potential of semantic overlap.
Association Name: procurer  
Association Type: procurement  
Association Stereotype: «class»  
Source Class: ProcurementOutcome [0..*]  
Target Class: theBusiness [1]  
Definition: The procurer association asserts that theBusiness is the procurer of the ProcurementOutcome  
Usage: The procurer LegalEntity is not necessarily the acceptor LegalEntity of the ProcurementOffering

7.3.5 Package: Process

7.3.5.1 Diagram: Process

Commented JR33]: Issue BACM-83 resolved by BACM-9
The Process diagram defines abstract syntax for a high level process model, for representing how components of a process implement a value stream's ValueStreamStages, and for relating Capabilities and Processes/Activities through their related Outcomes and the Outcome associated BusinessObjects and InformationItems.

Processes are modeled as Activities and Processes. A Process is an aggregator of other Processes and Activities.

### 7.3.5.2 Class Name: AbstractProcess  Class Type: Class  Class Stereotype:

**Base Classes:** AbstractAction, AbstractOperatingModel, APCICB, BACMPlainEntity

**Definition:** AbstractProcess is not intended to represent a business concept. It is a metamodeling technical device to share relationships with Process and Activity that would otherwise need to be duplicated.

**Usage:** AbstractProcess is an abstract meta-class that provides input and output Outcome connection abilities to both Process and Activity. It also provides the role link to PerformerRoles and ResourceRoles. It also provides the implements link to a ValueStream or some ValueStreamStages. Since implements aligns the scope of the Process with either a ValueStreamStage or a ValueStream, it should not link both a ValueStreamStage and the ValueStream the ValueStreamStage belongs to.

### 7.3.5.2.1 Attributes, Methods and Connectors:

- **Association Name:**  
- **Association Type:** Generalization  
- **Generalization Stereotype:**  
- **Source Class:** AbstractProcess  
- **Target Class:** APCICB
Association Name: Association Type: Generalization Stereotype:
Source Class: AbstractProcess [] Target Class: BACMPlanEntity []

Association Name: implements_0 Association Type: Association Stereotype: «shortcut»
Source Class: AbstractProcess [0..*] Target Class: AbstractCapability [0..*]
Definition: The implements shortcut represents that a CapabilityBehavior and an AbstractProcess have related Outcomes
Usage: It could also be justified by a common Performer playing a role in the CapabilityBehavior and the AbstractProcess
Constraint: Let P1 be a Process and C1 be a capability associated by an implements association. Then there should exist Outcomes O1 and O2 such that O1 is produced by (needed by) C1 and O2 is output (input) by P1 and O1 and O2 are related such that they are the same Outcome or one is in the extended aggregation of the other or one is the extended specialization of the other or any chain of relationships connecting the two where the chain consists exclusively of being aggregated by or being a specialization of the predecessor Outcome.

Association Name: Association Type: Generalization Stereotype:
Source Class: AbstractProcess [] Target Class: AbstractOperatingModel []

Association Name: input Association Type: Association Stereotype: «class»
Source Class: AbstractProcess [0..*] Target Class: Outcome [0..*]
Definition: The input association represents that the AbstractProcess inputs (requires or can use) the Outcome.
Usage: The input association in the process perspective corresponds to the needs association in the capability perspective. While it is possible that the same Outcome is input to a process and needed by a capability, it will usually be the case that a process inputs an Outcome that is related by generalization or aggregation (or another relation between Outcomes) to an Outcome needed by a capability. The process and capability in this case are semantically related by the relationship between their Outcomes.
For example, a CustomerInformationManagement Capability may need CustomerInformation_change_pending Outcome. A process that updates the CustomerAddress (a component of CustomerInformation) may input CustomerAddress_change_pending Outcome, that is related to the other Outcome by aggregation.

Association Name: output Association Type: Association Stereotype: «class»
Source Class: AbstractProcess [0..*] Target Class: Outcome [0..*]
Definition: The output association represents that the AbstractProcess outputs the Outcome.
Usage: The output association in the process perspective corresponds to the produces association in the capability perspective. While it is possible that the same Outcome is output from a process and produced by a capability, it will usually be the case that a process outputs an Outcome that is related by generalization or aggregation (or another relation between Outcomes) to an Outcome produced by a capability. The process and capability in this case are semantically related by the relationship between their Outcomes.
For example, a CustomerInformationManagement Capability may produce CustomerInformation_is_current and CustomerInformation_is_correct Outcomes. A process that updates the CustomerAddress (a component of CustomerInformation) may produce CustomerAddress_is_current and CustomerAddress_is_correct Outcomes, that are related to the other Outcomes by aggregation.

Association Name: Association Type: Generalization Stereotype:
Source Class: AbstractProcess [] Target Class: AbstractAction []

Association Name: implements_1 Association Type: Association Stereotype: «class»
Source Class: AbstractProcess [0..*] Target Class: VSVSS [0..*]
Definition: The implements association asserts that a Process or Activity implements a ValueStream and implies that Outcomes of the Process are valued as ValueItems incorporated into the ValueProposition delivered by the ValueStream.

BACM 1.0
Usage: It is not permitted for a Process or Activity to implement both a ValueStream and one or more ValueStreamStages of that ValueStream. A Process implementing a ValueStream may have aggregated Processes that implement ValueStreamStages of the ValueStream.

Association Name: aggregates Association Type: Association Stereotype:
Source Class: Process [0..*] Target Class: AbstractProcess [0..*]
A Process aggregates other Processes and Activities.

Association Name: Association Type: Generalization Stereotype:
Source Class: Activity [] Target Class: AbstractProcess []

Association Name: informs 1 Association Type: Association Stereotype: «class»
Source Class: InformationItem [0..*] Target Class: AbstractProcess [0..*]
Definition: The informs association represents the influence of information (represented by InformationItem) on a Process or Activity.
Usage: Information, such as weather, production targets, and results of a business analysis project will change how a business behaves and how a Process or Activity performs.

Association Name: Association Type: Generalization Stereotype:
Source Class: Process [] Target Class: AbstractProcess []

Association Name: ofProcess Association Type: Association Stereotype:
Source Class: Role [0..*] Target Class: AbstractProcess [0..1]
Definition: The ofProcess leg of the Role association links a PerformerRole or ResourceRole to a Process or Activity.

7.3.5.3 Class Name: Activity Class Type: Class Stereotype:
Base Classes: AbstractProcess
Definition: Activities represent atomic (non-decomposable) activities.

7.3.5.3.1 Attributes, Methods and Connectors:

Association Name: Association Type: Generalization Stereotype:
Source Class: Activity [] Target Class: AbstractProcess []

7.3.5.4 Class Name: Process Class Type: Class Stereotype:
Base Classes: AbstractProcess
Definition: Process represents an aggregation of Activities and other Processes.
Usage: A Process aggregated into another Process means that the aggregated Process may be executed as a part of executing the aggregator Process. The abstract syntax does not specify a starting or ending Process/Activity; consequently starting and ending Activities/Processes aggregated by another Process must be determined by analysis of the Outcome connections.

7.3.5.4.1 Attributes, Methods and Connectors:

Association Name: aggregates Association Type: Association Stereotype:
Source Class: Process [0..*] Target Class: AbstractProcess [0..*]
A Process aggregates other Processes and Activities.
7.3.5.5 **Class Name: VSVSS**  **Class Type: Class Stereotype:**

**Base Classes:**

**Usage:** This abstract class provides a union type for ValueStream and ValueStreamStage, allowing instances of the `implements_1` association to link to instances of any concrete subclass of either of these classes.

7.3.5.5.1 **Attributes, Methods and Connectors:**

**Association Name:** Association **Type:** Generalization **Stereotype:**

Source Class: ValueStream  Target Class: VSVSS

**Association Name:** implements_1 Association **Type:** Association **Stereotype:** «class»

Source Class: AbstractProcess  Target Class: VSVSS

**Definition:** The `implements_1` association asserts that a Process or Activity implements a ValueStream and implies that Outcomes of the Process are valued as ValueItems incorporated into the ValueProposition delivered by the ValueStream.

**Usage:** It is not permitted for a Process or Activity to implement both a ValueStream and one or more ValueStreamStages of that ValueStream. A Process implementing a ValueStream may have aggregated Processes that implement ValueStreamStages of the ValueStream.

**Association Name:** Association **Type:** Generalization **Stereotype:**

Source Class: ValueStreamStage  Target Class: VSVSS

7.3.6 **Package: Product**
7.3.6.1 **Diagram: MerchandiseOffering**

- **Organization:** LegalEntity
- **Offering**
  - accepts
  - provides
- **ProductOffering**
  - incorporates
- **MerchandiseOffering**
  - buyer
  - seller
- **Capability:** Outcome
  - stateOf
- **MerchandiseOutcome**
  - incorporates

Commented [JR34]: Issue BACM-83 resolved by BACM-9
7.3.6.2 **Diagram: OutsourcedServiceOffering**

Commented [JR35]: Issue BACM-83 resolved by BACM-9
Issue BACM-100 resolved by BACM-101 - introduces abstract APCICB class
7.3.6.3 **Diagram: ProcurementOffering**

7.3.6.4 **Diagram: Product**

[Diagram showing relationships between organizational entities and offerings, including services, merchandise, and contract relations.]

**Commented [JR37]:** Issue BACM-83 resolved by BACM-9.
The metamodel uses a single syntax to express three different patterns:

1. **GoodOffering** - Possession of an AbstractBusinessObject is changed and the customer experiences Outcomes associated with the AbstractCapabilities possessed by the AbstractBusinessObject in a post-change-of-possession JourneyStage. The Outcome incorporated in the GoodOffering represents the pledged state of the AbstractBusinessObject that is the object of the ProductOffering (e.g. that the AbstractBusinessObject is complete and functional).

2. **ServiceOffering** - The customer experiences the Outcome of a ServiceOffering provided by the business through some of its AbstractCapabilities. This experience is associated with both the activities at the Touchpoint (i.e. while the Service is being rendered) and at a JourneyStage subsequent to the completion of the Service. These situations would be modeled by two different Outcomes and not by a single Outcome that is experienced at both a Touchpoint and a JourneyStage. In the Service case, the Outcome incorporated in the ProductOffering is produced by the AbstractCapability that is the object of the Service.

3. **OutsourcingOffering** - the Customer is solicited to provide Outcomes to the business. The OutsourcingOffering may specify processes (CapabilityBehaviors) and resources (CapabilityImplementations) that the Customer is asked to follow and use respectively.
7.3.6.5 **Diagram: ServiceOffering**

Commented [JR38]: Issue BACM-83 resolved by BACM-9
Issue BACM-118 resolved by BACM-119
Modified diagram to separate "recordedAs" and "stateOf" associations between Outcome and AbstractBusinessObject
7.3.6.6  **Class Name**: APCICB  **Class Type**: Class  **Class Stereotype**:

**Base Classes**:

**Usage**: This abstract element defines a union type for AbstractProcess, CapabilityImplementation and CapabilityProvider, allowing the specifies association to connect any instances of any concrete subclasses of these classes.

**7.3.6.6.1 Attributes, Methods and Connectors:**

- **Association Name**: specifies  **Association Type**: Association  **Association Stereotype**: «class»  
  **Source Class**: OutsourcedServiceOffering [0..*]  **Target Class**: APCICB [0..*]  
  **Definition**: The specifies association represents a relationship between an OutsourcingOffering and a CapabilityBehavior or Process or CapabilityImplementation, in which the Customer would be required or advised to perform the CapabilityBehavior or Process or CapabilityImplementation, in which the Customer would be required or advised to perform the CapabilityBehavior or Process and/or provide Performers and Resources as specified by the CapabilityImplementation as an implementation of the CapabilityBehavior or Process.

- **Usage**: This association is effectively combined with the two other specifies relation whose source is OutsourcingOffering so that the range of the combined associations is the union of AbstractProcess, CapabilityBehavior and CapabilityImplementation.

- **Association Name**:  **Association Type**: Generalization  **Association Stereotype**: «class»  
  **Source Class**: CapabilityBehavior []  **Target Class**: APCICB []

BACM 1.0
Association Name: Association Type: Generalization Stereotype:
Source Class: CapabilityImplementation [] Target Class: APCICB []

Association Name: Association Type: Generalization Stereotype:
Source Class: AbstractProcess [] Target Class: APCICB []

7.3.6.7 Class Name: ContractRelation Class Type: Class Stereotype: «association»
Base Classes: AbstractOperatingModel
Definition: ContractRelation represents any kind of relationship between Offerings.
Usage: ContractRelation should be instanced as a relationship between Offerings whose arity is determined by the architect. Each leg of such an instance effectively inherits from the relation association.

7.3.6.7.1 Attributes, Methods and Connectors:

Association Name: Association Type: Generalization Stereotype:
Source Class: ContractRelation [] Target Class: AbstractOperatingModel []

Association Name: related_2 Association Type: Association Stereotype:
Source Class: ContractRelation [0..*] Target Class: Offering [0..*]
Definition: The relation association represents a leg of a potentially n-ary relationship that may exist among multiple Offerings.

7.3.6.8 Class Name: MerchandiseOffering Class Type: Class Stereotype:
Base Classes: ProductOffering
Definition: A MerchandiseOffering represents an offering to sell or lease a good to a customer who may use the good to produce Outcomes.
Usage: The MerchandiseOffering is characterized by some BusinessObjects or InformationItems that would be transferred to the Customer for use by the Customer. The BusinessObjects and/or InformationItems are objects of the MerchandiseOffering.

7.3.6.8.1 Attributes, Methods and Connectors:

Association Name: incorporates_2 Association Type: Association Stereotype: «class»
Source Class: MerchandiseOffering [0..*] Target Class: MerchandiseOutcome [0..*]
Definition: This incorporates association refines the incorporates association between the generalizing meta-classes ProductOffering and Outcome. It asserts that a MerchandiseOffering incorporates a MerchandiseOutcome.

Association Name: Association Type: Generalization Stereotype:
Source Class: MerchandiseOffering [] Target Class: ProductOffering []

Association Name: object_0 Association Type: Association Stereotype: «shortcut»
Source Class: MerchandiseOffering [0..*] Target Class: AbstractBusinessObject [0..*]
Definition: The object association represents a shortcut relationship between a MerchandiseOffering and a BusinessObject or InformationItem offered for sale or lease to the Customer.
Usage: This shortcut implies that there is an unspecified MerchandiseOutcome of the AbstractBusinessObject that would describe the terms of ownership/use incorporated in the MerchandiseOffering.

Constraint: Let MOF1 be a MerchandiseOffering and BO1 be a BusinessObject associated by o1 an "object" association. Then MOF1 should incorporate MerchandiseOutcomes \{MOj\} that represent either the change of ownership of BO1 or the establishment of a limited right to use BO1.

7.3.6.9 **Class Name:** MerchandiseOutcome  **Class Type:** Class  **Stereotype:**

**Base Classes:** Outcome

**Definition:** MerchandiseOutcome represents the transfer of ownership and/or use between the business that is selling the merchandise via the MerchandiseOffering and the LegalEntity who receives the possession and/or use of the merchandise. The LegalEntity may also be a Customer.

7.3.6.9.1 **Attributes, Methods and Connectors:**

**Association Name:** seller

**Association Type:** Association  **Stereotype:** «class»

**Source Class:** MerchandiseOutcome [0..*]  **Target Class:** theBusiness [1]

**Definition:** The seller association asserts that theBusiness is the seller of the MerchandiseOutcome incorporated in the MerchandiseOffering.

**Usage:** This association does not imply that the LegalEntity providing the MerchandiseOffering is the same as the seller of the MerchandiseOutcome.

**Association Name:** buyer

**Association Type:** Association  **Stereotype:** «class»

**Source Class:** MerchandiseOutcome [0..*]  **Target Class:** LegalEntity [0..*]

**Definition:** The buyer association is related to the accepts association and asserts that a LegalEntity (typically also a Customer) is the targeted buyer of the MerchandiseOutcome.

**Usage:** The buyer of the MerchandiseOutcome is not necessarily the LegalEntity that accepts the MerchandiseOffering in the case when the acceptor is acting as an agent for the buyer.

**Association Name:** incorporates.2

**Association Type:** Association  **Stereotype:** «class»

**Source Class:** MerchandiseOffering [0..*]  **Target Class:** MerchandiseOutcome [0..*]

**Definition:** This incorporates association refines the incorporates association between the generalizing meta-classes ProductOffering and Outcome. It asserts that a MerchandiseOffering incorporates a MerchandiseOutcome.

7.3.6.10 **Class Name:** Offering  **Class Type:** Class  **Stereotype:**

**Base Classes:** AbstractBusinessObject

**Definition:** Offering represents the solicitation of business from a Customer by presenting Outcomes and BusinessObjects that the business is willing to provide in return for items of value received from the Customer.

**Usage:** Offering is abstract because the metamodel may eventually include subtypes other than ProductOffering. Offering is provided by the business or a partner and the intended consumer is a type of Customer.

The business architecture does not include the concept of a sale directly. Sales are in the past of a business, and business architecture is focused on the possible futures of the business. Sales are useful as predictors of acceptance of future offering and as predictors of future liability for warranties.

7.3.6.10.1 **Attributes, Methods and Connectors:**
Association Name: Association Type: Generalization Stereotype:  
Source Class: Offering []  Target Class: AbstractBusinessObject []

Association Name: Association Type: Generalization Stereotype:  
Source Class: ProductOffering []  Target Class: Offering []

Association Name: related_2 Association Type: Association Stereotype:  
Source Class: ContractRelation [0..*]  Target Class: Offering [0..*]  
Definition: The relation association represents a leg of a potentially n-ary relationship that may exist among multiple Offerings.

Association Name: provides Association Type: Association Stereotype: «class»  
Source Class: LegalEntity [0..*]  Target Class: Offering [0..*]  
Definition: The provider relation represents a relationship between a LegalEntity and an Offering created by the LegalEntity that is intended to solicit the business of designated parties identified by the consumer relation.

Association Name: accepts Association Type: Association Stereotype: «class»  
Source Class: LegalEntity [0..*]  Target Class: Offering [0..*]  
Definition: The acceptor relation represents a relationship between a party external to the business and an Offering intended to solicit business from the acceptor party represented by the Customer.

Usage: Note that offering does not represent a sale; in a sale, each party gives something of value and receives something of value.

7.3.6.11 Class Name: OutsourcedServiceOffering  Class Type: Class Stereotype:  
Base Classes: ProductOffering  
Definition: OutsourcedServiceOffering represents an offering made by the business that solicits a service to be performed by another business.

7.3.6.11.1 Attributes, Methods and Connectors:

Association Name: specifies Association Type: Association Stereotype: «class»  
Source Class: OutsourcedServiceOffering [0..*]  Target Class: APCICB [0..*]  
Definition: The specifies association represents a relationship between an OutsourcingOffering and a CapabilityBehavior or Process or CapabilityImplementation, in which the Customer would be required or advised to perform the CapabilityBehavior or Process and/or provide Performers and Resources as specified by the CapabilityImplementation as an implementation of the CapabilityBehavior or Process...  
Usage: This association is effectively combined with the two other specifies relation whose source is OutsourcingOffering so that the range of the combined associations is the union of AbstractProcess, CapabilityBehavior and CapabilityImplementation.

Association Name: Association Type: Generalization Stereotype:  
Source Class: OutsourcedServiceOffering []  Target Class: ProductOffering []

Association Name: incorporates Association Type: Association Stereotype: «class»  
Source Class: OutsourcedServiceOffering [0..*]  Target Class: OutsourcedServiceOutcome [0..*]  
Definition: The incorporates association designates that an OutsourcedServiceOffering incorporates some OutsourcedServiceOutcomes.  
Usage: The incorporates association refines the incorporates association between ProductOffering and Outcome.
7.3.6.12 **Class Name:** OutsourcedServiceOutcome  **Class Type:** Class  **Stereotype:**

**Base Classes:** Outcome

**Definition:** OutsourcedServiceOutcome represents the expected Outcome of the performance of an outsourced service (i.e. a service performed for the business by another business).

### 7.3.6.12.1 Attributes, Methods and Connectors:

**Association Name:** provider_0  **Association Type:** Association  **Stereotype:** «class»

**Source Class:** OutsourcedServiceOutcome [0..*]  **Target Class:** LegalEntity [0..*]

**Definition:** The provider association asserts that a LegalEntity is the provider of the OutsourcedServiceOutcome.

**Usage:** The provider LegalEntity is not necessarily the same LegalEntity as the provider of the OutsourcedServiceOffering.

**Association Name:** recipient_0  **Association Type:** Association  **Stereotype:** «class»

**Source Class:** OutsourcedServiceOutcome [0..*]  **Target Class:** theBusiness [1]

**Definition:** The recipient association asserts that theBusiness is the intended recipient and beneficiary of the OutsourcedServiceOutcome.

**Usage:** The recipient is not necessarily the same LegalEntity that accepts the OutsourcedServiceOffering.

**Association Name:** incorporates_0  **Association Type:** Association  **Stereotype:** «class»

**Source Class:** OutsourcedServiceOffering [0..*]  **Target Class:** OutsourcedServiceOutcome [0..*]

**Definition:** The incorporates association designates that an OutsourcedServiceOffering incorporates some OutsourcedServiceOutcomes.

**Usage:** The incorporates association refines the incorporates association between ProductOffering and Outcome.

7.3.6.13 **Class Name:** ProcurementOffering  **Class Type:** Class  **Stereotype:**

**Base Classes:** ProductOffering

**Definition:** ProcurementOffering is an offering by theBusiness to purchase or lease a BusinessObject and/or InformationItem from a LegalEntity.

### 7.3.6.13.1 Attributes, Methods and Connectors:

**Association Name:** object_1  **Association Type:** Association  **Stereotype:** «shortcut»

**Source Class:** ProcurementOffering [0..*]  **Target Class:** AbstractBusinessObject [0..*]

**Definition:** The object shortcut association asserts that the ProcurementOffering incorporates unspecified Outcomes describing the states of AbstractBusinessObjects.

**Usage:** This association allows the business architect to omit the Outcome in the procurement of some AbstractBusinessObjects for use by theBusiness when those Outcomes are obvious or irrelevant to the purposes of the analysis that is using the business architecture model.

**Constraint:** Let PO1 be a ProcurementOffering and BO1 be a BusinessObject associated by o1 an "object" association. Then PO1 should incorporate ProcurementOutcomes {POj} that represent either the change of ownership of BO1 or the establishment of a limited right to use BO1.

**Association Name:** incorporates_1  **Association Type:** Association  **Stereotype:** «class»

**Source Class:** ProcurementOffering [0..*]  **Target Class:** ProcurementOutcome [0..*]
**Definition:** The *incorporates* association refines the *incorporates* association between the generalizing meta-classes *ProductOffering* and *Outcome* and asserts that the *ProcurementOffering* incorporates the *ProcurementOutcomes*.

**Association Name:**  **Association Type:** Generalization  **Stereotype:**  
**Source Class:** ProcurementOffering  
**Target Class:** ProductOffering

### 7.3.6.14 Class Name: ProcurementOutcome  
**Class Type:** Class  **Stereotype:**

**Base Classes:** Outcome  

**Definition:** ProcurementOutcome represents the expected Outcome of the procurement. E.g. that the BusinessObject/InformationItem received has the characteristics needed by the procuring business.  

**Usage:** ProcurementOutcome specifies such details and is associated with a ProcurementOffering that should not duplicate the details of the ProcurementOutcome.

#### 7.3.6.14.1 Attributes, Methods and Connectors:

**Association Name:** supplier  
**Association Type:** Association  **Stereotype:** «class»  
**Source Class:** ProcurementOutcome  
**Target Class:** LegalEntity  

**Definition:** The supplier association asserts that the LegalEntity is to be the supplier of the ProcurementOutcome.  

**Usage:** The supplier LegalEntity is not necessarily the same as the provider LegalEntity for the ProcurementOffering incorporating the ProcurementOutcome.

**Association Name:** procurer  
**Association Type:** Association  **Stereotype:** «class»  
**Source Class:** ProcurementOutcome  
**Target Class:** theBusiness  

**Definition:** The procurer association asserts that theBusiness is the procurer of the ProcurementOutcome  

**Usage:** The procurer LegalEntity is not necessarily the acceptor LegalEntity of the ProcurementOffering

**Association Name:** incorporates  
**Association Type:** Association  **Stereotype:** «class»  
**Source Class:** ProcurementOffering  
**Target Class:** ProcurementOutcome  

**Definition:** The *incorporates* association refines the *incorporates* association between the generalizing meta-classes *ProductOffering* and *Outcome* and asserts that the *ProcurementOffering* incorporates the *ProcurementOutcomes*.

### 7.3.6.15 Class Name: ProductOffering  
**Class Type:** Class  **Stereotype:**

**Base Classes:** Offering  

**Definition:** ProductOffering represents the terms and conditions associated with the acquisition of a product or service by a customer. It would typically include price, delivery terms, warranty and other aspects of these terms. The ProductOffering incorporates Outcomes such as change of possession for a product (BusinessObject or InformationItem) that is sold.  

**Usage:** A ProductOffering (and its specializations Good and Service) are a type of BusinessObject. This allows a Customer to experience the ProductOffering at a Touchpoint and develop a reaction (such as the ProductOffering being a good deal). Such a reaction can be represented as a CustomerSegment associated with the Customer and the JourneyStage that includes the Touchpoint.
7.3.6.15.1 Attributes, Methods and Connectors:

Association Name: incorporates_0  Association Type: Association  Stereotype: «class»
Source Class: ProductOffering  [0..*]  Target Class: Outcome  [0..*]

Definition: The incorporates association represents that an Outcome is included in a ProductOffering.
Usage: It may be implied that the BusinessObject whose state is represented by the Outcome is also included in the ProductOffering. In the case of a service offering, the Outcome instance represents the intended result of performing the capability as a service for a customer (as opposed to performing the capability for the immediate benefit of the business).

Association Name: Association Type: Generalization  Stereotype: Generalization
Source Class: OutsourcedServiceOffering  []  Target Class: ProductOffering  []

Association Name: Association Type: Generalization  Stereotype: Generalization
Source Class: MerchandiseOffering  []  Target Class: ProductOffering  []

Association Name: Association Type: Generalization  Stereotype: Generalization
Source Class: ProcurementOffering  []  Target Class: ProductOffering  []

Association Name: Association Type: Generalization  Stereotype: Generalization
Source Class: ServiceOffering  []  Target Class: ProductOffering  []

Association Name: of  Association Type: Association  Stereotype: «shortcut»
Source Class: ValueProposition  [0..*]  Target Class: ProductOffering  [0..*]

Definition: The of association links a ValueProposition to a ProductOffering and represents that is the ValueProposition is about the ProductOffering.
Constraint: Let VP1 be a ValueProposition and PO1 be a ProductOffering associated by o1, an "of" association. Then for some subset of ValueItems {VIj} aggregated by VP1 such that each VIj values an Outcome O1 that is incorporated in the ProductOffering PO1. Note that the ProductOfferings typically include Outcomes that are experienced by the Customer at a Touchpoint.

7.3.6.16 Class Name: ServiceOffering  Class Type: Class  Stereotype: Generalization
Base Classes: ProductOffering
Definition: ServiceOffering represents an offer to provide a service to a Customer. the business provides the CapabilityImplementations and CapabilityBehaviors needed to effect the Outcome promised to the Customer by the ServiceOffering.
Usage: A ServiceOffering is a specialization of a ProductOffering such that a Capability or CapabilityBehavior or Process or Activity is performed to produce an Outcome that is incorporated into the service. Unlike a sale or lease, where some incorporated Outcomes represent a change of ownership or possession/use of a business object, the incorporated Outcomes (such as a cleaned residence) are the primary Outcomes desired by the customer.
A business that offers a ServiceOffering must incorporate or arrange for the Capabilities and or Processes needed to produce the promised Outcomes.

7.3.6.16.1 Attributes, Methods and Connectors:

Association Name: incorporates_3  Association Type: Association  Stereotype: «class»
Source Class: ServiceOffering  [0..*]  Target Class: ServiceOutcome  [0..*]
**Definition:** The incorporates association refines the incorporates association between the generalizing meta-classes (ProductOffering and Outcome) and asserts that the ServiceOffering incorporates some ServiceOutcomes.

**Association Name:** object_2 **Association Type:** Association **Stereotype:** «shortcut»

**Source Class:** ServiceOffering [0..*] **Target Class:** AbstractCapability [0..*]

**Definition:** the object shortcut association designates an AbstractCapability possessed by theBusiness that is intended to produce the ServiceOutcome incorporated into the ServiceOffering.

**Constraint:** Let SO1 be a ServiceOffering and C1 be a Capability that is associated by o1 an object association. Then there should exist a ServiceOutcome SO1 such that SO1 is incorporated in SO1 and SO1 is produced by C1.

**Association Name:** incorporates_3 **Association Type:** Association **Stereotype:** «class»

**Source Class:** ServiceOffering [0..*] **Target Class:** ServiceOutcome [0..*]

**Definition:** The incorporates association refines the incorporates association between the generalizing meta-classes (ProductOffering and Outcome) and asserts that the ServiceOffering incorporates some ServiceOutcomes.

7.3.6.17 **Class Name:** ServiceOutcome **Class Type:** Class **Stereotype:**

**Base Classes:** Outcome

**Definition:** ServiceOutcome represents the expected Outcome of the performance of a service for a Customer.

7.3.6.17.1 Attributes, Methods and Connectors:

**Association Name:** recipient_1 **Association Type:** Association **Stereotype:** «class»

**Source Class:** ServiceOutcome [0..*] **Target Class:** LegalEntity [0..*]

**Definition:** The recipient association asserts that the LegalEntity is the recipient of ServiceOutcomes incorporated into a ServiceOffering.

**Usage:** It is not necessarily the case that the recipient LegalEntity is the same as the accepting LegalEntity of the incorporating ServiceOffering.

**Association Name:** provider_1 **Association Type:** Association **Stereotype:** «class»

**Source Class:** ServiceOutcome [0..*] **Target Class:** theBusiness [0..1]

**Definition:** The provider association asserts that theBusiness is the provider of the ServiceOutcome incorporated into a ServiceOffering.

**Usage:** The provider LegalEntity is not necessarily the same as the LegalEntity that provides the ServiceOffering that incorporates the ServiceOutcome.

**Association Name:** incorporates_3 **Association Type:** Generalization **Stereotype:**

**Source Class:** ServiceOffering [] **Target Class:** ProductOffering []

**Definition:**

7.3.7 **Package:** Strategy
7.3.7.1 **Diagram: Operating Model**
Commented [JR39]: Issue BACM-122 resolved by BACM.
The OperatingModel diagram defines metaclasses for the business entities that are changeable by a strategy and are part of the business operating model, but not part of the business value model (see the Strategy::ValueModel diagram)
7.3.7.2 **Diagram: Strategy**

The Strategy diagram defines abstract syntax for modeling strategy driven change.

BACM 1.0
A strategy is represented by a StrategyModel that contains Means and Ends. A StrategyModel also contains the Initiatives and Changes that implement the Means and Ends. Multiple StrategyModels are contained in StrategyChoices, allowing an analysis to evaluate StrategyModels and compare them with each other.

Ends represent desired changes to delivered value and/or business results. Means represent prospective ways to achieve the Ends.

Strategy is modeled at two levels:
1. High level strategy expressed as Means and Ends. Ends are statements primarily about value delivered to stakeholders (e.g., increased stock price, better customer satisfaction with a product). Means are high level statements about possible ways to achieve the Ends (e.g., reducing expenses, improving manufacturing quality).
2. Planned initiatives to implement a strategy, expressed as Initiatives and Changes. Changes represent specific objectives for Outcomes, BusinessObjects, ProductOfferings, ValuePropositions, ValueCharacteristics and ValueItems. Initiatives represent changes to be made to Capabilities, CapabilityBehaviors, CapabilityImplementations, Roles, Processes, Activities, Flows and assignments of Performers and Resources to achieve the Changes.

This abstract syntax does not distinguish the model elements changed by Ends from those changed by Means. For simplicity, it lumps these together as specializations of AbstractOperatingModel and AbstractValueModel. Implementors should follow the descriptions in items 1 and 2 above.

This abstract syntax also does not distinguish the model element changed by Change from those changed by Initiative.

### 7.3.7.3 Diagram: StrategyNeeds

The StrategyNeeds diagram displays the abstract syntax for representing that Means and Initiatives require Capabilities, CapabilityBehaviors and CapabilityImplementations for performance. This abstract syntax allows the business architect to establish abstract Capability requirements for performance of a Means and to establish both abstract and specific requirements for performance of an Initiative. The details of the requires and implements associations would be contained in strategy and initiative planning tools and their documents.
The ValueModel diagram defines the BACM meta-classes whose instances can be used to model aspects of the business which represent value or which represent characteristics of the customer. The concrete specializations of AbstractValueModel can be changed by the Ends instance of a StrategyModel instance.

Class Name: AbstractOperatingModel  Class Type: Class  Stereotype: 

Base Classes:
Definition: AbstractOperatingModel is an abstract metaclass whose concrete specializations are the model elements of the operating model (see the AbstractOperatingModel diagram). This metaclass groups together the concrete metaclasses that may be impacted by a Means or Initiative or baselined by Ends or Changes.
Usage: Means and Initiatives describe behaviors that will impact parts of the operating model of the business to achieve the Ends and Changes associated with the Means and Initiatives. While the behaviors are described by the Means and Initiatives, the affected operating model components are represented by the impacts relationship to facilitate analysis of these impacts for feasibility, risk, cost and other measures. Ends and Change describe the new state and behavior of the baselined parts of the operating model of the business. For example, an End may be the improvement of throughput and reduction of wait time for a CapabilityImplementation. The Means may be the addition of personnel and upgrading of an application. The End describes a new baseline for the CapabilityImplementation (relative to the existing baseline associated with the CapabilityImplementation). The Means describes the behaviors to be carried out with respect to the staffing and resourcing of the CapabilityImplementation.

7.3.7.5.1 Attributes, Methods and Connectors:

Association Name: Association Type: Generalization Stereotype:
Source Class: Role [] Target Class: AbstractOperatingModel []

Association Name: Association Type: Generalization Stereotype:
Source Class: AbstractProcess [] Target Class: AbstractOperatingModel []

Association Name: Association Type: Generalization Stereotype:
Source Class: Responsible [] Target Class: AbstractOperatingModel []

Association Name: Association Type: Generalization Stereotype:
Source Class: AbstractBusinessObject [] Target Class: AbstractOperatingModel []

Association Name: baseline_0 Association Type: Association Stereotype: «class»
Source Class: Ends [0..*] Target Class: AbstractOperatingModel [0..*]
Definition: The baseline association links one or more operating model elements representing business results to change objectives represented by the Ends.
Usage: An operating model Outcome (e.g. cost of executing an activity) is the baseline for an End (e.g. a 10% reduction in the cost of executing the activity).

Association Name: impacts_0 Association Type: Association Stereotype: «class»
Source Class: Means [0..*] Target Class: AbstractOperatingModel [0..*]
Definition: The impacts association links a Means (description of changes to be made to business operations) to the operating model elements that will be impacted (changed).

Association Name: Association Type: Generalization Stereotype:
Source Class: ObjectRelation [] Target Class: AbstractOperatingModel []

Association Name: baseline_2 Association Type: Association Stereotype: «class»
Source Class: Change [0..*] Target Class: AbstractOperatingModel [0..*]
Definition: The baseline association links one or more operating model elements representing business results to change objectives represented by the Changes.
Usage: An operating model Outcome (e.g. cost of executing an activity) is the baseline for a Change (e.g. a 5% reduction in the cost of executing the activity as a result of purchasing a new robot).

Association Name: Association Type: Generalization Stereotype:
Source Class: AbstractCapability [] Target Class: AbstractOperatingModel []

Association Name: Association Type: Generalization Stereotype:
Source Class: CapabilityImplementation [] Target Class: AbstractOperatingModel []
Association Name: impacts_1 Association Type: Association Stereotype: «class»
Source Class: Initiative [0..*] Target Class: AbstractOperatingModel [0..*]
Definition: The impacts association links Initiatives (planned changes to operating model elements) to the operating model elements impacted (changed) by the Initiatives.

Association Name:  Association Type: Generalization Stereotype:
Source Class: OutcomeRelation [] Target Class: AbstractOperatingModel []

Association Name:  Association Type: Generalization Stereotype:
Source Class: Outcome [] Target Class: AbstractOperatingModel []

Association Name:  Association Type: Generalization Stereotype:
Source Class: ContractRelation [] Target Class: AbstractOperatingModel []

Association Name:  Association Type: Generalization Stereotype:
Source Class: Jurisdiction [] Target Class: AbstractOperatingModel []

7.3.7.6 Class Name: AbstractValueModel  Class Type: Class Stereotype:
Base Classes:
Definition: The AbstractValueModel represents the value-related concepts that the Means and Initiative behaviors seek to achieve by changes made to the AbstractOperatingModel.
Usage: AbstractValueModel model elements represent perceptions of value as seen by a Customer or imagined by the Business to be seen by the Customer. As such, they cannot be directly changed by the business, so Means and Initiatives do not directly impact them. For example, the ValueProposition and ValueCharacteristic of an Offering may be improved by lowering its price, but this result is not guaranteed as the price action may be viewed as a signal of inflated worth or diminished quality. The architect may express a conviction that this result will occur in the expects association that links the price Means to the new Ends baseline for the ValueProposition and ValueCharacteristic.

7.3.7.6.1 Attributes, Methods and Connectors:
Association Name: baseline_1 Association Type: Association Stereotype: «class»
Source Class: Ends [0..*] Target Class: AbstractValueModel [0..*]
Definition: The baseline association links a value model element (e.g. a ValueProposition where the price of a product is equal to the competitive average price) to an End (e.g. an End that reduces the price of a product to 5% below the competitive average).

Association Name:  Association Type: Generalization Stereotype:
Source Class: ValueCharacteristic [] Target Class: AbstractValueModel []

Association Name:  Association Type: Generalization Stereotype:
Source Class: ValueProposition [] Target Class: AbstractValueModel []

Association Name:  Association Type: Generalization Stereotype:
Source Class: ValueItem [] Target Class: AbstractValueModel []


Source Class: Customer []  Target Class: AbstractValueModel []

Association Name: Association Type: Generalization Stereotype: Source Class: CustomerSegment []  Target Class: AbstractValueModel []

Association Name: Association Type: Generalization Stereotype: Source Class: ValueStream []  Target Class: AbstractValueModel []

Association Name: Association Type: Generalization Stereotype: Source Class: ValueStreamStage []  Target Class: AbstractValueModel []

Association Name: baseline_3 Association Type: Association Stereotype: =class Source Class: Change [0..*]  Target Class: AbstractValueModel [0..*]
Definition: The baseline association links a value model element (e.g. a ValueProposition where the price of a product is equal to the competitive average price) to a change (e.g. a change that reduces the price of a product by 5%).

7.3.7.7  **Class Name**: Change  **Class Type**: Class **Stereotype**: Base Classes: AbstractResult, BACMPlainEntity
Definition: Change represents desired states of business value and results as represented by the baselined elements of the AbstractOperatingModel and the AbstractValueModel. These states are expected to result from the changes described by the Initiatives.
Usage: Changes can be decomposed and share sub-Changes.

7.3.7.7.1 Attributes, Methods and Connectors:

Association Name: implements_2 Association Type: Association Stereotype: =class Source Class: Change [0..*]  Target Class: Ends [0..*]
This "implements" meta-association links a desired end of a strategy to the specific changes that are expected to result in the achievement of the end.

Association Name: baseline_2 Association Type: Association Stereotype: =class Source Class: Change [0..*]  Target Class: AbstractOperatingModel [0..*]
Definition: The baseline association links one or more operating model elements representing business results to change objectives represented by the Changes.
Usage: An operating model Outcome (e.g. cost of executing an activity) is the baseline for a Change (e.g., a 5% reduction in the cost of executing the activity as a result of purchasing a new robot).

Association Name: Association Type: Generalization Stereotype: Source Class: Change []  Target Class: BACMPlainEntity []

Association Name: baseline_3 Association Type: Association Stereotype: =class Source Class: Change [0..*]  Target Class: AbstractValueModel [0..*]
Definition: The baseline association links a value model element (e.g. a ValueProposition where the price of a product is equal to the competitive average price) to a change (e.g. a change that reduces the price of a product by 5%).

Association Name: Association Type: Generalization Stereotype: Source Class: Change []  Target Class: AbstractResult []
Association Name: expects_1 Association Type: Association Stereotype: «class»
Source Class: Initiative [0..*] Target Class: Change [0..*]
Definition: The expects association links one or more Changes that are expected to result from the Means described changes.

Association Name: needs_2 Association Type: Association Stereotype: «class»
Source Class: Initiative [0..*] Target Class: Change [0..*]
Definition: The needs association represents that one or more Changes are needed to enable the performance of the Initiatives.
Usage: This association must be instanced as an association classifier so that the modeler can express:
• a rationale for the expectation;
• note the likely influences of environmental factors, including competitive responses and regulatory actions
• risks and risk avoidance activities
Expressing these concerns may require the modeler to define additional properties and association legs at the M1 model level.

7.3.7.8 Class Name: Ends Class Type: Class Stereotype:
Base Classes: AbstractResult, BACMPlainEntity
Definition: Ends represent changes to elements representing business values, such as ValuePropositions., ValueItems and ValueCharacteristics. Ends also represent changes to business results (i.e. Outcomes, BusinessObjects, InformationItems and ProductOfferings). These element types derive from AbstractOperatingModel and AbstractValueModel.
Usage: A Ends element will typically state the desired result (e.g. improved customer satisfaction) relative to the currently achieved (baselined) result (customer satisfaction represented as an Outcome).
Ends can be decomposed into subordinate Ends. Subordinate Ends may be shared by one or more aggregator Ends.

7.3.7.8.1 Attributes, Methods and Connectors:

Association Name: baseline_1 Association Type: Association Stereotype: «class»
Source Class: Ends [0..*] Target Class: AbstractValueModel [0..*]
Definition: The baseline association links a value model element (e.g. a ValueProposition where the price of a product is equal to the competitive average price) to an End (e.g. an End that reduces the price of a product to 5% below the competitive average).

Association Name: needs_1 Association Type: Association Stereotype: «class»
Source Class: Ends [0..*] Target Class: Means [0..*]
Definition: The needs association represents that one or more Ends are needed to enable the performance of the Means
Usage: This association must be instanced as an association classifier so that the modeler can express:
• a rationale for the expectation;
• note the likely influences of environmental factors, including competitive responses and regulatory actions
• risks and risk avoidance activities
Expressing these concerns may require the modeler to define additional properties and association legs at the M1 model level.

Association Name: baseline_0 Association Type: Association Stereotype: «class»
Source Class: Ends [0..*] Target Class: AbstractOperatingModel [0..*]
Definition: The baseline association links one or more operating model elements representing business results to change objectives represented by the Ends.
Usage: An operating model Outcome (e.g. cost of executing an activity) is the baseline for an End (e.g. a 10% reduction in the cost of executing the activity).
**Association Name:** Association Type: Generalization Stereotype:  
**Source Class:** Ends [ ] **Target Class:** BACMPlainEntity [ ]

**Association Name:** Association Type: Generalization Stereotype:  
**Source Class:** Ends [ ] **Target Class:** AbstractResult [ ]

**Association Name:** expects_0 Association Type: Association Stereotype: «class»  
**Source Class:** Means [0..*] **Target Class:** Ends [0..*]  
**Definition:** The expects association represents that one or more Ends are expected to result from the changes described in the Means.  
**Usage:** This association must be instanced as an association classifier so that the modeler can express:  
- a rationale for the expectation;  
- note the likely influences of environmental factors, including competitive responses and regulatory actions  
- risks and risk avoidance activities  
Expressing these concerns may require the modeler to define additional properties and association legs at the M1 model level.

**Association Name:** implements_2 Association Type: Association Stereotype: «class»  
**Source Class:** Change [0..*] **Target Class:** Ends [0..*]  
This “implements” meta-association links a desired end of a strategy to the specific changes that are expected to result in the achievement of the end.

**Association Name:** aggregates_5 Association Type: Association Stereotype:  
**Source Class:** StrategyModel [0..*] **Target Class:** Ends [0..*]  
This “aggregates” association represents participation of End instances in a StrategyModel instance. This "aggregates" association and the "aggregates" association that summarizes End instancea to other End instances are not exclusive.

### 7.3.7.9 **Class Name:** Initiative **Class Type:** Class Stereotype:  
**Base Classes:** AbstractAction, BACMPlainEntity  
**Definition:** Initiatives represent plans to change business functions in order to achieve the business results described by Changes. Initiatives should be linked to the expected Changes with the expects association.  
**Usage:** Initiatives may be decomposed and may share sub-Initiatives.

#### 7.3.7.9.1 Attributes, Methods and Connectors:

**Association Name:** Association Type: Generalization Stereotype:  
**Source Class:** Initiative [ ] **Target Class:** BACMPlainEntity [ ]

**Association Name:** require_1 Association Type: Association Stereotype: «class»  
**Source Class:** Initiative [0..*] **Target Class:** AbstractCapability [0..*]  
**Definition:** The require association represents that a Capability and/or CapabilityBehavior is required for performance of the Initiative.  
**Usage:** Definition of this association in an M1 level model allows the business architect to record and analyze such requirements.

**Association Name:** expects_1 Association Type: Association Stereotype: «class»  
**Source Class:** Initiative [0..*] **Target Class:** Change [0..*]  
**Definition:** The expects association links one or more Changes that are expected to result from the Means described changes.
Association Name: needs_2 Association Type: Association Stereotype: «class»
Source Class: Initiative [0..*] Target Class: Change [0..*]
Definition: The needs association represents that one or more Changes are needed to enable the performance of the Initiatives.
Usage: This association must be instanced as an association classifier so that the modeler can express:
• a rationale for the expectation;
• note the likely influences of environmental factors, including competitive responses and regulatory actions
• risks and risk avoidance activities
Expressing these concerns may require the modeler to define additional properties and association legs at the M1 model level.

Association Name: implements_4 Association Type: Association Stereotype: «class»
Source Class: Initiative [0..*] Target Class: Means [0..*]
Definition: The implements association represents the assertion that an initiative implements a Means.

Association Name: impacts_1 Association Type: Association Stereotype: «class»
Source Class: Initiative [0..*] Target Class: AbstractOperatingModel [0..*]
Definition: The impacts association links Initiatives (planned changes to operating model elements) to the operating model elements impacted (changed) by the Initiatives.

Association Name: Association Type: Generalization Stereotype:
Source Class: Initiative [] Target Class: AbstractAction []

Association Name: implements_3 Association Type: Association Stereotype: «class»
Source Class: CapabilityImplementation [0..*] Target Class: Initiative [0..*]
Definition: The implements association represents an assertion that one or more CapabilityImplementations are required to perform the Initiative.
Usage: Definition of this association in an M1 level model allows the business architect to record that specific CapabilityImplementations are needed to perform the Initiative.

7.3.7.10 Class Name: Means Class Type: Class Stereotype:
Base Classes: AbstractAction, BACMPlainEntity
Definition: Means represent possible behaviors that will change functional elements of the business (represented by Capabilities, CapabilityBehaviors, CapabilityImplementations, Processes, Activities, Roles, Performers and Resources). These changes are expected to produce the changes represented by the Ends. Each End should be expected to result from the changes described by one or more Means.
Usage: Means can be decomposed and subordinate Means may be shared by aggregator Means.

7.3.7.10.1 Attributes, Methods and Connectors:

Association Name: expects_0 Association Type: Association Stereotype: «class»
Source Class: Means [0..*] Target Class: Ends [0..*]
Definition: The expects association represents that one or more Ends are expected to result from the changes described in the Means.
Usage: This association must be instanced as an association classifier so that the modeler can express:
• a rationale for the expectation;
• note the likely influences of environmental factors, including competitive responses and regulatory actions
• risks and risk avoidance activities
Expressing these concerns may require the modeler to define additional properties and association legs at the M1 model level.
Association Name: Association Type: Generalization Stereotype:  
Source Class: Means [] Target Class: BACMPlainEntity []

Association Name: impacts_0 Association Type: Association Stereotype: «class»  
Source Class: Means [0..*] Target Class: AbstractOperatingModel [0..*]  
Definition: The impacts association links a Means (description of changes to be made to business operations) to the operating model elements that will be impacted (changed).

Association Name: Association Type: Generalization Stereotype:  
Source Class: Means [] Target Class: AbstractAction []

Association Name: require_0 Association Type: Association Stereotype: «class»  
Source Class: Means [0..*] Target Class: AbstractCapability [0..*]  
Definition: The require association represents that a Capability and/or CapabilityBehavior is required for performance of the Means.  
Usage: Definition of this association in an M1 level model allows the business architect to record and analyze such requirements.

Association Name: needs_1 Association Type: Association Stereotype: «class»  
Source Class: Ends [0..*] Target Class: Means [0..*]  
Definition: The needs association represents that one or more Ends are needed to enable the performance of the Means  
Usage: This association must be instanced as an association classifier so that the modeler can express:  
• a rationale for the expectation;  
• note the likely influences of environmental factors, including competitive responses and regulatory actions  
• risks and risk avoidance activities  
Expressing these concerns may require the modeler to define additional properties and association legs at the M1 model level.

Association Name: implements_4 Association Type: Association Stereotype: «class»  
Source Class: Initiative [0..*] Target Class: Means [0..*]  
Definition: The implements association represents the assertion that an initiative implements a Means.

Association Name: aggregates_4 Association Type: Association Stereotype:  
Source Class: StrategyModel [0..*] Target Class: Means [0..*]  
The "aggregates" meta-association represents the inclusion of Means instances into a StrategyModel instance.

7.3.7.11 Class Name: StrategyChoices Class Type: Class Stereotype:  
Base Classes: BACMPlainEntity  
Definition: The StrategyChoices represents a suite of strategies that can be evaluated for selection. Each StrategyModel in a StrategyChoices element shall be considered as alternatives. Alternative StrategyModels may share Means, Ends, Initiatives and Changes.  
Usage: There may be at most a single instance of StrategyChoices in a BACM model.

7.3.7.11.1 Attributes, Methods and Connectors:

Association Name: alternative Association Type: Association Stereotype:  
Source Class: StrategyChoices [0..*] Target Class: StrategyModel [0..*]
The alternatives association connects two or more StrategyModels to a StrategyChoices. Each StrategyModel alternative contained in a StrategyChoices should be taken as alternative strategies for evaluation and comparison.

Association Name: Association Type: Generalization Stereotype:
Source Class: StrategyChoices [] Target Class: BACMPlainEntity []

Association Name: strategy_choices Association Type: Association Stereotype:
Source Class: BACM_Model [1] Target Class: StrategyChoices [0..*]
Definition: strategy_choices links a set of StrategyChoices to a BACMModel.
Usage: To facilitate reuse of the BACM model in different strategy situations, multiple StrategyChoices may be associated with a BACMModel.

7.3.7.12 Class Name: StrategyModel Class Type: Class Stereotype:
Base Classes: BACMPlainEntity
Definition: StrategyModel is a collection of Means and Ends and the Initiatives and Changes implementing the Means and Ends. It represents a single, coherent and complete strategy.
Usage: StrategyModels each represent a particular strategy choice. StrategyModels may share sub-StrategyModels. The set of StrategyModels as prepared by the architect and strategist is represented by the StrategyChoices model element and the alternative associations linking it to each StrategyModel

7.3.7.12.1 Attributes, Methods and Connectors:

Association Name: Association Type: Generalization Stereotype:
Source Class: StrategyModel [] Target Class: BACMPlainEntity []

Association Name: aggregates_5 Association Type: Association Stereotype:
Source Class: StrategyModel [0..*] Target Class: Ends [0..*]
This "aggregates" association represents participation of End instances in a StrategyModel instance. This "aggregates" association and the "aggregates" association that summarizes End instances to other End instances are not exclusive.

Association Name: aggregates_4 Association Type: Association Stereotype:
Source Class: StrategyModel [0..*] Target Class: Means [0..*]
The "aggregates" meta-association represents the inclusion of Means instances into a StrategyModel instance.

Association Name: alternative Association Type: Association Stereotype:
Source Class: StrategyChoices [0..*] Target Class: StrategyModel [0..*]
The alternatives association connects two or more StrategyModels to a StrategyChoices. Each StrategyModel alternative contained in a StrategyChoices should be taken as alternative strategies for evaluation and comparison.

8 Shortcuts and Touchpoints (normative)
8.1 Shortcuts

8.1.1 Definition

UML allows properties such as attributes and owned ends to be defined as virtual; associations may also be marked as virtual. This means that the value of the property or the links of the association are to be computed according to some specification rather than being represented explicitly. However, the computation may simply consist of retrieving the stored value or retrieving an explicit link. This mechanism can be used, along with a constraint language such as OCL, to insure that a high level association is grounded in a chain of lower level associations and classes.

The mechanism is represented in the metamodel abstract syntax by applying a "<<shortcut>>" stereotype to a UML association. The stereotype is accompanied by documentation that describes the constraint that should be applied to insure that details involving the same endpoint classes are consistent with the intent of the <<shortcut>> association. Since the architect is not required to provide details when asserting a <<shortcut>> association, these constraints are not invariants, but should be evaluated on demand by the architect to check the model for consistency and completeness.

The normative XMI expresses these constraints in OCL. A conforming implementation may use OCL or may use an equivalent mechanism. The MOF XMI expresses shortcut constraints as OpaqueConstraints whose language is OCL 2.0. The OCL expresses the semantic of the “hasDetail” Boolean function defined in the BACMShortcut abstract class.

8.1.2 Compliance

An implementor may but is not required to implement a mechanism to evaluate the consistency of a shortcut constraint with respect to a model as described in this section. However, an implementor must represent, make visible and preserve across model saves, import and export, any shortcuts specified in this metamodel or defined by business architecture modelers. A conforming implementation may advise a business architecture modeler that a model contains shortcuts that will not be validated by the implementation. For example, a conforming implementation may implement a meta-model shortcut as a class-association or an n-ary class-association and preserve the specification of the shortcut semantics as a tagged text value or a similar scheme.

8.2 Touchpoints

Touchpoints are intended to link a BACM model to one or more other models. A touchpoint shall be able to access elements of another model and specify the potentially complex relationship that may exist between multiple BACM model elements and multiple elements or sections in the external model, document or dataset. Touchpoints are specified as external relationships. As a default, an IRI may be used to identify or dereference a resource and a natural language description may be given as the external reference specification that describes the mapping between the BACM elements and information or model elements in the external model.

A BACM model does not represent everything that is interesting about a business. It does not adequately represent strategic planning, resource management, business processes, IT architecture or market campaigns for example. It should be able to link to models of such domains and extract information from those models. In addition, the BACM model should serve as a guide to details about the business that are represented in other models. This guide function reduces the need for the analyst to search through unorganized business models looking for information relevant to the current analysis project.
Annex A:
(normative)

A.1 Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>AbstractAction</td>
<td><em>AbstractAction</em> is used to classify entities that should be disjoint from <em>Capability</em>, <em>AbstractResult</em> and <em>AbstractThing</em>. It is not used for any other purpose in the metamodel.</td>
</tr>
<tr>
<td>AbstractResult</td>
<td><em>AbstractResult</em> is used to classify entities that should be disjoint from <em>Capability</em>, <em>AbstractAction</em> and <em>AbstractThing</em>. It is not used for any other purpose in the metamodel.</td>
</tr>
<tr>
<td>AbstractThing</td>
<td><em>AbstractThing</em> is used to classify entities that should be disjoint from <em>Capability</em>, <em>AbstractResult</em> and <em>AbstractAction</em>. It is not used for any other purpose in the metamodel.</td>
</tr>
<tr>
<td>Annotation</td>
<td><strong>Definition:</strong> <em>Annotation</em> provides the modeler an ability to associate tag/value pairs to any <em>BACMElement</em> in a BACM model.</td>
</tr>
<tr>
<td>BACM_Model</td>
<td><strong>Definition:</strong> The BACMModel represents the root element of a BACM model (i.e. the element from which a tool or person can navigate to every other element in the model)</td>
</tr>
<tr>
<td>BACMBinDirRelation</td>
<td><strong>Definition:</strong> BACMBinDirRelation is an abstract class that generalizes the classes resulting from the transformation of model associations stereotyped as &lt;&lt;class&gt;&gt; or &lt;&lt;shortcut&gt;&gt;. It specializes BACMRelation to represent binary directed relations and redefines the association between BACMRelation and BACMEntity to designate the start (from_bacm_entity) and end (to_bacm_entity) of the relation direction</td>
</tr>
<tr>
<td>BACMElement</td>
<td><strong>Definition:</strong> The BACMElement represents the class of all elements in a BACM model. It provides elements with a name and description and allows elements to be annotated.</td>
</tr>
<tr>
<td>BACMEntity</td>
<td><strong>Definition:</strong> BACMEntity is an abstract class that is characterized by participating in relationships defined by BACMRelation and BACMBinDirRelation. BACMEntity is also a generalization of all classes intended to represent concepts of the modeled business. See the normative XMI file for details.</td>
</tr>
<tr>
<td>BACMPlainEntity</td>
<td><strong>Definition:</strong> BACMPlainEntity is an abstract class disjoint from BACMRelation that classifies all BACM classes representing concepts of the modeled business that are not specializations of BACMRelation.</td>
</tr>
<tr>
<td>BACMRelation</td>
<td><strong>Definition:</strong> BACMRelation is an abstract class that models n-ary relations with features and the ability to participate in other specializations and instances of this class as bacm_entity ends.</td>
</tr>
<tr>
<td>BACMShortcut</td>
<td><strong>Definition:</strong> BACMShortcut is an abstract class inherited by the transformation of all metamodel classes stereotyped as &lt;&lt;shortcut&gt;&gt; and all generated classes that result from the transformation of model classes stereotyped as &lt;&lt;shortcut&gt;&gt;. It declares a string (constr) that defines the shortcut constraint and a boolean valued function (hasDetail) that evaluates the constraint string and determines whether it is true or false.</td>
</tr>
<tr>
<td>BusinessElement</td>
<td><strong>Definition:</strong> BusinessElement represents a concept or entity that existing or is planned to exist in the business.</td>
</tr>
<tr>
<td>ExternalData</td>
<td><strong>Definition:</strong> ExternalData is a class that wraps an IRI. An ExternalRelationship instance may be associated with multiple ExternalData instances.</td>
</tr>
</tbody>
</table>
| ExternalRelationship  | **Definition:** ExternalRelationship represents a relationship between a BusinessElement in a provider tool or repository to ExternalData in another tool or Repository. The external data may be a BusinessElement (or a linked collection of BusinessElements) or some other element (or
linked collection of elements) from a model that is not a BACM model. The IRI must identify a resource to which the specification String can be applied to identify the element (or linked set of elements) in that resource. The language attribute of the `ExternalRelationship` identifies the language of the specification String.

Note that `BusinessElement` classifies all BACM classes and associations that are intended to represent business concepts (as opposed to model concepts or analysis concepts).

<table>
<thead>
<tr>
<th>IRI</th>
<th>Definition: Specializes PrimitiveTypes#String to match the regular expression defining a legal IRI.</th>
</tr>
</thead>
<tbody>
<tr>
<td>AbstractBusinessObject</td>
<td>Definition: AbstractBusinessObject represents BusinessObjects or InformationItems.</td>
</tr>
<tr>
<td>AbstractCapability</td>
<td>Definition: AbstractCapability is not intended to represent a business concept. It is a metamodeling device to provide relationships to Capability and CapabilityBehavior that would otherwise be duplicated.</td>
</tr>
<tr>
<td>BusinessObject</td>
<td>Definition: BusinessObject represents a tangible thing that is of significance to a business.</td>
</tr>
<tr>
<td>Capability</td>
<td>Definition: Capability represents generalization over variations in behavior and variations in structure applied to the behavior where the same general Outcome is produced by the behavior. A Capability represents the ability a business has to produce an Outcome without specifying how that Outcome is produced.</td>
</tr>
<tr>
<td>CapabilityBehavior</td>
<td>Definition: CapabilityBehavior represents a behavior description or specification, such as process diagrams, procedures manuals and other means of recording and publishing expected business practices.</td>
</tr>
<tr>
<td>CapabilityImplementation</td>
<td>Definition: The CapabilityImplementation represents a collection of Resources and Performers that may be used to implement a Capability or CapabilityBehavior (see the Roles diagram).</td>
</tr>
<tr>
<td>InformationItem</td>
<td>Definition: The InformationItem represents a kind of information.</td>
</tr>
<tr>
<td>ObjectRelation</td>
<td>Definition: ObjectRelation represents any relationship of any arity among BusinessObjects and InformationItems.</td>
</tr>
<tr>
<td>Outcome</td>
<td>Definition: An Outcome represents a fact or collection of facts about an experienced state of affairs pertaining to one or more BusinessObjects and/or InformationItems. Outcomes are produced needed by land outputs/inputs of AbstractProcesses.</td>
</tr>
<tr>
<td>OutcomeRelation</td>
<td>Definition: OutcomeRelation represents any kind of semantic relationship between Outcomes.</td>
</tr>
<tr>
<td>PerformerRole</td>
<td>Definition: PerformerRole represents skills, knowledge and willingness to use these in the production of the Outcomes of a Capability.</td>
</tr>
<tr>
<td>ResourceRole</td>
<td>Definition: ResourceRole represents the set of roles that must be fulfilled by business entities that are passive participants in the Capability, CapabilityBehavior, Process or Activity. This includes tools, locations and materials that are used in the behavior but do not become incorporated into the Outcome of the behavior. Any materials or entities that are incorporated into a BusinessObject or InformationItem whose Outcomes are produced by the Capability or CapabilityBehavior should be represented as BusinessObjects or InformationItems associated with Outcomes needed by the Capability and not represented as Resources in this context.</td>
</tr>
<tr>
<td>Role</td>
<td>Definition: Role represents a specified way for an entity to participate in producing the Outcome of a Capability or a Process. However, only the concrete subclasses of Role may be used in a model.</td>
</tr>
<tr>
<td>Customer</td>
<td>Definition: Customer represents a customer type or a class of customers. Customer also represents partner businesses and other forms of contracted business relationships.</td>
</tr>
<tr>
<td>Term</td>
<td>Definition</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CustomerJourney</td>
<td>A CustomerJourney represents a sequence of stages through which a Customer may pass with respect to a ProductOffering and its ValueProposition. The CustomerJourneyStages of the CustomerJourney capture the notion that the customer experience is cumulative.</td>
</tr>
<tr>
<td>CustomerJourneyStage</td>
<td>The CustomerJourneyStage represents a significant stage in the CustomerJourney. An example of the stages of a customer journey would be: awareness, seeking a solution, weighting alternatives, acquiring the solution, using the solution, disposing the solution.</td>
</tr>
<tr>
<td>CustomerSegment</td>
<td>The CustomerSegment represents a characteristic of the Customer or a component of customer state of mind. CustomerSegments are owned by the Customer they describe.</td>
</tr>
<tr>
<td>JSTP</td>
<td></td>
</tr>
<tr>
<td>Touchpoint</td>
<td>The Touchpoint represents an interaction between the business and the Customer.</td>
</tr>
<tr>
<td>ValueCharacteristic</td>
<td>ValueCharacteristic represents the fit between the ValueProposition of a ProductOffering targeted at a Customer.</td>
</tr>
<tr>
<td>ValueItem</td>
<td>A ValueItem represents the business belief that a Customer will value one or more Outcomes that are experienced by the Customer.</td>
</tr>
<tr>
<td>ValueProposition</td>
<td>The ValueProposition represents a collection of values the business believes it is offering to customers, partners and other stakeholders through a ProductOffering.</td>
</tr>
<tr>
<td>ValueStream</td>
<td>A ValueStream represents a set of stages that accumulate value represented by the ValueProposition.</td>
</tr>
<tr>
<td>ValueStreamStage</td>
<td>ValueStreamStages represent significant points of value creation in a ValueStream.</td>
</tr>
<tr>
<td>Jurisdiction</td>
<td>The Jurisdiction represents a legal jurisdictions with powers to charter and/or regulate businesses.</td>
</tr>
<tr>
<td>LegalEntity</td>
<td>LegalEntity represents a human organization that is subject to the laws and regulations of a Jurisdiction.</td>
</tr>
<tr>
<td>OrgUnit</td>
<td>The OrgUnit meta-class represents the various types of human organizations and individuals capable of acting as performers.</td>
</tr>
<tr>
<td>Performer</td>
<td>The Performer represents entities that are capable of performing PerformerRoles. Performer has two specializations: OrgUnit and System, representing a human components of the business or a system.</td>
</tr>
<tr>
<td>Resource</td>
<td>Resource represents an entity that is required or needed by a ResourceRole but is not a Performer and does not become a part of a BusinessObject or InformationItem associated with any Outcome produced by the Capability or CapabilityBehavior.</td>
</tr>
<tr>
<td>Responsible</td>
<td>Responsible represents an unspecified kind of responsibility relationship between a source OrgUnit and a target OrgUnit. This relationship may also include a BusinessElement that defines the nature of the association.</td>
</tr>
<tr>
<td>System</td>
<td>The System represents the concept of a non-human performer, such as an IT system or a robot. Tools such as jigs and drills are not considered Performers for the purpose of business architecture. They should be modeled as Resources.</td>
</tr>
<tr>
<td>theBusiness</td>
<td>theBusiness represents the particular business that is the subject of the business architecture model.</td>
</tr>
<tr>
<td>AbstractProcess</td>
<td>AbstractProcess is not intended to represent a business concept. It is a metamodeling technical device to share relationships with Process and Activity that would otherwise need to be duplicated.</td>
</tr>
<tr>
<td>Activity</td>
<td>Activities represent atomic (non-decomposable) activities.</td>
</tr>
<tr>
<td>Process</td>
<td>Process represents an aggregation of Activities and other Processes.</td>
</tr>
<tr>
<td><strong>APCICB</strong></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td><strong>ContractRelation</strong></td>
<td><strong>Definition:</strong> ContractRelation represents any kind of relationship between Offerings.</td>
</tr>
<tr>
<td><strong>MerchandiseOffering</strong></td>
<td><strong>Definition:</strong> A MerchandiseOffering represents an offering to sell or lease a good to a customer who may use the good to produce Outcomes.</td>
</tr>
<tr>
<td><strong>MerchandiseOutcome</strong></td>
<td><strong>Definition:</strong> MerchandiseOutcome represents the transfer of ownership and/or use between the business that is selling the merchandise via the MerchandiseOffering and the LegalEntity who receives the possession and/or use of the merchandise. The LegalEntity may also be a Customer.</td>
</tr>
<tr>
<td><strong>Offering</strong></td>
<td><strong>Definition:</strong> Offering represents the solicitation of business from a Customer by presenting Outcomes and BusinessObjects that the business is willing to provide in return for items of value received from the Customer.</td>
</tr>
<tr>
<td><strong>OutsourcedServiceOffering</strong></td>
<td><strong>Definition:</strong> OutsourcedServiceOffering represents an offering made by the business that solicits a service to be performed by another business.</td>
</tr>
<tr>
<td><strong>OutsourcedServiceOutcome</strong></td>
<td><strong>Definition:</strong> OutsourcedServiceOutcome represents the expected Outcome of the performance of an outsourced service (i.e. a service performed for the business by another business).</td>
</tr>
<tr>
<td><strong>ProcurementOffering</strong></td>
<td><strong>Definition:</strong> ProcurementOffering is an offering by the Business to purchase or lease a BusinessObject and/or InformationItem from a LegalEntity.</td>
</tr>
<tr>
<td><strong>ProcurementOutcome</strong></td>
<td><strong>Definition:</strong> ProcurementOutcome represents the expected Outcome of the procurement. E.g. that the BusinessObject/InformationItem received has the characteristics needed by the procuring business.</td>
</tr>
<tr>
<td><strong>ProductOffering</strong></td>
<td><strong>Definition:</strong> ProductOffering represents the terms and conditions associated with the acquisition of a product or service by a customer. It would typically include price, delivery terms, warranty and other aspects of these terms. The ProductOffering incorporates Outcomes such as change of possession for a product (BusinessObject or InformationItem) that is sold.</td>
</tr>
<tr>
<td><strong>ServiceOffering</strong></td>
<td><strong>Definition:</strong> ServiceOffering represents an offer to provide a service to a Customer. The business provides the CapabilityImplementations and CapabilityBehaviors needed to effect the Outcome promised to the Customer by the ServiceOffering.</td>
</tr>
<tr>
<td><strong>ServiceOutcome</strong></td>
<td><strong>Definition:</strong> ServiceOutcome represents the expected Outcome of the performance of a service for a Customer.</td>
</tr>
<tr>
<td><strong>MeasureLibrary</strong></td>
<td><strong>Definition:</strong> The MeasureLibrary represents a collection of measures (measurement types) that define measurements associated with BACM model elements.</td>
</tr>
<tr>
<td><strong>Measurement</strong></td>
<td><strong>Definition:</strong> Measurement represents an actual measurement, which includes the time and other attributes of the measurement and the measure (measurement type).</td>
</tr>
<tr>
<td><strong>Scope</strong></td>
<td><strong>Definition:</strong> The Scope class represents a collection of BACM BusinessElements that define the scope of a Measurement.</td>
</tr>
<tr>
<td><strong>SmmModel</strong></td>
<td><strong>Definition:</strong> An SmmModel represents the collection of MeasureLibraries and other entities needed to implement a set of measures appropriate to a business architecture model based on the BACM specification.</td>
</tr>
<tr>
<td><strong>AbstractOperatingModel</strong></td>
<td><strong>Definition:</strong> AbstractOperatingModel is an abstract metaclass whose concrete specializations are the model elements of the operating model (see the AbstractOperatingModel diagram). This metaclass groups together the concrete metaclasses that may be impacted by a Means or Initiative or baselined by Ends or Changes.</td>
</tr>
<tr>
<td><strong>AbstractValueModel</strong></td>
<td><strong>Definition:</strong> The AbstractValueModel represents the value-related concepts that the Means and Initiative behaviors seek to achieve by changes made to the AbstractOperatingModel.</td>
</tr>
<tr>
<td><strong>Change</strong></td>
<td><strong>Definition:</strong> Change represents desired states of business value and results as represented by the baselined elements of the AbstractOperatingModel and the AbstractValueModel. These states are expected to result from the changes described by the Initiatives.</td>
</tr>
<tr>
<td>Term</td>
<td>Definition</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Ends</td>
<td>Definition: Ends represent changes to elements representing business values, such as ValuePropositions, ValueItems and ValueCharacteristics. Ends also represent changes to business results (i.e. Outcomes, BusinessObjects, InformationItems and ProductOfferings). These element types derive from AbstractOperatingModel and AbstractValueModel.</td>
</tr>
<tr>
<td>Initiative</td>
<td>Definition: Initiatives represent plans to change business functions in order to achieve the business results described by Changes. Initiatives should be linked to the expected Changes with the expects association.</td>
</tr>
<tr>
<td>Means</td>
<td>Definition: Means represent possible behaviors that will change functional elements of the business (represented by Capabilities, CapabilityBehaviors, CapabilityImplementations, Processes, Activities, Roles, Performers and Resources). These changes are expected to produce the changes represented by the Ends. Each End should be expected to result from the changes described by one or more Means.</td>
</tr>
<tr>
<td>StrategyModel</td>
<td>Definition: StrategyModel is a collection of Means and Ends and the Initiatives and Changes implementing the Means and Ends. It represents a single, coherent and complete strategy.</td>
</tr>
</tbody>
</table>