



An OMG® Systems Modeling Publication



# OMG Systems Modeling Language™ (SysML®)

Version 2.0

## Part 2: SysML v1 to SysML v2 Transformation

---

**OMG Document Number:** formal/2025-09-05

**Date:** September 2025

**Standard document URL:** <https://www.omg.org/spec/SysML/2.0/Transformation/>

**Machine Readable File(s):** <https://www.omg.org/spec/SysML/20250201/>

---



Copyright © 2019-2025, 88solutions Corporation  
Copyright © 2019-2025, Airbus  
Copyright © 2019-2025, Aras Corporation  
Copyright © 2019-2025, Association of Universities for Research in Astronomy (AURA)  
Copyright © 2019-2025, BigLever Software  
Copyright © 2019-2025, Boeing  
Copyright © 2022-2025, Budapest University of Technology and Economics  
Copyright © 2021-2025, Commissariat à l'énergie atomique et aux énergies alternatives (CEA)  
Copyright © 2019-2025, Contact Software GmbH  
Copyright © 2019-2025, Dassault Systèmes (No Magic)  
Copyright © 2019-2025, DSC Corporation  
Copyright © 2020-2025, DEKonsult  
Copyright © 2020-2025, Delligatti Associates LLC  
Copyright © 2019-2025, The Charles Stark Draper Laboratory, Inc.  
Copyright © 2020-2025, ESTACA  
Copyright © 2023-2025, Galois, Inc.  
Copyright © 2019-2025, GfSE e.V.  
Copyright © 2019-2025, George Mason University  
Copyright © 2019-2025, IBM  
Copyright © 2019-2025, Idaho National Laboratory  
Copyright © 2019-2025, INCOSE  
Copyright © 2019-2025, Intercax LLC  
Copyright © 2019-2025, Jet Propulsion Laboratory (California Institute of Technology)  
Copyright © 2019-2025, Kenntnis LLC  
Copyright © 2020-2025, Kungliga Tekniska högskolan (KTH)  
Copyright © 2019-2025, LightStreet Consulting LLC  
Copyright © 2019-2025, Lockheed Martin Corporation  
Copyright © 2019-2025, Maplesoft  
Copyright © 2021-2025, MID GmbH  
Copyright © 2020-2025, MITRE  
Copyright © 2019-2025, Model Alchemy Consulting  
Copyright © 2019-2025, Model Driven Solutions, Inc.  
Copyright © 2019-2025, Model Foundry Pty. Ltd.  
Copyright © 2023-2025, Object Management Group, Inc.  
Copyright © 2019-2025, On-Line Application Research Corporation (OAC)  
Copyright © 2019-2025, oose eG  
Copyright © 2019-2025, Østfold University College  
Copyright © 2019-2025, PTC  
Copyright © 2020-2025, Qualtech Systems, Inc.  
Copyright © 2019-2025, SAF Consulting  
Copyright © 2019-2025, Simula Research Laboratory AS  
Copyright © 2019-2025, System Strategy, Inc.  
Copyright © 2019-2025, Thematix Partners, LLC  
Copyright © 2019-2025, Tom Sawyer  
Copyright © 2023-2025, Tucson Embedded Systems, Inc.  
Copyright © 2019-2025, Universidad de Cantabria  
Copyright © 2019-2025, University of Alabama in Huntsville  
Copyright © 2019-2025, University of Detroit Mercy  
Copyright © 2019-2025, University of Kaiserslauten  
Copyright © 2020-2025, Willert Software Tools GmbH (SodiusWillert)



## 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<sup>®</sup>, CORBA logos<sup>®</sup>, FIBO<sup>®</sup>, Financial Industry Business Ontology<sup>®</sup>, Financial Instrument Global Identifier<sup>®</sup>, IIOP<sup>®</sup>, IMM<sup>®</sup>, Model Driven Architecture<sup>®</sup>, MDA<sup>®</sup>, Object Management Group<sup>®</sup>, OMG<sup>®</sup>, OMG Logo<sup>®</sup>, SoaML<sup>®</sup>, SOAML<sup>®</sup>, SysML<sup>®</sup>, UAF<sup>®</sup>, Unified Modeling Language<sup>™</sup>, UML<sup>®</sup>, UML Cube Logo<sup>®</sup>, VSIPL<sup>®</sup>, and XMI<sup>®</sup> are registered trademarks of the Object Management Group, Inc.

For a complete list of trademarks, see: [https://www.omg.org/legal/tm\\_list.htm](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<sup>®</sup> (MDA<sup>®</sup>), 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<sup>®</sup> (Unified Modeling Language<sup>™</sup>); CORBA<sup>®</sup> (Common Object Request Broker Architecture); CWM<sup>™</sup> (Common Warehouse Metamodel); and industry-specific standards for dozens of vertical markets.

More information on the OMG is available at <https://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](mailto:pubs@omg.org)

Certain OMG specifications are also available as ISO standards. Please consult <https://www.iso.org>

## Issues

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 Specifications, Report an Issue.



# Table of Contents

1 Scope .....	1
2 Conformance .....	3
3 Normative References .....	5
4 Terms and Definitions .....	7
5 Symbols .....	9
6 Introduction .....	11
6.1 Mapping Approach .....	11
6.2 Acknowledgements .....	11
7 Mappings .....	13
7.1 Overview .....	13
7.2 Foundations .....	13
7.2.1 Overview .....	13
7.2.2 Foundational class specifications .....	14
7.2.2.1 UniqueMapping .....	14
7.2.2.2 Factory .....	14
7.2.2.3 Mapping .....	14
7.2.2.4 MainMapping .....	15
7.2.2.5 Initializer .....	16
7.3 Mapping Helper and Library .....	16
7.3.1 Helper .....	16
7.3.2 SysML v1 Library .....	22
7.4 Initializers .....	25
7.4.1 Overview .....	25
7.4.2 Mapping Specifications .....	25
7.4.2.1 KerML Initializers .....	25
7.4.2.1.1 ToAnnotatingElement_Init .....	25
7.4.2.1.2 ToAnnotation_Init .....	25
7.4.2.1.3 ToAssociation_Init .....	26
7.4.2.1.4 ToBehavior_Init .....	26
7.4.2.1.5 ToClassifier_Init .....	26
7.4.2.1.6 ToComment_Init .....	27
7.4.2.1.7 ToConjugation_Init .....	27
7.4.2.1.8 ToConnector_Init .....	27
7.4.2.1.9 ToDocumentation_Init .....	28
7.4.2.1.10 ToElement_Init .....	28
7.4.2.1.11 ToEndFeatureMembership_Init .....	29
7.4.2.1.12 ToExpression_Init .....	29
7.4.2.1.13 ToFeature_Init .....	29
7.4.2.1.14 ToFeatureChainExpression_Init .....	30
7.4.2.1.15 ToFeatureChaining_Init .....	31
7.4.2.1.16 ToFeatureMembership_Init .....	31
7.4.2.1.17 ToFeatureReferenceExpression_Init .....	31
7.4.2.1.18 ToFeatureTyping_Init .....	32
7.4.2.1.19 ToFeatureValue_Init .....	32
7.4.2.1.20 ToFlow_Init .....	33
7.4.2.1.21 ToFunction_Init .....	33
7.4.2.1.22 ToImport_Init .....	33
7.4.2.1.23 ToInteraction_Init .....	34
7.4.2.1.24 ToInvocationExpression_Init .....	34
7.4.2.1.25 ToMembership_Init .....	34
7.4.2.1.26 ToMembershipImport_Init .....	35
7.4.2.1.27 ToNamespace_Init .....	35
7.4.2.1.28 ToNamespaceImport_Init .....	36
7.4.2.1.29 ToOperatorExpression_Init .....	36

7.4.2.1.30 ToOwningMembership_Init.....	36
7.4.2.1.31 ToPackage_Init.....	37
7.4.2.1.32 ToParameterMembership_Init.....	37
7.4.2.1.33 ToPredicate_Init.....	37
7.4.2.1.34 ToRedefinition_Init.....	38
7.4.2.1.35 ToReferenceSubsetting_Init.....	38
7.4.2.1.36 ToRelationship_Init.....	38
7.4.2.1.37 ToReturnParameterMembership_Init.....	39
7.4.2.1.38 ToSpecialization_Init.....	39
7.4.2.1.39 ToStep_Init.....	40
7.4.2.1.40 ToSubclassification_Init.....	40
7.4.2.1.41 ToSubsetting_Init.....	40
7.4.2.1.42 ToSuccession_Init.....	41
7.4.2.1.43 ToSuccessionItemFlow_Init.....	41
7.4.2.1.44 ToTextualRepresentation_Init.....	41
7.4.2.1.45 ToType_Init.....	42
7.4.2.1.46 ToTypeFeaturing_Init.....	42
7.4.2.2 System Initializers.....	43
7.4.2.2.1 ToActionUsage_Init.....	43
7.4.2.2.2 ToActorMembership_Init.....	43
7.4.2.2.3 ToAssignmentActionUsage_Init.....	43
7.4.2.2.4 ToBindingConnectorAsUsage_Init.....	44
7.4.2.2.5 ToCalculationUsage_Init.....	44
7.4.2.2.6 ToConjugatedPortDefinition_Init.....	44
7.4.2.2.7 ToConjugatedPortTyping_Init.....	44
7.4.2.2.8 ToConnectionUsage_Init.....	45
7.4.2.2.9 ToConstraintDefinition_Init.....	45
7.4.2.2.10 ToConstraintUsage_Init.....	45
7.4.2.2.11 ToDefinition_Init.....	45
7.4.2.2.12 ToEventOccurrenceUsage_Init.....	46
7.4.2.2.13 ToFlowUsage_Init.....	46
7.4.2.2.14 ToItemDefinition_Init.....	47
7.4.2.2.15 ToItemFeature_Init.....	47
7.4.2.2.16 ToItemUsage_Init.....	47
7.4.2.2.17 ToMetadataUsage_Init.....	47
7.4.2.2.18 ToObjectiveMembership_Init.....	48
7.4.2.2.19 ToOccurrenceDefinition_Init.....	48
7.4.2.2.20 ToOccurrenceUsage_Init.....	48
7.4.2.2.21 ToPartUsage_Init.....	49
7.4.2.2.22 ToPerformActionUsage_Init.....	49
7.4.2.2.23 ToPortConjugation_Init.....	49
7.4.2.2.24 ToPortDefinition_Init.....	50
7.4.2.2.25 ToReferenceUsage_Init.....	50
7.4.2.2.26 ToRequirementUsage_Init.....	50
7.4.2.2.27 ToStateSubactionMembership_Init.....	50
7.4.2.2.28 ToStateUsage_Init.....	51
7.4.2.2.29 ToSubjectMembership_Init.....	51
7.4.2.2.30 ToTransitionUsage_Init.....	51
7.4.2.2.31 ToTriggerInvocationExpression_Init.....	51
7.4.2.2.32 ToUsage_Init.....	52
7.5 Factories.....	52
7.5.1 Overview.....	52
7.5.2 Mapping Specifications.....	52
7.5.2.1 LiteralString_Factory.....	52
7.5.2.2 StringParameterFeature_Factory.....	53
7.5.2.3 StringParameterFeatureValue_Factory.....	53

7.5.2.4	StringParameterMembership_Factory	54
7.5.2.5	SubjectMembership_Factory	54
7.5.2.6	AssignmentActionUsage_Factory	54
7.5.2.7	AssignmentActionUsageFeatureMembership2_Factory	55
7.5.2.8	AssignmentActionUsageFeatureMembership3_Factory	55
7.5.2.9	AssignmentActionUsageOwningMembership_Factory	55
7.5.2.10	AssignmentActionUsageParameterMembership_Factory	56
7.5.2.11	AssignmentActionUsageReferenceUsageIn1_Factory	56
7.5.2.12	AssignmentActionUsageTargetReferenceUsageIn2_Factory	57
7.5.2.13	AssignmentActionUsageTargetReferenceUsageIn3_Factory	57
7.5.2.14	DirectedReferenceUsage_Factory	57
7.5.2.15	DirectedReferenceUsageParameterMembership_Factory	58
7.5.2.16	EmptyObjectiveMembership_Factory	58
7.5.2.17	EmptyRequirementUsage_Factory	58
7.5.2.18	EmptySubject_Factory	59
7.5.2.19	EmptySubjectMembership_Factory	59
7.5.2.20	FeatureTyping_Factory	59
7.5.2.21	FlowEndParameterMembership_Factory	60
7.5.2.22	FlowItem_Factory	60
7.5.2.23	FlowItemFeatureMembership_Factory	61
7.5.2.24	FlowUsage_Factory	61
7.5.2.25	FlowUsageFeatureMembership_Factory	62
7.5.2.26	InformationFlowEventOccurrenceUsage_Factory	62
7.5.2.27	InformationFlowReferenceSubsetting_Factory	63
7.5.2.28	LiteralBoolean_Factory	63
7.5.2.29	LiteralNull_Factory	64
7.5.2.30	LiteralRational_Factory	64
7.5.2.31	LowerBound_Factory	65
7.5.2.32	MultiplicityElement_Factory	65
7.5.2.33	MultiplicityLowerBoundMembership_Factory	65
7.5.2.34	MultiplicityMembership_Factory	66
7.5.2.35	MultiplicityUpperBoundMembership_Factory	67
7.5.2.36	ObjectFlowItemFlowEndRedefinition_Factory	67
7.5.2.37	ParameterMembership_Factory	67
7.5.2.38	ReferenceSubsetting_Factory	68
7.5.2.39	ReferenceUsage_Factory	68
7.5.2.40	ReturnParameterFeature_Factory	68
7.5.2.41	ReturnParameterFeatureMembership_Factory	69
7.5.2.42	Subsetting_Factory	69
7.5.2.43	UpperBound_Factory	69
7.6	Generic Mappings	70
7.6.1	Overview	70
7.6.2	Common Mappings	70
7.6.2.1	CommonFeatureReferenceExpression_Mapping	70
7.6.2.2	CommonMembership_Mapping	71
7.6.2.3	CommonParameterReferenceUsageInMembership_Mapping	71
7.6.2.4	CommonParameterReferenceUsageIn_Mapping	72
7.6.2.5	CommonParameterReferenceUsageInFeatureTyping_Mapping	73
7.6.2.6	CommonParameterReferenceUsageInUntyped_Mapping	73
7.6.2.7	CommonReturnParameterFeature_Mapping	74
7.6.2.8	CommonReturnParameterFeatureTyping_Mapping	75
7.6.2.9	CommonReturnParameterFeatureUntyped_Mapping	75
7.6.2.10	CommonReturnParameterFeatureMembership_Mapping	76
7.6.2.11	CommonReturnParameterReferenceUsageMembership_Mapping	77
7.6.2.12	CommonReturnParameterReferenceUsage_Mapping	78
7.6.2.13	CommonReturnParameterReferenceUsageFeatureTyping_Mapping	78

7.6.2.14 CommonReturnParameterReferenceUsageUntyped_Mapping .....	79
7.6.2.15 CommonReferenceUsageIn_Mapping .....	80
7.6.2.16 CommonReferenceUsageInFeatureMembership_Mapping .....	80
7.6.2.17 CommonReferenceUsageInFeatureTyping_Mapping .....	81
7.6.2.18 CommonReferenceUsageInUntyped_Mapping .....	82
7.7 Mappings from UML4SysML metaclasses .....	82
7.7.1 Overview .....	82
7.7.2 Actions .....	83
7.7.2.1 Overview .....	83
7.7.2.2 UML4SysML::Actions elements not mapped .....	84
7.7.2.3 Mapping Specifications .....	85
7.7.2.3.1 Accept Event Actions .....	85
7.7.2.3.1.1 AcceptCallAction_Mapping .....	85
7.7.2.3.1.2 AcceptEventAction_Mapping .....	86
7.7.2.3.1.3 AEChangeExpressionMembership_Mapping .....	87
7.7.2.3.1.4 AEChangeParameter_Mapping .....	87
7.7.2.3.1.5 AEChangeParameterFeatureValue_Mapping .....	88
7.7.2.3.1.6 AEChangeParameterTrigger_Mapping .....	89
7.7.2.3.1.7 AEChangeParameterTriggerExpression_Mapping .....	89
7.7.2.3.1.8 AEChangeParameterResultExpressionMembership_Mapping .....	90
7.7.2.3.1.9 AEChangeParameterFeatureChainExpression_Mapping .....	91
7.7.2.3.1.10 AEChangeParameterFeatureMembership_Mapping .....	91
7.7.2.3.1.11 AEChangeParameterFeature_Mapping .....	92
7.7.2.3.1.12 AEChangeParameterExpressionFeatureValue_Mapping .....	92
7.7.2.3.1.13 AEChangeParameterFeatureReferenceExpression_Mapping .....	93
7.7.2.3.1.14 AEChangeParameterMembership_Mapping .....	94
7.7.2.3.1.15 AEChangeParameterParameterMembership_Mapping .....	94
7.7.2.3.1.16 AEAReceiverParameter_Mapping .....	95
7.7.2.3.1.17 AEAReceiverParameterMembership_Mapping .....	96
7.7.2.3.1.18 AEAReceiverFeatureValue_Mapping .....	96
7.7.2.3.1.19 AEASignalParameter_Mapping .....	97
7.7.2.3.1.20 AEASignalParameterFeatureTyping_Mapping .....	98
7.7.2.3.1.21 AEAParameterMembership_Mapping .....	98
7.7.2.3.1.22 AEAReceiverFeatureReferenceExpression_Mapping .....	99
7.7.2.3.1.23 AEAReceiverFeatureReferenceExpressionMembership_Mapping .....	100
7.7.2.3.1.24 ReplyAction_Mapping .....	101
7.7.2.3.1.25 UnmarshallAction_Mapping .....	101
7.7.2.3.2 Actions .....	102
7.7.2.3.2.1 CommonAction_Mapping .....	102
7.7.2.3.2.2 OpaqueAction_Mapping .....	102
7.7.2.3.2.3 OABody_Mapping .....	103
7.7.2.3.2.4 OABodyMembership_Mapping .....	104
7.7.2.3.2.5 Pin_Mapping .....	105
7.7.2.3.2.6 ValuePin_Mapping .....	106
7.7.2.3.2.7 ValuePinFeatureValue_Mapping .....	107
7.7.2.3.2.8 ValuePinUntyped_Mapping .....	107
7.7.2.3.3 Invocation Actions .....	108
7.7.2.3.3.1 BroadcastSignalAction_Mapping .....	108
7.7.2.3.3.2 CallBehaviorAction_Mapping .....	109
7.7.2.3.3.3 CBAFeatureTyping_Mapping .....	109
7.7.2.3.3.4 CallOperationAction_Mapping .....	110
7.7.2.3.3.5 COAOutputPinFeature_Mapping .....	111
7.7.2.3.3.6 COAOutputPinFeatureChainExpression_Mapping .....	112
7.7.2.3.3.7 COAOutputPinFeatureChainExpressionMembership_Mapping .....	112
7.7.2.3.3.8 COAOutputPinFeatureFeature_Mapping .....	113
7.7.2.3.3.9 COAOutputPinFeatureFeatureMembership_Mapping .....	113

7.7.2.3.3.10 COAOutputPinFeatureFeatureValue_Mapping .....	114
7.7.2.3.3.11 COAOutputPinFeatureMembership_Mapping.....	115
7.7.2.3.3.12 COAOutputPinFeatureReferenceExpression_Mapping.....	115
7.7.2.3.3.13 COAOutputPinFeatureReferenceExpressionMembership_Mapping.....	116
7.7.2.3.3.14 COAOutputPinParameterMembership_Mapping .....	116
7.7.2.3.3.15 COAOutputPinReferenceUsage_Mapping.....	117
7.7.2.3.3.16 COAOutputPinReferenceUsageFeatureValue_Mapping .....	118
7.7.2.3.3.17 COAPerformAction_Mapping .....	118
7.7.2.3.3.18 COAPerformActionFeatureMembership_Mapping .....	119
7.7.2.3.3.19 COAPerformActionReferenceSubsetting_Mapping .....	120
7.7.2.3.3.20 COAPerformActionFeature_Mapping .....	120
7.7.2.3.3.21 COAPerformActionFeatureChainingOperation_Mapping.....	121
7.7.2.3.3.22 COAPerformActionFeatureChainingTarget_Mapping .....	122
7.7.2.3.3.23 SendObjectAction_Mapping.....	122
7.7.2.3.3.24 SendSignalAction_Mapping.....	123
7.7.2.3.3.25 SSAFeatureMembership_Mapping .....	124
7.7.2.3.3.26 SSAParameterMembership_Mapping .....	124
7.7.2.3.3.27 SSAResourceUsage_Mapping .....	125
7.7.2.3.3.28 SSAItemParameterMembership_Mapping.....	125
7.7.2.3.3.29 SSAItemResourceUsage_Mapping .....	126
7.7.2.3.3.30 SSAItemResourceUsageFeatureValue_Mapping .....	127
7.7.2.3.3.31 SSAItemResourceUsageFeatureTyping_Mapping .....	127
7.7.2.3.3.32 SSAItemResourceUsageInvocationExpression_Mapping .....	128
7.7.2.3.3.33 SSATargetParameterMembership_Mapping .....	129
7.7.2.3.3.34 SSATargetResourceUsage_Mapping.....	129
7.7.2.3.3.35 SSATargetResourceUsageFeatureValue_Mapping .....	130
7.7.2.3.3.36 SSATargetResourceUsageFeatureValueMembership_Mapping.....	131
7.7.2.3.3.37 SSATargetResourceUsageFeatureValueExpression_Mapping .....	131
7.7.2.3.3.38 SSASendActionUsage_Mapping .....	132
7.7.2.3.3.39 StartClassifierBehaviorAction_Mapping .....	133
7.7.2.3.3.40 StartObjectBehaviorAction_Mapping .....	133
7.7.2.3.4 Link Actions .....	134
7.7.2.3.4.1 ClearAssociationAction_Mapping .....	134
7.7.2.3.4.2 CreateLinkAction_Mapping.....	134
7.7.2.3.4.3 CreateLinkObjectAction_Mapping .....	135
7.7.2.3.4.4 DestroyLinkAction_Mapping.....	135
7.7.2.3.4.5 ReadLinkAction_Mapping.....	136
7.7.2.3.4.6 ReadLinkObjectEndAction_Mapping.....	137
7.7.2.3.4.7 ReadLinkObjectEndQualifierAction_Mapping .....	137
7.7.2.3.5 Object Actions .....	138
7.7.2.3.5.1 CreateObjectAction_Mapping.....	138
7.7.2.3.5.2 COAInvocationExpressionFeatureTyping_Mapping .....	139
7.7.2.3.5.3 COAInvocationExpression_Mapping .....	139
7.7.2.3.5.4 COAPin_Mapping .....	140
7.7.2.3.5.5 COAPinFeatureValue_Mapping .....	140
7.7.2.3.5.6 DestroyObjectAction_Mapping .....	141
7.7.2.3.5.7 DOADestroyActionUsage_Mapping .....	142
7.7.2.3.5.8 DOADestroyActionUsageFeatureMembership_Mapping .....	143
7.7.2.3.5.9 DOADestroyActionUsageFeatureReferenceExpression_Mapping .....	143
7.7.2.3.5.10 DOADestroyActionUsageMembership_Mapping .....	144
7.7.2.3.5.11 DOADestroyActionUsageFeatureTyping_Mapping .....	144
7.7.2.3.5.12 DOADestroyActionUsageFeatureValue_Mapping.....	145
7.7.2.3.5.13 DOADestroyActionUsageResourceUsage_Mapping .....	146
7.7.2.3.5.14 DOADestroyFeatureMembership_Mapping .....	146
7.7.2.3.5.15 ReadIsClassifiedObjectAction_Mapping .....	147
7.7.2.3.5.16 RICOAFeatureValue_Mapping.....	148

7.7.2.3.5.17 RICOAFeatureValueOperatorExpression_Mapping.....	148
7.7.2.3.5.18 RICOAFeatureValueOperatorExpressionFeature_Mapping.....	149
7.7.2.3.5.19 RICOAFeatureValueOperatorExpressionFeatureValue_Mapping .....	150
7.7.2.3.5.20 RICOAFeatureValueOperatorFeatureReferenceExpression_Mapping .....	150
7.7.2.3.5.21 RICOAFeatureValueOperatorMembership_Mapping .....	151
7.7.2.3.5.22 RICOAFeatureValueOperatorParameterMembership_Mapping .....	151
7.7.2.3.5.23 RICOAOutputPin_Mapping.....	152
7.7.2.3.5.24 ReadExtentAction_Mapping .....	153
7.7.2.3.5.25 REAFeatureValue_Mapping.....	154
7.7.2.3.5.26 REAFeatureValueOperatorExpression_Mapping .....	154
7.7.2.3.5.27 REAFeatureValueOperatorExpressionFeature_Mapping .....	155
7.7.2.3.5.28 REAFeatureValueOperatorExpressionFeatureTyping_Mapping.....	156
7.7.2.3.5.29 REAFeatureValueOperatorExpressionMembership_Mapping .....	156
7.7.2.3.5.30 REAOutputPin_Mapping .....	157
7.7.2.3.5.31 ReadSelfAction_Mapping .....	158
7.7.2.3.5.32 RSAFeatureValue_Mapping .....	158
7.7.2.3.5.33 RSAFeatureValueFeatureReferenceExpression_Mapping .....	159
7.7.2.3.5.34 RSAFeatureValueMembership_Mapping .....	159
7.7.2.3.5.35 RSAOutputPin_Mapping .....	160
7.7.2.3.5.36 ReclassifyObjectAction_Mapping .....	161
7.7.2.3.5.37 TestIdentityAction_Mapping .....	161
7.7.2.3.5.38 TIAOperatorExpression_Mapping .....	162
7.7.2.3.5.39 TIAResultExpressionMembership_Mapping .....	163
7.7.2.3.5.40 ValueSpecificationAction_Mapping .....	164
7.7.2.3.5.41 VSAOutputPin_Mapping .....	165
7.7.2.3.5.42 VSAOutputPinFeatureValue_Mapping.....	165
7.7.2.3.6 Other Actions .....	166
7.7.2.3.6.1 RaiseExceptionAction_Mapping.....	166
7.7.2.3.6.2 ReduceAction_Mapping.....	167
7.7.2.3.7 Structural Feature Actions.....	167
7.7.2.3.7.1 AddStructuralFeatureValueAction_Mapping.....	167
7.7.2.3.7.2 ASFVAFeatureTyping_Mapping.....	168
7.7.2.3.7.3 ASFVAObjectFeatureMembership_Mapping.....	169
7.7.2.3.7.4 ASFVAObjectReferenceUsage_Mapping.....	169
7.7.2.3.7.5 ASFVAObjectReferenceUsageFeatureTyping_Mapping .....	170
7.7.2.3.7.6 ASFVAObjectReferenceUsageRedefinition_Mapping.....	170
7.7.2.3.7.7 ASFVATargetFeatureChainExpression_Mapping.....	171
7.7.2.3.7.8 ASFVATargetFeatureMembership_Mapping.....	172
7.7.2.3.7.9 ASFVATargetFeatureValue_Mapping.....	172
7.7.2.3.7.10 ASFVATargetParameterExpressionFeature_Mapping .....	173
7.7.2.3.7.11 ASFVATargetParameterExpressionFeatureMembership_Mapping .....	174
7.7.2.3.7.12 ASFVATargetParameterExpressionMembership_Mapping .....	174
7.7.2.3.7.13 ASFVATargetParameterFeature_Mapping .....	175
7.7.2.3.7.14 ASFVATargetParameterFeatureExpressionMembership_Mapping .....	175
7.7.2.3.7.15 ASFVATargetParameterFeatureReferenceExpression_Mapping .....	176
7.7.2.3.7.16 ASFVATargetParameterFeatureValue_Mapping .....	177
7.7.2.3.7.17 ASFVATargetParameterMembership_Mapping.....	177
7.7.2.3.7.18 ASFVATargetReferenceUsage_Mapping .....	178
7.7.2.3.7.19 ASFVATargetReferenceUsageRedefinition_Mapping.....	179
7.7.2.3.7.20 ClearStructuralFeatureAction_Mapping .....	179
7.7.2.3.7.21 ReadStructuralFeatureAction_Mapping.....	180
7.7.2.3.7.22 RSFAReferenceUsage_Mapping .....	181
7.7.2.3.7.23 RSFAReferenceUsageExpressionFeature_Mapping.....	181
7.7.2.3.7.24 RSFAReferenceUsageExpressionFeatureMembership_Mapping .....	182
7.7.2.3.7.25 RSFAReferenceUsageExpressionFeatureReferenceExpression_Mapping .....	183
7.7.2.3.7.26 RSFAReferenceUsageExpressionFeatureValue_Mapping .....	183



7.7.2.3.7.27 RSFReferenceUsageFeatureChainExpression_Mapping .....	184
7.7.2.3.7.28 RSFReferenceUsageFeatureChainExpressionFeature_Mapping .....	185
7.7.2.3.7.29 RSFReferenceUsageFeatureChainExpressionMembership_Mapping .....	185
7.7.2.3.7.30 RSFReferenceUsageFeatureMembership_Mapping .....	186
7.7.2.3.7.31 RSFReferenceUsageFeatureValue_Mapping .....	186
7.7.2.3.7.32 RSFReferenceUsageMembership_Mapping .....	187
7.7.2.3.7.33 RSFReferenceUsageParameterMembership_Mapping .....	188
7.7.2.3.7.34 RemoveStructuralFeatureValueAction_Mapping .....	188
7.7.2.3.8 Structured Actions .....	189
7.7.2.3.8.1 LoopNode_Mapping .....	189
7.7.2.3.8.2 SequenceNode_Mapping .....	189
7.7.2.3.8.3 StructuredActivityNode_Mapping .....	190
7.7.2.3.9 Variable Actions .....	191
7.7.2.3.9.1 AddVariableValueAction_Mapping .....	191
7.7.2.3.9.2 AVVAFeatureTyping_Mapping .....	192
7.7.2.3.9.3 AVVAFeatureValue_Mapping .....	192
7.7.2.3.9.4 AVVAIsReplaceAll_Mapping .....	193
7.7.2.3.9.5 AVVAIsReplaceAllFeatureMembership_Mapping .....	194
7.7.2.3.9.6 AVVAIsReplaceAllRedefinition_Mapping .....	194
7.7.2.3.9.7 AVVAIsReplaceAllValue_Mapping .....	195
7.7.2.3.9.8 AVVAValueExpressionMembership_Mapping .....	196
7.7.2.3.9.9 AVVAValueFeatureReferenceExpression_Mapping .....	196
7.7.2.3.9.10 AVVAVariable_Mapping .....	197
7.7.2.3.9.11 AVVAVariableFeatureMembership_Mapping .....	198
7.7.2.3.9.12 AVVAVariableRedefinition_Mapping .....	198
7.7.2.3.9.13 ClearVariableAction_Mapping .....	199
7.7.2.3.9.14 CVAFeatureMembership_Mapping .....	200
7.7.2.3.9.15 CVAResourceUsage_Mapping .....	200
7.7.2.3.9.16 CVAResourceUsageFeatureValue_Mapping .....	201
7.7.2.3.9.17 ReadVariableAction_Mapping .....	202
7.7.2.3.9.18 RVAFeatureMembership_Mapping .....	202
7.7.2.3.9.19 RVAResourceUsage_Mapping .....	203
7.7.2.3.9.20 RVAResourceUsageFeatureReferenceExpression_Mapping .....	204
7.7.2.3.9.21 RVAResourceUsageFeatureTyping_Mapping .....	204
7.7.2.3.9.22 RVAResourceUsageFeatureValue_Mapping .....	205
7.7.2.3.9.23 RVAResourceUsageExpressionMembership_Mapping .....	206
7.7.2.3.9.24 RemoveVariableValueAction_Mapping .....	206
7.7.2.3.9.25 RVVAFeatureTyping_Mapping .....	207
7.7.2.3.9.26 RVVAVariable_Mapping .....	208
7.7.2.3.9.27 RVVAVariableExpressionMembership_Mapping .....	208
7.7.2.3.9.28 RVVAVariableFeatureMembership_Mapping .....	209
7.7.2.3.9.29 RVVAVariableFeatureReferenceExpression_Mapping .....	210
7.7.2.3.9.30 RVVAVariableFeatureValue_Mapping .....	210
7.7.2.3.9.31 RVVAVariableRedefinition_Mapping .....	211
7.7.3 Activities .....	211
7.7.3.1 Overview .....	212
7.7.3.2 UML4SysML::Activities elements not mapped .....	212
7.7.3.3 Mapping Specifications .....	213
7.7.3.3.1 ActivityAsDefinition_Mapping .....	213
7.7.3.3.2 ActivityEdgeInitialNodeFeatureMembership_Mapping .....	213
7.7.3.3.3 ActivityEdgeMetadata_Mapping .....	214
7.7.3.3.4 ActivityEdgeMetadataFeatureMembership_Mapping .....	215
7.7.3.3.5 ActivityEdgeMetadataFeatureTyping_Mapping .....	215
7.7.3.3.6 ActivityEdgeMetadataFeatureValue_Mapping .....	216
7.7.3.3.7 ActivityEdgeMetadataOwningMembership_Mapping .....	217
7.7.3.3.8 ActivityEdgeMetadataRedefinition_Mapping .....	217

7.7.3.3.9 ActivityEdgeMetadataReferenceUsage_Mapping .....	218
7.7.3.3.10 ActivityEdgeSourceEndFeature_Mapping .....	219
7.7.3.3.11 ActivityEdgeSourceInitialNode_Mapping .....	219
7.7.3.3.12 ActivityEdgeSourceEndFeatureMembership_Mapping .....	220
7.7.3.3.13 ActivityEdgeSourceInitialNodeSubsetting_Mapping .....	221
7.7.3.3.14 ActivityEdgeSourceEndSubsetting_Mapping .....	221
7.7.3.3.15 ActivityEdgeTransitionUsageSourceMembership_Mapping .....	222
7.7.3.3.16 ActivityFinalNode_Mapping .....	223
7.7.3.3.17 CentralBufferNode_Mapping .....	223
7.7.3.3.18 CommonActivityEdgeSuccessionAsUsage_Mapping .....	224
7.7.3.3.19 CommonVariable_Mapping .....	225
7.7.3.3.20 ControlFlowTransitionUsage_Mapping .....	226
7.7.3.3.21 ControlFlowFinalNodeFeatureMembership_Mapping .....	227
7.7.3.3.22 ControlFlowTargetFinalNodeSubsetting_Mapping .....	228
7.7.3.3.23 ControlFlowSuccessionAsUsage_Mapping .....	228
7.7.3.3.24 ControlFlowTargetFinalNode_Mapping .....	230
7.7.3.3.25 ControlFlowTargetEndFeature_Mapping .....	230
7.7.3.3.26 ControlFlowTargetFeatureMembership_Mapping .....	231
7.7.3.3.27 ControlFlowTargetEndSubsetting_Mapping .....	232
7.7.3.3.28 ControlFlowTransitionUsageFeatureMembership_Mapping .....	232
7.7.3.3.29 ControlNodeObjectFlowFeatureMembership_Mapping .....	233
7.7.3.3.30 ControlNodeObjectFlowFeatureValue_Mapping .....	234
7.7.3.3.31 ControlNodeObjectFlowReferenceUsage_Mapping .....	235
7.7.3.3.32 DataStoreNode_Mapping .....	236
7.7.3.3.33 DecisionNode_Mapping .....	236
7.7.3.3.34 FlowFinalNodeMembership_Mapping .....	237
7.7.3.3.35 ForkNode_Mapping .....	238
7.7.3.3.36 ForkNodeObjectFlowFeatureReferenceExpression_Mapping .....	239
7.7.3.3.37 ForkNodeObjectFlowMembership_Mapping .....	240
7.7.3.3.38 JoinMergeNodeObjectFlowFeature_Mapping .....	241
7.7.3.3.39 JoinMergeNodeObjectFlowFeatureReferenceExpression_Mapping .....	241
7.7.3.3.40 JoinMergeNodeObjectFlowFeatureValue_Mapping .....	242
7.7.3.3.41 JoinMergeNodeObjectFlowMembership_Mapping .....	243
7.7.3.3.42 JoinMergeNodeObjectFlowOperatorExpression_Mapping .....	243
7.7.3.3.43 JoinMergeNodeObjectFlowParameterMembership_Mapping .....	244
7.7.3.3.44 InitialNodeMembership_Mapping .....	245
7.7.3.3.45 JoinNode_Mapping .....	245
7.7.3.3.46 MergeNode_Mapping .....	247
7.7.3.3.47 ObjectFlow_Mapping .....	248
7.7.3.3.48 ObjectFlowFeatureMembership_Mapping .....	250
7.7.3.3.49 ObjectFlowGuardFeatureMembership_Mapping .....	250
7.7.3.3.50 ObjectFlowGuard_Mapping .....	251
7.7.3.3.51 ObjectFlowGuardSuccessionTargetEndFeature_Mapping .....	252
7.7.3.3.52 ObjectFlowGuardSuccessionTargetEndFeatureMembership_Mapping .....	253
7.7.3.3.53 ObjectFlowGuardSuccessionTargetEndSubsetting_Mapping .....	253
7.7.3.3.54 ObjectFlowItemFeature_Mapping .....	254
7.7.3.3.55 ObjectFlowItemFeatureMembership_Mapping .....	255
7.7.3.3.56 ObjectFlowItemFeatureTyping_Mapping .....	255
7.7.3.3.57 ObjectFlowItemFeatureUntyped_Mapping .....	256
7.7.3.3.58 ObjectFlowEndFeatureMembership_Mapping .....	256
7.7.3.3.59 ObjectFlowItemFlowEnd_Mapping .....	257
7.7.3.3.60 ObjectFlowItemFlowEndReferenceUsage_Mapping .....	258
7.7.3.3.61 ObjectFlowItemFlowEndFeatureMembership_Mapping .....	259
7.7.3.3.62 ObjectFlowItemFlowEndRedefinition_Mapping .....	260
7.7.3.3.63 ObjectFlowItemFlowEndSubsetting_Mapping .....	260
7.7.3.3.64 ObjectFlowTransitionUsageFeatureMembership_Mapping .....	261

7.7.3.3.65 VariableAttribute_Mapping .....	262
7.7.3.3.66 VariableFeatureTyping_Mapping .....	262
7.7.3.3.67 VariableItem_Mapping .....	263
7.7.3.3.68 VariableMembership_Mapping.....	264
7.7.4 Classification .....	264
7.7.4.1 Overview .....	264
7.7.4.2 Mapping Specifications.....	265
7.7.4.2.1 BehavioralFeature_Mapping .....	265
7.7.4.2.2 Classifier_Mapping .....	265
7.7.4.2.3 DefaultLowerBound_Mapping .....	266
7.7.4.2.4 DefaultMultiplicityBoundFeatureMembership_Mapping.....	267
7.7.4.2.5 DefaultMultiplicityElement_Mapping .....	267
7.7.4.2.6 DefaultMultiplicityLowerBoundFeatureMembership_Mapping .....	268
7.7.4.2.7 DefaultMultiplicityMembership_Mapping .....	269
7.7.4.2.8 DefaultMultiplicityUpperBoundFeatureMembership_Mapping.....	269
7.7.4.2.9 DefaultUpperBound_Mapping.....	270
7.7.4.2.10 DefaultValue_Mapping .....	271
7.7.4.2.11 ElementFeatureMembership_Mapping .....	271
7.7.4.2.12 Generalization_Mapping .....	272
7.7.4.2.13 InstanceSpecificationLink_Mapping.....	273
7.7.4.2.14 InstanceSpecification_Mapping .....	274
7.7.4.2.15 InstanceSpecificationFeatureTyping_Mapping.....	275
7.7.4.2.16 InstanceValue_Mapping.....	276
7.7.4.2.17 InstanceValueMembership_Mapping .....	277
7.7.4.2.18 LowerBoundValueFeatureMembership_Mapping.....	277
7.7.4.2.19 MultiplicityElement_Mapping .....	278
7.7.4.2.20 MultiplicityLowerBoundOwningMembership_Mapping .....	279
7.7.4.2.21 MultiplicityMembership_Mapping .....	279
7.7.4.2.22 MultiplicityUpperBoundOwningMembership_Mapping.....	280
7.7.4.2.23 Operation_Mapping.....	281
7.7.4.2.24 Parameter_Mapping .....	282
7.7.4.2.25 ParameterDefaultValue_Mapping.....	283
7.7.4.2.26 ParameterMembership_Mapping .....	284
7.7.4.2.27 ParameterSet_Mapping .....	284
7.7.4.2.28 ParameterSetMembership_Mapping.....	286
7.7.4.2.29 ParameterSetParameterFeatureMembership_Mapping.....	286
7.7.4.2.30 ParameterSetParameterReferenceUsage_Mapping.....	287
7.7.4.2.31 ParameterSetParameterReferenceUsageFeatureValue_Mapping .....	287
7.7.4.2.32 ParameterSetParameterReferenceUsageFeatureValueExpression_Mapping .....	288
7.7.4.2.33 ParameterSetParameterReferenceUsageMembership_Mapping.....	289
7.7.4.2.34 ParameterToFeatureTyping_Mapping .....	289
7.7.4.2.35 PropertyCommon_Mapping .....	290
7.7.4.2.36 PropertySubsetting_Mapping.....	291
7.7.4.2.37 PropertyTypedByClassInterface_Mapping .....	292
7.7.4.2.38 PropertyUntyped_Mapping .....	293
7.7.4.2.39 Realization_Mapping .....	294
7.7.4.2.40 Slot_Mapping .....	294
7.7.4.2.41 SlotMembership_Mapping.....	294
7.7.4.2.42 SlotFeatureTyping_Mapping.....	295
7.7.4.2.43 SlotValue_Mapping.....	296
7.7.4.2.44 StructuralFeature_Mapping.....	297
7.7.4.2.45 StructuralFeatureMembership_Mapping.....	298
7.7.4.2.46 StructuralFeatureToFeatureTyping_Mapping.....	298
7.7.4.2.47 TypedElementFeatureTyping_Mapping .....	299
7.7.4.2.48 UpperBoundValueFeatureMembership_Mapping .....	300

7.7.5 CommonBehavior .....	300
7.7.5.1 Overview .....	300
7.7.5.2 UML4SysML::CommonBehavior elements not mapped .....	301
7.7.5.3 Mapping Specifications .....	301
7.7.5.3.1 Behavior_Mapping .....	301
7.7.5.3.2 ChangeEvent_Mapping .....	302
7.7.5.3.3 ChangeEventReturnParameter_Mapping .....	302
7.7.5.3.4 ChangeEventReturnParameterMembership_Mapping .....	303
7.7.5.3.5 ChangeTriggerBindingConnector_Mapping .....	304
7.7.5.3.6 ChangeTriggerConstraintUsage_Mapping .....	304
7.7.5.3.7 ChangeTriggerEndFeatureMembership_Mapping .....	305
7.7.5.3.8 ChangeTriggerEventChainingFeature_Mapping .....	306
7.7.5.3.9 ChangeTriggerEventReturnParameterChainingFeature_Mapping .....	306
7.7.5.3.10 ChangeTriggerExpressionFeature_Mapping .....	307
7.7.5.3.11 ChangeTriggerExpressionFeatureMembership_Mapping .....	308
7.7.5.3.12 ChangeTriggerExpressionFeatureReferenceExpression_Mapping .....	308
7.7.5.3.13 ChangeTriggerExpressionFeatureTyping_Mapping .....	309
7.7.5.3.14 ChangeTriggerExpressionFeatureValue_Mapping .....	309
7.7.5.3.15 ChangeTriggerExpressionInvocationExpression_Mapping .....	310
7.7.5.3.16 ChangeTriggerExpressionParameterMembership_Mapping .....	311
7.7.5.3.17 ChangeTriggerFeature_Mapping .....	311
7.7.5.3.18 ChangeTriggerFeatureMembership_Mapping .....	312
7.7.5.3.19 ChangeTriggerFeatureValue_Mapping .....	313
7.7.5.3.20 ChangeTriggerInvocationExpression_Mapping .....	313
7.7.5.3.21 ChangeTriggerReferenceSubsetting_Mapping .....	314
7.7.5.3.22 ChangeTriggerReferenceUsage_Mapping .....	315
7.7.5.3.23 ChangeTriggerReturnEndFeatureMembership_Mapping .....	315
7.7.5.3.24 ChangeTriggerReturnParameter_Mapping .....	316
7.7.5.3.25 ChangeTriggerReturnParameterMembership_Mapping .....	317
7.7.5.3.26 ChangeTriggerReturnReferenceSubsetting_Mapping .....	317
7.7.5.3.27 ChangeTriggerReturnReferenceUsage_Mapping .....	318
7.7.5.3.28 OpaqueBehavior_Mapping .....	319
7.7.5.3.29 OpaqueBehaviorMembership_Mapping .....	320
7.7.5.3.30 OpaqueBehaviorSpecification_Mapping .....	320
7.7.5.3.31 SignalTriggerReferenceUsage_Mapping .....	321
7.7.5.3.32 SignalTriggerReferenceUsageFeatureTyping_Mapping .....	322
7.7.5.3.33 TimeEvent_Mapping .....	322
7.7.5.3.34 TimeTriggerBindingConnector_Mapping .....	323
7.7.5.3.35 TimeTriggerCalculationUsage_Mapping .....	324
7.7.5.3.36 TimeTriggerEndFeatureMembership_Mapping .....	325
7.7.5.3.37 TimeTriggerEventChainingFeature_Mapping .....	325
7.7.5.3.38 TimeTriggerEventReturnParameterChainingFeature_Mapping .....	326
7.7.5.3.39 TimeTriggerExpressionFeature_Mapping .....	327
7.7.5.3.40 TimeTriggerExpressionFeatureTyping_Mapping .....	327
7.7.5.3.41 TimeTriggerExpressionFeatureValue_Mapping .....	328
7.7.5.3.42 TimeTriggerExpressionInvocationExpression_Mapping .....	328
7.7.5.3.43 TimeTriggerExpressionParameterMembership_Mapping .....	329
7.7.5.3.44 TimeTriggerFeature_Mapping .....	330
7.7.5.3.45 TimeTriggerFeatureMembership_Mapping .....	330
7.7.5.3.46 TimeTriggerFeatureTyping_Mapping .....	331
7.7.5.3.47 TimeTriggerFeatureValue_Mapping .....	332
7.7.5.3.48 TimeTriggerInvocationExpression_Mapping .....	332
7.7.5.3.49 TimeTriggerReferenceSubsetting_Mapping .....	333
7.7.5.3.50 TimeTriggerReferenceUsage_Mapping .....	334
7.7.5.3.51 TimeTriggerReturnEndFeatureMembership_Mapping .....	335
7.7.5.3.52 TimeTriggerReturnParameter_Mapping .....	335

7.7.5.3.53 TimeTriggerReturnParameterMembership_Mapping .....	336
7.7.5.3.54 TimeTriggerReturnReferenceSubsetting_Mapping .....	336
7.7.5.3.55 TimeTriggerReturnReferenceUsage_Mapping .....	337
7.7.5.3.56 Trigger_Mapping .....	338
7.7.5.3.57 TriggerParameterMembership_Mapping .....	338
7.7.6 CommonStructure .....	339
7.7.6.1 Overview .....	339
7.7.6.2 Mapping Specifications .....	340
7.7.6.2.1 Abstraction_Mapping .....	340
7.7.6.2.2 Comment_Mapping .....	340
7.7.6.2.3 CommentAnnotation_Mapping .....	341
7.7.6.2.4 CommentOwnership_Mapping .....	342
7.7.6.2.5 Constraint_Mapping .....	343
7.7.6.2.6 ConstrainedElementFeatureMembership_Mapping .....	344
7.7.6.2.7 ConstraintUsageFeatureTyping_Mapping .....	344
7.7.6.2.8 ConstraintUsage_Mapping .....	345
7.7.6.2.9 Dependency_Mapping .....	346
7.7.6.2.10 DirectedRelationship_Mapping .....	346
7.7.6.2.11 ElementMain_Mapping .....	347
7.7.6.2.12 ElementMembership_Mapping .....	348
7.7.6.2.13 ElementOwnership_Mapping .....	349
7.7.6.2.14 ElementOwningMembership_Mapping .....	349
7.7.6.2.15 NamedElementMain_Mapping .....	350
7.7.6.2.16 Namespace_Mapping .....	351
7.7.6.2.17 Relationship_Mapping .....	351
7.7.6.2.18 Usage_Mapping .....	352
7.7.7 InformationFlows .....	353
7.7.7.1 Overview .....	353
7.7.7.2 Mapping Specifications .....	353
7.7.7.2.1 InformationFlow_Mapping .....	353
7.7.7.2.2 InformationFlowConveyedFeatureMembership_Mapping .....	354
7.7.7.2.3 InformationFlowEnd_Mapping .....	354
7.7.7.2.4 InformationFlowEndFeatureMembership_Mapping .....	355
7.7.7.2.5 InformationFlowFeatureTyping_Mapping .....	356
7.7.7.2.6 InformationFlowSubclassification_Mapping .....	356
7.7.7.2.7 InformationItem_Mapping .....	357
7.7.7.2.8 InformationItemFlowConveyedItemUsage_Mapping .....	357
7.7.7.2.9 InformationItemFlowConveyedItemUsageFeatureTyping_Mapping .....	358
7.7.8 Interactions .....	359
7.7.8.1 Overview .....	359
7.7.8.2 UML4SysML::Interactions elements not mapped .....	359
7.7.8.3 Mapping Specifications .....	360
7.7.8.3.1 ActionExecutionSpecification_Mapping .....	360
7.7.8.3.2 BehaviorExecutionSpecification_Mapping .....	360
7.7.8.3.3 CombinedFragment_Mapping .....	361
7.7.8.3.4 CombinedFragmentMembership_Mapping .....	362
7.7.8.3.5 ExecutionSpecificationMembership_Mapping .....	362
7.7.8.3.6 Interaction_Mapping .....	363
7.7.8.3.7 InteractionOperand_Mapping .....	364
7.7.8.3.8 InteractionOperandMembership_Mapping .....	365
7.7.8.3.9 InteractionUse_Mapping .....	366
7.7.8.3.10 InteractionUseMembership_Mapping .....	366
7.7.8.3.11 InteractionUseFeatureTyping_Mapping .....	367
7.7.8.3.12 LifelineMembership_Mapping .....	368
7.7.8.3.13 LifelinePartUsage_Mapping .....	368
7.7.8.3.14 LifelineFeatureTyping_Mapping .....	369

7.7.8.3.15 Message_Mapping.....	370
7.7.8.3.16 MessageMembership_Mapping .....	370
7.7.8.3.17 StateInvariant_Mapping .....	371
7.7.8.3.18 StateInvariantMembership_Mapping .....	372
7.7.8.3.19 StateInvariantFeatureTyping_Mapping.....	372
7.7.9 Packages.....	373
7.7.9.1 Overview .....	373
7.7.9.2 UML4SysML::Packages elements not mapped .....	373
7.7.9.3 Mapping Specifications.....	374
7.7.9.3.1 ElementImport_Mapping .....	374
7.7.9.3.2 Model_Mapping .....	375
7.7.9.3.3 ModelViewpointMetadataUsage_Mapping .....	375
7.7.9.3.4 ModelViewpointMetadataFeatureMembership_Mapping .....	376
7.7.9.3.5 ModelViewpointMetadataReferenceUsage_Mapping .....	377
7.7.9.3.6 ModelViewpointMetadataFeatureTyping_Mapping.....	377
7.7.9.3.7 ModelViewpointMetadataMembership_Mapping .....	378
7.7.9.3.8 ModelViewpointMetadataFeatureValue_Mapping .....	379
7.7.9.3.9 ModelViewpointMetadataRedefinition_Mapping .....	379
7.7.9.3.10 ModelViewpointValue_Mapping.....	380
7.7.9.3.11 Package_Mapping .....	381
7.7.9.3.12 PackageImport_Mapping .....	382
7.7.9.3.13 PackageURIMetadataUsage_Mapping .....	382
7.7.9.3.14 PackageURIFeatureMembership_Mapping .....	383
7.7.9.3.15 PackageURIFeatureTyping_Mapping.....	384
7.7.9.3.16 PackageURIMetadataReferenceUsage_Mapping .....	385
7.7.9.3.17 PackageURIMetadataFeatureValue_Mapping.....	385
7.7.9.3.18 PackageURIMetadataMembership_Mapping .....	386
7.7.9.3.19 PackageURIRedefinition_Mapping .....	387
7.7.9.3.20 PackageURIValue_Mapping.....	387
7.7.9.3.21 Profile_Mapping.....	388
7.7.9.3.22 ProfileMetadataMembership_Mapping.....	389
7.7.9.3.23 ProfileMetadataUsage_Mapping .....	389
7.7.9.3.24 StereotypeMetadataDefinition_Mapping .....	390
7.7.9.3.25 StereotypeMetadataDefinitionMembership_Mapping.....	390
7.7.9.3.26 StereotypeOccurrenceUsage_Mapping .....	391
7.7.9.3.27 StereotypeOccurrenceUsageFeatureTyping_Mapping.....	392
7.7.9.3.28 StereotypeOccurrenceUsageMembership_Mapping .....	392
7.7.9.3.29 StereotypeOccurrenceUsageMultiplicityMembership_Mapping .....	393
7.7.9.3.30 StereotypeOccurrenceUsageMultiplicityRange_Mapping .....	394
7.7.9.3.31 StereotypeOccurrenceUsageMultiplicityRangeInfinity_Mapping .....	394
7.7.9.3.32 StereotypeOccurrenceUsageInfinityReturnParameter_Mapping .....	395
7.7.9.3.33 StereotypeOccurrenceUsageInfinityReturnParameterMembership_Mapping .....	395
7.7.9.3.34 StereotypeOccurrenceUsageMultiplicityRangeMembership_Mapping .....	396
7.7.10 SimpleClassifiers.....	397
7.7.10.1 Overview .....	397
7.7.10.2 Mapping Specifications.....	397
7.7.10.2.1 Attribute_Mapping .....	397
7.7.10.2.2 AttributeRedefined_Mapping.....	398
7.7.10.2.3 AttributeRedefinedRedefinition_Mapping.....	399
7.7.10.2.4 AttributeRedefinedMembership_Mapping .....	400
7.7.10.2.5 AttributeRedefinedFeatureTyping_Mapping .....	400
7.7.10.2.6 BehavioredClassifier_Mapping.....	401
7.7.10.2.7 BehavioredClassifierFeatureMembership_Mapping.....	402
7.7.10.2.8 BehavioredClassifierFeatureTyping_Mapping .....	403
7.7.10.2.9 BehavioredClassifierActionUsage_Mapping.....	403
7.7.10.2.10 DataType_Mapping.....	404

7.7.10.2.11 Enumeration_Mapping.....	404
7.7.10.2.12 EnumerationLiteral_Mapping.....	405
7.7.10.2.13 EnumerationVariantMembership_Mapping.....	406
7.7.10.2.14 Interface_Mapping.....	406
7.7.10.2.15 InterfaceConjugatedPortDefinition_Mapping.....	407
7.7.10.2.16 InterfaceConjugatedPortDefinitionMembership_Mapping.....	408
7.7.10.2.17 InterfacePortConjugation_Mapping.....	409
7.7.10.2.18 InterfaceRealization_Mapping.....	409
7.7.10.2.19 PrimitiveType_Mapping.....	410
7.7.10.2.20 Reception_Mapping.....	411
7.7.10.2.21 ReceptionFeatureTyping_Mapping.....	411
7.7.10.2.22 Signal_Mapping.....	412
7.7.11 StateMachines.....	412
7.7.11.1 Overview.....	412
7.7.11.2 Mapping Specifications.....	413
7.7.11.2.1 ChangeTriggerReferenceUsage_Mapping.....	413
7.7.11.2.2 CommonPseudostate_Mapping.....	413
7.7.11.2.3 ConnectionPointReference_Mapping.....	414
7.7.11.2.4 DoBehaviorStateSubactionMembership_Mapping.....	415
7.7.11.2.5 EntryBehaviorStateSubactionMembership_Mapping.....	416
7.7.11.2.6 ExitBehaviorStateSubactionMembership_Mapping.....	416
7.7.11.2.7 FinalState_Mapping.....	417
7.7.11.2.8 InitialState_Mapping.....	417
7.7.11.2.9 InitialStateSubactionMembership_Mapping.....	418
7.7.11.2.10 PseudoState_Mapping.....	419
7.7.11.2.11 Region_Mapping.....	419
7.7.11.2.12 State_Mapping.....	420
7.7.11.2.13 StateBehaviorPerformActionUsage_Mapping.....	422
7.7.11.2.14 StateBehaviorPerformActionUsageFeatureTyping_Mapping.....	422
7.7.11.2.15 StateBehaviorStateSubactionMembership_Mapping.....	423
7.7.11.2.16 StateDefinition_Mapping.....	423
7.7.11.2.17 TimeTriggerReferenceUsage_Mapping.....	424
7.7.11.2.18 Transition_Mapping.....	425
7.7.11.2.19 TransitionSuccession_Mapping.....	426
7.7.11.2.20 TransitionSourceToSubsetting_Mapping.....	427
7.7.11.2.21 TransitionSuccessionSource_Mapping.....	427
7.7.11.2.22 TransitionSuccessionSourceMembership_Mapping.....	428
7.7.11.2.23 TransitionSuccessionTarget_Mapping.....	429
7.7.11.2.24 TransitionSuccessionTargetMembership_Mapping.....	430
7.7.11.2.25 TransitionTargetToSubsetting_Mapping.....	430
7.7.11.2.26 TransitionTriggerFeatureMembership_Mapping.....	431
7.7.12 StructuredClassifiers.....	432
7.7.12.1 Overview.....	432
7.7.12.2 Mapping Specifications.....	432
7.7.12.2.1 AssociationClass_Mapping.....	432
7.7.12.2.2 AssociationCommon_Mapping.....	433
7.7.12.2.3 AssociationMetadataUsage_Mapping.....	434
7.7.12.2.4 AssociationMetadataUsageFeatureMembership_Mapping.....	435
7.7.12.2.5 AssociationMetadataUsageFeatureTyping_Mapping.....	435
7.7.12.2.6 AssociationMetadataUsageFeature_Mapping.....	436
7.7.12.2.7 AssociationMetadataUsageFeatureValue_Mapping.....	436
7.7.12.2.8 AssociationMetadataUsageMembership_Mapping.....	437
7.7.12.2.9 AssociationMetadataUsageRedefinition_Mapping.....	438
7.7.12.2.10 Class_Mapping.....	438
7.7.12.2.11 ConnectionDefEnd_Mapping.....	439
7.7.12.2.12 ConnectionDefEndMembership_Mapping.....	440

7.7.12.2.13	ConnectionEndToSubsetting_Mapping .....	441
7.7.12.2.14	Connector_Mapping .....	442
7.7.12.2.15	ConnectorEndToFeatureCommon_Mapping .....	443
7.7.12.2.16	ConnectorEndToMembership_Mapping .....	443
7.7.12.2.17	ConnectorEndToOwnedFeature_Mapping .....	444
7.7.12.2.18	ConnectorEndToSubsettedFeature_Mapping .....	445
7.7.12.2.19	ConnectorEndToSubsettedFeatureMembership_Mapping .....	445
7.7.12.2.20	ConnectorType_Mapping .....	446
7.7.12.2.21	ConnectorTypeDerived_Mapping .....	447
7.7.12.2.22	CrossSubsetting_Mapping .....	448
7.7.12.2.23	End_Mapping .....	448
7.7.12.2.24	EndMembership_Mapping .....	449
7.7.12.2.25	EndToSubsettedFeature_Mapping .....	449
7.7.12.2.26	EndToSubsettedFeatureChaining_Mapping .....	450
7.7.12.2.27	MultiplicityReferenceUsage_Mapping .....	451
7.7.12.2.28	NonOwnedEndSubsetting_Mapping .....	451
7.7.12.2.29	NonOwnedEndToSubsettedFeatureMembership_Mapping .....	452
7.7.12.2.30	NonOwnedEnd_Mapping .....	453
7.7.12.2.31	NonOwnedEndMembership_Mapping .....	454
7.7.12.2.32	NonOwnedEndSubsettingMembership_Mapping .....	454
7.7.12.2.33	NonOwnedEndFeatureTyping_Mapping .....	455
7.7.12.2.34	OwnedEnd_Mapping .....	455
7.7.12.2.35	OwnedEndMembership_Mapping .....	457
7.7.12.2.36	Port_Mapping .....	457
7.7.12.2.37	PortUntyped_Mapping .....	458
7.7.12.2.38	PropertyToFeatureChaining_Mapping .....	459
7.7.12.2.39	QualifierMembership_Mapping .....	459
7.7.13	UseCases .....	460
7.7.13.1	Overview .....	460
7.7.13.2	UML4SysML::UseCases elements not mapped .....	460
7.7.13.3	Mapping Specifications .....	460
7.7.13.3.1	Actor_Mapping .....	460
7.7.13.3.2	Include_Mapping .....	461
7.7.13.3.3	IncludeFeatureTyping_Mapping .....	462
7.7.13.3.4	UseCase_Mapping .....	462
7.7.13.3.5	UseCaseActor_Mapping .....	464
7.7.13.3.6	UseCaseActorFeatureTyping_Mapping .....	464
7.7.13.3.7	UseCaseActorMembership_Mapping .....	465
7.7.13.3.8	UseCaseEmptySubjectReferenceUsage_Mapping .....	466
7.7.13.3.9	UseCaseObjectiveMembership_Mapping .....	466
7.7.13.3.10	UseCaseObjectiveRequirementUsage_Mapping .....	467
7.7.13.3.11	UseCaseObjectiveSubjectMembership_Mapping .....	467
7.7.13.3.12	UseCaseSubjectFeatureTyping_Mapping .....	468
7.7.13.3.13	UseCaseSubjectMembership_Mapping .....	469
7.7.13.3.14	UseCaseSubjectReferenceUsage_Mapping .....	469
7.7.14	Values .....	470
7.7.14.1	Overview .....	470
7.7.14.2	UML4SysML::Values elements not mapped .....	471
7.7.14.3	Mapping Specifications .....	471
7.7.14.3.1	EqualOperatorExpressionFeature_Mapping .....	471
7.7.14.3.2	EqualOperatorExpressionFeatureValue_Mapping .....	472
7.7.14.3.3	EqualOperatorExpressionOperandParameterMembership_Mapping .....	472
7.7.14.3.4	Expression_Mapping .....	473
7.7.14.3.5	ExpressionElse_Mapping .....	474
7.7.14.3.6	ExpressionElseMembership_Mapping .....	474
7.7.14.3.7	ExpressionElseSpecification_Mapping .....	475



7.7.14.3.8 LiteralBoolean_Mapping.....	476
7.7.14.3.9 LiteralInteger_Mapping.....	476
7.7.14.3.10 LiteralNull_Mapping.....	477
7.7.14.3.11 LiteralReal_Mapping.....	477
7.7.14.3.12 LiteralSpecificationCommon_Mapping.....	478
7.7.14.3.13 LiteralSpecificationFeatureTyping_Mapping.....	479
7.7.14.3.14 LiteralString_Mapping.....	479
7.7.14.3.15 LiteralUnlimitedUnbounded_Mapping.....	480
7.7.14.3.16 LiteralUnlimitedInteger_Mapping.....	480
7.7.14.3.17 OpaqueExpressionAsValue_Mapping.....	481
7.7.14.3.18 OpaqueExpression_Mapping.....	482
7.7.14.3.19 OpaqueExpressionFeature_Mapping.....	483
7.7.14.3.20 OpaqueExpressionFeatureFeature_Mapping.....	483
7.7.14.3.21 OpaqueExpressionFeatureFeatureMembership_Mapping.....	484
7.7.14.3.22 OpaqueExpressionFeatureValue_Mapping.....	484
7.7.14.3.23 OpaqueExpressionFeatureValueExpression_Mapping.....	485
7.7.14.3.24 OpaqueExpressionFeatureValueExpressionMembership_Mapping.....	486
7.7.14.3.25 OpaqueExpressionMembership_Mapping.....	486
7.7.14.3.26 OpaqueExpressionParameterMembership_Mapping.....	487
7.7.14.3.27 OpaqueExpressionReferenceUsageReturnParameterMembership_Mapping.....	488
7.7.14.3.28 OpaqueExpressionReferenceUsage_Mapping.....	488
7.7.14.3.29 OpaqueExpressionReferenceUsageFeatureTyping_Mapping.....	489
7.7.14.3.30 OpaqueExpressionReferenceUsageUntyped_Mapping.....	489
7.7.14.3.31 OpaqueExpressionSpecification_Mapping.....	490
7.7.14.3.32 TimeExpression_Mapping.....	491
7.7.14.3.33 ValueSpecification_Mapping.....	492
7.8 Mappings from SysML v1.7 stereotypes.....	492
7.8.1 Overview.....	492
7.8.2 Activities.....	493
7.8.2.1 Overview.....	493
7.8.2.2 SysML::Activities elements not mapped.....	493
7.8.2.3 Mapping Specifications.....	493
7.8.2.3.1 ProbabilityMetadataUsage_Mapping.....	493
7.8.2.3.2 ProbabilityMetadataUsageFeatureMembership_Mapping.....	494
7.8.2.3.3 ProbabilityMetadataUsageFeatureTyping_Mapping.....	495
7.8.2.3.4 ProbabilityMetadataUsageReferenceUsage_Mapping.....	495
7.8.2.3.5 ProbabilityMetadataUsageReferenceUsageFeatureValue_Mapping.....	496
7.8.2.3.6 ProbabilityMetadataUsageReferenceUsageRedefinition_Mapping.....	497
7.8.2.3.7 ProbabilityOwningMembership_Mapping.....	498
7.8.2.3.8 RateMetadataUsage_Mapping.....	498
7.8.2.3.9 RateMetadataUsageContinuousFeatureMembership_Mapping.....	499
7.8.2.3.10 RateMetadataUsageFeatureValue_Mapping.....	500
7.8.2.3.11 RateMetadataUsageContinuousReferenceUsage_Mapping.....	501
7.8.2.3.12 RateMetadataUsageContinuousReferenceUsageRedefinition_Mapping.....	502
7.8.2.3.13 RateMetadataUsageDiscreteFeatureMembership_Mapping.....	502
7.8.2.3.14 RateMetadataUsageDiscreteReferenceUsage_Mapping.....	503
7.8.2.3.15 RateMetadataUsageDiscreteReferenceUsageRedefinition_Mapping.....	504
7.8.2.3.16 RateMetadataUsageFeatureTyping_Mapping.....	504
7.8.2.3.17 RateOwningMembership_Mapping.....	505
7.8.2.3.18 Model Libraries.....	506
7.8.2.3.18.1 ControlValues.....	506
7.8.2.3.18.1.1 ControlValueKind.....	506
7.8.3 Allocations.....	506
7.8.3.1 Overview.....	506
7.8.3.2 SysML::Allocations elements not mapped.....	506

7.8.3.3 Mapping Specifications .....	506
7.8.3.3.1 Allocation_Mapping.....	506
7.8.3.3.2 AllocationFeatureMembership_Mapping .....	507
7.8.3.3.3 AllocationFeatureTyping_Mapping .....	508
7.8.3.3.4 AllocationReferenceUsage_Mapping .....	509
7.8.3.3.5 AllocationSourceReferenceUsageRedefinition_Mapping .....	510
7.8.3.3.6 AllocationTargetFeatureMembership_Mapping .....	510
7.8.3.3.7 AllocationTargetReferenceUsage_Mapping .....	511
7.8.3.3.8 AllocationTargetReferenceUsageRedefinition_Mapping .....	511
7.8.3.3.9 AllocationUsage_Mapping.....	512
7.8.3.3.10 AllocationUsageEndFeatureMembership_Mapping.....	513
7.8.3.3.11 AllocationUsageFeature_Mapping.....	513
7.8.3.3.12 AllocationUsageFeatureChaining_Mapping .....	514
7.8.3.3.13 AllocationUsageFeatureChainingChainedFeature_Mapping.....	515
7.8.3.3.14 AllocationUsageFeatureMembership_Mapping .....	515
7.8.3.3.15 AllocationUsageFeatureSubsetting_Mapping .....	516
7.8.3.3.16 AllocationUsageFeatureSubsettingFeature_Mapping.....	517
7.8.3.3.17 AllocationUsageTargetEndFeatureMembership_Mapping.....	517
7.8.3.3.18 AllocationUsageTargetFeature_Mapping .....	518
7.8.3.3.19 AllocationUsageTargetFeatureChaining_Mapping.....	518
7.8.3.3.20 AllocationUsageTargetFeatureSubsetting_Mapping .....	519
7.8.3.3.21 AllocationUsageTargetFeatureSubsettingFeature_Mapping .....	520
7.8.4 Blocks.....	520
7.8.4.1 Overview .....	520
7.8.4.2 SysML::Blocks elements not mapped.....	521
7.8.4.3 Mapping Specifications .....	522
7.8.4.3.1 AssociationBlock_Mapping .....	522
7.8.4.3.2 BindingConnector_Mapping .....	522
7.8.4.3.3 Block_Mapping .....	523
7.8.4.3.4 EncapsulatedBlock_Mapping.....	524
7.8.4.3.5 EncapsulatedBlockMetadataMembership_Mapping.....	525
7.8.4.3.6 EncapsulatedBlockMetadata_Mapping.....	526
7.8.4.3.7 EncapsulatedBlockMetadataFeatureMembership_Mapping.....	526
7.8.4.3.8 EncapsulatedBlockMetadataFeatureTyping_Mapping .....	527
7.8.4.3.9 EncapsulatedBlockMetadataReferenceUsage_Mapping.....	527
7.8.4.3.10 EncapsulatedBlockMetadataFeatureValue_Mapping .....	528
7.8.4.3.11 EncapsulatedBlockMetadataRedefinition_Mapping.....	529
7.8.4.3.12 FlowPropertyPart_Mapping .....	529
7.8.4.3.13 PartProperty_Mapping .....	530
7.8.4.3.14 Model Libraries .....	531
7.8.4.3.14.1 PrimitiveValueTypes .....	531
7.8.4.3.14.1.1 Boolean.....	531
7.8.4.3.14.1.2 Complex.....	531
7.8.4.3.14.1.3 Integer .....	531
7.8.4.3.14.1.4 Number .....	531
7.8.4.3.14.1.5 Real.....	532
7.8.4.3.14.1.6 String .....	532
7.8.4.3.14.2 UnitAndQuantityKind .....	532
7.8.4.3.14.2.1 QuantityKind .....	532
7.8.4.3.14.2.2 Unit .....	532
7.8.4.3.15 ValueType_Mapping.....	532
7.8.5 ConstraintBlocks .....	533
7.8.5.1 Overview .....	533
7.8.5.2 Mapping Specifications .....	533
7.8.5.2.1 ConstraintBlock_Mapping .....	533
7.8.5.2.2 ConstraintParameter_Mapping.....	534

7.8.6 Model Elements .....	535
7.8.6.1 Overview .....	535
7.8.6.2 SysML::ModelElements elements not mapped .....	535
7.8.6.3 Mapping Specifications .....	535
7.8.6.3.1 ProblemRationaleMetadataFeatureMembership_Mapping .....	535
7.8.6.3.2 ProblemRationaleMetadataFeatureTyping_Mapping .....	536
7.8.6.3.3 ProblemRationaleMetadataReferenceUsage_Mapping .....	537
7.8.6.3.4 ProblemRationaleMetadataFeatureValue_Mapping .....	537
7.8.6.3.5 ProblemRationaleMetadataMembership_Mapping .....	538
7.8.6.3.6 Concern_Mapping .....	538
7.8.6.3.7 ConcernDocumentation_Mapping .....	540
7.8.6.3.8 ConcernOwningMembership_Mapping .....	540
7.8.6.3.9 ConcernStakeholderMembership_Mapping .....	541
7.8.6.3.10 ConcernStakeholderPartUsage_Mapping .....	542
7.8.6.3.11 ConcernStakeholderPartUsageFeatureTyping_Mapping .....	542
7.8.6.3.12 ConcernStakeholderPartUsageOwningMembership_Mapping .....	543
7.8.6.3.13 ConcernStakeholderPartUsageFeature_Mapping .....	544
7.8.6.3.14 ElementGroup_Mapping .....	544
7.8.6.3.15 ElementGroupMetadaMembership_Mapping .....	545
7.8.6.3.16 ElementGroupMetadataFeatureMembership_Mapping .....	546
7.8.6.3.17 ElementGroupMetadataFeatureTyping_Mapping .....	547
7.8.6.3.18 ElementGroupMetadataFeatureValue_Mapping .....	547
7.8.6.3.19 ElementGroupMetadataRedefinition_Mapping .....	548
7.8.6.3.20 ElementGroupMetadataReferenceUsage_Mapping .....	549
7.8.6.3.21 ElementGroupMetadataUsage_Mapping .....	549
7.8.6.3.22 ProblemRationale_Mapping .....	550
7.8.6.3.23 ProblemRationaleMetadataRedefinition_Mapping .....	551
7.8.6.3.24 ProblemRationaleMetadataUsage_Mapping .....	552
7.8.6.3.25 Stakeholder_Mapping .....	552
7.8.6.3.26 StakeholderMetadataUsage_Mapping .....	554
7.8.6.3.27 StakeholderMetadataFeatureMembership_Mapping .....	554
7.8.6.3.28 StakeholderMetadataFeatureTyping_Mapping .....	555
7.8.6.3.29 StakeholderMetadataOwningMembership .....	555
7.8.6.3.30 StakeholderMetadataReferenceUsage_Mapping .....	556
7.8.6.3.31 StakeholderMetadataReferenceUsageFeatureValue_Mapping .....	557
7.8.6.3.32 StakeholderMetadataReferenceUsageRedefinition_Mapping .....	557
7.8.6.3.33 Viewpoint_Mapping .....	558
7.8.6.3.34 ViewpointConcernReferenceSubsetting_Mapping .....	560
7.8.6.3.35 ViewpointConcernUsage_Mapping .....	560
7.8.6.3.36 ViewpointConstraintUsage_Mapping .....	561
7.8.6.3.37 ViewpointConstraintUsageDocumentation_Mapping .....	562
7.8.6.3.38 ViewpointConstraintUsageOwningMembership_Mapping .....	562
7.8.6.3.39 ViewpointFramedConcernMembership_Mapping .....	563
7.8.6.3.40 ViewpointLanguagesMetadataFeatureMembership_Mapping .....	564
7.8.6.3.41 ViewpointLanguagesMetadataFeatureValue_Mapping .....	564
7.8.6.3.42 ViewpointLanguagesMetadataRedefinition_Mapping .....	565
7.8.6.3.43 ViewpointLanguagesMetadataReferenceUsage_Mapping .....	565
7.8.6.3.44 ViewpointMetadataFeatureTyping_Mapping .....	566
7.8.6.3.45 ViewpointLanguagesMetadataOperatorExpression_Mapping .....	567
7.8.6.3.46 ViewpointMetadataOwningMembership_Mapping .....	567
7.8.6.3.47 ViewpointMetadataUsage_Mapping .....	568
7.8.6.3.48 ViewpointPresentationsMetadataFeatureMembership_Mapping .....	569
7.8.6.3.49 ViewpointPresentationsMetadataFeatureValue_Mapping .....	569
7.8.6.3.50 ViewpointPresentationsMetadataOperatorExpression_Mapping .....	570
7.8.6.3.51 ViewpointPresentationsMetadataRedefinition_Mapping .....	571
7.8.6.3.52 ViewpointPresentationsMetadataReferenceUsage_Mapping .....	571

7.8.6.3.53 ViewpointRenderingFeatureMembership_Mapping.....	572
7.8.6.3.54 ViewpointRenderingUsage_Mapping.....	573
7.8.6.3.55 ViewpointRenderingUsageActionUsage_Mapping.....	573
7.8.6.3.56 ViewpointRenderingUsageActionUsageFeatureMembership_Mapping.....	574
7.8.6.3.57 ViewpointRenderingUsageActionUsageFeatureTyping_Mapping.....	575
7.8.6.3.58 ViewpointRequirementConstraintMembership_Mapping.....	575
7.8.6.3.59 ViewpointSatisfyFeatureMembership_Mapping.....	576
7.8.6.3.60 ViewpointSatisfyRequirementUsage_Mapping.....	576
7.8.6.3.61 ViewpointSatisfyRequirementUsageReferenceSubsetting_Mapping.....	577
7.8.6.3.62 ViewpointViewpointUsage_Mapping.....	578
7.8.6.3.63 ViewpointViewpointUsageFeatureMembership_Mapping.....	578
7.8.7 PortsAndFlows.....	579
7.8.7.1 Overview.....	579
7.8.7.2 SysML::Ports&Flows elements not mapped.....	580
7.8.7.3 Mapping Specifications.....	580
7.8.7.3.1 AcceptChangeStructuralFeatureEventAction_Mapping.....	580
7.8.7.3.2 CommonFullPort_Mapping.....	581
7.8.7.3.3 ConjugatedPortDefinition_Mapping.....	581
7.8.7.3.4 FlowProperty_Mapping.....	582
7.8.7.3.5 FlowPropertyAttribute_Mapping.....	583
7.8.7.3.6 FlowPropertyUntyped_Mapping.....	584
7.8.7.3.7 FullPort_Mapping.....	585
7.8.7.3.8 FullPortMetadata_Mapping.....	585
7.8.7.3.9 FullPortMetadataFeatureMembership_Mapping.....	586
7.8.7.3.10 FullPortMetadataFeatureTyping_Mapping.....	587
7.8.7.3.11 FullPortMetadataOwningMembership_Mapping.....	587
7.8.7.3.12 FullPortMetadataReferenceUsage_Mapping.....	588
7.8.7.3.13 FullPortMetadataReferenceUsageFeatureValue_Mapping.....	588
7.8.7.3.14 FullPortMetadataReferenceUsageRedefinition_Mapping.....	589
7.8.7.3.15 FullPortUntyped_Mapping.....	590
7.8.7.3.16 InterfaceBlock_Mapping.....	591
7.8.7.3.17 InterfaceBlockConjugated_Mapping.....	591
7.8.7.3.18 InterfaceBlockOwningMembership_Mapping.....	592
7.8.7.3.19 OperationDirectedFeature_Mapping.....	593
7.8.7.3.20 PortConjugation_Mapping.....	593
7.8.8 Requirements.....	594
7.8.8.1 Overview.....	594
7.8.8.2 SysML::Requirements elements not mapped.....	595
7.8.8.3 Mapping Specifications.....	595
7.8.8.3.1 DeriveReq_Mapping.....	595
7.8.8.3.2 DeriveReqFeatureTyping_Mapping.....	596
7.8.8.3.3 DeriveReqSourceEndFeatureMembership_Mapping.....	596
7.8.8.3.4 DeriveReqSourceFeature_Mapping.....	597
7.8.8.3.5 DeriveReqSourceFeatureReferenceSubsetting_Mapping.....	598
7.8.8.3.6 DeriveReqTargetEndFeatureMembership_Mapping.....	598
7.8.8.3.7 DeriveReqTargetFeature_Mapping.....	599
7.8.8.3.8 DeriveReqTargetFeatureReferenceSubsetting_Mapping.....	600
7.8.8.3.9 Refine_Mapping.....	600
7.8.8.3.10 RefineAnnotation_Mapping.....	601
7.8.8.3.11 RefineMetadataFeatureMembership_Mapping.....	602
7.8.8.3.12 RefineMetadataReferenceUsage_Mapping.....	602
7.8.8.3.13 RefineMetadataReferenceUsageFeatureValue_Mapping.....	603
7.8.8.3.14 RefineMetadataReferenceUsageRedefinition_Mapping.....	604
7.8.8.3.15 RefineMetadataUsage_Mapping.....	604
7.8.8.3.16 RefineMetadataUsageFeatureTyping_Mapping.....	605
7.8.8.3.17 Requirement_Mapping.....	605

7.8.8.3.18 RequirementDocumentation_Mapping .....	607
7.8.8.3.19 RequirementDocumentationMembership_Mapping .....	607
7.8.8.3.20 RequirementSubject_Mapping .....	608
7.8.8.3.21 RequirementSubjectMembership_Mapping .....	608
7.8.8.3.22 Satisfy_Mapping .....	609
7.8.8.3.23 SatisfyReferenceUsage_Mapping .....	610
7.8.8.3.24 SatisfyReferenceUsageFeatureMembership_Mapping .....	611
7.8.8.3.25 SatisfySubjectReferenceUsage_Mapping .....	612
7.8.8.3.26 SatisfySubjectReferenceUsageValue_Mapping .....	612
7.8.8.3.27 SatisfySubjectReferenceUsageValueFeature_Mapping .....	613
7.8.8.3.28 SatisfySubjectReferenceUsageFeatureChaining_Mapping .....	614
7.8.8.3.29 SatisfySubjectReferenceUsageValueFeatureChainingProperty_Mapping .....	614
7.8.8.3.30 SatisfySubjectReferenceUsageFeatureValue_Mapping .....	615
7.8.8.3.31 SatisfySubjectReferenceUsageValueOwningMembership_Mapping .....	616
7.8.8.3.32 SatisfySubjectSubjectMembership_Mapping .....	616
7.8.8.3.33 SatisfyFeatureTyping_Mapping .....	617
7.8.8.3.34 SatisfyReferenceUsageFeatureTyping_Mapping .....	618
7.8.8.3.35 TestCaseActivity_Mapping .....	618
7.8.8.3.36 TestCaseActivityReturnParameterMembership_Mapping .....	619
7.8.8.3.37 TestCaseVerifyObjectiveMembership_Mapping .....	620
7.8.8.3.38 TestCaseVerifyObjectiveRequirementUsage_Mapping .....	620
7.8.8.3.39 TestCaseVerifyRequirementUsageReferenceSubsetting_Mapping .....	621
7.8.8.3.40 TestCaseVerifyRequirementUsage_Mapping .....	622
7.8.8.3.41 Trace_Mapping .....	622
7.8.8.3.42 TraceAnnotation_Mapping .....	623
7.8.8.3.43 TraceMetadataFeatureMembership_Mapping .....	624
7.8.8.3.44 TraceMetadataReferenceUsage_Mapping .....	625
7.8.8.3.45 TraceMetadataReferenceUsageFeatureValue_Mapping .....	625
7.8.8.3.46 TraceMetadataReferenceUsageRedefinition_Mapping .....	626
7.8.8.3.47 TraceMetadataUsage_Mapping .....	626
7.8.8.3.48 TraceMetadataUsageFeatureTyping_Mapping .....	627
7.8.8.3.49 Verify_Mapping .....	628
7.8.8.3.50 Model Libraries .....	629
7.8.8.3.50.1 Verdicts .....	629
7.8.8.3.50.1.1 VerdictKind .....	629

# List of Tables

1. List of all mappings .....	83
2. List of SysML v1 elements not mapped of this section.....	84
3. List of all mappings .....	212
4. List of SysML v1 elements not mapped of this section.....	212
5. List of all mappings .....	264
6. List of all mappings .....	300
7. List of SysML v1 elements not mapped of this section.....	301
8. List of all mappings .....	339
9. List of all mappings .....	353
10. List of all mappings .....	359
11. List of SysML v1 elements not mapped of this section.....	359
12. List of all mappings .....	373
13. List of SysML v1 elements not mapped of this section.....	373
14. List of all mappings .....	397
15. List of all mappings .....	412
16. List of all mappings .....	432
17. List of all mappings .....	460
18. List of SysML v1 elements not mapped of this section.....	460
19. List of all mappings .....	470
20. List of SysML v1 elements not mapped of this section.....	471
21. List of all mappings .....	493
22. List of SysML v1 elements not mapped of this section.....	493
23. List of all mappings .....	506
24. List of SysML v1 elements not mapped of this section.....	506
25. List of all mappings .....	520
26. List of SysML v1 elements not mapped of this section.....	521
27. List of all mappings .....	533
28. List of all mappings .....	535
29. List of SysML v1 elements not mapped of this section.....	535
30. List of all mappings .....	579
31. List of SysML v1 elements not mapped of this section.....	580
32. List of all mappings .....	594
33. List of SysML v1 elements not mapped of this section.....	595







# 1 Scope

This specification describes a transformation for a semantic translation from SysML v1 [SysMLv1] to SysML v2 [SysMLv2] in a precise way. (In this document, "SysML v1" refers to SysML v1.7, the last version of SysML prior to v2.0, and "SysML v2" refers to SysML v2.0, or whatever version corresponds to the current version of this specification.)

The main intent is to provide the rules on which automated conversions of SysML v1 models to the SysML v2 standard can be developed. In addition, this annex can be considered an educational document that provides useful information for people who would like to compare using SysML v2 and using SysML v1.

More sophisticated applications of this transformation can also be envisaged. For instance, a SysML v1 conformant tool could use this transformation to implement a limited subset of the SysML v2 API that will provide "SysMLv2-like" read-only access to its SysMLv1 models for external applications.



## 2 Conformance

A tool shall demonstrate *conformance* with this specification by meeting all of the following requirements.

1. The tool shall implement the UML4SysML abstract syntax and SysML v1 profile conformant with [SysMLv1]. The tool should, but is not required, to provide the ability to import a SysML v1 model using standard XMI Model Interchange format [XMI].
2. The tool shall implement the SysML v2 abstract syntax conformant with [SysML v2]. The tool should, but is not required, to provide the ability to export a SysML v2 model KerML-standard model interchange project (see [KerML], Clause 10; see also [SysML v2], Clause 2).
3. The tool shall implement a transformation from an abstract syntax representation of an input SysML v1 model to the abstract syntax representation of an output SysML v2, as specified in of this specification.

A tool may claim *partial conformance* with this specification by satisfying the first two requirements above, but only implementing an identified subset of the mappings specified in and . (Note that care must also be taken that certain mappings depend on other mappings, and so cannot reasonably be implemented separately.)

**Note.** A tool that conforms to [SysMLv2] is not required to necessarily implement a transformation conformant with this specification, or it may implement a SysML v1 to v2 transformation that is not claimed to conform with the transformation defined in this specification.



## 3 Normative References

The following normative documents contain provisions which, through reference in this text, constitute provisions of this specification.

[KerML] *Kernel Modeling Language (KerML)*, Version 1.0  
<https://www.omg.org/spec/KerML/1.0>

[MOF] *Meta Object Facility*, Version 2.5.1  
<https://www.omg.org/spec/MOF/2.5.1>

[OCL] *Object Constraint Language*, Version 2.4  
<https://www.omg.org/spec/OCL/2.4>

[SysML v1] *OMG Systems Modeling Language (SysML)*, Version 1.7  
<https://www.omg.org/spec/SysML/1.7>

[SysML v2] *OMG Systems Modeling Language (SysML)*, Version 2.0  
<https://www.omg.org/spec/SysML/2.0>

[UML] *Unified Modeling Language (UML)*, Version 2.5.1  
<https://www.omg.org/spec/UML/2.5.1>

[XMI] *XML Metadata Interchange*, Version 2.5.1  
<https://www.omg.org/spec/XMI/2.5.1>



## 4 Terms and Definitions

Various terms and definitions are specified throughout the body of this specification.





## 5 Symbols

No special symbols are defined in this specification.



# 6 Introduction

## 6.1 Mapping Approach

The SysML v1 to v2 transformation is specified by directional mappings between UML metaclasses or stereotypes that are part of the SysML v1 specification [SysMLv1] (referenced below as the "SysML v1 scope") on the one hand, and the set of the metaclasses defined in the KerML [KerML] and SysMLv2 [SysMLv2] specifications (referenced below as "SysML v2") in the other hand. Some library classes are also involved.

Each mapping is a directed relationship that reifies a semantic link between a concept belonging to the SysML v1 scope on the source side and one concept belonging to SysML v2 (or one conforming library element) on the target side. As a set, those mappings constitute a declarative specification of a formal transformation that describes how the information encoded by the SysML v1 concepts can be reliably represented using constructs of SysML v2 metaclass instances.

In this approach, a mapping is represented by a UML class that has a pair of associations. One provides the `from` end that designates the source SysML v1 concept, while the other provides the `to` end that designates the target SysML v2 metaclass.

In addition to those associations, a mapping class provides a set of operations defining how the values of non-derived properties of the target metaclass instance have to be computed based on property values reachable from the source object. The computation algorithm is provided by the body condition of those operations and expressed using OCL code.

Note that the values assigned to the properties of the target object shall be instances of SysML v2 metaclasses, coming themselves from transformations of SysMLv1 objects to SysMLv2 objects. Since the specification is declarative, the order in which the individual transformations shall happen is not imposed. It is up to a conforming implementation to deal with this. Instead, the `getMapped` static operation is provided for referring to the result of a transformation from within an OCL rule. It returns a (possibly undefined) value, that is typed by the target metaclass of the mapping class from which it is invoked.

Each mapping class enables the transformation of any object that has the type specified by the `from` role to an object of the type specified by the `to` role, as long as it is not overloaded by a more specific mapping definition. In other words, assume a mapping is specified for the class `A` (i.e., it has `A` typing its `from` property), then it applies to any instance of a class `B` if `B` is a subclass of `A` and if there is no specialization of that mapping class specified for `B` (i.e., that has `B` typing its `from` property).

It is possible to restrict the applicability of a mapping specification to a specific subset of objects. This is achieved by the `filter` static operation that is evaluated against each candidate object. Only objects of the appropriate type for which this `filter` operation returns `true` shall be translated according to the specifications of that mapping class. The default `filter` operation always returns `true`.

Some mapping classes have one or more qualifiers for their `to` attribute. In such a case, each of those qualifiers reflects the specific property of the source type (i.e. the type of the `from` attribute) that has the same name and the same type. For those specific mappings, it is expected to get one instance of the target class (as specified by the type of the `to` attribute") for each actual combination of value of those properties for a given instance of object of the source type, assuming they pass the applicability filter as described above.

## 6.2 Acknowledgements

The primary authors of this specification document (and also developers of a proof-of-concept implementation of it) are:

- Yves Bernard, Airbus

- Tim Weilkiens, oose

The specification was formally submitted for standardization by the following organizations:

- 88solutions Corporation
- Dassault Systèmes
- GfSE e.V.
- IBM
- INCOSE
- Intercax LLC
- Lockheed Martin Corporation
- MITRE
- Model Driven Solutions, Inc.
- PTC
- Simula Research Laboratory AS
- Thematrix Partners LLC

However, work on the specification was also supported by over 200 people in over 80 organizations that participated in the SysML v2 Submission Team (SST), by contributing use cases, providing critical review and comment, and validating the language design. The following individuals had leadership roles in the SST:

- Manas Bajaj, Intercax LLC (API and services development lead)
- Yves Bernard, Airbus (v1 to v2 transformation co-lead)
- Bjorn Cole, Lockheed Martin Corporation (metamodel development co-lead)
- Sanford Friedenthal, SAF Consulting (SST co-lead, requirements V&V lead)
- Charles Gale, Lockheed Martin Corporation (metamodel development co-lead)
- Karen Ryan, Siemens (metamodel development co-lead)
- Ed Seidewitz, Model Driven Solutions (SST co-lead, pilot implementation lead)
- Tim Weilkiens, oose (v1 to v2 transformation co-lead)

The specification was prepared using CATIA Magic/No Magic modeling tools and the OpenMBEE system for model publication (<http://www.openmbee.org>), supported by the 3DEXPERIENCE platform, with the invaluable support of the following individuals:

- Tyler Anderson, Dassault Systèmes
- Christopher Delp, Jet Propulsion Laboratory
- Jackson Galloway, Dassault Systèmes
- Ivan Gomes, Twingineer
- Doris Lam, Jet Propulsion Laboratory
- Robert Karban, Jet Propulsion Laboratory
- Christopher Klotz, Dassault Systèmes
- John Watson, Lightstreet Consulting

# 7 Mappings

## 7.1 Overview

This Clause is organized in order to match the packages that subdivide the model of the transformation. The `Foundations` package gathers the abstract classes that represent the concepts on top of which the mapping approach is built. The next subclause presents a utility class named `Helper` that provides reusable operations that simplify the OCL statements defining the computation rules of target properties and make them more readable. Libraries play an important role in SysML v2, and a specific one has been created in order to represent semantics equivalent to those of UML/SysML concepts, where needed. It is presented in this subclause as well.

The three next subclauses are dedicated to initializers, factories and generic mappings, respectively. They do not specify mappings, strictly speaking. Instead, they factorize more or less advanced OCL code that will be reused by the actual mapping specifications that are contained in the two last subclauses. The first of them is dedicated to UML metaclass from the UML4SYSML scope, while the second deals with SysML stereotypes more specifically.

## 7.2 Foundations

### 7.2.1 Overview

The concepts defined by KerML/SysML v2 are relatively similar to those of UML/SysML v1, but the ways they are built are different. This makes the specification of the global transformation quite complex. In order to keep it manageable, specific kinds of foundational classes are provided. They represent concepts on which classical "model to model" transformation technologies rely:

- The mappings built on top of the abstract class `Mapping` shall be executed only when they are explicitly called. Each call shall produce a new target element, whatever the source element. It specifies a `from` property typed by the `UML::CommonStructure::Element` metaclass that shall be redefined by any of its subclass for specifying the convenient type of source element. Also it specifies a default (neutral) filter and a set of `getMapped` operations for various purposes: regular mapping result, qualified mapping result and mapping result for a collection of elements.
- The mappings built on top of the abstract class `UniqueMapping`, specified as a specialization of the `Mapping` class, shall produce only one target element for a given source element, whatever the number of time they are called.
- The mappings built on top of the abstract class `MainMapping`, specified as a specialization of the `UniqueMapping` class, shall be systematically executed (i.e. implicitly called) for all the elements that match both their source type and filter. There can be at most one main mapping for a given source type and only one target element shall be produced for a given source element.

The corresponding classes are located in the `Foundations` package.

Sometimes, it is necessary to be able to generate elements in the target model without having to provide an explicit link with a source element. In such a case, a mapping class is not appropriate. Instead the mapping framework provides the concept of a `Factory`.

Last, the concept of an `Initializer` allows the factorization of the specification of properties' default values that can be inherited by mappings and factories, as convenient.

In the model of the transformation that is specified here, all of the abstract classes of this `Foundations` package are subject to direct or indirect subclassing. In other words, this specification is built as a set of interrelated initializers, factories, regular, unique and main mappings, where the initializers' operation factorizes the specification of default values for their target element, wherever possible. Those "default operations" are either used as-is or redefined by mappings or factories that can inherit for a specific initializer, as appropriate.

## 7.2.2 Foundational class specifications

### 7.2.2.1 UniqueMapping

#### Description

The mappings built on top of the abstract class UniqueMapping are a specific kind of Mappings that are intended to produce only one target element for a given source element, whatever the number of time they are called. If a getMapped is called several time with the same source element, the target element returned shall always be the same.

#### General Classes

- Mapping (from Foundations)

### 7.2.2.2 Factory

#### Description

Similarly to the well-known to the homonyms software design pattern, a Factory can be used for specifying the production of a target element without any link with a source element. Factories have in common with mapping classes the operations that specify how the properties of the target element shall be computed and the "to" property that specifies the type of the target element. However factories do not define source element. Instead, they can have parameters. Those parameters, if any, shall be specified by properties with appropriate types and multiplicities. Factories are expected to provide a "create" operation with parameters matching in type and multiplicity the properties that are intended to specify them.

#### General Classes

- Initializer (from Foundations)

### 7.2.2.3 Mapping

#### Description

This is the generic abstract class that provides the basic features of any mapping class mapping. The mappings built on top of the abstract class Mapping are intended to be executed only when explicitly called (e.g. by the rule of another mapping class). It specifies a "from" property typed by the UML::CommonStructure::Element metaclass that shall be redefined by any of its subclass for specifying the convenient type of source element. Also it specifies a default (neutral) filter and a set of getMapped operations for various purposes: regular mapping result, qualified mapping result and mapping result for a collection of elements. Each call to the getMapped operation shall produce a new target element, whatever the source element provided. Instances of Mapping class are represent a link between one source element and the target element produced by the transformation specified by that mapping class.

#### General Classes

- Initializer (from Foundations)

#### Association Ends

- from : Element [1]

#### Operations

- filter (in src : Element) : Boolean [1]  
returns "true" if the element provided as the actual parameter value can have a mapping to an instance of

the type specified by the "to" attribute (i.e. can be used as a value for the "from" attribute)

```
true
```

- `getMapped (in fromVar : Element) : Element [1]`

**postConditions:**

```
self.filter(fromVar) and
self.to.allFeatures()->selectByKind(UML::Property)->reject(isDerived)
->forAll(p | let ops: Operation = self.allFeatures()
  ->selectByKind(UML::Operation)->any(o | o.name = p.name) in
  p = ops()) and
result = self.to
```

- `getMapped (in fromVar : Element, in qual : Element) : Element [1]`

**postConditions:**

```
self.filter(fromVar) and
self.to.allFeatures()->selectByKind(UML::Property)->reject(isDerived)
->forAll(p | let ops: Operation = self.allFeatures()
  ->selectByKind(UML::Operation)->any(o | o.name = p.name) in
  if ops.ownedParameter
    ->select(p | p.direction = UML::ParameterDirectionKind::_in')
    ->size()=1 then
      p = ops(qual)
  else if ops.ownedParameter
    ->select(p | p.direction = UML::ParameterDirectionKind::_in')
    ->size()=0 then
      p = ops()
  else
    invalid
  endif endif) and
result = self.to
```

- `getMappedColl (in fromColl : Element) : Element [0..*]`

**postConditions:**

```
result = fromColl->collect(e | self.getMapped(e))
```

## 7.2.2.4 MainMapping

### Description

The mappings built on top of the abstract class `MainMapping` are a specific kind of `UniqueMappings` class that are always implicitly called for any element in the source model that match both their source type (as specified by their "from" property) and their filter condition. If more than one main mapping is specified for a given source type, they shall have filters that specify mutually exclusive conditions. Also, as with any unique mapping, only one target element shall be produced for a given source element.

### General Classes

- `UniqueMapping` (from `Foundations`)

### 7.2.2.5 Initializer

#### Description

The abstract class Initializer is the common ancestor of Mapping and Factory. It specifies a "to" property typed by the KerML::Root::Element metaclass that shall be redefined by any of its subclass for specifying the convenient type of target element. Initializers are intended to specify reusable properties' computation rules, mainly for initializing them with default values. Those rules will be inherited or redefined by the sub-classes, as appropriate.

#### Attributes

- /inputs [0..\*]

#### Association Ends

- to : Element [1]

## 7.3 Mapping Helper and Library

### 7.3.1 Helper

#### Description

The Helper class contains operations that are used by multiple mapping classes. The specification is in the bodyCondition.

#### Operations

- actionOwnedRelationship (in src : Element) : Relationship [0..\*]  
Reusable mapping rule for owned relationships of a UML4SysML::Action mapping.

```
let actionInputPin: Set(UML::Element) =
  src.ownedElement->select(e | e.ocIsTypeOf(UML::ActionInputPin)) in
let triggers: Set(UML::Element) =
  src.ownedElement->select(e | e.ocIsKindOf(UML::Trigger)) in
let toElementFMS: Set(UML::Element) =
  src.ownedElement->select(e | e.ocIsKindOf(UML::Pin)) in
let toElementOMS: Set(UML::Element) =
  (((src.ownedElement - toElementFMS) - actionInputPin) - triggers) in
toElementOMS->collect(e | ElementOwningMembership_Mapping.getMapped(e))
->union(toElementFMS->collect(e | ElementFeatureMembership_Mapping.getMapped(e)))
```

- activityOwnedRelationship (in src : Element) : Relationship [0..\*]  
Reusable mapping rule for owned relationships of a UML4SysML::Activity mapping.

```
let initialNodes: Set(UML!Element) = src.ownedElement
->select(e | e.ocIsKindOf(UML!InitialNode)) in
let flowFinalNodes: Set(UML!Element) = src.ownedElement
->select(e | e.ocIsKindOf(UML!FlowFinalNode)) in
let elementsFMS: Set(UML!Element) = (src.ownedElement
->select(e | e.ocIsKindOf(UML!ControlNode)
  or e.ocIsKindOf(UML!Action)
  or e.ocIsKindOf(UML!ActivityEdge)
  or e.ocIsKindOf(UML!Property))-initialNodes)-flowFinalNodes in
let parameters: Set(UML!Parameter) = src.ownedElement
```



```

->select(e | e.ocIsKindOf(UML!Parameter)) in
let ignoreParameterNodes: Set(UML!ActivityParameterNode) = src.ownedElement
->select(e | e.ocIsKindOf(UML!ActivityParameterNode)) in
let ignoreActivityPartition: Set(UML!ActivityPartition) = src.ownedElement
->select(e | e.ocIsKindOf(UML!ActivityPartition)) in
let ignoreInterruptibleActivityRegion: Set(UML!InterruptibleActivityRegion)
= src.ownedElement
->select(e | e.ocIsKindOf(UML!InterruptibleActivityRegion)) in
let ownedClassifier: Sequence(UML!Classifier) = src.ownedElement
->select(e | e.ocIsKindOf(UML!Classifier)) in
let variables: Sequence(UML!Variable) = src.ownedElement
->select(e | e.ocIsKindOf(UML!Variable)) in
let parameterSets: Set(UML!ParameterSet) = src.ownedElement
->select(e | e.ocIsKindOf(UML!ParameterSet)) in
let elementsOMS: Set(UML!Element) = (((((((((src.ownedElement-initialNodes)-
flowFinalNodes)-elementsFMS)-parameters)-ignoreParameterNodes)-
ignoreActivityPartition)-ignoreInterruptibleActivityRegion)-ownedClassifier)-
variables)-parameterSets)-Set{src.classifierBehavior}) in
let memberships: Sequence(UML!Element) = elementsOMS
->collect(e | thisModule.ElementOwningMembership_Mapping((e)))
->union(initialNodes
->collect(e | thisModule.InitialNodeMembership_Mapping((e))))
->union(flowFinalNodes
->collect(e | thisModule.FlowFinalNodeMembership_Mapping((e))))
->union(elementsFMS
->collect(e | thisModule.ElementFeatureMembership_Mapping((e))))
->union(variables
->collect(e | thisModule.VariableMembership_Mapping((e))))
->union(parameterSets
->collect(e | thisModule.ParameterSetMembership_Mapping((e))))
->union(ownedClassifier
->collect(e | thisModule.ElementOwningMembership_Mapping((e)))) in
if src.classifierBehavior.ocIsUndefined() then
memberships
else
memberships
->append(thisModule.BehavioredClassifierFeatureMembership_Mapping((src)))
endif
endif

```

- **createUUID () : String [1]**  
Creates a UUID. The specification is implementation-specific and therefore cannot provided here.
- **excludedPin (in pin : Pin) : Boolean [1]**  
Checks if a pin is excluded from the transformation, because it is already defined as a parameter in the SysMLv1Library.

```

if (pin.owner.ocIsTypeOf(UML::AddVariableValueAction) and
(pin.name = 'value' or pin.name = 'insertAt')) then
true
else if (pin.owner.ocIsTypeOf(UML::AddStructuralFeatureValueAction) and
(pin.name = 'value' or pin.name = 'insertAt' or pin.name = 'object')) then
true
else
false
endif endif

```

- **getAppliedStereotypes (in element : Element) : Stereotype [0..\*]**  
Returns the list of applied stereotypes. The specification is implementation-specific and therefore cannot provided here.

- `getEnumerationType (in t : Enumeration) : EnumerationDefinition [1]`  
Maps a given UML4SysM::Enumeration to the appropriate SysML v2 EnumerationDefinition.

```

let enum: SYSML2::EnumerationDefinition =
  Enumeration_Mapping.getMapped(t) in
if enum.ocIsKindOf(SYSML2::EnumerationDefinition) then
  enum
else if t.name = 'VerdictKind' then
  SYSML2::EnumerationDefinition.allInstances()
  ->any(e | e.qualifiedName = 'VerificationCases::VerdictKind')

  else if t = UML::ParameterDirectionKind then
    KerML::FeatureDirectionKind

    else if t.qualifiedName =
      'SysML::Libraries::ControlValues::ControlValueKind' then
      SYSML2::EnumerationDefinition.allInstances()
      ->any(e | e.qualifiedName =
        'SysMLv1Library::Enumerations::ControlValueKind')

      else
        SYSML2::EnumerationDefinition.allInstances()
        ->any(e | e.qualifiedName =
          'SysMLv1Library::Enumerations::' + t.name)
      endif
    endif
  endif
endif

```

- `getFlowDirectionKind (in v : EnumerationLiteral) : FeatureDirectionKind [1]`  
Maps a given SysMLv1 feature direction enumeration literal to a SysML v2 FeatureDirectionKind enumeration literal.

```

if v.enumeration.qualifiedName =
  'SysML::Ports&Flows::FlowDirectionKind' then
  if v.name = 'out' then
    KerML::FeatureDirectionKind::_'out'
  else if v.name = 'in' then
    KerML::FeatureDirectionKind::_'in'
  else if v.name = 'inout' then
    KerML::FeatureDirectionKind::inout
  else
    invalid
  endif endif endif
else
  invalid
endif

```

- `getID (in src : Element) : String [1]`  
Returns the identifier of a UML4SysML::Element. The specification is implementation-specific and therefore cannot be provided here.
- `getKerMLFeatureDirectionKind (in v : EnumerationLiteral) : FeatureDirectionKind [1]`  
Maps a given SysMLv1 feature direction enumeration literal to a SysML v2 FeatureDirectionKind enumeration literal.

```

if v.enumeration.qualifiedName =
  'SysML::Ports&Flows::FeatureDirectionKind' or

```

```

    v.enumeration.qualifiedName = 'SysML::Ports&Flows::FeatureDirection' then
    if v = SysML::FeatureDirectionKind::provided then
        KerML::FeatureDirectionKind::_out'
    else if (v = SysML::FeatureDirectionKind::required) then
        KerML::FeatureDirectionKind::_in'
    else if (v = SysML::FeatureDirectionKind::providedRequired) then
        KerML::FeatureDirectionKind::inout
    else
        invalid
    endif endif endif
else
    invalid
endif

```

- **getKerMLParameterDirectionKind (in v : ParameterDirectionKind) : FeatureDirectionKind [1]**  
Maps a given SysMLv1 parameter direction enumeration literal to a SysML v2 FeatureDirectionKind enumeration literal.

```

    if v = UML::ParameterDirectionKind::_in' then
        KerML::FeatureDirectionKind::_in'
    else if (v = UML::ParameterDirectionKind::return) then
        KerML::FeatureDirectionKind::out
    else if (v = UML::ParameterDirectionKind::out) then
        KerML::FeatureDirectionKind::out
    else if (v = UML::ParameterDirectionKind::inout) then
        KerML::FeatureDirectionKind::inout
    else
        invalid
    endif endif endif endif

```

- **getKerMLVisibilityKind (in v : VisibilityKind) : VisibilityKind [1]**  
Maps a given UML4SysML::VisibilityKind enumeration literal to a SysML v2 VisibilityKind enumeration literal.

```

    if (v = UML::VisibilityKind::public) then
        KerML::VisibilityKind::public
    else if (v = UML::VisibilityKind::protected) then
        KerML::VisibilityKind::protected
    else if (v = UML::VisibilityKind::private) then
        KerML::VisibilityKind::private
    else if (v = UML::VisibilityKind::package) then
        KerML::VisibilityKind::public
    else
        invalid
    endif endif endif endif

```

- **getMetadataByName (in mdName : String) : AttributeDefinition [1]**  
Returns the metadata attribute definition element for a given metadata name.

```

SYSML2::AttributeDefiniton.allInstances()->any(e | e.name = mdName)

```

- **getMultiplicityRangeByName (in name : String) : MultiplicityRange [0..1]**  
This operation retrieve a frequently used multiplicity range defiend in the KerML Base Library

```

SYSML2::DataType.allInstances()
->any(e | e.qualifiedName = 'Base::' + name)

```

- **getRequirementStereotype** (in element : NamedElement) : Stereotype [0..1]  
Returns the requirement stereotype for a given element.

```
let stereotypes: Set(UML::Stereotype) =
  Helper.getAppliedStereotypes(element) in
stereotypes->any(s | s.general->collect(g | g.qualifiedName)
->includes('SysML::Requirements::AbstractRequirement'))
```

- **getScalarValueType** (in t : DataType) : DataType [1]  
Maps a given SysMLv1 primitive type to a SysMLv2 scalar value type.

```
if t.ocIsUndefined()
  or t.name = ''
  or t.name.ocIsUndefined()
  or t.qualifiedName.ocIsUndefined() then
  OclUndefined
else if (t.qualifiedName = 'SysML::Libraries::PrimitiveValueTypes::UnlimitedNatural')
  or t.qualifiedName.indexOf('PrimitiveTypes::UnlimitedNatural') > 0 then
  SysML2::DataType.allInstances()
  ->any(e | e.qualifiedName = 'ScalarValues::Natural')
else
  SysML2::DataType.allInstances()
  ->any(e | e.qualifiedName = 'ScalarValues::' + t.name)
endif endif
```

- **getScalarValueTypeByName** (in ptName : String) : DataType [1]  
Maps a given SysMLv1 primitive type name string to a SysMLv2 scalar value type.

```
SysML2::DataType.allInstances()
->any(e | e.qualifiedName = 'ScalarValues::' + ptName)
```

- **getTagValue** (in element : Element, in stereotypeName : String, in tagValueName : String) [1]  
Returns the value of a stereotype property. The specification is implementation-specific and therefore cannot provided here.
- **getTagValueAsElement** (in element : Element, in stereotypeName : String, in tagValueName : String) : Element [1]  
Returns the value of a stereotype property. The specification is implementation-specific and therefore cannot provided here.
- **getTagValueAsElementColl** (in element : Element, in stereotypeName : String, in tagValueName : String) : Element [0..\*]  
Returns the value of a stereotype property as a collection. The specification is implementation-specific and therefore cannot provided here.
- **getTagValueAsString** (in element : Element, in stereotypeName : String, in tagValueName : String) : String [1]  
Returns the value of a stereotype property as a string. The specification is implementation-specific and therefore cannot provided here.
- **getTagValueAsStringColl** (in element : Element, in stereotypeName : String, in tagValueName : String) : String [0..\*]  
Returns the value of a stereotype property as a string collection. The specification is implementation-specific and therefore cannot provided here.

- `globalNamespace () : Namespace [1]`

```
KerML::Package.allInstances()->any(p | p.owningNamespace->isEmpty())
```

- `hasMainMapping (in element : Element) : Boolean [1]`
- `hasStereotypeApplied (in element : Element, in stereotypeName : String) : Boolean [1]`  
Returns true if the given stereotype is applied to the element. The specification is implementation-specific and therefore cannot be provided here.
- `isConnectionDef (in association : Association) : Boolean [1]`  
Checks if a `UML4SysML::Association` is mapped to a SysML v2 `ConnectionDefinition`.

```
-- Case 1: composite association with
-- multiplicity 1..1 on owner side
let case1: Boolean = association.memberEnd
->exists(e | not e.isComposite and e.lower=1) and
association.memberEnd->exists(e | e.isComposite) in

-- Case 2: association is not composite and
-- there is no owned end with multiplicity 0..*
let case2: Boolean = not association.memberEnd
->exists(e | e.isComposite) and
not association.ownedEnd
->exists(e | e.lower = 0 and e.upper = -1) in

association.oclIsTypeOf(UML::AssociationClass) or
case1 or
case2
```

- `isInScope (in element : Element) : Boolean [1]`  
The `isInScope` operation is intended to define the scope on which the transformation will apply. If the `isInScope` operation returns "true" for a given model element, this element shall be considered by the transformation. Especially, main mappings - if any - will apply to it. It shall be ignored otherwise.
- `isRequirement (in element : Element) : Boolean [1]`  
Checks whether the stereotype `AbstractRequirement` is applied to the given element.

```
let stereotypes: Set(UML::Stereotype) =
  Helper.getAppliedStereotypes(element) in
stereotypes->exists(s | s.general->collect(g | g.qualifiedName)
->includes('SysML::Requirements::AbstractRequirement'))
```

- `packageOwnedRelationship (in src : Element) : Relationship [0..*]`  
Reusable mapping rule for owned relationships of a `UML4SysML::Package` mapping.

```
let pkg: UML::Package = src.oclAsType(UML::Package) in
if pkg.oclIsUndefined() then
  Set{}
else
  let useCaseAssociations : Set(UML::Association) =
    pkg.ownedType->select(e | e.oclIsKindOf(UML::Association))
    ->select(a | a.memberEnd->exists(e | e.type.oclIsKindOf(UML::UseCase))) in
  let unmappedAssociations : Set(UML::Association) = pkg.ownedType
    ->select(e | e.oclIsKindOf(UML::Association))
    ->reject(a | Helper.isConnectionDef(a)) in
  let imports: Set(UML::PackageImport) = pkg.packageImport
```

```

->select(pi | Helper.isInScope(pi.importedPackage)) in
let informationFlows: Set(UML::InformationFlow) = pkg.packagedElement
->select(e | e.ocIsKindOf(UML::InformationFlow))
->reject(i | i.realization->isEmpty() and i.realizingConnector->isEmpty()) in
let fromIF: Set(SysMLv2::ConnectionUsage) = informationFlows
->collect(i | i.realization->collect(r | InformationFlow_Mapping.getMapped(i, r)))
->flatten()->asSet()
->union(informationFlows->collect(i | i.realizingConnector
->collect(r | InformationFlow_Mapping.getMapped(i, r)))
->flatten()->asSet()->asSet() in
let relationships: Set(SysMLv2::Relationship) = pkg.ownedComment
->reject(c | c.annotatedElement->includes(pkg))
->collect(c | CommentOwnership_Mapping.getMapped(c))->asSet()
->union((pkg.ownedType-useCaseAssociations)-unmappedAssociations)
->collect(e | ElementOwningMembership_Mapping.getMapped(e))->asSet()
->union(imports->collect(i | PackageImport_Mapping.getMapped(i))->asSet())
->union(pkg.ownedElement
->select(e | e.ocIsKindOf(UML::Dependency)
or e.ocIsKindOf(UML::Package)
or (e.ocIsKindOf(UML::InstanceSpecification)
and e.ocAsType(UML::InstanceSpecification).classifier->notEmpty()))
->collect(e | ElementOwningMembership_Mapping.getMapped(e))->asSet()
->union(fromIF)->asSet() in
if pkg.URI.ocIsUndefined() or pkg.URI = '' then
relationships
else
relationships->including(PackageURIMetadataMembership_Mapping.getMapped(pkg))
endif endif

```

- **stateOwnedRelationship (in src : Element) : Relationship [0..\*]**  
Reusable mapping rule for owned relationships of a UML4SysML::State mapping.

```

let initialState : Set(UML::Element) =
from.ownedElement->select(e | e.ocIsKindOf(UML::Pseudostate) and
e.ocAsType(UML::Pseudostate).kind = UML::PseudostateKind::initial) in
let toElementOMS : Set(UML::Element) = from.ownedElement - initialState in
toElementOMS->collect(e | ElementOwningMembership_Mapping.getMapped(e))
->union(initialState->collect(e | InitialStateMembership_Mapping.getMapped(e)))

```

## 7.3.2 SysML v1 Library

The SysML v1 library is a SysML v2 model library with metadata definitions for annotating some model elements resulting from a transformation from a SysML v1 model using the SysML v1 to SysML v2 transformation.

```

package SysMLv1Library {

doc /*
* The SysMLv1Library defines library elements and metadata for
* SysML elements which cannot mapped to a SysML v2 element.
*/

// Library elements

action def AddValueAction {
in insertAt : ScalarValues::Natural [0..1];
in value : ScalarValues::Integer;
in isReplaceAll : ScalarValues::Boolean = false;
in target;

if not isReplaceAll {

```

```

        if insertAt == * {
            assign target := SequenceFunctions::including(target, value);
        }
        else {
            assign target :=
                SequenceFunctions::includingAt(target, value, insertAt);
        }
    } else {
        target := value;
    }
}

action def AddStructuralFeatureValueAction :> AddValueAction {
    in object;
}

action def RemoveVariableValueAction :> Actions::AssignmentAction {
    in removeAt: ScalarValues::Integer [0..1];
    in value : ScalarValues::Integer;
    in isRemoveDuplicates : ScalarValues::Boolean = false;
    in variable;

    // isRemoveDuplicates not covered yet
    if isRemoveDuplicates {
        if removeAt {
            assign variable :=
                SequenceFunctions::excludingAt(variable, value, removeAt);
        } else {
            assign variable := SequenceFunctions::excluding(variable, value);
        }
    }
}

// Metadata

metadata def ActivityEdgeData {
    doc /* Metadata definition for UML::ActivityEdge::weight property */
    attribute weight : ScalarValues::Natural;
}

metadata def AssociationData {
    doc /* Metadata definition for
        * UML::StructuredClassifiers::Association::isDerived property mapping
        */
    attribute isDerived : ScalarValues::Boolean;
}

metadata def BlockData {
    doc /* Metadata definition for
        * SysML::Blocks::Block::isEncapsulated property
        */
    attribute isEncapsulated : ScalarValues::Boolean;
}

metadata def ElementGroupData {
    doc /* Metadata definition for the criterion
        * of a SysML::ModelElements::ElementGroup
        */
    attribute criterion : ScalarValues::String;
}

metadata def ModelData :> PackageData {

```

```

    doc /* Metadata definition for the UML::Model::viewpoint property */
    :> annotatedElement : SysML::Package;
    attribute 'viewpoint' : ScalarValues::String;
}

metadata def PackageData {
    doc /* Metadata definition for the UML::Package::URI property */
    :> annotatedElement : SysML::Package;
    attribute URI : ScalarValues::String;
}

metadata def ParameterSetData {
    doc /* Metadata definition for tagging parameters
    * mapped from a UML::ParameterSet
    */
    attribute isParameterSet : ScalarValues::Boolean;
}

metadata def PortData {
    doc /* Metadata definition for tagging SysML v2 ports
    * mapped from a SysML::Ports&Flows::FullPort element
    */
    :> annotatedElement : SysML::PartUsage;
    attribute isFullPort : ScalarValues::Boolean;
}

metadata def ProbabilityData {
    doc /* Metadata definition for SysML::Activities::Probability stereotype */
    attribute probability : ScalarValues::Real;
}

metadata def RateData {
    doc /* Metadata definition for SysML::Activities::Rate and
    * specialized Discrete and Continuous stereotypes
    */
    :> annotatedElement : SysML::PartUsage;
    part rate;
    attribute isDiscrete : ScalarValues::Boolean;
    attribute isConcrete : ScalarValues::Boolean;
}

metadata def RefineData {
    doc /* Metadata definition for tagging SysML v2 dependencies
    * mapped from a SysML::Requirements::Refine relationship
    */
    :> annotatedElement : SysML::Dependency;
    attribute isRefine : ScalarValues::Boolean;
}

metadata def StakeholderData {
    doc /* Metadata definition for tagging SysML v2 item definitions
    * mapped from a SysML::ModelElements::Stakeholder element
    */
    :> annotatedElement : SysML::ItemDefinition;
    attribute isStakeholder : ScalarValues::Boolean;
}

metadata def traceData {
    doc /* Metadata definition for tagging SysML v2 dependencies
    * mapped from a SysML::Requirements::Trace relationship
    */
    :> annotatedElement : SysML::Dependency;
    attribute isTrace : ScalarValues::Boolean;
}

```



```

    }

    metadata def ViewpointData {
      doc /* Metadata definition for SysML::ModelElements::Viewpoint properties */
      attribute languages [0..*] : ScalarValues::String;
      attribute presentations [0..*] : ScalarValues::String;
    }

    package Enumerations {
      enum def ControlValueKind {
        doc /* The ControlValueKind enumeration is a type for
           * treating control values as data and for UML control pins.
           */
        enum disable;
        enum enable;
      }
    }
  }
}

```

## 7.4 Initializers

### 7.4.1 Overview

The classes presented in this subclause provide set of rules that provide default values for all non-derived features of their target metaclasses. Intentionally, initializers do not specify any "source" element. This makes them easier to specialize but prevents them from being able to provide a computation algorithm for some target features. In such a case, the operation matching the feature will be specified as abstract.

### 7.4.2 Mapping Specifications

#### 7.4.2.1 KerML Initializers

##### 7.4.2.1.1 ToAnnotatingElement\_Init

###### Description

Initializes the properties of the SysML v2 element AnnotatingElement.

###### General Classes

- ToElement\_Init (from KerMLInitializers)

###### Association Ends

- to : AnnotatingElement [1]  
{redefines: ToElement\_Init::to}

###### Operations

- annotation () : Annotation [0..\*]

Set {}

##### 7.4.2.1.2 ToAnnotation\_Init

###### Description

Initializes the properties of the SysML v2 element Annotation.

## General Classes

- ToRelationship\_Init (from KerMLInitializers)

## Association Ends

- to : Annotation [1]  
{redefines: ToRelationship\_Init::to}

## Operations

- annotatedElement () : Element [1] {redefines target, abstract}
- annotatingElement () : AnnotatingElement [1] {redefines source, abstract}
- owningAnnotatedElement () : Element [0..1]

null

### 7.4.2.1.3 ToAssociation\_Init

#### Description

Initializes the properties of the SysML v2 element Association.

## General Classes

- ToClassifier\_Init (from KerMLInitializers)
- ToRelationship\_Init (from KerMLInitializers)

## Association Ends

- to : Association [1]  
{redefines: ToRelationship\_Init::to}  
{redefines: ToClassifier\_Init::to}

### 7.4.2.1.4 ToBehavior\_Init

#### Description

Initializes the properties of the SysML v2 element Behavior.

## General Classes

- ToClassifier\_Init (from KerMLInitializers)

## Association Ends

- to : Behavior [1]  
{redefines: ToClassifier\_Init::to}

### 7.4.2.1.5 ToClassifier\_Init

#### Description

Initializes the properties of the SysML v2 element Classifier.

## General Classes

- ToType\_Init (from KerMLInitializers)

#### Association Ends

- to : Classifier [1]  
{redefines: ToType\_Init::to}

#### 7.4.2.1.6 ToComment\_Init

##### Description

Initializes the properties of the SysML v2 element Comment.

##### General Classes

- ToAnnotatingElement\_Init (from KerMLInitializers)

#### Association Ends

- to : Comment [1]  
{redefines: ToAnnotatingElement\_Init::to}

##### Operations

- body () : String [1] {abstract}
- locale () : String [1]

null

#### 7.4.2.1.7 ToConjugation\_Init

##### Description

Initializes the properties of the SysML v2 element Conjugation.

##### General Classes

- ToRelationship\_Init (from KerMLInitializers)

#### Association Ends

- to : Conjugation [1]  
{redefines: ToRelationship\_Init::to}

##### Operations

- conjugatedType () : Type [1] {redefines source, abstract}
- originalType () : Type [1] {redefines target, abstract}

#### 7.4.2.1.8 ToConnector\_Init

##### Description

Initializes the properties of the SysML v2 element Connector.

##### General Classes

- ToFeature\_Init (from KerMLInitializers)
- ToRelationship\_Init (from KerMLInitializers)

#### Association Ends

- to : Connector [1]  
 {redefines: ToFeature\_Init::to}  
 {redefines: ToRelationship\_Init::to}

#### Operations

- isDirected () : Boolean [1]

false

#### 7.4.2.1.9 ToDocumentation\_Init

##### Description

Initializes the properties of the SysML v2 element Documentation.

##### General Classes

- ToComment\_Init (from KerMLInitializers)

#### Association Ends

- to : Documentation [1]  
 {redefines: ToComment\_Init::to}

#### 7.4.2.1.10 ToElement\_Init

##### Description

This is the general abstract class to be used as an ancestor for any class mapping specification.

##### General Classes

- Initializer (from Foundations)

#### Association Ends

- to : Element [1]  
 {redefines: Initializer::to}

#### Operations

- aliasId () : String [0..\*]

Set {}

- declaredName () : String [0..1]

null

- `elementId () : String [1]`

```
Helper.createUUID()
```

- `ownedRelationship () : Relationship [0..*]`

```
Set{}
```

- `shortName () : String [0..1]`

```
null
```

### Constraints

- `from_and_to_types`  
`from.ocIsKindOf(factory.srcType) and to.ocIsKindOf(factory.tgtType)`

#### 7.4.2.1.11 ToEndFeatureMembership\_Init

##### Description

Initializes the properties of the SysML v2 element EndFeatureMembership.

##### General Classes

- ToFeatureMembership\_Init (from KerMLInitializers)

##### Association Ends

- `to : EndFeatureMembership [1]`  
`{redefines: ToFeatureMembership_Init::to}`

#### 7.4.2.1.12 ToExpression\_Init

##### Description

Initializes the properties of the SysML v2 element Expression.

##### General Classes

- ToStep\_Init (from KerMLInitializers)

##### Association Ends

- `to : Expression [1]`  
`{redefines: ToStep_Init::to}`

#### 7.4.2.1.13 ToFeature\_Init

##### Description

Initializes the properties of the SysML v2 element Feature.

##### General Classes

- ToType\_Init (from KerMLInitializers)

#### Association Ends

- to : Feature [1]  
  {redefines: ToType\_Init::to}

#### Operations

- direction () : FeatureDirectionKind [0..1]

null

- isComposite () : Boolean [1]

false

- isDerived () : Boolean [1]

false

- isEnd () : Boolean [1]

false

- isOrdered () : Boolean [1]

false

- isPortion () : Boolean [1]

false

- isReadOnly () : Boolean [1]

false

- isUnique () : Boolean [1]

true

#### 7.4.2.1.14 ToFeatureChainExpression\_Init

##### Description

Initializes the properties of the SysML v2 element FeatureChainExpression.

##### General Classes

- ToOperatorExpression\_Init (from KerMLInitializers)

## Association Ends

- to : FeatureChainExpression [1]  
{redefines: ToOperatorExpression\_Init::to}

### 7.4.2.1.15 ToFeatureChaining\_Init

#### Description

Initializes the properties of the SysML v2 element FeatureChaining.

#### General Classes

- ToRelationship\_Init (from KerMLInitializers)

## Association Ends

- to : FeatureChaining [1]  
{redefines: ToRelationship\_Init::to}

#### Operations

- chainingFeature () : Feature [1] {redefines target, abstract}

### 7.4.2.1.16 ToFeatureMembership\_Init

#### Description

Initializes the properties of the SysML v2 element FeatureMembership.

#### General Classes

- ToOwningMembership\_Init (from KerMLInitializers)
- ToTypeFeaturing\_Init (from KerMLInitializers)

## Association Ends

- to : FeatureMembership [1]  
{redefines: ToTypeFeaturing\_Init::to}  
{redefines: ToOwningMembership\_Init::to}

#### Operations

- ownedMemberFeature () : Feature [1] {redefines ownedMemberElement}  
  
self.upperBound
- ownedRelatedElement () : Element [0..\*] {redefines ownedRelatedElement}  
  
Set {self.ownedMemberFeature () }

### 7.4.2.1.17 ToFeatureReferenceExpression\_Init

#### Description

Initializes the properties of the SysML v2 element FeatureReferenceExpression.

#### General Classes

- ToExpression\_Init (from KerMLInitializers)

#### Association Ends

- to : FeatureReferenceExpression [1]  
{redefines: ToExpression\_Init::to}

#### 7.4.2.1.18 ToFeatureTyping\_Init

##### Description

Initializes the properties of the SysML v2 element FeatureTyping.

#### General Classes

- ToSpecialization\_Init (from KerMLInitializers)

#### Association Ends

- to : FeatureTyping [1]  
{redefines: ToSpecialization\_Init::to}

#### Operations

- type () : Type [1] {redefines general, abstract}
- typedFeature () : Feature [1] {redefines specific, abstract}

#### 7.4.2.1.19 ToFeatureValue\_Init

##### Description

Initializes the properties of the SysML v2 element FeatureValue.

#### General Classes

- ToOwningMembership\_Init (from KerMLInitializers)

#### Association Ends

- to : FeatureValue [1]  
{redefines: ToOwningMembership\_Init::to}

#### Operations

- featureWithValue () : Feature [1] {redefines ownedMemberElement, abstract}
- isDefault () : Boolean [1]

false

- isInitial () : Boolean [1]



`false`

- `ownedRelatedElement () : Element [0..*]` {redefines `ownedRelatedElement`}

`Set {self.value () }`

- `value () : Expression [1]` {redefines `ownedMemberElement`, abstract}

#### **7.4.2.1.20 ToFlow\_Init**

##### **Description**

Initializes the properties of the SysML v2 element Flow.

##### **General Classes**

- `ToConnector_Init` (from `KerMLInitializers`)

##### **Association Ends**

- `to : Flow [1]`  
{redefines: `ToConnector_Init::to`}

#### **7.4.2.1.21 ToFunction\_Init**

##### **Description**

Initializes the properties of the SysML v2 element Function.

##### **General Classes**

- `ToBehavior_Init` (from `KerMLInitializers`)

##### **Association Ends**

- `to : Function [1]`  
{redefines: `ToBehavior_Init::to`}

#### **7.4.2.1.22 ToImport\_Init**

##### **Description**

Initializes the properties of the SysML v2 element Import.

##### **General Classes**

- `ToRelationship_Init` (from `KerMLInitializers`)

##### **Association Ends**

- `to : Import [1]`  
{redefines: `ToRelationship_Init::to`}

## Operations

- importedMemberName () : String [0..1]

null

- isImportAll () : Boolean [1]

false

- isRecursive () : Boolean [1]

false

- visibility () : VisibilityKind [1]

KerML::VisibilityKind::public

### 7.4.2.1.23 ToInteraction\_Init

#### Description

Initializes the properties of the SysML v2 element Interaction.

#### General Classes

- ToAssociation\_Init (from KerMLInitializers)
- ToBehavior\_Init (from KerMLInitializers)

#### Association Ends

- to : Interaction [1]  
{redefines: ToAssociation\_Init::to}  
{redefines: ToBehavior\_Init::to}

### 7.4.2.1.24 ToInvocationExpression\_Init

#### Description

Initializes the properties of the SysML v2 element InvocationExpression.

#### General Classes

- ToExpression\_Init (from KerMLInitializers)

#### Association Ends

- to : InvocationExpression [1]  
{redefines: ToExpression\_Init::to}

### 7.4.2.1.25 ToMembership\_Init

#### Description

Initializes the properties of the SysML v2 element Membership.

#### General Classes

- ToRelationship\_Init (from KerMLInitializers)

#### Association Ends

- to : Membership [1]  
{redefines: ToRelationship\_Init::to}

#### Operations

- memberElement () : Element [1] {redefines target, abstract}
- memberName () : String [0..1]

null

- memberShortName () : String [0..1]

null

- membershipOwningNamespace () : Element [0..\*] {redefines source, abstract}
- visibility () : VisibilityKind [1]

KerML::VisibilityKind::public

#### 7.4.2.1.26 ToMembershipImport\_Init

##### Description

Initializes the properties of the SysML v2 element MembershipImport.

#### General Classes

- ToImport\_Init (from KerMLInitializers)

#### Association Ends

- to : MembershipImport [1]  
{redefines: ToImport\_Init::to}

#### Operations

- importedMembership () : Namespace [1] {redefines target, abstract}

#### 7.4.2.1.27 ToNamespace\_Init

##### Description

Initializes the properties of the SysML v2 element Namespace.

#### General Classes

- ToElement\_Init (from KerMLInitializers)

#### **Association Ends**

- to : Namespace [1]  
{redefines: ToElement\_Init::to}

#### **7.4.2.1.28 ToNamespaceImport\_Init**

##### **Description**

Initializes the properties of the SysML v2 element NamespaceImport.

##### **General Classes**

- ToImport\_Init (from KerMLInitializers)

#### **Association Ends**

- to : NamespaceImport [1]  
{redefines: ToImport\_Init::to}

##### **Operations**

- importedNamespace () : Namespace [1] {redefines target, abstract}

#### **7.4.2.1.29 ToOperatorExpression\_Init**

##### **Description**

Initializes the properties of the SysML v2 element OperatorExpression.

##### **General Classes**

- ToExpression\_Init (from KerMLInitializers)

#### **Association Ends**

- to : OperatorExpression [1]  
{redefines: ToExpression\_Init::to}

##### **Operations**

- operator () : String [1]{abstract}

#### **7.4.2.1.30 ToOwningMembership\_Init**

##### **Description**

Initializes the properties of the SysML v2 element OwningMembership.

##### **General Classes**

- ToMembership\_Init (from KerMLInitializers)

#### **Association Ends**

- to : OwingMembership [1]  
{redefines: ToMembership\_Init::to}

### Operations

- ownedMemberElement () : Element [1] {redefines memberElement, abstract}
- ownedRelatedElement () : Element [0..\*] {redefines ownedRelatedElement}

```
Set{self.ownedMemberElement() }
```

#### 7.4.2.1.31 ToPackage\_Init

##### Description

Initializes the properties of the SysML v2 element Package.

##### General Classes

- ToNamespace\_Init (from KerMLInitializers)

##### Association Ends

- to : Package [1]  
{redefines: ToNamespace\_Init::to}

#### 7.4.2.1.32 ToParameterMembership\_Init

##### Description

Initializes the properties of the SysML v2 element ParameterMembership.

##### General Classes

- ToFeatureMembership\_Init (from KerMLInitializers)

##### Association Ends

- to : ParameterMembership [1]  
{redefines: ToFeatureMembership\_Init::to}  
{redefines: ElementOwningMembership\_Mapping::to}

### Operations

- ownedMemberParameter () : Feature [1] {redefines ownedMemberFeature}

```
null
```

- ownedRelatedElement () : Element [0..\*] {redefines ownedRelatedElement}

```
Set{self.ownedMemberParameter() }
```

#### 7.4.2.1.33 ToPredicate\_Init

##### Description

Initializes the properties of the SysML v2 element Predicate.

#### General Classes

- ToFunction\_Init (from KerMLInitializers)

#### Association Ends

- to : Predicate [1]  
{redefines: ToFunction\_Init::to}

#### 7.4.2.1.34 ToRedefinition\_Init

##### Description

Initializes the properties of the SysML v2 element Redefinition.

#### General Classes

- ToSubsetting\_Init (from KerMLInitializers)

#### Association Ends

- to : Redefinition [1]  
{redefines: ToSubsetting\_Init::to}

#### Operations

- redefinedFeature () : Feature [1] {redefines subsettedFeature, abstract}
- redefiningFeature () : Feature [1]{abstract}

#### 7.4.2.1.35 ToReferenceSubsetting\_Init

##### Description

Initializes the properties of the SysML v2 element ReferenceSubsetting.

#### General Classes

- ToSubsetting\_Init (from KerMLInitializers)

#### Association Ends

- to : ReferenceSubsetting [1]  
{redefines: ToSubsetting\_Init::to}

#### Operations

- referencedFeature () : Feature [1] {redefines subsettedFeature, abstract}

#### 7.4.2.1.36 ToRelationship\_Init

##### Description

Initializes the properties of the SysML v2 element Relationship.

#### General Classes

- ToElement\_Init (from KerMLInitializers)

#### Association Ends

- to : Relationship [1]  
{redefines: ToElement\_Init::to}

#### Operations

- ownedRelatedElement () : Element [0..\*]

Set {}

- source () : Element [0..\*]

Set {}

- target () : Element [0..\*]

Set {}

#### 7.4.2.1.37 ToReturnParameterMembership\_Init

##### Description

Initializes the properties of the SysML v2 element ReturnParameterMembership.

##### General Classes

- ToParameterMembership\_Init (from KerMLInitializers)

#### Association Ends

- to : ReturnParameterMembership [1]  
{redefines: ToParameterMembership\_Init::to}

#### Operations

- isComposite (in src : Element) : Boolean [1]  
returns "true" if the element provided as the actual parameter value can have a mapping to an instance of the type specified by the "to" attribute (i.e. can be used as a value for the "from" attribute)

false

#### 7.4.2.1.38 ToSpecialization\_Init

##### Description

Initializes the properties of the SysML v2 element Specialization.

##### General Classes

- ToRelationship\_Init (from KerMLInitializers)

### Association Ends

- to : Specialization [1]  
{redefines: ToRelationship\_Init::to}

### Operations

- general () : Type [1] {redefines target, abstract}
- specific () : Type [1] {redefines source, abstract}

#### 7.4.2.1.39 ToStep\_Init

### Description

Initializes the properties of the SysML v2 element Step.

### General Classes

- ToFeature\_Init (from KerMLInitializers)

### Association Ends

- to : Step [1]  
{redefines: ToFeature\_Init::to}

#### 7.4.2.1.40 ToSubclassification\_Init

### Description

Initializes the properties of the SysML v2 element Subclassification.

### General Classes

- ToSpecialization\_Init (from KerMLInitializers)

### Association Ends

- to : Subclassification [1]  
{redefines: ToSpecialization\_Init::to}

### Operations

- subclassifier () : Classifier [1]  
  
null
- superclassifier () : Classifier [1]  
  
null

#### 7.4.2.1.41 ToSubsetting\_Init

### Description

Initializes the properties of the SysML v2 element Subsetting.



## General Classes

- ToSpecialization\_Init (from KerMLInitializers)

## Association Ends

- to : Subsetting [1]  
{redefines: ToSpecialization\_Init::to}

## Operations

- ownedRelatedElement () : Element [0..\*] {redefines ownedRelatedElement}

Set {}

- subsettingFeature () : Feature [1] {redefines general, abstract}

### 7.4.2.1.42 ToSuccession\_Init

#### Description

Initializes the properties of the SysML v2 element Succession.

## General Classes

- ToConnector\_Init (from KerMLInitializers)

## Association Ends

- to : Succession [1]  
{redefines: ToConnector\_Init::to}

### 7.4.2.1.43 ToSuccessionItemFlow\_Init

#### Description

Initializes the properties of the SysML v2 element SuccessionFlow.

## General Classes

- ToItemFlow\_Init (from KerMLInitializers)
- ToSuccession\_Init (from KerMLInitializers)

## Association Ends

- to : SuccessionFlow [1]  
{redefines: ToSuccession\_Init::to}  
{redefines: ToItemFlow\_Init::to}

### 7.4.2.1.44 ToTextualRepresentation\_Init

#### Description

Initializes the properties of the SysML v2 element TextualRepresentation.

## General Classes

- ToAnnotatingElement\_Init (from KerMLInitializers)

#### Association Ends

- to : TextualRepresentation [1]  
{redefines: ToAnnotatingElement\_Init::to}

#### Operations

- body () : String [1]{abstract}
- language () : String [1]{abstract}

#### 7.4.2.1.45 ToType\_Init

##### Description

Initializes the properties of the SysML v2 element Type.

##### General Classes

- ToNamespace\_Init (from KerMLInitializers)

#### Association Ends

- to : Type [1]  
{redefines: ToNamespace\_Init::to}

#### Operations

- isAbstract () : Boolean [1]

false

- isSufficient () : Boolean [1]

false

#### 7.4.2.1.46 ToTypeFeaturing\_Init

##### Description

Initializes the properties of the SysML v2 element TypeFeaturing.

##### General Classes

- ToRelationship\_Init (from KerMLInitializers)

#### Association Ends

- to : TypeFeaturing [1]  
{redefines: ToRelationship\_Init::to}

#### Operations

- featureOfType () : Feature [1] {redefines source, abstract}

- `featuringType () : Type [1] {redefines target, abstract}`

### 7.4.2.2 System Initializers

#### 7.4.2.2.1 ToActionUsage\_Init

##### Description

Initializes the properties of the SysML v2 element ActionUsage.

##### General Classes

- `ToStep_Init` (from `KerMLInitializers`)
- `ToUsage_Init` (from `SystemInitializers`)

##### Association Ends

- `to : ActionUsage [1]`  
`{redefines: ToStep_Init::to}`  
`{redefines: ToUsage_Init::to}`

##### Operations

- `isComposite () : Boolean [1] {redefines isComposite}`

`true`

#### 7.4.2.2.2 ToActorMembership\_Init

##### Description

Initializes the properties of the SysML v2 element ActorMembership.

##### General Classes

- `ToParameterMembership_Init` (from `KerMLInitializers`)

##### Association Ends

- `to : ActorMembership [1]`  
`{redefines: ToParameterMembership_Init::to}`

#### 7.4.2.2.3 ToAssignmentActionUsage\_Init

##### Description

Initializes the properties of the SysML v2 element AssignmentActionUsage.

##### General Classes

- `ToActionUsage_Init` (from `SystemInitializers`)

##### Association Ends

- `to : AssignmentActionUsage [1]`  
`{redefines: ToActionUsage_Init::to}`

#### 7.4.2.2.4 ToBindingConnectorAsUsage\_Init

##### Description

Initializes the properties of the SysML v2 element BindingConnectorAsUsage.

##### General Classes

- ToConnectionUsage\_Init (from SystemInitializers)

##### Association Ends

- to : BindingConnectorAsUsage [1]  
{redefines: ToConnectionUsage\_Init::to}

#### 7.4.2.2.5 ToCalculationUsage\_Init

##### Description

Initializes the properties of the SysML v2 element CalculationUsage.

##### General Classes

- ToActionUsage\_Init (from SystemInitializers)

##### Association Ends

- to : CalculationUsage [1]  
{redefines: ToActionUsage\_Init::to}

#### 7.4.2.2.6 ToConjugatedPortDefinition\_Init

##### Description

Initializes the properties of the SysML v2 element ConjugatedPortDefinition.

##### General Classes

- ToPortDefinition\_Init (from SystemInitializers)

##### Association Ends

- to : ConjugatedPortDefinition [1]  
{redefines: ToPortDefinition\_Init::to}

#### 7.4.2.2.7 ToConjugatedPortTyping\_Init

##### Description

Initializes the properties of the SysML v2 element ConjugatedPortTyping.

##### General Classes

- ToFeatureTyping\_Init (from KerMLInitializers)

##### Association Ends

- to : ConjugatedPortTyping [1]  
{redefines: ToFeatureTyping\_Init::to}

#### Operations

- conjugatedPortDefinition () : ConjugatedPortDefinition [1] {redefines type, abstract}
- portDefinition () : PortDefinition [1] {abstract}

#### 7.4.2.2.8 ToConnectionUsage\_Init

##### Description

Initializes the properties of the SysML v2 element ConnectionUsage.

##### General Classes

- ToPartUsage\_Init (from SystemInitializers)

##### Association Ends

- to : ConnectionUsage [1]  
{redefines: ToPartUsage\_Init::to}

#### 7.4.2.2.9 ToConstraintDefinition\_Init

##### Description

Initializes the properties of the SysML v2 element ConstraintDefinition.

##### General Classes

- ToDefinition\_Init (from SystemInitializers)

##### Association Ends

- to : ConstraintDefinition [1]  
{redefines: ToDefinition\_Init::to}  
{redefines: ToFunction\_Init::to}

#### 7.4.2.2.10 ToConstraintUsage\_Init

##### Description

Initializes the properties of the SysML v2 element ConstraintUsage.

##### General Classes

- ToUsage\_Init (from SystemInitializers)

##### Association Ends

- to : ConstraintUsage [1]  
{redefines: ToUsage\_Init::to}

#### 7.4.2.2.11 ToDefinition\_Init

##### Description

Initializes the properties of the SysML v2 element Definition.

#### General Classes

- ToClassifier\_Init (from KerMLInitializers)

#### Association Ends

- to : Definition [1]  
{redefines: ToClassifier\_Init::to}

#### Operations

- isVariation () : Boolean [1]

false

### 7.4.2.2.12 ToEventOccurrenceUsage\_Init

#### Description

Initializes the properties of the SysML v2 element EventOccurrenceUsage.

#### General Classes

- ToOccurrenceUsage\_Init (from SystemInitializers)

#### Association Ends

- to : EventOccurrenceUsage [1]  
{redefines: ToOccurrenceUsage\_Init::to}

### 7.4.2.2.13 ToFlowUsage\_Init

#### Description

Initializes the properties of the SysML v2 element FlowUsage.

#### General Classes

- ToActionUsage\_Init (from SystemInitializers)
- ToConnector\_Init (from KerMLInitializers)

#### Association Ends

- to : FlowUsage [1]  
{redefines: ToConnector\_Init::to}  
{redefines: ToActionUsage\_Init::to}

#### Operations

- isDirected () : Boolean [1] {redefines isDirected}

true

#### **7.4.2.2.14 ToltemDefinition\_Init**

##### **Description**

Initializes the properties of the SysML v2 element ItemDefinition.

##### **General Classes**

- ToDefinition\_Init (from SystemInitializers)

##### **Association Ends**

- to : ItemDefinition [1]  
{redefines: ToDefinition\_Init::to}

#### **7.4.2.2.15 ToltemFeature\_Init**

##### **Description**

Initializes the properties of the SysML v2 element ItemFeature.

##### **General Classes**

- ToFeature\_Init (from KerMLInitializers)

##### **Association Ends**

- to : PayloadFeature [1]  
{redefines: ToFeature\_Init::to}

#### **7.4.2.2.16 ToltemUsage\_Init**

##### **Description**

Generic mapping class for mappings to the SysML v2 element ItemUsage.

##### **General Classes**

- ToOccurrenceUsage\_Init (from SystemInitializers)

##### **Association Ends**

- to : ItemUsage [1]  
{redefines: ToOccurrenceUsage\_Init::to}

#### **7.4.2.2.17 ToMetadataUsage\_Init**

##### **Description**

Initializes the properties of the SysML v2 element MetadataUsage.

##### **General Classes**

- ToUsage\_Init (from SystemInitializers)

#### Association Ends

- to : MetadataUsage [1]  
{redefines: ToUsage\_Init::to}

#### 7.4.2.2.18 ToObjectiveMembership\_Init

##### Description

Initializes the properties of the SysML v2 element ObjectiveMembership.

##### General Classes

- ToFeatureMembership\_Init (from KerMLInitializers)

#### Association Ends

- to : ObjectiveMembership [1]  
{redefines: ToFeatureMembership\_Init::to}

#### 7.4.2.2.19 ToOccurrenceDefinition\_Init

##### Description

Initializes the properties of the SysML v2 element OccurrenceDefinition.

##### General Classes

- ToDefinition\_Init (from SystemInitializers)

#### Association Ends

- to : OccurrenceDefinition [1]  
{redefines: ToDefinition\_Init::to}

##### Operations

- isIndividual () : Boolean [1]

false

#### 7.4.2.2.20 ToOccurrenceUsage\_Init

##### Description

Initializes the properties of the SysML v2 element OccurrenceUsage.

##### General Classes

- ToUsage\_Init (from SystemInitializers)

#### Association Ends

- to : OccurrenceUsage [1]  
{redefines: ToUsage\_Init::to}



## Operations

- `isIndividual () : Boolean [1]`

`false`

- `portionKind () : PortionKind [1]`

`invalid`

### 7.4.2.2.21 ToPartUsage\_Init

#### Description

Initializes the properties of the SysML v2 element PartUsage.

#### General Classes

- `ToUsage_Init` (from `SystemInitializers`)

#### Association Ends

- `to : PartUsage [1]`  
{redefines: `ToUsage_Init::to`}

### 7.4.2.2.22 ToPerformActionUsage\_Init

#### Description

Initializes the properties of the SysML v2 element PerformActionUsage.

#### General Classes

- `ToActionUsage_Init` (from `SystemInitializers`)

#### Association Ends

- `to : PerformActionUsage [1]`  
{redefines: `ToActionUsage_Init::to`}

### 7.4.2.2.23 ToPortConjugation\_Init

#### Description

Initializes the properties of the SysML v2 element PortConjugation.

#### General Classes

- `ToConjugation_Init` (from `KerMLInitializers`)

#### Association Ends

- `to : PortConjugation [1]`  
{redefines: `ToConjugation_Init::to`}

## Operations

- originalPortDefinition () : PortDefinition [1] {redefines originalType, abstract}

### 7.4.2.2.24 ToPortDefinition\_Init

#### Description

Initializes the properties of the SysML v2 element PortDefinition.

#### General Classes

- ToDefinition\_Init (from SystemInitializers)

#### Association Ends

- to : PortDefinition [1]  
{redefines: ToDefinition\_Init::to}

### 7.4.2.2.25 ToReferenceUsage\_Init

#### Description

Provides the basic features to map to a ReferenceUsage element.

#### General Classes

- ToUsage\_Init (from SystemInitializers)

#### Association Ends

- to : ReferenceUsage [1]  
{redefines: ToUsage\_Init::to}

### 7.4.2.2.26 ToRequirementUsage\_Init

#### Description

Initializes the properties of the SysML v2 element RequirementUsage.

#### General Classes

- ToUsage\_Init (from SystemInitializers)

#### Association Ends

- to : RequirementUsage [1]  
{redefines: ToUsage\_Init::to}

### 7.4.2.2.27 ToStateSubactionMembership\_Init

#### Description

Initializes the properties of the SysML v2 element StateSubactionMembership.

#### General Classes

- ToFeatureMembership\_Init (from KerMLInitializers)

#### **Association Ends**

- to : StateSubactionMembership [1]  
{redefines: ToFeatureMembership\_Init::to}

#### **7.4.2.2.28 ToStateUsage\_Init**

##### **Description**

Initializes the properties of the SysML v2 element StateUsage.

##### **General Classes**

- ToActionUsage\_Init (from SystemInitializers)

#### **Association Ends**

- to : StateUsage [1]  
{redefines: ToActionUsage\_Init::to}

#### **7.4.2.2.29 ToSubjectMembership\_Init**

##### **Description**

Initializes the properties of the SysML v2 element SubjectMembership.

##### **General Classes**

- ToParameterMembership\_Init (from KerMLInitializers)

#### **Association Ends**

- to : SubjectMembership [1]  
{redefines: ToParameterMembership\_Init::to}

#### **7.4.2.2.30 ToTransitionUsage\_Init**

##### **Description**

Initializes the properties of the SysML v2 element TransitionUsage.

##### **General Classes**

- ToActionUsage\_Init (from SystemInitializers)

#### **Association Ends**

- to : TransitionUsage [1]  
{redefines: ToActionUsage\_Init::to}

#### **7.4.2.2.31 ToTriggerInvocationExpression\_Init**

##### **Description**

Initializes the properties of the SysML v2 element TriggerInvocationExpression.

## General Classes

- ToInvocationExpression\_Init (from KerMLInitializers)

## Association Ends

- to : TriggerInvocationExpression [1]  
{redefines: ToInvocationExpression\_Init::to}

## Operations

- kind () : TriggerKind [0..1] {redefines direction, abstract}

### 7.4.2.2.32 ToUsage\_Init

#### Description

Initializes the properties of the SysML v2 element Usage.

## General Classes

- ToFeature\_Init (from KerMLInitializers)

## Association Ends

- to : Usage [1]  
{redefines: ToFeature\_Init::to}

## Operations

- isVariation () : Boolean [1]

false

## 7.5 Factories

### 7.5.1 Overview

The classes presented in this subclause specify facilities for creating elements in the target model form an arbitrary set of zero to many input parameters. After the target element is created, no link between it and an the value of inputs parameter (if any) will be preserved.

### 7.5.2 Mapping Specifications

#### 7.5.2.1 LiteralString\_Factory

#### Description

Factory class to create a LiteralString element.

## General Classes

- Factory (from Foundations)
- ToExpression\_Init (from KerMLInitializers)

## Association Ends

- string : String [1]
- to : LiteralString [1]  
    {redefines: ToExpression\_Init::to}

### Operations

- create (in string : String) : LiteralString [1]
- ownedRelationship () : Relationship [0..\*] {redefines ownedRelationship}

```
Set{ReturnParameterFeatureMembership_Factory.create() }
```

## 7.5.2.2 StringParameterFeature\_Factory

### Description

Factory class to create a feature element representing a string.

### General Classes

- Factory (from Foundations)
- ToFeature\_Init (from KerMLInitializers)

### Association Ends

- string : String [1]

### Operations

- create (in string : String) : Feature [1]
- ownedRelationship () : Relationship [0..\*] {redefines ownedRelationship}

```
Set{StringParameterFeatureValue_Factory.create(string) }
```

## 7.5.2.3 StringParameterFeatureValue\_Factory

### Description

Factory class to create a string feature value relationship for a feature element.

### General Classes

- Factory (from Foundations)
- ToFeatureValue\_Init (from KerMLInitializers)

### Association Ends

- string : String [1]

### Operations

- create (in string : String) : FeatureValue [1]
- value () : Expression [1] {redefines value}

```
LiteralString_Factory.create(string)
```

#### 7.5.2.4 StringParameterMembership\_Factory

##### Description

Factory class to create a parameter membership relationship for a feature element representing a string.

##### General Classes

- Factory (from Foundations)
- ToParameterMembership\_Init (from KerMLInitializers)

##### Association Ends

- string : String [1]

##### Operations

- create (in string : String) : ParameterMembership [1]
- ownedMemberParameter () : Feature [1] {redefines ownedMemberParameter}

```
StringParameterFeature_Factory.create(string)
```

#### 7.5.2.5 SubjectMembership\_Factory

##### Description

Factory class to create a subject membership relationship for a given subject.

##### General Classes

- Factory (from Foundations)
- ToSubjectMembership\_Init (from SystemInitializers)

##### Association Ends

- subject : Type [1]

##### Operations

- create (in subject : Type) : SubjectMembership [1]
- ownedMemberParameter () : Feature [1] {redefines ownedMemberParameter}

```
subject
```

#### 7.5.2.6 AssignmentActionUsage\_Factory

##### Description

Factory to create an assignment action usage.

##### General Classes

- Factory (from Foundations)
- ToAssignmentActionUsage\_Init (from SystemInitializers)

## Operations

- `create () : AssignmentActionUsage [1]`
- `ownedRelationship () : Relationship [0..*]` {redefines `ownedRelationship`}

```
Set{AssignmentActionUsageParameterMembership_Factory.create(),  
DirectedReferenceUsageParameterMembership_Factory.create(  
    KerML::FeatureDirectionKind::_'in')}
```

### 7.5.2.7 AssignmentActionUsageFeatureMembership2\_Factory

#### Description

Factory class to create a feature membership relationship for a feature element created by the factory class `AssignmentActionUsageTargetReferenceUsageIn2_Factory`.

#### General Classes

- `Factory` (from `Foundations`)
- `ToFeatureMembership_Init` (from `KerMLInitializers`)

## Operations

- `create () : FeatureMembership [1]`
- `ownedMemberFeature () : Feature [1]` {redefines `ownedMemberFeature`}

```
AssignmentActionUsageTargetReferenceUsageIn2_Factory.create()
```

### 7.5.2.8 AssignmentActionUsageFeatureMembership3\_Factory

#### Description

Factory class to create a feature membership relationship for a feature element created by the factory class `AssignmentActionUsageTargetReferenceUsageIn3_Factory`.

#### General Classes

- `Factory` (from `Foundations`)
- `ToFeatureMembership_Init` (from `KerMLInitializers`)

## Operations

- `create () : FeatureMembership [1]`
- `ownedMemberFeature () : Feature [1]` {redefines `ownedMemberFeature`}

```
AssignmentActionUsageTargetReferenceUsageIn3_Factory.create()
```

### 7.5.2.9 AssignmentActionUsageOwningMembership\_Factory

#### Description

Factory class to create an owning membership relationship for an element created by the factory class `AssignmentActionUsage_Factory`.

## General Classes

- Factory (from Foundations)
- ToOwningMembership\_Init (from KerMLInitializers)

## Operations

- create () : OwningMembership [1]
- ownedMemberElement () : Element [1] {redefines ownedMemberElement}

```
AssignmentActionUsage_Factory.create()
```

### 7.5.2.10 AssignmentActionUsageParameterMembership\_Factory

#### Description

Factory class to create a parameter membership relationship for a feature element created by the factory class AssignmentActionUsageReferenceUsageIn1\_Factory.

## General Classes

- Factory (from Foundations)
- ToParameterMembership\_Init (from KerMLInitializers)

## Operations

- create () : ParameterMembership [1]
- ownedMemberParameter () : Feature [1] {redefines ownedMemberParameter}

```
AssignmentActionUsageReferenceUsageIn1_Factory.create()
```

### 7.5.2.11 AssignmentActionUsageReferenceUsageIn1\_Factory

#### Description

Factory class creating a reference usage element with direction "in" as parameter of an assignment action usage.

## General Classes

- Factory (from Foundations)
- ToReferenceUsage\_Init (from SystemInitializers)

## Operations

- create () : ReferenceUsage [1]
- direction () : FeatureDirectionKind [0..1] {redefines direction}

```
KerML::FeatureDirectionKind::_in'
```

- ownedRelationship () : Relationship [0..\*] {redefines ownedRelationship}

```
Set{AssignmentActionUsageFeatureMembership2_Factory.create() }
```



### 7.5.2.12 AssignmentActionUsageTargetReferenceUsageIn2\_Factory

#### Description

Factory class creating a reference usage element as an owned feature of the reference usage of an assignment action usage.

#### General Classes

- Factory (from Foundations)
- ToReferenceUsage\_Init (from SystemInitializers)

#### Operations

- create () : ReferenceUsage [1]
- ownedRelationship () : Relationship [0..\*] {redefines ownedRelationship}

```
Set{AssignmentActionUsageFeatureMembership3_Factory.create() }
```

### 7.5.2.13 AssignmentActionUsageTargetReferenceUsageIn3\_Factory

#### Description

Factory class creating a reference usage element as an owned feature of the reference usage of an assignment action usage.

#### General Classes

- Factory (from Foundations)
- ToReferenceUsage\_Init (from SystemInitializers)

#### Operations

- create () : ReferenceUsage [1]

### 7.5.2.14 DirectedReferenceUsage\_Factory

#### Description

Factory class creating a reference usage element with a given direction and without owned relationships.

#### General Classes

- Factory (from Foundations)
- ToReferenceUsage\_Init (from SystemInitializers)

#### Association Ends

- featureDirectionKind : FeatureDirectionKind [1]

#### Operations

- create (in featureDirectionKind : FeatureDirectionKind) : ReferenceUsage [1]
- direction () : FeatureDirectionKind [0..1] {redefines direction}

featureDirectionKind

#### 7.5.2.15 DirectedReferenceUsageParameterMembership\_Factory

##### Description

Factory class to create a parameter membership relationship for a feature element created by the factory class DirectedReferenceUsage\_Factory.

##### General Classes

- Factory (from Foundations)
- ToParameterMembership\_Init (from KerMLInitializers)

##### Association Ends

- featureDirectionKind : FeatureDirectionKind [1]

##### Operations

- create (in featureDirectionKind : FeatureDirectionKind) : ParameterMembership [1]
- ownedMemberParameter () : Feature [1] {redefines ownedMemberParameter}

```
DirectedReferenceUsage_Factory.create(featureDirectionKind)
```

#### 7.5.2.16 EmptyObjectiveMembership\_Factory

##### Description

Factory class to create an objective membership without a source in the SysML v1 model.

##### General Classes

- Factory (from Foundations)
- ToObjectiveMembership\_Init (from SystemInitializers)

##### Operations

- create () : ObjectiveMembership [1]
- ownedMemberFeature () : Feature [1] {redefines ownedMemberFeature}

```
EmptyRequirementUsage_Factory.create()
```

#### 7.5.2.17 EmptyRequirementUsage\_Factory

##### Description

Factory class to create a requirement usage without a source in the SysML v1 model.

##### General Classes

- Factory (from Foundations)
- ToRequirementUsage\_Init (from SystemInitializers)

## Operations

- `create () : RequirementUsage [1]`
- `ownedRelationship () : Relationship [0..*]` {redefines `ownedRelationship`}

```
Set {  
  EmptySubjectMembership_Factory.create(),  
  ReturnParameterFeatureMembership_Factory.create() }
```

### 7.5.2.18 EmptySubject\_Factory

#### Description

Factory class to create a reference usage representing a subject without a source in the SysML v1 model.

#### General Classes

- `Factory` (from `Foundations`)
- `ToReferenceUsage_Init` (from `SystemInitializers`)

#### Operations

- `create () : ReferenceUsage [1]`
- `direction () : FeatureDirectionKind [0..1]` {redefines `direction`}

```
KerML::FeatureDirectionKind::_in'
```

### 7.5.2.19 EmptySubjectMembership\_Factory

#### Description

Factory class to create a membership relationship for a reference usage representing a subject without a source in the SysML v1 model.

#### General Classes

- `Factory` (from `Foundations`)
- `ToSubjectMembership_Init` (from `SystemInitializers`)

#### Operations

- `create () : SubjectMembership [1]`
- `ownedMemberParameter () : Feature [1]` {redefines `ownedMemberParameter`}

```
EmptySubject_Factory.create()
```

### 7.5.2.20 FeatureTyping\_Factory

#### Description

Factory class to create a `FeatureTyping` relationship. The `create` parameter is set as the type.

#### General Classes

- Factory (from Foundations)
- ToFeatureTyping\_Init (from KerMLInitializers)

#### Association Ends

- type : NamedElement [1]

#### Operations

- create (in type : NamedElement) : FeatureTyping [1]
- type () : Type [1] {redefines type}

type

### 7.5.2.21 FlowEndParameterMembership\_Factory

#### Description

Factory class to create a ParameterMembership relationship for an end of a FlowUsage as a target element for a UML4SysML::InformationFlow that is realized by a UML4SysML::Connector.

#### General Classes

- Factory (from Foundations)
- ToParameterMembership\_Init (from KerMLInitializers)

#### Association Ends

- end : NamedElement [1]
- informationFlow : InformationFlow [1]

#### Operations

- create (in informationFlow : InformationFlow, in end : NamedElement) : ParameterMembership [1]
- ownedMemberParameter () : Feature [1] {redefines ownedMemberParameter}

```
InformationFlowEventOccurrenceUsage_Factory.create(informationFlow, end)
```

### 7.5.2.22 FlowItem\_Factory

#### Description

Factory class to create a ItemFeature element as a target element for the flowing entity specified by an UML4SysML::InformationFlow.

#### General Classes

- Factory (from Foundations)
- ToItemFeature\_Init (from SystemInitializers)

#### Association Ends

- item : NamedElement [1]

## Operations

- create (in item : NamedElement) : PayloadFeature [1]
- ownedRelationship () : Relationship [0..\*] {redefines ownedRelationship}

```
if item.ocIsKindOf(UML::Classifier) then
    Set{FeatureTyping_Factory.create(item)}
else if item.ocIsKindOf(UML::Property) then
    Set{ReferenceSubsetting_Factory.create(item)}
else
    Set{}
endif
endif
```

### 7.5.2.23 FlowItemFeatureMembership\_Factory

#### Description

Factory class to create a FeatureMembership relationship for an ItemFeature as a target element for the flowing entity specified by an UML4SysML::InformationFlow.

#### General Classes

- Factory (from Foundations)
- ToFeatureMembership\_Init (from KerMLInitializers)

#### Association Ends

- item : NamedElement [1]

## Operations

- create (in item : NamedElement) : FeatureMembership [1]
- ownedMemberFeature () : Feature [1] {redefines ownedMemberFeature}

```
FlowItem_Factory.create(item)
```

### 7.5.2.24 FlowUsage\_Factory

#### Description

Factory class to create a FlowUsage as a target element for a UML4SysML::InformationFlow that is realized by a UML4SysML::Connector. The factory class only supports UML4SysML::InformationFlows which have exactly one source and one target element, which is implicitly assured since connectors in SysML may only ever have two ends.

#### General Classes

- Factory (from Foundations)
- ToFlowUsage\_Init (from SystemInitializers)

#### Association Ends

- informationFlow : InformationFlow [1]

## Operations

- create (in informationFlow : InformationFlow) : FlowUsage [1]
- ownedRelationship () : Relationship [0..\*] {redefines ownedRelationship}

```
let relationships : Set(KerML::Relationship) =
  informationFlow.realizingConnector->collect(c|Subsetting_Factory.create(c))
  ->including(FeatureTyping_Factory.create(informationFlow))
  ->including(FlowEndParameterMembership_Factory.create(
    informationFlow,informationFlow.source.get(0))
  ->including(FlowEndParameterMembership_Factory.create(
    informationFlow,informationFlow.target.get(0))) in
let itemProperty : UML::Property =
  if Helper.hasStereotypeApplied(informationFlow,
    'SysML::Ports&Flows::ItemFlow') then
    Helper.getTagValueAsElement(informationFlow,
      'SysML::Ports&Flows::ItemFlow', 'itemProperty')
  else
    invalid
  endif in

if itemProperty.oclIsUndefined() then
  relationships->union(informationFlow.conveyed->flatten()
    ->collect(i | FlowItemFeatureMembership_Factory.create(i)))
else
  relationships->including(
    FlowItemFeatureMembership_Factory.create(itemProperty))
endif
```

### 7.5.2.25 FlowUsageFeatureMembership\_Factory

#### Description

Factory class to create a FeatureMembership relationship for a FlowUsage as a target element for a UML4SysML::InformationFlow that is realized by a UML4SysML::Connector.

#### General Classes

- Factory (from Foundations)
- ToFeatureMembership\_Init (from KerMLInitializers)

#### Association Ends

- informationFlow : InformationFlow [1]

#### Operations

- create (in informationFlow : InformationFlow) : FeatureMembership [1]
- ownedMemberFeature () : Feature [1] {redefines ownedMemberFeature}

```
FlowUsage_Factory.create(informationFlow)
```

### 7.5.2.26 InformationFlowEventOccurrenceUsage\_Factory

#### Description

## General Classes

- Factory (from Foundations)
- ToEventOccurrenceUsage\_Init (from SystemInitializers)

## Association Ends

- end : NamedElement [1]
- informationFlow : InformationFlow [1]

## Operations

- create (in informationFlow : InformationFlow, in end : NamedElement) : EventOccurrenceUsage [1]
- ownedRelationship () : Relationship [0..\*] {redefines ownedRelationship}

```
Set{InformationFlowReferenceSubsetting_Factory.create(informationFlow, end)}
```

### 7.5.2.27 InformationFlowReferenceSubsetting\_Factory

#### Description

Factory class to create a ReferenceSubsetting relationship for an end of a FlowUsage subsetting the target element of an end element of an UML4SysML::InformationFlow.

## General Classes

- Factory (from Foundations)
- ToReferenceSubsetting\_Init (from KerMLInitializers)

## Association Ends

- end : NamedElement [1]
- informationFlow : InformationFlow [1]

## Operations

- create (in informationFlow : InformationFlow, in end : NamedElement) : ReferenceSubsetting [1]
- referencedFeature () : Feature [1] {redefines referencedFeature}

```
InformationFlowEnd_Mapping.getMapped(informationFlow, end)
```

### 7.5.2.28 LiteralBoolean\_Factory

#### Description

Factory class to create a LiteralBoolean element.

## General Classes

- Factory (from Foundations)
- ToExpression\_Init (from KerMLInitializers)

## Association Ends

- boolean : Boolean [1]

- to : LiteralBoolean [1]  
{redefines: ToExpression\_Init::to}

### Operations

- create (in boolean : Boolean) : LiteralBoolean [1]
- ownedRelationship () : Relationship [0..\*] {redefines ownedRelationship}

```
Set{ReturnParameterFeatureMembership_Factory.create() }
```

## 7.5.2.29 LiteralNull\_Factory

### Description

Factory class to create a LiteralNull element.

### General Classes

- Factory (from Foundations)
- ToExpression\_Init (from KerMLInitializers)

### Association Ends

- to : NullExpression [1]  
{redefines: ToExpression\_Init::to}

### Operations

- create () : NullExpression [1]
- ownedRelationship () : Relationship [0..\*] {redefines ownedRelationship}

```
Set{ReturnParameterFeatureMembership_Factory.create() }
```

## 7.5.2.30 LiteralRational\_Factory

### Description

Factory class to create a LiteralRational element.

### General Classes

- Factory (from Foundations)
- ToExpression\_Init (from KerMLInitializers)

### Association Ends

- real : Real [1]
- to : LiteralRational [1]  
{redefines: ToExpression\_Init::to}

### Operations

- create (in real : Real) : LiteralReal [1]
- ownedRelationship () : Relationship [0..\*] {redefines ownedRelationship}



```
Set{ReturnParameterFeatureMembership_Factory.create() }
```

### 7.5.2.31 LowerBound\_Factory

#### Description

#### General Classes

- Factory (from Foundations)

#### Association Ends

- multiplicityLowerBoundMembership : MultiplicityLowerBoundMembership\_Factory [1]
- to : LiteralInteger [1]  
{redefines: Initializer::to}

#### Operations

- create (in lowerValue : Integer) : LiteralInteger [1]
- ownedRelationship () : Relationship [0..\*]

```
Set{ReturnParameterFeatureMembership_Factory.create() }
```

- value () : Integer [1]

```
lowerValue
```

### 7.5.2.32 MultiplicityElement\_Factory

#### Description

#### General Classes

- Factory (from Foundations)
- ToFeature\_Init (from KerMLInitializers)

#### Association Ends

- lowerValue : Integer [1]
- multiplicityLowerBoundMembership : MultiplicityLowerBoundMembership\_Factory [1]
- multiplicityUpperBoundMembership : MultiplicityUpperBoundMembership\_Factory [1]
- upperValue : Integer [1]

#### Operations

- create (in lowerValue : Integer, in upperValue : Integer) : Feature [1]
- ownedRelationship () : Relationship [0..\*] {redefines ownedRelationship}

```
Set{self.multiplicityLowerBoundMembership, self.multiplicityUpperBoundMembership}
```

### 7.5.2.33 MultiplicityLowerBoundMembership\_Factory

#### Description

## General Classes

- Factory (from Foundations)
- ToFeatureMembership\_Init (from KerMLInitializers)

## Association Ends

- lowerBound : LowerBound\_Factory [1]
- multiplicityElement : MultiplicityElement\_Factory [1]

## Operations

- create () : FeatureMembership [1]
- ownedMemberFeature () : Feature [1] {redefines ownedMemberFeature}

```
self.lowerBound
```

### 7.5.2.34 MultiplicityMembership\_Factory

## Description

## General Classes

- Factory (from Foundations)
- ToFeatureMembership\_Init (from KerMLInitializers)

## Association Ends

- lowerValue : Integer [1]
- upperValue : Integer [1]

## Operations

- create (in lowerValue : Integer, in upperValue : Integer) : FeatureMembership [1]
- ownedMemberFeature () : Feature [1] {redefines ownedMemberFeature}

```
if upperValue = 1 then
  if lowerValue = 0 then
    Helper.getMultiplicityRangeByName('zeroOrOne')
  else if lowerValue = 1 then
    Helper.getMultiplicityRangeByName('exactlyOne')
  else
    MultiplicityElement_Factory.create(lowerValue, upperValue)
  endif endif
else if upperValue = -1 then
  if lowerValue = 0 then
    Helper.getMultiplicityRangeByName('zeroToMany')
  else if lowerValue = 1 then
    Helper.getMultiplicityRangeByName('oneToMany')
  else
    MultiplicityElement_Factory.create(lowerValue, upperValue)
  endif endif
else
  MultiplicityElement_Factory.create(lowerValue, upperValue)
endif endif
```

### 7.5.2.35 MultiplicityUpperBoundMembership\_Factory

#### Description

#### General Classes

- Factory (from Foundations)
- ToFeatureMembership\_Init (from KerMLInitializers)

#### Association Ends

- multiplicityElement : MultiplicityElement\_Factory [1]
- upperBound : UpperBound\_Factory [1]

#### Operations

- create (in upperValue : Integer) : FeatureMembership [1]

### 7.5.2.36 ObjectFlowItemFlowEndRedefinition\_Factory

#### Description

#### General Classes

- Factory (from Foundations)
- ToRedefinition\_Init (from KerMLInitializers)

#### Association Ends

- feature : Feature [1]

#### Operations

- create (in feature : Feature) : Redefinition [1]
- redefinedFeature () : Feature [1] {redefines redefinedFeature}

`feature`

### 7.5.2.37 ParameterMembership\_Factory

#### Description

Factory class to create a ParameterMembership relationship.

#### General Classes

- Factory (from Foundations)
- ToParameterMembership\_Init (from KerMLInitializers)

#### Operations

- create () : ParameterMembership [1]
- ownedMemberParameter () : Feature [1] {redefines ownedMemberParameter}

`ReferenceUsage_Factory.create()`

### 7.5.2.38 ReferenceSubsetting\_Factory

#### Description

Factory class to create a ReferenceSubsetting relationship. The create parameter is set as the referenced feature.

#### General Classes

- Factory (from Foundations)
- ToReferenceSubsetting\_Init (from KerMLInitializers)

#### Association Ends

- property : Property [1]

#### Operations

- create (in property : Property) : ReferenceSubsetting [1]
- referencedFeature () : Feature [1] {redefines referencedFeature}

property

### 7.5.2.39 ReferenceUsage\_Factory

#### Description

Factory class to create a ReferenceUsage element with direction 'in'.

#### General Classes

- Factory (from Foundations)
- ToReferenceUsage\_Init (from SystemInitializers)

#### Operations

- create () : ReferenceUsage [1]
- direction () : FeatureDirectionKind [0..1] {redefines direction}

KerML::FeatureDirectionKind::\_in'

### 7.5.2.40 ReturnParameterFeature\_Factory

#### Description

Factory class to create a feature element with direction 'out' representing a return parameter.

#### General Classes

- Factory (from Foundations)
- ToFeature\_Init (from KerMLInitializers)

#### Operations

- create () : Feature [1]

- direction () : FeatureDirectionKind [0..1] {redefines direction}

```
KerML::FeatureDirectionKind::_out'
```

#### 7.5.2.41 ReturnParameterFeatureMembership\_Factory

##### Description

Factory class to create a feature membership relationship for a feature element with direction 'out' representing a return parameter.

##### General Classes

- Factory (from Foundations)
- ToReturnParameterMembership\_Init (from KerMLInitializers)

##### Operations

- create () : ReturnParameterMembership [1]
- ownedMemberParameter () : Feature [1] {redefines ownedMemberParameter}

```
ReturnParameterFeature_Factory.create()
```

#### 7.5.2.42 Subsetting\_Factory

##### Description

Factory class to create a Subsetting relationship. The create parameter is set as the subsetting feature.

##### General Classes

- Factory (from Foundations)
- ToSubsetting\_Init (from KerMLInitializers)

##### Association Ends

- subsetting : NamedElement [1]

##### Operations

- create (in subsetting : NamedElement) : Subsetting [1]
- subsettingFeature () : Feature [1] {redefines subsettingFeature}

```
subsetting
```

#### 7.5.2.43 UpperBound\_Factory

##### Description

##### General Classes

- Factory (from Foundations)

##### Association Ends

- multiplicityUpperBoundMembership : MultiplicityUpperBoundMembership\_Factory [1]
- to : LiteralInteger [1]  
    {redefines: Initializer::to}

## Operations

- create (in upperValue : Integer) : LiteralInteger [1]
- ownedRelationship () : Relationship [0..\*]

```
Set {ReturnParameterFeatureMembership_Factory.create () }
```

- value () : Integer [1]

```
upperValue
```

## 7.6 Generic Mappings

### 7.6.1 Overview

Generic mappings are partial definitions of transformation rules that are intended to factorize reusable algorithms for making the global specification more compact and easier to read and maintain. Basically, they provide a default value for all the non-derived attributes of their target metaclass wherever possible, or declare an abstract operation for them otherwise. They are similar to initializers, except that they have a source element defined. The operations provided by the generic mappings can be redefined by their specialization, as appropriate according to the source type specified by the redefinition of their *from* attribute.

All of these generic mappings are abstract.

### 7.6.2 Common Mappings

#### 7.6.2.1 CommonFeatureReferenceExpression\_Mapping

##### Description

Common mapping class for a feature reference expression.

##### General Mappings

ToFeatureReferenceExpression\_Init  
Mapping

##### Mapping Source

TypedElement

##### Mapping Target

FeatureReferenceExpression

##### Owned Mappings

(none)

##### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureReferenceExpression::ownedRelationship () : Relationship [0..*]`  
`Set { CommonMembership_Mapping.getMapped (from) ,`  
`CommonReturnParameterFeatureMembership_Mapping.getMapped (from) }`

#### 7.6.2.2 CommonMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

##### General Mappings

ToMembership\_Init  
Mapping

##### Mapping Source

TypedElement

##### Mapping Target

Membership

##### Owned Mappings

(none)

##### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Membership::memberElement () : Element [1]`  
`from`

#### 7.6.2.3 CommonParameterReferenceUsageInMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

##### General Mappings

ToParameterMembership\_Init  
Mapping

## Mapping Source

Element

## Mapping Target

ParameterMembership

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ParameterMembership::ownedMemberParameter () : Feature [1]  

```
if not from.ocIsKindOf(UML::TypedElement) then
    CommonParameterReferenceUsageIn_Mapping.getMapped(from)
else if from.ocIsType(UML::TypedElement).type.ocIsUndefined() then
    CommonParameterReferenceUsageIn_Mapping.getMapped(from)
else
    CommonParameterReferenceUsageInUntyped_Mapping.getMapped(from)
endif
endif
```

### 7.6.2.4 CommonParameterReferenceUsageIn\_Mapping

#### Description

Common mapping class that creates a parameter reference usage element with direction 'in' and with a type.

#### General Mappings

CommonParameterReferenceUsageInUntyped\_Mapping  
Mapping

## Mapping Source

Element

## Mapping Target

ReferenceUsage

## Owned Mappings

(none)

## Applicable filters

(none)



## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceUsage::ownedRelationship () : Relationship [0..*]`

```
if from.ooclIsKindOf(UML::TypedElement) then
  Set{CommonParameterReferenceUsageInFeatureTyping_Mapping.getMapped(from)}
else Set{} endif
```

### 7.6.2.5 CommonParameterReferenceUsageInFeatureTyping\_Mapping

#### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

#### General Mappings

ToFeatureTyping\_Init  
Mapping

#### Mapping Source

Element

#### Mapping Target

FeatureTyping

#### Owned Mappings

(none)

#### Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureTyping::type () : Type [1]`

```
if from.ooclIsKindOf(UML::TypedElement)
then
  if from.ooclAsType(UML::TypedElement).type.ooclIsKindOf(UML::PrimitiveType) then
    Helper.getScalarValueType(from.ooclAsType(UML::TypedElement).type)
  else
    from.ooclAsType(UML::TypedElement).type
  endif
else invalid endif
```

### 7.6.2.6 CommonParameterReferenceUsageInUntyped\_Mapping

#### Description

Common mapping class that creates a parameter reference usage element with direction 'in' and without a type.

## General Mappings

ToReferenceUsage\_Init  
Mapping

## Mapping Source

Element

## Mapping Target

ReferenceUsage

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReferenceUsage::direction () : FeatureDirectionKind [0..1]

KerML::FeatureDirectionKind::\_in'

### 7.6.2.7 CommonReturnParameterFeature\_Mapping

#### Description

Common mapping class that creates a parameter feature element with a type.

## General Mappings

CommonReturnParameterFeatureUntyped\_Mapping  
Mapping

## Mapping Source

Element

## Mapping Target

Feature

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Feature::ownedRelationship () : Relationship [0..\*]

```
if from.ocIsKindOf(UML::Property) then
    Set{CommonReturnParameterFeatureTyping_Mapping.getMapped(from)}
else
    Set{}
endif
```

### 7.6.2.8 CommonReturnParameterFeatureTyping\_Mapping

#### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

#### General Mappings

ToFeatureTyping\_Init  
Mapping

#### Mapping Source

Element

#### Mapping Target

FeatureTyping

#### Owned Mappings

(none)

#### Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureTyping::type () : Type [1]

```
if from.ocIsKindOf(UML::Property)
then
if from.ocAsType(UML::TypedElement).type.ocIsKindOf(UML::PrimitiveType) then
    Helper.getScalarValueType(from.ocAsType(UML::TypedElement).type)
else
    from.ocAsType(UML::TypedElement).type
endif
else invalid endif
```

### 7.6.2.9 CommonReturnParameterFeatureUntyped\_Mapping

#### Description

Common mapping class that creates a parameter feature element without a type.

### General Mappings

ToFeature\_Init  
Mapping

### Mapping Source

Element

### Mapping Target

Feature

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Feature::direction () : FeatureDirectionKind [0..1]

```
KerML::FeatureDirectionKind::_out'
```

## 7.6.2.10 CommonReturnParameterFeatureMembership\_Mapping

### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

### General Mappings

ToReturnParameterMembership\_Init  
Mapping

### Mapping Source

Element

### Mapping Target

ReturnParameterMembership

### Owned Mappings

(none)

### Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReturnParameterMembership::ownedMemberParameter () : Feature [1]

```
if not from.ocIsKindOf(UML::TypedElement) then
    CommonReturnParameterFeatureUntyped_Mapping.getMapped(from)
else if from.ocIsType(UML::TypedElement).type.ocIsUndefined() then
    CommonReturnParameterFeatureUntyped_Mapping.getMapped(from)
else
    CommonReturnParameterFeatureUntyped_Mapping.getMapped(from)
endif
endif
```

### 7.6.2.11 CommonReturnParameterReferenceUsageMembership\_Mapping

#### Description

Creates a membership relationship for *memberElement()*.

#### General Mappings

ToReturnParameterMembership\_Init  
Mapping

#### Mapping Source

Element

#### Mapping Target

ReturnParameterMembership

#### Owned Mappings

(none)

#### Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReturnParameterMembership::ownedMemberParameter () : Feature [0..1]

```
if not from.ocIsKindOf(UML::TypedElement) then
    CommonReturnParameterReferenceUsageUntyped_Mapping.getMapped(from)
else if from.ocIsType(UML::TypedElement).type.ocIsUndefined() then
    CommonReturnParameterReferenceUsageUntyped_Mapping.getMapped(from)
else
```

```

        CommonReturnParameterReferenceUsageUntyped_Mapping.getMapped(from)
    endif
endif

```

### 7.6.2.12 CommonReturnParameterReferenceUsage\_Mapping

#### Description

Creates a reference usage.

#### General Mappings

CommonReturnParameterReferenceUsageUntyped\_Mapping  
Mapping

#### Mapping Source

Element

#### Mapping Target

ReferenceUsage

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReferenceUsage::ownedRelationship () : Relationship [0..\*]

```

if from.ocIsKindOf(UML::TypedElement) then
Set{CommonReturnParameterReferenceUsageFeatureTyping_Mapping.getMapped(from)}
else Set{} endif

```

### 7.6.2.13 CommonReturnParameterReferenceUsageFeatureTyping\_Mapping

#### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

#### General Mappings

ToFeatureTyping\_Init  
Mapping

#### Mapping Source

Element

#### Mapping Target

FeatureTyping

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureTyping::type () : Type [1]

```
if from.ocIsKindOf(UML::TypedElement)
then
if from.ocAsType(UML::TypedElement).type.ocIsKindOf(UML::PrimitiveType) then
  Helper.getScalarValueType(from.ocAsType(UML::TypedElement).type)
else
  from.ocAsType(UML::TypedElement).type
endif
else invalid endif
```

## 7.6.2.14 CommonReturnParameterReferenceUsageUntyped\_Mapping

### Description

Creates a reference usage.

### General Mappings

ToReferenceUsage\_Init  
Mapping

### Mapping Source

Element

### Mapping Target

ReferenceUsage

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceUsage::direction () : FeatureDirectionKind [0..1]`

`KerML::FeatureDirectionKind::_'out'`

### 7.6.2.15 CommonReferenceUsageIn\_Mapping

#### Description

Common mapping class that creates a reference usage element with direction 'in'.

#### General Mappings

CommonReferenceUsageInUntyped\_Mapping  
Mapping

#### Mapping Source

TypedElement

#### Mapping Target

ReferenceUsage

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceUsage::ownedRelationship () : Relationship [0..*]`

Common mapping class that creates a reference usage element with direction 'in'.

`Set { CommonReferenceUsageInFeatureTyping_Mapping.getMapped (from) }`

### 7.6.2.16 CommonReferenceUsageInFeatureMembership\_Mapping

#### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

#### General Mappings

ToFeatureMembership\_Init  
Mapping

#### Mapping Source



TypedElement

### Mapping Target

FeatureMembership

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureMembership::ownedMemberFeature () : Feature [1]  

```
if from.type.ocIsUndefined() then
    CommonReferenceUsageInUntyped_Mapping.getMapped(from)
else
    CommonReferenceUsageIn_Mapping.getMapped(from)
endif
```

## 7.6.2.17 CommonReferenceUsageInFeatureTyping\_Mapping

### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

### General Mappings

ToFeatureTyping\_Init  
Mapping

### Mapping Source

TypedElement

### Mapping Target

FeatureTyping

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureTyping::type () : Type [1]`

```
if from.type.ocIsKindOf(UML::PrimitiveType) then
    Helper.getScalarValueType(from.type)
else
    from.type
endif
```

### 7.6.2.18 CommonReferenceUsageInUntyped\_Mapping

#### Description

Common mapping class that creates an untyped reference usage element with direction 'in'.

#### General Mappings

ToReferenceUsage\_Init  
Mapping

#### Mapping Source

TypedElement

#### Mapping Target

ReferenceUsage

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceUsage::declaredName () : String [0..1]`

```
from.name
```

- `ReferenceUsage::direction () : FeatureDirectionKind [0..1]`

```
KerML::FeatureDirectionKind::_in'
```

## 7.7 Mappings from UML4SysML metaclasses

### 7.7.1 Overview

UML4SysML is the subset of UML containing all model elements that are reused by SysML. The complete list of model elements is defined in [SysMLv1], subclause 4.1.

## 7.7.2 Actions

### 7.7.2.1 Overview

**Table 1. List of all mappings**

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
AcceptCallAction	AcceptActionUsage
AcceptEventAction	AcceptActionUsage
ActionInputPin	ReferenceUsage
AddStructuralFeatureValueAction	ActionUsage
AddVariableValueAction	ActionUsage
BroadcastSignalAction	ActionUsage
CallBehaviorAction	ActionUsage
CallOperationAction	ActionUsage
Clause	not mapped; see next section
ClearAssociationAction	ActionUsage
ClearStructuralFeatureAction	ActionUsage
ClearVariableAction	ActionUsage
ConditionalNode	Namespace ActionUsage
CreateLinkAction	ActionUsage
CreateLinkObjectAction	ActionUsage
CreateObjectAction	ActionUsage
DestroyLinkAction	ActionUsage
DestroyObjectAction	ActionUsage
InputPin	ReferenceUsage
LinkEndCreationData	not mapped; see next section
LinkEndData	not mapped; see next section
LinkEndDestructionData	not mapped; see next section
LoopNode	Namespace ActionUsage
OpaqueAction	ActionUsage
OutputPin	ReferenceUsage
RaiseExceptionAction	ActionUsage
ReadExtentAction	ActionUsage
ReadIsClassifiedObjectAction	ActionUsage
ReadLinkAction	ActionUsage
ReadLinkObjectEndAction	ActionUsage

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
ReadSelfAction	ActionUsage
ReadStructuralFeatureAction	ActionUsage
ReadVariableAction	ActionUsage
ReclassifyObjectAction	ActionUsage
ReduceAction	ActionUsage
RemoveStructuralFeatureValueAction	ActionUsage
RemoveVariableValueAction	ActionUsage
ReplyAction	ActionUsage
SendObjectAction	ActionUsage
SendSignalAction	ActionUsage
SequenceNode	Namespace ActionUsage
StartClassifierBehaviorAction	ActionUsage
StartObjectBehaviorAction	ActionUsage
StructuredActivityNode	Namespace ActionUsage
TestIdentityAction	CalculationUsage
UnmarshallAction	ActionUsage
ValuePin	ReferenceUsage
ValueSpecificationAction	ActionUsage

### 7.7.2.2 UML4SysML::Actions elements not mapped

**Table 2. List of SysML v1 elements not mapped of this section**

SysML v1 Concept	Rationale
AcceptCallAction	Since the CallEvent is not supported by SysML v2, the AcceptCallAction is also not covered. It is mapped to an empty action usage to keep the connections within the activity respectively action definition.
ActionInputPin	The UML4SysML::ActionInputPin concept is not covered by SysML v2. The model element is mapped as a input or output pin, but without the special action input pin semantics.
Clause	Mapping is not specified yet.
ConditionalNode	Mapping is not specified yet.
LinkEndCreationData	Mapping is not specified yet.
LinkEndData	Mapping is not specified yet.
LinkEndDestructionData	Mapping is not specified yet.

SysML v1 Concept	Rationale
ReclassifyObjectAction	The UML4SysML::ReclassifyObjectAction is not supported by SysML v2. It is mapped to an empty action usage to keep the connections within the activity respectively action definition.
ReplyAction	The UML4SysML::ReplyAction is only used with UML4SysML::AcceptCallAction. Since we have no mapping of AcceptCallAction to SysML v2, there is also no mapping for ReplyAction. However, it is mapped to an empty action usage to keep the connections within the activity respectively action definition.
StartClassifierBehaviorAction	The UML4SysML::StartClassifierBehaviorAction is not supported by SysML v2. It is mapped to an empty action usage to keep the connections within the activity respectively action definition.
StartObjectBehaviorAction	The UML4SysML::StartObjectBehaviorAction is not supported by SysML v2. It is mapped to an empty action usage to keep the connections within the activity respectively action definition.
UnmarshallAction	Mapping is not specified yet.

### 7.7.2.3 Mapping Specifications

#### 7.7.2.3.1 Accept Event Actions

##### 7.7.2.3.1.1 AcceptCallAction\_Mapping

##### Description

Since the CallEvent is not supported by SysML v2, the AcceptCallAction is also not covered. It is mapped to an empty action usage to keep the connections within the activity respectively action definition.

##### General Mappings

AcceptEventAction\_Mapping

##### Mapping Source

AcceptCallAction

##### Mapping Target

AcceptActionUsage

##### Owned Mappings

(none)

##### Applicable filters

(none)

### 7.7.2.3.1.2 AcceptEventAction\_Mapping

#### Description

The UML4SysML::AcceptEventAction is mapped to a AcceptActionUsage element.

If the trigger is a signal, it is mapped to an accept parameter typed by the signal.

SysMLv2 does not support more than one trigger. Therefore only the first specified trigger of the action is transformed. All further triggers are ignored.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
action acceptEventActionSignalEvent1 accept : SysMLv1Signal via sysMLv1Port;
action acceptEventActionChangeEvent1 accept when when changeExpression.result {
    calc changeExpression {
        return : ScalarValues::Boolean;
        language "OCL"
        /*
        * x > 0
        */
    }
}
```

#### General Mappings

CommonAction\_Mapping

#### Mapping Source

AcceptEventAction

#### Mapping Target

AcceptActionUsage

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- AcceptActionUsage::ownedRelationship () : Relationship [0..\*]

```
let relationships : Set(KerML::Relationship) = Helper.actionOwnedRelationship(from)
->including(AEARReceiverParameterMembership_Mapping.getMapped(from)) in
let relationshipsWithParameter : Set(KerML::Relationship) =
if (from.trigger.get(0).event.ocIsTypeOf(UML::SignalEvent) or
    from.trigger.get(0).event.ocIsTypeOf(UML::ChangeEvent)) then
```

```

        relationships->including(AEAPParameterMembership_Mapping.getMapped(from))
    else
        relationships
    endif in
    if from.trigger.get(0).event.oclIsTypeOf(UML::ChangeEvent) then
        relationshipsWithParameter
        ->including(ElementFeatureMembership_Mapping.getMapped(
            from.trigger.get(0).event.oclAsType(UML::ChangeEvent).changeExpression))
    else relationshipsWithParameter
    endif

```

### 7.7.2.3.1.3 AEChangeExpressionMembership\_Mapping

#### Description

Creates a membership relationship for *memberElement()*.

#### General Mappings

ToFeatureMembership\_Init  
Mapping

#### Mapping Source

AcceptEventAction

#### Mapping Target

FeatureMembership

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureMembership::ownedMemberFeature () : Feature [1]

```

from.trigger.get(0).event.oclAsType(UML::ChangeEvent).changeExpression

```

### 7.7.2.3.1.4 AEChangeParameter\_Mapping

#### Description

The mapping class transforms the change event specified at the AcceptEventAction.

#### General Mappings

ToReferenceUsage\_Init  
Mapping

#### Mapping Source

AcceptEventAction

### Mapping Target

ReferenceUsage

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReferenceUsage::ownedRelationship () : Relationship [0..\*]  
`Set{AEChangeParameterFeatureValue_Mapping.getMapped(from)}`
- ReferenceUsage::direction () : FeatureDirectionKind [0..1]  
`KerML::FeatureDirectionKind::_in'`

#### 7.7.2.3.1.5 AEChangeParameterFeatureValue\_Mapping

### Description

Creates a feature value relationship.

### General Mappings

ToFeatureValue\_Init  
Mapping

### Mapping Source

AcceptEventAction

### Mapping Target

FeatureValue

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules



In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureValue::value () : Expression [1]`

`AEChangeParameterTrigger_Mapping.getMapped(from)`

#### **7.7.2.3.1.6 AEChangeParameterTrigger\_Mapping**

##### **Description**

The mapping class creates a `TriggerInvocationExpression` from the change event specified at the `AcceptEventAction`.

##### **General Mappings**

`ToInvocationExpression_Init`  
Mapping

##### **Mapping Source**

`AcceptEventAction`

##### **Mapping Target**

`TriggerInvocationExpression`

##### **Owned Mappings**

(none)

##### **Applicable filters**

(none)

##### **Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `TriggerInvocationExpression::ownedRelationship () : Relationship [0..*]`

`Set{AEChangeParameterFeatureMembership_Mapping.getMapped(from)}`

#### **7.7.2.3.1.7 AEChangeParameterTriggerExpression\_Mapping**

##### **Description**

The mapping class creates the trigger expression element for the change parameter of the SysML v2 `AcceptActionUsage` element.

##### **General Mappings**

`ToExpression_Init`  
Mapping

##### **Mapping Source**

AcceptEventAction

### Mapping Target

Expression

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Expression::ownedRelationship () : Relationship [0..\*]

```
Set { AEChangeParameterResultExpressionMembership_Mapping.getMapped (from) }
```

#### 7.7.2.3.1.8 AEChangeParameterResultExpressionMembership\_Mapping

### Description

Creates a membership relationship for *memberElement()*.

### General Mappings

ToFeatureMembership\_Init  
Mapping

### Mapping Source

AcceptEventAction

### Mapping Target

ResultExpressionMembership

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ResultExpressionMembership::ownedMemberFeature () : Feature [1]

```
AEChangeParameterFeatureChainExpression_Mapping.getMapped(from)
```

#### 7.7.2.3.1.9 AEChangeParameterFeatureChainExpression\_Mapping

##### Description

The mapping class creates the feature chain expression element for the change parameter of the SysML v2 `AcceptActionUsage` element.

##### General Mappings

ToInvocationExpression\_Init  
Mapping

##### Mapping Source

AcceptEventAction

##### Mapping Target

FeatureChainExpression

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureChainExpression::ownedRelationship () : Relationship [0..*]`

```
Set{AEChangeParameterParameterMembership_Mapping.getMapped(from) }
```

#### 7.7.2.3.1.10 AEChangeParameterFeatureMembership\_Mapping

##### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

##### General Mappings

ToFeatureMembership\_Init  
Mapping

##### Mapping Source

AcceptEventAction

##### Mapping Target

FeatureMembership

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureMembership::ownedMemberFeature () : Feature [1]`  
`AEChangeParameterTriggerExpression_Mapping.getMapped(from)`

### 7.7.2.3.1.11 AEChangeParameterFeature\_Mapping

#### Description

The mapping class creates the feature for the feature chain expression element for the change parameter of the SysML v2 `AcceptActionUsage` element.

#### General Mappings

ToFeature\_Init  
Mapping

#### Mapping Source

AcceptEventAction

#### Mapping Target

Feature

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Feature::ownedRelationship () : Relationship [0..*]`  
`Set{AEChangeParameterExpressionFeatureValue_Mapping.getMapped(from) }`

### 7.7.2.3.1.12 AEChangeParameterExpressionFeatureValue\_Mapping

#### Description

Creates a feature value relationship.

### General Mappings

ToFeatureValue\_Init  
Mapping

### Mapping Source

AcceptEventAction

### Mapping Target

FeatureValue

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureValue::value () : Expression [1]

```
AEChangeParameterFeatureReferenceExpression_Mapping.getMapped(from)
```

#### 7.7.2.3.1.13 AEChangeParameterFeatureReferenceExpression\_Mapping

### Description

The mapping class creates the feature reference expression for the feature chain expression element for the change parameter of the SysML v2 AcceptActionUsage element.

### General Mappings

ToFeatureReferenceExpression\_Init  
Mapping

### Mapping Source

AcceptEventAction

### Mapping Target

FeatureReferenceExpression

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureReferenceExpression::ownedRelationship () : Relationship [0..*]`  
`Set{AEChangeParameterMembership_Mapping.getMapped(from) }`

#### 7.7.2.3.1.14 AEChangeParameterMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

##### General Mappings

ToMembership\_Init  
Mapping

##### Mapping Source

AcceptEventAction

##### Mapping Target

Membership

##### Owned Mappings

(none)

##### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Membership::memberElement () : Element [1]`  
`from.trigger.get(0).event.oclAsType(UML::ChangeEvent).changeExpression`

#### 7.7.2.3.1.15 AEChangeParameterParameterMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

##### General Mappings

ToParameterMembership\_Init  
Mapping

**Mapping Source**

AcceptEventAction

**Mapping Target**

ParameterMembership

**Owned Mappings**

(none)

**Applicable filters**

(none)

**Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ParameterMembership::ownedMemberParameter () : Feature [1]  
`AEChangeParameterFeature_Mapping.getMapped(from)`

**7.7.2.3.1.16 AEAREceiverParameter\_Mapping****Description**

The mapping class creates the reference usage element for the receiver parameter of the SysML v2 AcceptActionUsage element.

**General Mappings**

ToReferenceUsage\_Init  
Mapping

**Mapping Source**

AcceptEventAction

**Mapping Target**

ReferenceUsage

**Owned Mappings**

(none)

**Applicable filters**

(none)

**Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceUsage::direction () : FeatureDirectionKind [0..1]`

```
KerML::FeatureDirectionKind::_in'
```

- `ReferenceUsage::ownedRelationship () : Relationship [0..*]`

```
if from.trigger.get(0).port->size() > 0
then Set{AEARceiverFeatureValue_Mapping.getMapped(from)}
else Set{}
endif
```

#### 7.7.2.3.1.17 AEARceiverParameterMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

##### General Mappings

ToParameterMembership\_Init  
Mapping

##### Mapping Source

AcceptEventAction

##### Mapping Target

ParameterMembership

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ParameterMembership::ownedMemberParameter () : Feature [1]`

```
AEARceiverParameter_Mapping.getMapped(from)
```

#### 7.7.2.3.1.18 AEARceiverFeatureValue\_Mapping

##### Description

Creates a feature value relationship.

##### General Mappings



ToFeatureValue\_Init  
Mapping

#### **Mapping Source**

AcceptEventAction

#### **Mapping Target**

FeatureValue

#### **Owned Mappings**

(none)

#### **Applicable filters**

(none)

#### **Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureValue::value () : Expression [1]

`AEARceiverFeatureReferenceExpression_Mapping.getMapped (from)`

#### **7.7.2.3.1.19 AEASignalParameter\_Mapping**

##### **Description**

The mapping class creates the reference usage element for the signal parameter of the SysML v2 AcceptActionUsage element.

##### **General Mappings**

ToReferenceUsage\_Init  
Mapping

#### **Mapping Source**

AcceptEventAction

#### **Mapping Target**

ReferenceUsage

#### **Owned Mappings**

(none)

#### **Applicable filters**

(none)

#### **Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReferenceUsage::ownedRelationship () : Relationship [0..\*]  
`Set{AEASignalParameterFeatureTyping_Mapping.getMapped(from)}`
- ReferenceUsage::direction () : FeatureDirectionKind [0..1]  
`KerML::FeatureDirectionKind::_in'`

#### 7.7.2.3.1.20 AEASignalParameterFeatureTyping\_Mapping

##### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

##### General Mappings

ToFeatureTyping\_Init  
Mapping

##### Mapping Source

AcceptEventAction

##### Mapping Target

FeatureTyping

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureTyping::type () : Type [1]  

```
let event : UML::Event = from.trigger.get(0).event in
if event.ocIsTypeOf(UML::SignalEvent) then
    event.ocIsType(UML::SignalEvent).signal
else invalid endif
```

#### 7.7.2.3.1.21 AEAParameterMembership\_Mapping

##### Description

The mapping class creates the parameter membership relationship for the element that can be received by the accept action. The source of the element is the trigger of the UML4SysML::AcceptEventAction.

Currently, more than one trigger is not supported by the transformation.

## General Mappings

ToParameterMembership\_Init  
Mapping

## Mapping Source

AcceptEventAction

## Mapping Target

ParameterMembership

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ParameterMembership::ownedMemberParameter () : Feature [1]  

```
if from.trigger.get(0).event.ocliIsTypeOf(UML::SignalEvent) then
    AEASignalParameter_Mapping.getMapped(from)
else if from.trigger.get(0).event.ocliIsTypeOf(UML::ChangeEvent) then
    AEASignalParameter_Mapping.getMapped(from)
else
    invalid
endif endif
```

### 7.7.2.3.1.22 AEASignalParameter\_Mapping

#### Description

The mapping class creates the feature reference expression for the reference usage element for the receiver parameter of the SysML v2 AcceptActionUsage element.

## General Mappings

ToFeatureReferenceExpression\_Init  
Mapping

## Mapping Source

AcceptEventAction

## Mapping Target

FeatureReferenceExpression

## Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureReferenceExpression::ownedRelationship () : Relationship [0..*]`  

```
Set{AEARceiverFeatureReferenceExpressionMembership_Mapping.getMapped(from),  
ReturnParameterFeatureMembership_Factory.create() }
```

#### 7.7.2.3.1.23 AEARceiverFeatureReferenceExpressionMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

##### General Mappings

ToMembership\_Init  
Mapping

##### Mapping Source

AcceptEventAction

##### Mapping Target

Membership

##### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Membership::memberElement () : Element [1]`  

```
if from.trigger.get(0).port->size() > 0 then  
    from.trigger.get(0).port.get(0)  
else  
    invalid  
endif
```

#### 7.7.2.3.1.24 ReplyAction\_Mapping

##### Description

The UML4SysML::ReplyAction is only used with UML4SysML::AcceptCallAction. Since we have no mapping of AcceptCallAction to SysML v2, there is also no mapping for ReplyAction. However, it is mapped to an empty action usage to keep the connections within the activity respectively action definition.

##### General Mappings

CommonAction\_Mapping

##### Mapping Source

ReplyAction

##### Mapping Target

ActionUsage

##### Owned Mappings

(none)

##### Applicable filters

(none)

#### 7.7.2.3.1.25 UnmarshallAction\_Mapping

##### Description

The mapping of UML4SysML::UnmarshallAction is not specified yet. It is currently mapped to an empty action usage to keep the connections within the activity respectively action definition.

##### General Mappings

CommonAction\_Mapping

##### Mapping Source

UnmarshallAction

##### Mapping Target

ActionUsage

##### Owned Mappings

(none)

##### Applicable filters

(none)

### 7.7.2.3.2 Actions

#### 7.7.2.3.2.1 CommonAction\_Mapping

##### Description

Base mapping class for model elements of kind UML4SysML::Action. The target element is a SysML v2 ActionUsage.

##### General Mappings

ToActionUsage\_Init  
NamedElementMain\_Mapping

##### Mapping Source

Action

##### Mapping Target

ActionUsage

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ActionUsage::ownedRelationship () : Relationship [0..\*]

```
let actionInputPin: Set(UML::Element) =
    from.ownedElement->select(e | e.ocIsTypeOf(UML::ActionInputPin)) in
let triggers: Set(UML::Element) =
    from.ownedElement->select(e | e.ocIsKindOf(UML::Trigger)) in
let toElementFMS: Set(UML::Element) =
    from.ownedElement->select(e | e.ocIsKindOf(UML::Pin)) in
let toElementOMS: Set(UML::Element) =
    (((from.ownedElement-toElementFMS)-
    actionInputPin)-triggers)-from.ownedElement in
toElementOMS->collect(e | ElementOwningMembership_Mapping.getMapped(e))
->asSet()
->union(self.ocAsType(ElementMain_Mapping).ownedRelationship())
->union(toElementFMS
->collect(e | ElementFeatureMembership_Mapping.getMapped(e))->asSet())
```

- ActionUsage::isComposite () : Boolean [1]

```
true
```

#### 7.7.2.3.2.2 OpaqueAction\_Mapping

##### Description

The UML4SysML::OpaqueAction is mapped to a SysML v2 ActionUsage with a textual representation.

The following shows an example of the expected SysMLv2 textual syntax of a UML4SysML::OpaqueAction.

```
action thisIsAOpaqueAction {
  in x : ScalarValues::Integer;
  in y : ScalarValues::Integer;
  out result : ScalarValues::Boolean;

  language "OCL"
  /*
   * x = y + 1;
   */
}
```

## General Mappings

CommonAction\_Mapping

### Mapping Source

OpaqueAction

### Mapping Target

ActionUsage

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ActionUsage::ownedRelationship () : Relationship [0..\*]

```
if from.body->size() > 0 then
  Helper.actionOwnedRelationship (from) ->append (OABodyMembership_Mapping.getMapped (from))
else
  Helper.actionOwnedRelationship (from)
endif
```

#### 7.7.2.3.2.3 OABody\_Mapping

### Description

The languages and bodies of a UML4SysML::OpaqueAction are mapped to SysMLv2 TextualRepresentations.

### General Mappings

ToAnnotatingElement\_Init  
Mapping

#### Mapping Source

OpaqueAction

#### Mapping Target

TextualRepresentation

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- TextualRepresentation::body () : String [1]  

```
if from.body.notEmpty() then from.body.first() else invalid endif
```
- TextualRepresentation::language () : String [1]  

```
if from.language.notEmpty() then from.language.first() else invalid endif
```

#### 7.7.2.3.2.4 OABodyMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

##### General Mappings

ToOwningMembership\_Init  
Mapping

#### Mapping Source

OpaqueAction

#### Mapping Target

OwningMembership

#### Owned Mappings

(none)

#### Applicable filters



(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `OwningMembership::ownedMemberElement () : Element [1]`  
`OABody_Mapping.getMapped(from)`

#### 7.7.2.3.2.5 Pin\_Mapping

##### Description

Mapping class for model elements of kind `UML4SysML::Pin`. The operation `ownedRelationship()` makes a distinction between typed and untyped pins. The target element is a SysMLv2 `ReferenceUsage`.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
action def SysMLv1Activity {  
    action sysMLv1Action {  
        in sysMLv1InputPin : ScalarValues::Integer;  
        out sysMLv1UntypedOutputPin;  
    }  
}
```

##### General Mappings

`ToReferenceUsage_Init`  
`NamedElementMain_Mapping`

##### Mapping Source

`Pin`

##### Mapping Target

`ReferenceUsage`

##### Owned Mappings

(none)

##### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
not Helper.excludedPin(src)
```

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReferenceUsage::direction () : FeatureDirectionKind [0..1]

```
if from.ocIsTypeOf(UML::InputPin) then
    KerML::FeatureDirectionKind::_in'
else if from.ocIsTypeOf(UML::OutputPin) then
    KerML::FeatureDirectionKind::_out'
else
    invalid
endif endif
```

- ReferenceUsage::ownedRelationship () : Relationship [0..\*]

```
self.ocAsType(ElementMain_Mapping).ownedRelationship()
->including(MultiplicityMembership_Mapping.getMapped(from))
```

### 7.7.2.3.2.6 ValuePin\_Mapping

#### Description

A UML4SysML::ValuePin is mapped to a SysML v2 ReferenceUsage with assigned value.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
action sysMLv1Action {
    in sysMLv1ValuePin1 : ScalarValues::Integer = 42;
    in sysMLv1ValuePin2 = {
        return result;
        language "English"
        /*
         * this is a opaque expression
         */
    }.result;
}
```

#### General Mappings

Pin\_Mapping

#### Mapping Source

ValuePin

#### Mapping Target

ReferenceUsage

#### Owned Mappings

(none)

#### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
not Helper.excludedPin(src) and not src.type.oclIsUndefined()
```

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceUsage::ownedRelationship () : Relationship [0..*]`  

```
Set{PinFeatureTyping_Mapping.getMapped(from),  
ValuePinFeatureValue_Mapping.getMapped(from),  
MultiplicityMembership_Mapping.getMapped(from)}
```

### 7.7.2.3.2.7 ValuePinFeatureValue\_Mapping

#### Description

The mapping class creates the value expression for the reference usage element.

#### General Mappings

ToFeatureValue\_Init  
Mapping

#### Mapping Source

ValuePin

#### Mapping Target

FeatureValue

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureValue::value () : Expression [1]`  

```
if from.value.oclIsUndefined() then invalid else from.value endif
```

### 7.7.2.3.2.8 ValuePinUntyped\_Mapping

#### Description

Same as ValuePin\_Mapping, but for UML4SysML::ValuePins without a specified type.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```

action sysMLv1Action {
    in sysMLv1ValuePin1 = 42;
}

```

## General Mappings

Pin\_Mapping

## Mapping Source

ValuePin

## Mapping Target

ReferenceUsage

## Owned Mappings

(none)

## Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
not Helper.excludedPin(src) and src.type.oclIsUndefined()
```

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReferenceUsage::ownedRelationship () : Relationship [0..\*]

```

self.oclAsType(Pin_Mapping).ownedRelationship()
->including(ValuePinFeatureValue_Mapping.getMapped(from))

```

### 7.7.2.3.3 Invocation Actions

#### 7.7.2.3.3.1 BroadcastSignalAction\_Mapping

##### Description

The UML4SysML::BroadcastSignalAction is mapped to a SysML v2 ActionUsage. The details of the mapping are not defined yet.

## General Mappings

CommonAction\_Mapping

## Mapping Source

BroadcastSignalAction

## Mapping Target

ActionUsage

### Owned Mappings

(none)

### Applicable filters

(none)

#### 7.7.2.3.3.2 CallBehaviorAction\_Mapping

##### Description

A UML4SysML::CallBehaviorAction is mapped to a SysML v2 ActionUsage.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
action def SysMLv1Activity1 {  
    action sysMLv1CallBehaviorAction : SysMLv1Activity2;  
}  
action def SysMLv1Activity2;
```

### General Mappings

CommonAction\_Mapping

### Mapping Source

CallBehaviorAction

### Mapping Target

ActionUsage

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ActionUsage::ownedRelationship () : Relationship [0..\*]  
  
    Helper.actionOwnedRelationship (from)  
    ->append (CBAFeatureTyping\_Mapping.getMapped (from) )

#### 7.7.2.3.3.3 CBAFeatureTyping\_Mapping

##### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

### General Mappings

ToFeatureTyping\_Init  
Mapping

### Mapping Source

CallBehaviorAction

### Mapping Target

FeatureTyping

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureTyping::type () : Type [1]  
`from.behavior`

#### 7.7.2.3.3.4 CallOperationAction\_Mapping

### Description

A UML4SysML::CallOperationAction is mapped to a SysML v2 ActionUsage which calls the operation.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
action sysMLv1CallOperationAction {  
  in paramIn;  
  in target : ThisIsABlock;  
  out paramReturn = target.sysMLv1Operation;  
}
```

### General Mappings

CommonAction\_Mapping

### Mapping Source

CallOperationAction

### Mapping Target

ActionUsage

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ActionUsage::ownedRelationship () : Relationship [0..\*]

```
Helper.actionOwnedRelationship (from)  
->including (COAPerformActionFeatureMembership_Mapping.getMapped (from))
```

#### 7.7.2.3.3.5 COAOutputPinFeature\_Mapping

### Description

The mapping class creates the feature element for the output parameter.

### General Mappings

ToFeature\_Init  
Mapping

### Mapping Source

OutputPin

### Mapping Target

Feature

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Feature::direction () : FeatureDirectionKind [0..1]

```
KerML::FeatureDirectionKind::_in'
```

- Feature::ownedRelationship () : Relationship [0..\*]

```
Set{COAOutputPinFeatureFeatureValue_Mapping.getMapped(from),
COAOutputPinFeatureFeatureMembership_Mapping.getMapped(from)}
```

#### 7.7.2.3.3.6 COAOutputPinFeatureChainExpression\_Mapping

##### Description

The mapping class creates the feature chain expression for the output parameter feature value.

##### General Mappings

ToInvocationExpression\_Init  
Mapping

##### Mapping Source

OutputPin

##### Mapping Target

FeatureChainExpression

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureChainExpression::ownedRelationship () : Relationship [0..\*]

```
Set{COAOutputPinParameterMembership_Mapping.getMapped(from),
COAOutputPinFeatureChainExpressionMembership_Mapping.getMapped(from),
ReturnParameterFeatureMembership_Factory.create() }
```

#### 7.7.2.3.3.7 COAOutputPinFeatureChainExpressionMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

##### General Mappings

ToMembership\_Init  
Mapping

##### Mapping Source

OutputPin

##### Mapping Target



Membership

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Membership::memberElement () : Element [1]  
`from.owner.oclAsType (UML::CallOperationAction).operation`

#### 7.7.2.3.3.8 COAOutputPinFeatureFeature\_Mapping

##### Description

Creates a feature element for the UML4SysML::CallOperationAction mapping.

##### General Mappings

ToFeature\_Init  
Mapping

##### Mapping Source

OutputPin

##### Mapping Target

Feature

##### Owned Mappings

(none)

##### Applicable filters

(none)

#### 7.7.2.3.3.9 COAOutputPinFeatureFeatureMembership\_Mapping

##### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

##### General Mappings

ToFeatureMembership\_Init  
Mapping

##### Mapping Source

OutputPin

### Mapping Target

FeatureMembership

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureMembership::ownedMemberFeature () : Feature [1]`  
`COAOutputPinFeatureFeature_Mapping.getMapped(from)`

#### 7.7.2.3.3.10 COAOutputPinFeatureFeatureValue\_Mapping

### Description

Creates a feature value relationship.

### General Mappings

ToFeatureValue\_Init  
Mapping

### Mapping Source

OutputPin

### Mapping Target

FeatureValue

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureValue::value () : Expression [1]`

```
COAOutputPinFeatureReferenceExpression_Mapping.getMapped(from)
```

#### 7.7.2.3.3.11 COAOutputPinFeatureMembership\_Mapping

##### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

##### General Mappings

ToFeatureMembership\_Init  
Mapping

##### Mapping Source

OutputPin

##### Mapping Target

FeatureMembership

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureMembership::ownedMemberFeature () : Feature [1]

```
COAOutputPinReferenceUsage_Mapping.getMapped(from)
```

#### 7.7.2.3.3.12 COAOutputPinFeatureReferenceExpression\_Mapping

##### Description

The mapping class creates the feature reference expression for the output parameter.

##### General Mappings

ToFeatureReferenceExpression\_Init  
Mapping

##### Mapping Source

OutputPin

##### Mapping Target

FeatureReferenceExpression

##### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureReferenceExpression::ownedRelationship () : Relationship [0..*]`

```
Set{COAOutputPinFeatureReferenceExpressionMembership_Mapping.getMapped(from),  
ReturnParameterFeatureMembership_Factory.create() }
```

#### 7.7.2.3.3.13 COAOutputPinFeatureReferenceExpressionMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

##### General Mappings

ToMembership\_Init  
Mapping

##### Mapping Source

OutputPin

##### Mapping Target

Membership

##### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Membership::memberElement () : Element [1]`

```
from.owner.oclAsType(UML::CallOperationAction).target
```

#### 7.7.2.3.3.14 COAOutputPinParameterMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

## General Mappings

ToParameterMembership\_Init  
Mapping

## Mapping Source

OutputPin

## Mapping Target

ParameterMembership

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ParameterMembership::visibility () : VisibilityKind [1]  
`KerML::VisibilityKind::private`
- ParameterMembership::ownedMemberParameter () : Feature [1]  
`COAOutputPinFeature_Mapping.getMapped(from)`

### 7.7.2.3.3.15 COAOutputPinReferenceUsage\_Mapping

#### Description

Creates a reference usage.

## General Mappings

ToReferenceUsage\_Init  
Mapping

## Mapping Source

OutputPin

## Mapping Target

ReferenceUsage

## Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceUsage::ownedRelationship () : Relationship [0..*]`

```
Set{COAOutputPinReferenceUsageFeatureValue_Mapping.getMapped(from) }
```

#### 7.7.2.3.3.16 COAOutputPinReferenceUsageFeatureValue\_Mapping

##### Description

Creates a feature value relationship.

##### General Mappings

ToFeatureValue\_Init  
Mapping

##### Mapping Source

OutputPin

##### Mapping Target

FeatureValue

##### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureValue::value () : Expression [1]`

```
COAOutputPinFeatureChainExpression_Mapping.getMapped(from)
```

#### 7.7.2.3.3.17 COAPerformAction\_Mapping

##### Description

The mapping class creates the PerformActionUsage element.

##### General Mappings

ToPerformActionUsage\_Init  
Mapping

#### Mapping Source

CallOperationAction

#### Mapping Target

PerformActionUsage

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- PerformActionUsage::ownedRelationship () : Relationship [0..\*]  
`Set{COAPPerformActionReferenceSubsetting_Mapping.getMapped(from) }`

### 7.7.2.3.3.18 COAPPerformActionFeatureMembership\_Mapping

#### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

#### General Mappings

ToEndFeatureMembership\_Init  
Mapping

#### Mapping Source

CallOperationAction

#### Mapping Target

EndFeatureMembership

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- EndFeatureMembership::ownedMemberFeature () : Feature [1]

`COAPerformAction_Mapping.getMapped(from)`

#### **7.7.2.3.3.19 COAPerformActionReferenceSubsetting\_Mapping**

##### **Description**

Creates a subsetting relationship.

##### **General Mappings**

ToReferenceSubsetting\_Init  
Mapping

##### **Mapping Source**

CallOperationAction

##### **Mapping Target**

ReferenceSubsetting

##### **Owned Mappings**

(none)

##### **Applicable filters**

(none)

##### **Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReferenceSubsetting::ownedRelatedElement () : Element [0..\*]

`Set{COAPerformActionFeature_Mapping.getMapped(from) }`

#### **7.7.2.3.3.20 COAPerformActionFeature\_Mapping**

##### **Description**

The mapping class creates the feature element for the perform action usage.

##### **General Mappings**

ToFeature\_Init  
Mapping

##### **Mapping Source**

CallOperationAction



### Mapping Target

Feature

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Feature::ownedRelationship () : Relationship [0..\*]

```
Set{COAPerformActionFeatureChainingTarget_Mapping.getMapped(from),  
COAPerformActionFeatureChainingOperation_Mapping.getMapped(from)}
```

#### 7.7.2.3.3.21 COAPerformActionFeatureChainingOperation\_Mapping

### Description

The mapping class creates the feature chaining element for the operation of the perform action usage.

### General Mappings

ToFeatureChaining\_Init  
Mapping

### Mapping Source

CallOperationAction

### Mapping Target

FeatureChaining

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureChaining::chainingFeature () : Feature [1]

```
from.operation
```

#### 7.7.2.3.3.22 COAPPerformActionFeatureChainingTarget\_Mapping

##### Description

The mapping class creates the feature chaining element for the target element of the perform action usage.

##### General Mappings

ToFeatureChaining\_Init  
Mapping

##### Mapping Source

CallOperationAction

##### Mapping Target

FeatureChaining

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureChaining::chainingFeature () : Feature [1]  
  
from.target

#### 7.7.2.3.3.23 SendObjectAction\_Mapping

##### Description

A UML4SysML::SendObjectAction is mapped to a SysMLv2 ActionUsage that includes a SendActionUsage.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
action sysMLv1SendObjectAction {  
    in target : SysMLv1Block;  
    send SysMLv1Object1() to target;  
}  
part def SysMLv1Block;  
item def SysMLv1Object;
```

##### General Mappings

SendSignalAction\_Mapping

##### Mapping Source

SendObjectAction

### Mapping Target

ActionUsage

### Owned Mappings

(none)

### Applicable filters

(none)

#### 7.7.2.3.3.24 SendSignalAction\_Mapping

### Description

A UML4SysML::SendSignalAction is mapped to a SysMLv2 ActionUsage that includes a SendActionUsage.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
action sysMLv1SendSignalAction {  
    in target : SysMLv1Block;  
    send SysMLv1Signal() to target;  
}  
part def SysMLv1Block;  
item def SysMLv1Signal;
```

### General Mappings

CommonAction\_Mapping

### Mapping Source

SendSignalAction

### Mapping Target

ActionUsage

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ActionUsage::ownedRelationship () : Relationship [0..\*]

```
Helper.actionOwnedRelationship(from)
->including(SSAFeatureMembership_Mapping.getMapped(from))
```

#### 7.7.2.3.3.25 SSAFeatureMembership\_Mapping

##### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

##### General Mappings

ToFeatureMembership\_Init  
Mapping

##### Mapping Source

InvocationAction

##### Mapping Target

FeatureMembership

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureMembership::ownedMemberFeature () : Feature [1]

```
SSASendActionUsage_Mapping.getMapped(from)
```

#### 7.7.2.3.3.26 SSAParameterMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

##### General Mappings

ToParameterMembership\_Init  
Mapping

##### Mapping Source

InvocationAction

##### Mapping Target

ParameterMembership

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ParameterMembership::ownedMemberParameter () : Feature [1]`  
`SSAReferenceUsage_Mapping.getMapped (from)`

### 7.7.2.3.3.27 SSAReferenceUsage\_Mapping

#### Description

Creates a reference usage.

#### General Mappings

ToReferenceUsage\_Init  
Mapping

#### Mapping Source

InvocationAction

#### Mapping Target

ReferenceUsage

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceUsage::direction () : FeatureDirectionKind [0..1]`  
`KerML::FeatureDirectionKind::_in'`

### 7.7.2.3.3.28 SSALtemParameterMembership\_Mapping

#### Description

Creates a membership relationship for *memberElement()*.

## General Mappings

ToParameterMembership\_Init  
Mapping

## Mapping Source

InvocationAction

## Mapping Target

ParameterMembership

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ParameterMembership::ownedMemberParameter () : Feature [1]  
`SSAItemReferenceUsage_Mapping.getMapped(from)`

### 7.7.2.3.3.29 SSAItemReferenceUsage\_Mapping

## Description

Creates a reference usage.

## General Mappings

ToReferenceUsage\_Init  
Mapping

## Mapping Source

InvocationAction

## Mapping Target

ReferenceUsage

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceUsage::ownedRelationship () : Relationship [0..*]`  
`Set {SSAItemReferenceUsageFeatureValue_Mapping.getMapped (from) }`
- `ReferenceUsage::direction () : FeatureDirectionKind [0..1]`  
`KerML::FeatureDirectionKind::_in'`

### 7.7.2.3.3.30 SSAItemReferenceUsageFeatureValue\_Mapping

#### Description

Creates a feature value relationship.

#### General Mappings

ToFeatureValue\_Init  
Mapping

#### Mapping Source

InvocationAction

#### Mapping Target

FeatureValue

#### Owned Mappings

(none)

#### Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureValue::value () : Expression [1]`  
`SSAItemReferenceUsageInvocationExpression_Mapping.getMapped (from)`

### 7.7.2.3.3.31 SSAItemReferenceUsageFeatureTyping\_Mapping

#### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

#### General Mappings

ToFeatureTyping\_Init  
Mapping

#### Mapping Source

InvocationAction

#### Mapping Target

FeatureTyping

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureTyping::type () : Type [1]  

```
if from.ocIsTypeOf(UML::SendSignalAction) then
    from.signal
else if from.ocIsTypeOf(UML::SendObjectAction) then
    from.request
else
    invalid
endif endif
```

#### 7.7.2.3.3.32 SSItemReferenceUsageInvocationExpression\_Mapping

##### Description

The mapping class creates the invocation expression for the SysML v2 SendActionUsage.

##### General Mappings

ToInvocationExpression\_Init  
Mapping

#### Mapping Source

InvocationAction

#### Mapping Target

InvocationExpression

#### Owned Mappings

(none)

#### Applicable filters



(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `InvocationExpression::ownedRelationship () : Relationship [0..*]`  
`Set{SSAItemReferenceUsageFeatureTyping_Mapping.getMapped(from) ,`  
`ReturnParameterFeatureMembership_Factory.create() }`

#### 7.7.2.3.33 SSATargetParameterMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

##### General Mappings

ToParameterMembership\_Init  
Mapping

##### Mapping Source

InvocationAction

##### Mapping Target

ParameterMembership

##### Owned Mappings

(none)

##### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ParameterMembership::ownedMemberParameter () : Feature [1]`  
`SSATargetReferenceUsage_Mapping.getMapped(from)`

#### 7.7.2.3.34 SSATargetReferenceUsage\_Mapping

##### Description

Creates a reference usage.

##### General Mappings

ToReferenceUsage\_Init  
Mapping

### Mapping Source

InvocationAction

### Mapping Target

ReferenceUsage

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReferenceUsage::direction () : FeatureDirectionKind [0..1]  
`KerML::FeatureDirectionKind::_in'`
- ReferenceUsage::ownedRelationship () : Relationship [0..\*]  
`Set{SSATargetReferenceUsageFeatureValue_Mapping.getMapped(from) }`

#### 7.7.2.3.3.35 SSATargetReferenceUsageFeatureValue\_Mapping

### Description

Creates a feature value relationship.

### General Mappings

ToFeatureValue\_Init  
Mapping

### Mapping Source

InvocationAction

### Mapping Target

FeatureValue

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureValue::value () : Expression [1]`

`SSATargetReferenceUsageFeatureValueExpression_Mapping.getMapped(from)`

#### 7.7.2.3.36 SSATargetReferenceUsageFeatureValueMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

##### General Mappings

ToMembership\_Init  
Mapping

##### Mapping Source

InvocationAction

##### Mapping Target

Membership

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Membership::memberElement () : Element [1]`

`from.target`

#### 7.7.2.3.37 SSATargetReferenceUsageFeatureValueExpression\_Mapping

##### Description

The mapping class creates the feature reference expression for the target reference usage element of the SysML v2 `SendActionUsage`.

##### General Mappings

ToFeatureReferenceExpression\_Init  
Mapping

##### Mapping Source

InvocationAction

### Mapping Target

FeatureReferenceExpression

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureReferenceExpression::ownedRelationship () : Relationship [0..\*]

```
Set{SSATargetReferenceUsageFeatureValueMembership_Mapping.getMapped(from) ,  
ReturnParameterFeatureMembership_Factory.create() }
```

#### 7.7.2.3.3.38 SSASendActionUsage\_Mapping

### Description

The mapping class creates the SysML v2 element SendActionUsage for the UML4SysML::SendSignalAction mapping.

### General Mappings

ToActionUsage\_Init  
Mapping

### Mapping Source

InvocationAction

### Mapping Target

SendActionUsage

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- SendActionUsage::ownedRelationship () : Relationship [0..\*]

```
Set{SSAItemParameterMembership_Mapping.getMapped(from),  
SSAParameterMembership_Mapping.getMapped(from),  
SSATargetParameterMembership_Mapping.getMapped(from)}
```

#### **7.7.2.3.3.39 StartClassifierBehaviorAction\_Mapping**

##### **Description**

The UML4SysML::StartClassifierBehaviorAction is not supported by SysML v2. It is mapped to an empty action usage to keep the connections within the activity respectively action definition.

##### **General Mappings**

CommonAction\_Mapping

##### **Mapping Source**

StartClassifierBehaviorAction

##### **Mapping Target**

ActionUsage

##### **Owned Mappings**

(none)

##### **Applicable filters**

(none)

#### **7.7.2.3.3.40 StartObjectBehaviorAction\_Mapping**

##### **Description**

The UML4SysML::StartObjectBehaviorAction is not supported by SysML v2. It is mapped to an empty action usage to keep the connections within the activity respectively action definition.

##### **General Mappings**

CommonAction\_Mapping

##### **Mapping Source**

StartObjectBehaviorAction

##### **Mapping Target**

ActionUsage

##### **Owned Mappings**

(none)

##### **Applicable filters**

(none)

#### **7.7.2.3.4 Link Actions**

##### **7.7.2.3.4.1 ClearAssociationAction\_Mapping**

###### **Description**

The UML4SysML::ClearAssociationAction is mapped to a SysML v2 ActionUsage. The details of the mapping are not defined yet.

###### **General Mappings**

CommonAction\_Mapping

###### **Mapping Source**

ClearAssociationAction

###### **Mapping Target**

ActionUsage

###### **Owned Mappings**

(none)

###### **Applicable filters**

(none)

##### **7.7.2.3.4.2 CreateLinkAction\_Mapping**

###### **Description**

The UML4SysML::CreateLinkAction is mapped to a SysML v2 ActionUsage. The details of the mapping are not completely defined yet.

###### **General Mappings**

CommonAction\_Mapping

###### **Mapping Source**

CreateLinkAction

###### **Mapping Target**

ActionUsage

###### **Owned Mappings**

(none)

###### **Applicable filters**

(none)

###### **Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ActionUsage::ownedRelationship () : Relationship [0..\*]

```
let linkEndCreationData : Set(UML::Element) =
    from.ownedElement->select(e | e.ocliIsTypeOf(UML::LinkEndCreationData)) in
let actionInputPin: Set(UML::Element) =
    from.ownedElement->select(e | e.ocliIsTypeOf(UML::ActionInputPin)) in
let triggers: Set(UML::Element) =
    from.ownedElement->select(e | e.ocliIsKindOf(UML::Trigger)) in
let toElementFMS: Set(UML::Element) =
    from.ownedElement->select(e | e.ocliIsKindOf(UML::Pin)) in
let toElementOMS: Set(UML::Element) =
    (((from.ownedElement - toElementFMS) - actionInputPin)
    - triggers) - linkEndCreationData) in
toElementOMS->collect(e | ElementOwningMembership_Mapping.getMapped(e))
->union(toElementFMS->collect(e | ElementFeatureMembership_Mapping.getMapped(e)))
```

#### 7.7.2.3.4.3 CreateLinkObjectAction\_Mapping

##### Description

A UML4SysML::CreateLinkObjectAction is mapped to a SysML v2 ActionUsage. The details of the mapping are not defined yet.

##### General Mappings

CreateLinkAction\_Mapping

##### Mapping Source

CreateLinkObjectAction

##### Mapping Target

ActionUsage

##### Owned Mappings

(none)

##### Applicable filters

(none)

#### 7.7.2.3.4.4 DestroyLinkAction\_Mapping

##### Description

The UML4SysML::DestroyLinkAction is mapped to a SysML v2 ActionUsage. The details of the mapping are not completely defined yet.

##### General Mappings

CommonAction\_Mapping

##### Mapping Source

DestroyLinkAction

### Mapping Target

ActionUsage

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ActionUsage::ownedRelationship () : Relationship [0..\*]

```
let actionInputPin: Set(UML::Element) =
  from.ownedElement->select(e | e.ocIsTypeOf(UML::ActionInputPin)) in
let triggers: Set(UML::Element) =
  from.ownedElement->select(e | e.ocIsKindOf(UML::Trigger)) in
let linkData: Set(UML::Element) =
  from.ownedElement->select( e | e.ocIsKindOf(UML::LinkEndData) or
  e.ocIsKindOf(UML::LinkEndDestructionData)) in
let toElementFMS: Set(UML::Element) =
  from.ownedElement->select(e | e.ocIsKindOf(UML::Pin)) in
let toElementOMS: Set(UML::Element) =
  (((from.ownedElement - toElementFMS) - actionInputPin)
  - triggers) - linkData) in
toElementOMS->collect(e | ElementOwningMembership_Mapping.getMapped(e))
->union(toElementFMS->collect(e | ElementFeatureMembership_Mapping.getMapped(e)))
```

#### 7.7.2.3.4.5 ReadLinkAction\_Mapping

##### Description

The UML4SysML::ReadLinkAction is mapped to a SysML v2 ActionUsage. The details of the mapping are not completely defined yet.

##### General Mappings

CommonAction\_Mapping

##### Mapping Source

ReadLinkAction

##### Mapping Target

ActionUsage

##### Owned Mappings

(none)



## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ActionUsage::ownedRelationship () : Relationship [0..\*]

```
let actionInputPin: Set(UML::Element) =
    from.ownedElement->select(e | e.ocIsTypeOf(UML::ActionInputPin)) in
let triggers: Set(UML::Element) =
    from.ownedElement->select(e | e.ocIsKindOf(UML::Trigger)) in
let linkData: Set(UML::Element) =
    from.ownedElement->select(e | e.ocIsKindOf(UML::LinkEndData)) in
let toElementFMS: Set(UML::Element) =
    from.ownedElement->select(e | e.ocIsKindOf(UML::Pin)) in
let toElementOMS: Set(UML::Element) =
    (((from.ownedElement - toElementFMS) - actionInputPin)
    - triggers) - linkData in
toElementOMS->collect(e | ElementOwningMembership_Mapping.getMapped(e))
->union(toElementFMS->collect(e | ElementFeatureMembership_Mapping.getMapped(e)))
```

### 7.7.2.3.4.6 ReadLinkObjectEndAction\_Mapping

#### Description

The UML4SysML::ReadLinkObjectEndAction is mapped to a SysML v2 ActionUsage. The details of the mapping are not defined yet.

#### General Mappings

CommonAction\_Mapping

#### Mapping Source

ReadLinkObjectEndAction

#### Mapping Target

ActionUsage

#### Owned Mappings

(none)

## Applicable filters

(none)

### 7.7.2.3.4.7 ReadLinkObjectEndQualifierAction\_Mapping

#### Description

The UML4SysML::ReadLinkObjectEndQualifierAction is mapped to a SysML v2 ActionUsage. The details of the mapping are not defined yet.

## General Mappings

CommonAction\_Mapping

## Mapping Source

ReadLinkObjectEndQualifierAction

## Mapping Target

ActionUsage

## Owned Mappings

(none)

## Applicable filters

(none)

### 7.7.2.3.5 Object Actions

#### 7.7.2.3.5.1 CreateObjectAction\_Mapping

##### Description

A UML4SysML::CreateObjectAction is mapped to a SysML v2 ActionUsage.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
action def SysMLv1Activity {  
    action sysMLv1CreateObjectAction {  
        out result : SysMLv1Block = SysMLv1Block();  
    }  
}  
part def SysMLv1Block;
```

## General Mappings

CommonAction\_Mapping

## Mapping Source

CreateObjectAction

## Mapping Target

ActionUsage

## Owned Mappings

(none)

## Applicable filters

(none)

#### 7.7.2.3.5.2 COAInvocationExpressionFeatureTyping\_Mapping

##### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

##### General Mappings

ToFeatureTyping\_Init  
Mapping

##### Mapping Source

CreateObjectAction

##### Mapping Target

FeatureTyping

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureTyping::type () : Type [1]  
`from.classifier`

#### 7.7.2.3.5.3 COAInvocationExpression\_Mapping

##### Description

The mapping class creates the invocation expression to create the object.

##### General Mappings

ToInvocationExpression\_Init  
Mapping

##### Mapping Source

CreateObjectAction

##### Mapping Target

InvocationExpression

##### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `InvocationExpression::ownedRelationship () : Relationship [0..*]`  
`Set{COAInvocationExpressionFeatureTyping_Mapping.getMapped(from),`  
`CommonReturnParameterFeatureMembership_Mapping.getMapped(from,result) }`

#### 7.7.2.3.5.4 COAPin\_Mapping

##### Description

The mapping class creates the output parameter of the `ActionUsage` for the mapping of `UML4SysML::CreateObjectAction`.

##### General Mappings

Pin\_Mapping

##### Mapping Source

OutputPin

##### Mapping Target

ReferenceUsage

##### Owned Mappings

(none)

### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation `filter(src : Element) : Boolean` is verified:

```
src.owner.oclIsTypeOf(UML::CreateObjectAction)
```

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceUsage::ownedRelationship () : Relationship [0..*]`  
`Set{PinFeatureTyping_Mapping.getMapped(from),`  
`COAPinFeatureValue_Mapping.getMapped(from) }`

#### 7.7.2.3.5.5 COAPinFeatureValue\_Mapping

##### Description

Creates a feature value relationship.

### General Mappings

ToFeatureValue\_Init  
Mapping

### Mapping Source

OutputPin

### Mapping Target

FeatureValue

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureValue::value () : Expression [1]  
`COAInvocationExpression_Mapping.getMapped(from.owner)`

#### 7.7.2.3.5.6 DestroyObjectAction\_Mapping

##### Description

The UML4SysML::DestroyObjectAction is conceptually mapped to the SysML v2 library function OccurrenceFunctions::destroy.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
action def SysMLv1Activity {  
    action sysMLv1DestroyObjectAction {  
        in target : SysMLv1Block;  
        action : OccurrenceFunctions::destroy {  
            in occ = target;  
        }  
    }  
}  
part def SysMLv1Block;
```

### General Mappings

CommonAction\_Mapping

### Mapping Source

DestroyObjectAction

### Mapping Target

ActionUsage

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ActionUsage::ownedRelationship () : Relationship [0..\*]  
  

```
Helper.actionOwnedRelationship (from)  
->including (DOADestroyFeatureMembership_Mapping.getMapped (from))
```

#### 7.7.2.3.5.7 DOADestroyActionUsage\_Mapping

### Description

The mapping class creates the action usage for the destroy function.

### General Mappings

ToActionUsage\_Init  
Mapping

### Mapping Source

DestroyObjectAction

### Mapping Target

ActionUsage

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ActionUsage::ownedRelationship () : Relationship [0..\*]

```
Set{DOADestroyActionUsageFeatureTyping_Mapping.getMapped(from),  
DOADestroyActionUsageFeatureMembership_Mapping.getMapped(from)}
```

#### 7.7.2.3.5.8 DOADestroyActionUsageFeatureMembership\_Mapping

##### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

##### General Mappings

ToFeatureMembership\_Init  
Mapping

##### Mapping Source

DestroyObjectAction

##### Mapping Target

FeatureMembership

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureMembership::ownedMemberFeature () : Feature [1]

```
DOADestroyActionUsageReferenceUsage_Mapping.getMapped(from)
```

#### 7.7.2.3.5.9 DOADestroyActionUsageFeatureReferenceExpression\_Mapping

##### Description

The mapping class creates the feature reference expression for the UML4SysML::DestroyObjectAction mapping.

##### General Mappings

ToFeatureReferenceExpression\_Init  
Mapping

##### Mapping Source

DestroyObjectAction

##### Mapping Target

FeatureReferenceExpression

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureReferenceExpression::ownedRelationship () : Relationship [0..*]`  
`Set{DOADestroyActionUsageMembership_Mapping.getMapped(from) ,`  
`ReturnParameterFeatureMembership_Factory.create() }`

### 7.7.2.3.5.10 DOADestroyActionUsageMembership\_Mapping

#### Description

Creates a membership relationship for *memberElement()*.

#### General Mappings

ToMembership\_Init  
Mapping

#### Mapping Source

DestroyObjectAction

#### Mapping Target

Membership

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Membership::memberElement () : Element [1]`  
`from.target`

### 7.7.2.3.5.11 DOADestroyActionUsageFeatureTyping\_Mapping

#### Description



Creates a feature typing relationship owned by the element *typedFeature()*.

### General Mappings

ToFeatureTyping\_Init  
Mapping

### Mapping Source

DestroyObjectAction

### Mapping Target

FeatureTyping

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureTyping::type () : Type [1]  

```
SysMLv2::Function.allInstances(  
  )->any(e | e.qualifiedName = 'OccurrenceFunctions::destroy')
```

#### 7.7.2.3.5.12 DOADestroyActionUsageFeatureValue\_Mapping

### Description

Creates a feature value relationship.

### General Mappings

ToFeatureValue\_Init  
Mapping

### Mapping Source

DestroyObjectAction

### Mapping Target

FeatureValue

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureValue::value () : Expression [1]`

`DOADestroyActionUsageFeatureReferenceExpression_Mapping.getMapped(from)`

#### 7.7.2.3.5.13 DOADestroyActionUsageReferenceUsage\_Mapping

##### Description

Creates a reference usage.

##### General Mappings

ToReferenceUsage\_Init  
Mapping

##### Mapping Source

DestroyObjectAction

##### Mapping Target

ReferenceUsage

##### Owned Mappings

(none)

##### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceUsage::ownedRelationship () : Relationship [0..*]`

`Set{DOADestroyActionUsageFeatureValue_Mapping.getMapped(from) }`

#### 7.7.2.3.5.14 DOADestroyFeatureMembership\_Mapping

##### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

##### General Mappings

ToFeatureMembership\_Init  
Mapping

## Mapping Source

DestroyObjectAction

## Mapping Target

FeatureMembership

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureMembership::ownedMemberFeature () : Feature [1]  
DOADestroyActionUsage\_Mapping.getMapped(from)

### 7.7.2.3.5.15 ReadIsClassifiedObjectAction\_Mapping

#### Description

The UML4SysML::ReadIsClassifiedObjectAction is conceptually mapped to a SysML v2 ActionUsage.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
action def SysMLv1Activity {  
  action sysMLv1ReadIsClassifiedObjectActionDirect {  
    in object;  
    out result : ScalarValues::Boolean =  
      object istype ThisIsABlock;  
  }  
  
  action sysMLv1ReadIsClassifiedObjectActionNonDirect {  
    in object;  
    out result : ScalarValues::Boolean =  
      object hastype ThisIsABlock;  
  }  
}
```

## General Mappings

CommonAction\_Mapping

## Mapping Source

ReadIsClassifiedObjectAction

## Mapping Target

ActionUsage

### Owned Mappings

(none)

### Applicable filters

(none)

#### 7.7.2.3.5.16 RICOAFeatureValue\_Mapping

##### Description

Creates a feature value relationship.

##### General Mappings

ToFeatureValue\_Init  
Mapping

##### Mapping Source

ReadIsClassifiedObjectAction

##### Mapping Target

FeatureValue

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureValue::value () : Expression [1]

```
RICOAFeatureValueOperatorExpression_Mapping.getMapped(from)
```

#### 7.7.2.3.5.17 RICOAFeatureValueOperatorExpression\_Mapping

##### Description

The mapping class creates the operator expression for the UML4SysML::ReadIsClassifiedObjectAction mapping.

##### General Mappings

ToOperatorExpression\_Init  
Mapping

##### Mapping Source

ReadIsClassifiedObjectAction

### Mapping Target

OperatorExpression

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- OperatorExpression::operator () : String [1]  

```
if from.isDirect then 'istype' else 'hastype' endif
```
- OperatorExpression::ownedRelationship () : Relationship [0..\*]  

```
Set{RICOAFeatureValueOperatorParameterMembership_Mapping.getMapped(from) }
```

#### 7.7.2.3.5.18 RICOAFeatureValueOperatorExpressionFeature\_Mapping

### Description

The mapping class creates the feature for the operator expression of the UML4SysML::ReadIsClassifiedObjectAction mapping.

### General Mappings

ToFeature\_Init  
Mapping

### Mapping Source

ReadIsClassifiedObjectAction

### Mapping Target

Feature

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Feature::direction () : FeatureDirectionKind [0..1]`  
`KerML::FeatureDirectionKind::_in'`
- `Feature::ownedRelationship () : Relationship [0..*]`  
`Set{RICOAFeatureValueOperatorExpressionFeatureValue_Mapping.getMapped(from)}`

#### 7.7.2.3.5.19 RICOAFeatureValueOperatorExpressionFeatureValue\_Mapping

##### Description

Creates a feature value relationship.

##### General Mappings

ToFeatureValue\_Init  
Mapping

##### Mapping Source

ReadIsClassifiedObjectAction

##### Mapping Target

FeatureValue

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureValue::value () : Expression [1]`  
`RICOAFeatureValueOperatorFeatureReferenceExpression_Mapping.getMapped(from)`

#### 7.7.2.3.5.20 RICOAFeatureValueOperatorFeatureReferenceExpression\_Mapping

##### Description

The mapping class creates the feature reference expression for the UML4SysML::ReadIsClassifiedObjectAction mapping.

##### General Mappings

ToFeatureReferenceExpression\_Init  
Mapping

### Mapping Source

ReadIsClassifiedObjectAction

### Mapping Target

FeatureReferenceExpression

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureReferenceExpression::ownedRelationship () : Relationship [0..*]`  
`Set { RICOAFeatureValueOperatorMembership_Mapping.getMapped (from) ,  
CommonReturnParameterFeatureMembership_Mapping.getMapped (from) }`

#### 7.7.2.3.5.21 RICOAFeatureValueOperatorMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

##### General Mappings

ToMembership\_Init  
Mapping

##### Mapping Source

ReadIsClassifiedObjectAction

##### Mapping Target

Membership

##### Owned Mappings

(none)

##### Applicable filters

(none)

#### 7.7.2.3.5.22 RICOAFeatureValueOperatorParameterMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

## General Mappings

ToParameterMembership\_Init  
Mapping

## Mapping Source

ReadIsClassifiedObjectAction

## Mapping Target

ParameterMembership

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ParameterMembership::visibility () : VisibilityKind [1]  
`KerML::VisibilityKind::private`
- ParameterMembership::ownedMemberParameter () : Feature [1]  
`RICOAFeatureValueOperatorExpressionFeature_Mapping.getMapped(from)`

### 7.7.2.3.5.23 RICOAOutputPin\_Mapping

#### Description

The mapping class creates the output parameter of the ActionUsage element for the UML4SysML::ReadIsClassifiedObjectAction mapping.

## General Mappings

Pin\_Mapping

## Mapping Source

OutputPin

## Mapping Target

ReferenceUsage

## Owned Mappings

(none)



## Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
src.owner.oclIsTypeOf(UML::ReadIsClassifiedObjectAction)
```

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReferenceUsage::ownedRelationship () : Relationship [0..\*]  

```
Set{TypedElementFeatureTyping_Mapping.getMapped(from),  
RICOAFeatureValue_Mapping.getMapped(from.owner),  
MultiplicityMembership_Mapping.getMapped(from)}
```

### 7.7.2.3.5.24 ReadExtentAction\_Mapping

#### Description

A UML4SysML::ReadExtentAction is mapped to a SysML v2 ActionUsage.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
action def SysMLv1Activity {  
    action sysMLv1ReadExtentAction {  
        out thisIsTheOutputPin : SysMLv1Block =  
            all SysMLv1Block;  
    }  
}  
part def SysMLv1Block;
```

#### General Mappings

CommonAction\_Mapping

#### Mapping Source

ReadExtentAction

#### Mapping Target

ActionUsage

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ActionUsage::ownedRelationship () : Relationship [0..*]`

`Helper.actionOwnedRelationship (from)`

#### **7.7.2.3.5.25 REAFeatureValue\_Mapping**

##### **Description**

Creates a feature value relationship.

##### **General Mappings**

`ToFeatureValue_Init`  
`Mapping`

##### **Mapping Source**

`OutputPin`

##### **Mapping Target**

`FeatureValue`

##### **Owned Mappings**

(none)

##### **Applicable filters**

(none)

##### **Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureValue::value () : Expression [1]`

`REAFeatureValueOperatorExpression_Mapping.getMapped (from)`

#### **7.7.2.3.5.26 REAFeatureValueOperatorExpression\_Mapping**

##### **Description**

The mapping class creates the operator expression for the `UML4SysML::ReadExtentAction` mapping.

##### **General Mappings**

`ToOperatorExpression_Init`  
`Mapping`

##### **Mapping Source**

`OutputPin`

### Mapping Target

OperatorExpression

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- OperatorExpression::ownedRelationship () : Relationship [0..\*]  

```
Set{REAFeatureValueOperatorExpressionMembership_Mapping.getMapped(from) ,  
CommonReturnParameterFeatureMembership_Mapping.getMapped(from) }
```
- OperatorExpression::operator () : String [1]  

```
'all'
```

#### 7.7.2.3.5.27 REAFeatureValueOperatorExpressionFeature\_Mapping

### Description

The mapping class creates the feature for the operator expression for the UML4SysML::ReadExtentAction mapping.

### General Mappings

ToFeature\_Init  
Mapping

### Mapping Source

OutputPin

### Mapping Target

Feature

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Feature::ownedRelationship () : Relationship [0..*]`

```
Set{REAFeatureValueOperatorExpressionFeatureTyping_Mapping.getMapped(from)}
```

#### 7.7.2.3.5.28 REAFeatureValueOperatorExpressionFeatureTyping\_Mapping

##### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

##### General Mappings

ToFeatureTyping\_Init  
Mapping

##### Mapping Source

OutputPin

##### Mapping Target

FeatureTyping

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureTyping::type () : Type [1]`

```
from.owner.classifier
```

#### 7.7.2.3.5.29 REAFeatureValueOperatorExpressionMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

##### General Mappings

ToFeatureMembership\_Init  
Mapping

##### Mapping Source

OutputPin

### Mapping Target

FeatureMembership

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureMembership::ownedMemberFeature () : Feature [1]  
`REAFeatureValueOperatorExpressionFeature_Mapping.getMapped(from)`

#### 7.7.2.3.5.30 REAOutputPin\_Mapping

### Description

The mapping class creates the output parameter of the ActionUsage for the mapping of UML4SysML::ReadExtentAction.

### General Mappings

Pin\_Mapping

### Mapping Source

OutputPin

### Mapping Target

ReferenceUsage

### Owned Mappings

(none)

### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
src.owner.oclIsTypeOf(UML::ReadExtentAction)
```

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReferenceUsage::ownedRelationship () : Relationship [0..\*]

```

Set {TypedElementFeatureTyping_Mapping.getMapped(from),
    REAFeatureValue_Mapping.getMapped(from)}
->union(self.oclAsType(Pin_Mapping).ownedRelationship())

```

#### 7.7.2.3.5.31 ReadSelfAction\_Mapping

##### Description

A UML4SysML::ReadSelfAction is mapped to a SysML v2 ActionUsage.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```

action def SysMLv1Activity {
    action sysMLv1ReadSelfAction {
        out : Base::Anything = this;
    }
}

```

##### General Mappings

CommonAction\_Mapping

##### Mapping Source

ReadSelfAction

##### Mapping Target

ActionUsage

##### Owned Mappings

(none)

##### Applicable filters

(none)

#### 7.7.2.3.5.32 RSAFeatureValue\_Mapping

##### Description

Creates a feature value relationship.

##### General Mappings

ToFeatureValue\_Init  
Mapping

##### Mapping Source

OutputPin

##### Mapping Target

FeatureValue

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureValue::value () : Expression [1]`  
`RSAFeatureValueFeatureReferenceExpression_Mapping.getMapped (from)`

### 7.7.2.3.5.33 RSAFeatureValueFeatureReferenceExpression\_Mapping

#### Description

The mapping class creates the feature reference expression for the mapping of UML4SysML::ReadSelfAction.

#### General Mappings

ToFeatureReferenceExpression\_Init  
Mapping

#### Mapping Source

OutputPin

#### Mapping Target

FeatureReferenceExpression

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureReferenceExpression::ownedRelationship () : Relationship [0..*]`  
`Set {RSAFeatureValueMembership_Mapping.getMapped (from) ,`  
`CommonReturnParameterFeatureMembership_Mapping.getMapped (from) }`

### 7.7.2.3.5.34 RSAFeatureValueMembership\_Mapping

#### Description

Creates a membership relationship for *memberElement()*.

### General Mappings

ToMembership\_Init  
Mapping

### Mapping Source

OutputPin

### Mapping Target

Membership

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Membership::memberElement () : Element [1]

```
SYSML2::Feature.allInstances()  
->any(e | e.qualifiedName = 'Occurrences::Occurrence::this')
```

#### 7.7.2.3.5.35 RSAOutputPin\_Mapping

### Description

The mapping class creates the output parameter of the ActionUsage for the mapping of UML4SysML::ReadSelfAction.

### General Mappings

Pin\_Mapping

### Mapping Source

OutputPin

### Mapping Target

ReferenceUsage

### Owned Mappings

(none)

### Applicable filters



This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
src.owner.oclIsKindOf(UML::ReadSelfAction)
```

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReferenceUsage::ownedRelationship () : Relationship [0..\*]  

```
Set{TypedElementFeatureTyping_Mapping.getMapped(from),  
RSAFeatureValue_Mapping.getMapped(from)}  
->union(self.oclAsType(Pin_Mapping).ownedRelationship())
```
- ReferenceUsage::isUnique () : Boolean [1]  

```
false
```
- ReferenceUsage::isAbstract () : Boolean [1]  

```
true
```

#### 7.7.2.3.5.36 ReclassifyObjectAction\_Mapping

##### Description

The UML4SysML::ReclassifyObjectAction is not supported by SysML v2. It is mapped to an empty action usage to keep the connections within the activity respectively action definition.

##### General Mappings

CommonAction\_Mapping

##### Mapping Source

ReclassifyObjectAction

##### Mapping Target

ActionUsage

##### Owned Mappings

(none)

##### Applicable filters

(none)

#### 7.7.2.3.5.37 TestIdentityAction\_Mapping

##### Description

A UML4SysML::TestIdentityAction is mapped to a SysML v2 ActionUsage.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```

action def SysMLv1Activity {
    action sysMLv1TestIdentityAction {
        in firstParameter;
        in secondParameter;
        out result : ScalarValues::Boolean =
            firstParameter == secondParameter;
    }
}

```

## General Mappings

CommonAction\_Mapping

## Mapping Source

TestIdentityAction

## Mapping Target

CalculationUsage

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- CalculationUsage::ownedRelationship () : Relationship [0..\*]

```

Helper.actionOwnedRelationship (from)
->including (TIAResultExpressionMembership_Mapping.getMapped (from))

```

### 7.7.2.3.5.38 TIAOperatorExpression\_Mapping

## Description

The mapping class creates the operator expression for the UML4SysML::TestIdentityAction mapping.

## General Mappings

ToOperatorExpression\_Init  
Mapping

## Mapping Source

TestIdentityAction

## Mapping Target

OperatorExpression

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `OperatorExpression::operator () : String [1]`  
`'=='`
- `OperatorExpression::ownedRelationship () : Relationship [0..*]`  
`Set{EqualOperatorExpressionOperandParameterMembership_Mapping.getMapped(from.first),`  
`EqualOperatorExpressionOperandParameterMembership_Mapping.getMapped(from.second),`  
`CommonReturnParameterFeatureMembership_Mapping.getMapped(from.result) }`

### 7.7.2.3.5.39 TIAResultExpressionMembership\_Mapping

#### Description

Creates a membership relationship for *memberElement()*.

#### General Mappings

ToFeatureMembership\_Init  
Mapping

#### Mapping Source

TestIdentityAction

#### Mapping Target

ResultExpressionMembership

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ResultExpressionMembership::ownedMemberFeature () : Feature [0..1]`  
`TIAOperatorExpression_Mapping.getMapped(from)`

#### 7.7.2.3.5.40 ValueSpecificationAction\_Mapping

##### Description

A UML4SysML::ValueSpecificationAction is mapped to a SysML v2 ActionUsage.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
action def SysMLv1Acticity {
  action sysMLv1ValueSpecificationAction1 {
    out result : ScalarValues::Integer = 42;
  }

  action sysMLv1ValueSpecificationAction2 {
    out result = sysMLv1OpaqueExpression.result;
    calc sysMLv1OpaqueExpression {
      language "Math"
      /*
      * 42 + 23
      */
    }
  }
}
```

##### General Mappings

CommonAction\_Mapping

##### Mapping Source

ValueSpecificationAction

##### Mapping Target

ActionUsage

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ActionUsage::ownedRelationship () : Relationship [0..\*]

```
let toElementFMS: Set(UML::Element) =
  from.ownedElement->select(e | e.ocIsKindOf(UML::Pin)) in
let toElementOMS: Set(UML::Element) =
  (from.ownedElement - toElementFMS) - Set{from.value} in
toElementFMS->collect(e | ElementFeatureMembership_Mapping.getMapped(e))
->union(toElementOMS->collect(e | ElementOwningMembership_Mapping.getMapped(e)))
```

#### 7.7.2.3.5.41 VSAOutputPin\_Mapping

##### Description

The mapping class creates the output parameter of the ActionUsage for the mapping of UML4SysML::ValueSpecificationAction.

##### General Mappings

Pin\_Mapping

##### Mapping Source

OutputPin

##### Mapping Target

ReferenceUsage

##### Owned Mappings

(none)

##### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
src.owner.ocIsKindOf(UML::ValueSpecificationAction)
```

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReferenceUsage::ownedRelationship () : Relationship [0..\*]

```
let relationships : Set(KerML::Relationship) =  
  self.ocIsType(Pin_Mapping).ownedRelationship()  
  ->including(VSAOutputPinFeatureValue_Mapping.getMapped(from)) in  
if from.type.ocIsUndefined() then  
  relationships  
else  
  relationships  
  ->including(TypedElementFeatureTyping_Mapping.getMapped(from))  
endif
```

#### 7.7.2.3.5.42 VSAOutputPinFeatureValue\_Mapping

##### Description

Creates a feature value relationship.

##### General Mappings

ToFeatureValue\_Init  
Mapping

### Mapping Source

OutputPin

### Mapping Target

FeatureValue

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureValue::value () : Expression [1]  

```
if from.owner.value.oclIsTypeOf(UML::OpaqueExpression) then
    OpaqueExpressionAsValue_Mapping.getMapped(from.owner.value)
else
    from.owner.value
endif
```

## 7.7.2.3.6 Other Actions

### 7.7.2.3.6.1 RaiseExceptionAction\_Mapping

#### Description

The UML4SysML::RaiseExceptionAction is mapped to a SysML v2 ActionUsage. The details of the mapping are not defined yet.

#### General Mappings

CommonAction\_Mapping

### Mapping Source

RaiseExceptionAction

### Mapping Target

ActionUsage

### Owned Mappings

(none)

### Applicable filters

(none)

#### 7.7.2.3.6.2 ReduceAction\_Mapping

##### Description

The UML4SysML::ReduceAction is mapped to a SysML v2 ActionUsage. The details of the mapping are not defined yet.

##### General Mappings

CommonAction\_Mapping

##### Mapping Source

ReduceAction

##### Mapping Target

ActionUsage

##### Owned Mappings

(none)

##### Applicable filters

(none)

#### 7.7.2.3.7 Structural Feature Actions

##### 7.7.2.3.7.1 AddStructuralFeatureValueAction\_Mapping

##### Description

A UML4SysML::AddStructuralFeatureValueAction is mapped to a SysML v2 ActionUsage defined by the SysML v1 library action definition SysMLv1Library::AddStructuralFeatureValueAction.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
action thisIsAAddStructuralFeatureValueAction
: SysMLv1Library::AddStructuralFeatureValueAction {
  :>> target := object.thisIsAnAttribute;
  :>> object : ThisIsABlock;
}
part def SysMLv1Block {
  attribute sysMLv1Property;
}
```

##### General Mappings

CommonAction\_Mapping

##### Mapping Source

AddStructuralFeatureValueAction

##### Mapping Target

ActionUsage

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ActionUsage::ownedRelationship () : Relationship [0..\*]  

```
Set{ASFVAFeatureTyping_Mapping.getMapped(from),  
ASFVATargetFeatureMembership_Mapping.getMapped(from),  
ASFVAObjectFeatureMembership_Mapping.getMapped(from)}
```

#### 7.7.2.3.7.2 ASFVAFeatureTyping\_Mapping

### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

### General Mappings

ToFeatureTyping\_Init  
Mapping

### Mapping Source

AddStructuralFeatureValueAction

### Mapping Target

FeatureTyping

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureTyping::type () : Type [1]  

```
SYSML2::ActionDefinition.allInstances()  
->any(m | m.qualifiedName = 'SysMLv1Library::AddStructuralFeatureValueAction')
```



#### 7.7.2.3.7.3 ASFVAObjectFeatureMembership\_Mapping

##### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

##### General Mappings

ToFeatureMembership\_Init  
Mapping

##### Mapping Source

AddStructuralFeatureValueAction

##### Mapping Target

FeatureMembership

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureMembership::ownedMemberFeature () : Feature [1]  
`ASFVAObjectReferenceUsage_Mapping.getMapped (from)`

#### 7.7.2.3.7.4 ASFVAObjectReferenceUsage\_Mapping

##### Description

Creates a reference usage.

##### General Mappings

UniqueMapping  
ToReferenceUsage\_Init

##### Mapping Source

AddStructuralFeatureValueAction

##### Mapping Target

ReferenceUsage

##### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceUsage::ownedRelationship () : Relationship [0..*]`

```
Set{ASFVAObjectReferenceUsageRedefinition_Mapping.getMapped(from),  
ASFVAObjectReferenceUsageFeatureTyping_Mapping.getMapped(from)}
```

#### 7.7.2.3.7.5 ASFVAObjectReferenceUsageFeatureTyping\_Mapping

##### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

##### General Mappings

ToFeatureTyping\_Init  
Mapping

##### Mapping Source

AddStructuralFeatureValueAction

##### Mapping Target

FeatureTyping

##### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureTyping::type () : Type [1]`

```
from.structuralFeature.owner
```

#### 7.7.2.3.7.6 ASFVAObjectReferenceUsageRedefinition\_Mapping

##### Description

Creates a redefinition relationship for the *redefiningFeature()* and the *redefinedFeature()*.

##### General Mappings

ToRedefinition\_Init  
Mapping

#### Mapping Source

AddStructuralFeatureValueAction

#### Mapping Target

Redefinition

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Redefinition::redefinedFeature () : Feature [1]

```
SysML2::ReferenceUsage.allInstances()  
->any(m | m.qualifiedName = 'SysMLv1Library::AddStructuralFeatureValueAction::object')
```

#### 7.7.2.3.7.7 ASFVATargetFeatureChainExpression\_Mapping

##### Description

The mapping class creates the feature chain expression element for the target element of the UML4SysML::AddStructuralFeatureValueAction mapping.

##### General Mappings

ToFeatureChainExpression\_Init  
Mapping

#### Mapping Source

AddStructuralFeatureValueAction

#### Mapping Target

FeatureChainExpression

#### Owned Mappings

(none)

#### Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureChainExpression::ownedRelationship () : Relationship [0..*]`  

```
Set{ASFVATargetParameterMembership_Mapping.getMapped(from) ,  
ASFVATargetParameterFeatureExpressionMembership_Mapping.getMapped(from) ,  
ReturnParameterFeatureMembership_Factory.create() }
```

### 7.7.2.3.7.8 ASFVATargetFeatureMembership\_Mapping

#### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

#### General Mappings

ToFeatureMembership\_Init  
Mapping

#### Mapping Source

AddStructuralFeatureValueAction

#### Mapping Target

FeatureMembership

#### Owned Mappings

(none)

#### Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureMembership::ownedMemberFeature () : Feature [1]`  

```
ASFVATargetReferenceUsage_Mapping.getMapped(from)
```

### 7.7.2.3.7.9 ASFVATargetFeatureValue\_Mapping

#### Description

Creates a feature value relationship.

#### General Mappings

ToFeatureValue\_Init  
Mapping

### Mapping Source

AddStructuralFeatureValueAction

### Mapping Target

FeatureValue

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureValue::value () : Expression [1]  
`ASFVATargetFeatureChainExpression_Mapping.getMapped (from)`
- FeatureValue::isInitial () : Boolean [1]  
`true`

#### 7.7.2.3.7.10 ASFVATargetParameterExpressionFeature\_Mapping

### Description

The mapping class creates the feature element of the feature reference expression for the target element of the UML4SysML::AddStructuralFeatureValueAction mapping.

### General Mappings

ToFeature\_Init  
Mapping

### Mapping Source

AddStructuralFeatureValueAction

### Mapping Target

Feature

### Owned Mappings

(none)

### Applicable filters

(none)

#### 7.7.2.3.7.11 ASFVATargetParameterExpressionFeatureMembership\_Mapping

##### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

##### General Mappings

ToFeatureMembership\_Init  
Mapping

##### Mapping Source

AddStructuralFeatureValueAction

##### Mapping Target

FeatureMembership

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureMembership::ownedMemberFeature () : Feature [1]  
`ASFVATargetParameterExpressionFeature_Mapping.getMapped(from)`

#### 7.7.2.3.7.12 ASFVATargetParameterExpressionMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

##### General Mappings

ToMembership\_Init  
Mapping

##### Mapping Source

AddStructuralFeatureValueAction

##### Mapping Target

Membership

##### Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Membership::memberElement () : Element [1]  
`ASFVAObjectReferenceUsage_Mapping.getMapped(from)`

### 7.7.2.3.7.13 ASFVATargetParameterFeature\_Mapping

#### Description

The mapping class creates the feature element for the target element of the UML4SysML::AddStructuralFeatureValueAction mapping.

#### General Mappings

ToFeature\_Init  
Mapping

#### Mapping Source

AddStructuralFeatureValueAction

#### Mapping Target

Feature

#### Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Feature::direction () : FeatureDirectionKind [0..1]  
`KerML::FeatureDirectionKind::_in'`
- Feature::ownedRelationship () : Relationship [0..\*]  
`Set{ASFVATargetParameterFeatureValue_Mapping.getMapped(from),  
ASFVATargetParameterExpressionFeatureMembership_Mapping.getMapped(from) }`

### 7.7.2.3.7.14 ASFVATargetParameterFeatureExpressionMembership\_Mapping

#### Description

Creates a membership relationship for *memberElement()*.

### General Mappings

ToMembership\_Init  
Mapping

### Mapping Source

AddStructuralFeatureValueAction

### Mapping Target

Membership

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Membership::memberElement () : Element [1]  
`from.structuralFeature`

#### 7.7.2.3.7.15 ASFVATargetParameterFeatureReferenceExpression\_Mapping

### Description

The mapping class creates the feature reference expression element for the target element of the UML4SysML::AddStructuralFeatureValueAction mapping.

### General Mappings

ToFeatureReferenceExpression\_Init  
Mapping

### Mapping Source

AddStructuralFeatureValueAction

### Mapping Target

FeatureReferenceExpression

### Owned Mappings

(none)

### Applicable filters



(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureReferenceExpression::ownedRelationship () : Relationship [0..*]`

```
Set{ASFVATargetParameterExpressionMembership_Mapping.getMapped(from) ,  
ReturnParameterFeatureMembership_Factory.create() }
```

#### 7.7.2.3.7.16 ASFVATargetParameterFeatureValue\_Mapping

##### Description

Creates a feature value relationship.

##### General Mappings

ToFeatureValue\_Init  
Mapping

##### Mapping Source

AddStructuralFeatureValueAction

##### Mapping Target

FeatureValue

##### Owned Mappings

(none)

##### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureValue::value () : Expression [1]`

```
ASFVATargetParameterFeatureReferenceExpression_Mapping.getMapped(from)
```

#### 7.7.2.3.7.17 ASFVATargetParameterMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

##### General Mappings

ToParameterMembership\_Init  
Mapping

### Mapping Source

AddStructuralFeatureValueAction

### Mapping Target

ParameterMembership

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ParameterMembership::visibility () : VisibilityKind [1]  
`KerML::VisibilityKind::private`
- ParameterMembership::ownedMemberParameter () : Feature [1]  
`ASFVATargetParameterFeature_Mapping.getMapped(from)`

#### 7.7.2.3.7.18 ASFVATargetReferenceUsage\_Mapping

### Description

Creates a reference usage.

### General Mappings

ToReferenceUsage\_Init  
Mapping

### Mapping Source

AddStructuralFeatureValueAction

### Mapping Target

ReferenceUsage

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceUsage::ownedRelationship () : Relationship [0..*]`

```
Set{ASFVATargetReferenceUsageRedefinition_Mapping.getMapped(from),
ASFVATargetFeatureValue_Mapping.getMapped(from),
AssignmentActionUsageOwningMembership_Factory.create() }
```

#### 7.7.2.3.7.19 ASFVATargetReferenceUsageRedefinition\_Mapping

##### Description

Creates a redefinition relationship for the *redefiningFeature()* and the *redefinedFeature()*.

##### General Mappings

ToRedefinition\_Init  
Mapping

##### Mapping Source

AddStructuralFeatureValueAction

##### Mapping Target

Redefinition

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Redefinition::redefinedFeature () : Feature [1]`

```
SysML2::ReferenceUsage.allInstances()
->any(m | m.qualifiedName = 'SysMLv1Library::AddValueAction::target')
```

#### 7.7.2.3.7.20 ClearStructuralFeatureAction\_Mapping

##### Description

The UML4SysML::ClearStructuralFeatureAction is mapped to a SysML v2 ActionUsage. The details of the mapping are not defined yet.

##### General Mappings

CommonAction\_Mapping

##### Mapping Source

ClearStructuralFeatureAction

### **Mapping Target**

ActionUsage

### **Owned Mappings**

(none)

### **Applicable filters**

(none)

#### **7.7.2.3.7.21 ReadStructuralFeatureAction\_Mapping**

### **Description**

A UML4SysML::ReadStructuralFeatureAction is mapped to a SysML v2 ActionUsage that returns the value of the specified structural feature of the given object.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
action def SysMLv1Activity {
    action sysMLv1ReadStructuralFeatureAction {
        in object : SysMLv1Block;
        out result = object.sysMLv1Property;
    }
}
part def SysMLv1Block {
    attribute sysMLv1Property;
}
```

### **General Mappings**

CommonAction\_Mapping

### **Mapping Source**

ReadStructuralFeatureAction

### **Mapping Target**

ActionUsage

### **Owned Mappings**

(none)

### **Applicable filters**

(none)

### **Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ActionUsage::ownedRelationship () : Relationship [0..*]`

```
Helper.actionOwnedRelationship (from)  
->including (RSFAReferenceUsageFeatureMembership_Mapping.getMapped (from) )
```

#### 7.7.2.3.7.22 RSFAReferenceUsage\_Mapping

##### Description

Creates a reference usage.

##### General Mappings

ToReferenceUsage\_Init  
Mapping

##### Mapping Source

ReadStructuralFeatureAction

##### Mapping Target

ReferenceUsage

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceUsage::direction () : FeatureDirectionKind [0..1]`

```
KerML::FeatureDirectionKind::_out'
```

- `ReferenceUsage::ownedRelationship () : Relationship [0..*]`

```
Set {RSFAReferenceUsageFeatureValue_Mapping.getMapped (from) }
```

#### 7.7.2.3.7.23 RSFAReferenceUsageExpressionFeature\_Mapping

##### Description

The mapping class creates the feature of the feature chain expression for the reference usage of the UML4SysML::ReadStructuralFeatureValueAction mapping.

##### General Mappings

ToFeature\_Init  
Mapping

#### Mapping Source

ReadStructuralFeatureAction

#### Mapping Target

Feature

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Feature::ownedRelationship () : Relationship [0..\*]

```
Set { RSFReferenceUsageExpressionFeatureValue_Mapping.getMapped(from) ,  
      RSFReferenceUsageExpressionFeatureMembership_Mapping.getMapped(from) }
```

#### 7.7.2.3.7.24 RSFReferenceUsageExpressionFeatureMembership\_Mapping

##### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

##### General Mappings

ToFeatureMembership\_Init  
Mapping

#### Mapping Source

ReadStructuralFeatureAction

#### Mapping Target

FeatureMembership

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureMembership::ownedMemberFeature () : Feature [1]`

```
RSFAReferenceUsageFeatureChainExpressionFeature_Mapping.getMapped(from)
```

#### 7.7.2.3.7.25 RSFAReferenceUsageExpressionFeatureReferenceExpression\_Mapping

##### Description

The mapping class creates the feature reference expression element for the UML4SysML::RemoveStructuralFeatureValueAction mapping.

##### General Mappings

ToFeatureReferenceExpression\_Init  
Mapping

##### Mapping Source

ReadStructuralFeatureAction

##### Mapping Target

FeatureReferenceExpression

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureReferenceExpression::ownedRelationship () : Relationship [0..*]`

```
Set{RSFAReferenceUsageExpressionFeatureMembership_Mapping.getMapped(from),  
ReturnParameterFeatureMembership_Factory.create() }
```

#### 7.7.2.3.7.26 RSFAReferenceUsageExpressionFeatureValue\_Mapping

##### Description

Creates a feature value relationship.

##### General Mappings

ToFeatureValue\_Init  
Mapping

##### Mapping Source

ReadStructuralFeatureAction

### Mapping Target

FeatureValue

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureValue::value () : Expression [1]

`RSFAReferenceUsageExpressionFeatureReferenceExpression_Mapping.getMapped(from)`

#### 7.7.2.3.7.27 RSFAReferenceUsageFeatureChainExpression\_Mapping

### Description

The mapping class creates the feature chain expression element for the reference usage of the UML4SysML::ReadStructuralFeatureValueAction mapping.

### General Mappings

ToFeatureChainExpression\_Init  
Mapping

### Mapping Source

ReadStructuralFeatureAction

### Mapping Target

FeatureChainExpression

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureChainExpression::ownedRelationship () : Relationship [0..\*]



```

Set{RSFAReferenceUsageParameterMembership_Mapping.getMapped(from),
RSFAReferenceUsageMembership_Mapping.getMapped(from),
ReturnParameterFeatureMembership_Factory.create()}

```

#### 7.7.2.3.7.28 RSFAReferenceUsageFeatureChainExpressionFeature\_Mapping

##### Description

The mapping class creates the feature element for the feature chain expression for the UML4SysML::RemoveStructuralFeatureValueAction mapping.

##### General Mappings

ToFeature\_Init  
Mapping

##### Mapping Source

ReadStructuralFeatureAction

##### Mapping Target

Feature

##### Owned Mappings

(none)

##### Applicable filters

(none)

#### 7.7.2.3.7.29 RSFAReferenceUsageFeatureChainExpressionMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

##### General Mappings

ToMembership\_Init  
Mapping

##### Mapping Source

ReadStructuralFeatureAction

##### Mapping Target

Membership

##### Owned Mappings

(none)

##### Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Membership::memberElement () : Element [1]`

`from.structuralFeature`

### 7.7.2.3.7.30 RSFAReferenceUsageFeatureMembership\_Mapping

#### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

#### General Mappings

ToFeatureMembership\_Init  
Mapping

#### Mapping Source

ReadStructuralFeatureAction

#### Mapping Target

FeatureMembership

#### Owned Mappings

(none)

#### Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureMembership::ownedMemberFeature () : Feature [1]`

`RSFAReferenceUsageFeatureValue_Mapping.getMapped(from)`

### 7.7.2.3.7.31 RSFAReferenceUsageFeatureValue\_Mapping

#### Description

Creates a feature value relationship.

#### General Mappings

ToFeatureValue\_Init  
Mapping

#### Mapping Source

ReadStructuralFeatureAction

### Mapping Target

FeatureValue

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureValue::value () : Expression [1]

`RSFAReferenceUsageFeatureChainExpression_Mapping.getMapped (from)`

### 7.7.2.3.7.32 RSFAReferenceUsageMembership\_Mapping

#### Description

Creates a membership relationship for *memberElement()*.

#### General Mappings

ToMembership\_Init  
Mapping

#### Mapping Source

ReadStructuralFeatureAction

#### Mapping Target

Membership

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Membership::memberElement () : Element [1]

`from.object`

#### 7.7.2.3.7.33 RSFAReferenceUsageParameterMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

##### General Mappings

ToParameterMembership\_Init  
Mapping

##### Mapping Source

ReadStructuralFeatureAction

##### Mapping Target

ParameterMembership

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ParameterMembership::ownedMemberParameter () : Feature [1]

`RSFAReferenceUsageExpressionFeature_Mapping.getMapped(from)`

#### 7.7.2.3.7.34 RemoveStructuralFeatureValueAction\_Mapping

##### Description

The UML4SysML::RemoveStructuralFeatureValueAction is mapped to a SysML v2 ActionUsage. The details of the mapping are not defined yet.

##### General Mappings

CommonAction\_Mapping

##### Mapping Source

RemoveStructuralFeatureValueAction

##### Mapping Target

ActionUsage

##### Owned Mappings

(none)

#### **Applicable filters**

(none)

### **7.7.2.3.8 Structured Actions**

#### **7.7.2.3.8.1 LoopNode\_Mapping**

##### **Description**

The UML4SysML::LoopNode is mapped to a SysML v2 ActionUsage. The details of the mapping are not defined yet.

##### **General Mappings**

StructuredActivityNode\_Mapping

##### **Mapping Source**

LoopNode

##### **Mapping Target**

ActionUsage

##### **Owned Mappings**

(none)

#### **Applicable filters**

(none)

#### **7.7.2.3.8.2 SequenceNode\_Mapping**

##### **Description**

The UML4SysML::SequenceNode is mapped to a SysML v2 ActionUsage. The details of the mapping are not defined yet.

##### **General Mappings**

CommonAction\_Mapping

StructuredActivityNode\_Mapping

##### **Mapping Source**

SequenceNode

##### **Mapping Target**

ActionUsage

##### **Owned Mappings**

(none)

## Applicable filters

(none)

### 7.7.2.3.8.3 StructuredActivityNode\_Mapping

#### Description

The UML4SysML::StructuredActivityNode is mapped to a SysML v2 ActionUsage. The details of the mapping are not defined yet.

#### General Mappings

CommonAction\_Mapping

#### Mapping Source

StructuredActivityNode

#### Mapping Target

ActionUsage

#### Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ActionUsage::ownedRelationship () : Relationship [0..\*]

```
let initialNodes : Set(UML::Element) =
    from.ownedElement->select(e | e.ocIsKindOf(UML::InitialNode)) in
let finalNodes : Set(UML::Element) =
    from.ownedElement->select(e | e.ocIsKindOf(UML::FinalNode)) in
let objectFlowsWithGuard : Set(UML::ObjectFlow) =
    from.ownedElement->select(e | e.ocIsKindOf(UML::ObjectFlow)
        and not e.ocIsType(UML::ObjectFlow).guard.ocIsUndefined()) in
let objectFlows : Set(UML::ObjectFlow) =
    from.ownedElement->select(e | e.ocIsKindOf(UML::ObjectFlow)) in
let ignoreInterruptibleActivityRegion: Set(UML::InterruptibleActivityRegion) =
    from.ownedElement
        ->select(e | e.ocIsKindOf(UML::InterruptibleActivityRegion)) in
let elementsFMS : Set(UML::Element) =
    (((from.ownedElement->select(e | e.ocIsKindOf(UML::ControlNode) or
        e.ocIsKindOf(UML::Action) or (e.ocIsKindOf(UML::ControlFlow) or
        e.ocIsKindOf(UML::Pin)))) - initialNodes) - finalNodes) in
let elementsOMS: Set(UML::Element) =
    ((((((from.ownedElement-initialNodes)-finalNodes)-objectFlowsWithGuard)
        -objectFlows)-elementsFMS)-ignoreInterruptibleActivityRegion) in
elementsOMS->collect(e | ElementOwningMembership_Mapping.getMapped(e))
    ->union(elementsFMS->collect(e | ElementFeatureMembership_Mapping.getMapped(e)))
```

```

->union(initialNodes->collect(e | InitialNodeMembership_Mapping.getMapped(e)))
->union(finalNodes->collect(e | FlowFinalNodeMembership_Mapping.getMapped(e)))
->union(objectFlowsWithGuard
->collect(e | ObjectFlowGuardFeatureMembership_Mapping.getMapped(e)))
->union(objectFlows->collect(e | ObjectFlowFeatureMembership_Mapping.getMapped(e)))

```

### 7.7.2.3.9 Variable Actions

#### 7.7.2.3.9.1 AddVariableValueAction\_Mapping

##### Description

A UML4SysML::AddVariableValueAction is mapped to a SysML v2 ActionUsage defined by the SysML v1 library action definition SysMLv1Library::AddValueAction. The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```

action def SysMLv1Activity {
  private attribute sysMLv1Variable1 : ScalarValues::Integer;
  private attribute sysMLv1Variable2 [0..*] : ScalarValues::Integer;

  action sysMLv1AddVariableValueAction1 : SysMLv1Library::AddValueAction {
    :>> target := sysMLv1Variable1;
  }

  action sysMLv1AddVariableValueAction1 : SysMLv1Library::AddValueAction {
    :>> target := thisIsAVariable;
    :>> isReplaceAll := true;
  }
}

```

##### General Mappings

CommonAction\_Mapping

##### Mapping Source

AddVariableValueAction

##### Mapping Target

ActionUsage

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ActionUsage::ownedRelationship () : Relationship [0..\*]

```

let relationships : Set(KerML::Relationship) =
  Set{AVVAFeatureTyping_Mapping.getMapped(from)}
  ->including(AVVAVariableFeatureMembership_Mapping.getMapped(from)) in
if from.isReplaceAll then
  relationships
  ->including(AVVAIsReplaceAllFeatureMembership_Mapping.getMapped(from))
else
  relationships
endif

```

#### 7.7.2.3.9.2 AVVAFeatureTyping\_Mapping

##### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

##### General Mappings

ToFeatureTyping\_Init  
Mapping

##### Mapping Source

AddVariableValueAction

##### Mapping Target

FeatureTyping

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureTyping::type () : Type [1]

```

SYSML2::ActionDefinition.allInstances()
->any(m | m.qualifiedName = 'SysMLv1Library::AddValueAction')

```

#### 7.7.2.3.9.3 AVVAFeatureValue\_Mapping

##### Description

Creates a feature value relationship.

##### General Mappings

ToFeatureValue\_Init  
Mapping



### Mapping Source

AddVariableValueAction

### Mapping Target

FeatureValue

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureValue::value () : Expression [1]

```
AVVValueFeatureReferenceExpression_Mapping.getMapped(from)
```

#### 7.7.2.3.9.4 AVVAIsReplaceAll\_Mapping

### Description

The mapping class creates a reference usage element as mapping target for the AddVariableValueAction::isReplaceAll property.

### General Mappings

ToReferenceUsage\_Init  
Mapping

### Mapping Source

AddVariableValueAction

### Mapping Target

ReferenceUsage

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceUsage::ownedRelationship () : Relationship [0..*]`  

```
Set{AVVAIsReplaceAllRedefinition_Mapping.getMapped(from),
AVVAIsReplaceAllValue_Mapping.getMapped(from),
AssignmentActionUsageOwningMembership_Factory.create() }
```

#### 7.7.2.3.9.5 AVVAIsReplaceAllFeatureMembership\_Mapping

##### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

##### General Mappings

ToFeatureMembership\_Init  
Mapping

##### Mapping Source

AddVariableValueAction

##### Mapping Target

FeatureMembership

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureMembership::ownedMemberFeature () : Feature [1]`  

```
AVVAIsReplaceAll_Mapping.getMapped(from)
```

#### 7.7.2.3.9.6 AVVAIsReplaceAllRedefinition\_Mapping

##### Description

Creates a redefinition relationship for the *redefiningFeature()* and the *redefinedFeature()*.

##### General Mappings

ToRedefinition\_Init  
Mapping

##### Mapping Source

AddVariableValueAction

### Mapping Target

Redefinition

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Redefinition::redefinedFeature () : Feature [1]`  
`SysML2::ReferenceUsage.allInstances()`  
`->any(m | m.qualifiedName = 'SysMLv1Library::AddValueAction::isReplaceAll')`

#### 7.7.2.3.9.7 AVVAIsReplaceAllValue\_Mapping

### Description

The mapping class maps the value of the AddVariableValueAction::isReplaceAll property.

### General Mappings

ToFeatureValue\_Init  
Mapping

### Mapping Source

AddVariableValueAction

### Mapping Target

FeatureValue

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureValue::value () : Expression [1]`

```
LiteralBoolean_Factory.create(from.isReplaceAll)
```

#### 7.7.2.3.9.8 AVVAValueExpressionMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

##### General Mappings

ToMembership\_Init  
Mapping

##### Mapping Source

AddVariableValueAction

##### Mapping Target

Membership

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Membership::memberElement () : Element [1]

```
from.variable
```

#### 7.7.2.3.9.9 AVVAValueFeatureReferenceExpression\_Mapping

##### Description

The mapping class creates the feature reference expression element for the UML4SysML::AddStructuralFeatureValueAction mapping.

##### General Mappings

ToFeatureReferenceExpression\_Init  
Mapping

##### Mapping Source

AddVariableValueAction

##### Mapping Target

FeatureReferenceExpression

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureReferenceExpression::ownedRelationship () : Relationship [0..*]`  

```
Set{AVVAValueExpressionMembership_Mapping.getMapped(from),  
ReturnParameterFeatureMembership_Factory.create() }
```

### 7.7.2.3.9.10 AVVAVariable\_Mapping

#### Description

The mapping class creates a reference usage element for the UML4SysML::AddVariableValueAction mapping.

#### General Mappings

ToReferenceUsage\_Init  
Mapping

#### Mapping Source

AddVariableValueAction

#### Mapping Target

ReferenceUsage

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceUsage::ownedRelationship () : Relationship [0..*]`  

```
Set{AVVAVariableRedefinition_Mapping.getMapped(from),  
AVVAFeatureValue_Mapping.getMapped(from),  
AssignmentActionUsageOwningMembership_Factory.create() }
```

#### 7.7.2.3.9.11 AVVAVariableFeatureMembership\_Mapping

##### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

##### General Mappings

ToFeatureMembership\_Init  
Mapping

##### Mapping Source

AddVariableValueAction

##### Mapping Target

FeatureMembership

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureMembership::ownedMemberFeature () : Feature [1]  
`AVVAVariable_Mapping.getMapped (from)`

#### 7.7.2.3.9.12 AVVAVariableRedefinition\_Mapping

##### Description

Creates a redefinition relationship for the *redefiningFeature()* and the *redefinedFeature()*.

##### General Mappings

ToRedefinition\_Init  
Mapping

##### Mapping Source

AddVariableValueAction

##### Mapping Target

Redefinition

##### Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Redefinition::redefinedFeature () : Feature [1]`

```
SysML2::ReferenceUsage.allInstances()  
->any(m | m.qualifiedName = 'SysMLv1Library::AddValueAction::target')
```

### 7.7.2.3.9.13 ClearVariableAction\_Mapping

#### Description

The `UML4SysML::ClearVariableAction` is mapped to a SysML v2 `ActionUsage` that sets the attribute usage representing the variable to null.

The expected SysML v2 textual notation of a `SysMLv1::ClearVariableAction` is as follows

```
action def SysMLv1Activity {  
    private attribute sysMLv1Variable : ScalarValues::Integer;  
  
    action sysMLv1ClearVariableAction {  
        sysMLv1Variable := null;  
    }  
}
```

#### General Mappings

`CommonAction_Mapping`

#### Mapping Source

`ClearVariableAction`

#### Mapping Target

`ActionUsage`

#### Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ActionUsage::ownedRelationship () : Relationship [0..*]`

```
Helper.actionOwnedRelationship(from)
->including(CVAFeatureMembership_Mapping.getMapped(from))
```

#### 7.7.2.3.9.14 CVAFeatureMembership\_Mapping

##### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

##### General Mappings

ToFeatureMembership\_Init  
Mapping

##### Mapping Source

ClearVariableAction

##### Mapping Target

FeatureMembership

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureMembership::ownedMemberFeature () : Feature [1]  
CVAReferenceUsage\_Mapping.getMapped(from)

#### 7.7.2.3.9.15 CVAReferenceUsage\_Mapping

##### Description

Creates a reference usage.

##### General Mappings

ToReferenceUsage\_Init  
Mapping

##### Mapping Source

ClearVariableAction

##### Mapping Target

ReferenceUsage



## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceUsage::ownedRelationship () : Relationship [0..*]`  

```
Set{CVAReferenceUsageFeatureValue_Mapping.getMapped(from),  
AssignmentActionUsageOwningMembership_Factory.create() }
```
- `ReferenceUsage::declaredName () : String [0..1]`  

```
from.variable.name
```

### 7.7.2.3.9.16 CVAReferenceUsageFeatureValue\_Mapping

#### Description

Creates a feature value relationship.

#### General Mappings

ToFeatureValue\_Init  
Mapping

#### Mapping Source

ClearVariableAction

#### Mapping Target

FeatureValue

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureValue::value () : Expression [1]`  

```
LiteralNull_Factory.create()
```

#### 7.7.2.3.9.17 ReadVariableAction\_Mapping

##### Description

A UML4SysML::ReadVariableValueAction is mapped to a SysML v2 ActionUsage with an out parameter that returns the value of the attribute usage that is the transformation target of the UML4SysML::Variable.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
action def SysMLv1Activity {  
    private attribute sysMLv1Variable : ScalarValues::Integer;  
  
    action sysMLv1ReadVariableAction {  
        out result : ScalarValues::Integer = sysMLv1Variable;  
    }  
}
```

##### General Mappings

CommonAction\_Mapping

##### Mapping Source

ReadVariableAction

##### Mapping Target

ActionUsage

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ActionUsage::ownedRelationship () : Relationship [0..\*]  
`Set{RVAFeatureMembership_Mapping.getMapped(from)}`

#### 7.7.2.3.9.18 RVAFeatureMembership\_Mapping

##### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

##### General Mappings

ToFeatureMembership\_Init  
Mapping

**Mapping Source**

ReadVariableAction

**Mapping Target**

FeatureMembership

**Owned Mappings**

(none)

**Applicable filters**

(none)

**Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureMembership::ownedMemberFeature () : Feature [1]`  
`RVAReferenceUsage_Mapping.getMapped(from.result)`

**7.7.2.3.9.19 RVAReferenceUsage\_Mapping****Description**

Creates a reference usage.

**General Mappings**

ToReferenceUsage\_Init  
Mapping

**Mapping Source**

Pin

**Mapping Target**

ReferenceUsage

**Owned Mappings**

(none)

**Applicable filters**

(none)

**Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceUsage::ownedRelationship () : Relationship [0..*]`

```
let featureTyping : Set(KerML::FeatureTyping) =
  if from.type.ocIsUndefined() then
    Set{}
  else
    Set{RVReferenceUsageFeatureTyping_Mapping.getMapped(from)}
  endif in
featureTyping
->including(RVReferenceUsageFeatureValue_Mapping.getMapped(from))
```

#### 7.7.2.3.9.20 RVReferenceUsageFeatureReferenceExpression\_Mapping

##### Description

The mapping class creates the feature reference expression element for the UML4SysML::ReadVariableAction mapping.

##### General Mappings

ToFeatureReferenceExpression\_Init  
Mapping

##### Mapping Source

Pin

##### Mapping Target

FeatureReferenceExpression

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureReferenceExpression::ownedRelationship () : Relationship [0..*]`

```
Set{RVReferenceUsageExpressionMembership_Mapping.getMapped(from),
ReturnParameterFeatureMembership_Factory.create() }
```

#### 7.7.2.3.9.21 RVReferenceUsageFeatureTyping\_Mapping

##### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

## General Mappings

TypedElementFeatureTyping\_Mapping

## Mapping Source

Pin

## Mapping Target

FeatureTyping

## Owned Mappings

(none)

## Applicable filters

(none)

### 7.7.2.3.9.22 RVAReferenceUsageFeatureValue\_Mapping

## Description

Creates a feature value relationship.

## General Mappings

ToFeatureValue\_Init  
Mapping

## Mapping Source

Pin

## Mapping Target

FeatureValue

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureValue::value () : Expression [1]

```
RVAReferenceUsageFeatureReferenceExpression_Mapping.getMapped (from)
```

#### 7.7.2.3.9.23 RVAReferenceUsageExpressionMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

##### General Mappings

ToMembership\_Init  
Mapping

##### Mapping Source

Pin

##### Mapping Target

Membership

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Membership::memberElement () : Element [1]  
  
from.owner.oclAsType (UML::ReadVariableAction).variable

#### 7.7.2.3.9.24 RemoveVariableValueAction\_Mapping

##### Description

A UML4SysML::RemoveVariableValueAction is mapped to a SysML v2 ActionUsage defined by the SysML v1 library action definition SysMLv1Library::RemoveVariableValueAction.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
action def SysMLv1Activity {  
    private sysMLv1Variable : ScalarValues::Integer;  
  
    action sysMLv1RemoveVariableValueAction  
        : SysMLv1Library::RemoveVariableValueAction {  
            :>> variable := sysMLv1Variable;  
        }  
}
```

##### General Mappings

CommonAction\_Mapping

### Mapping Source

RemoveVariableValueAction

### Mapping Target

ActionUsage

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ActionUsage::ownedRelationship () : Relationship [0..\*]  
  
    Helper.actionOwnedRelationship (from)  
    ->including (RVVAFeatureTyping\_Mapping.getMapped (from) )  
    ->including (RVVAVariableFeatureMembership\_Mapping.getMapped (from) )

#### 7.7.2.3.9.25 RVVAFeatureTyping\_Mapping

### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

### General Mappings

ToFeatureTyping\_Init  
Mapping

### Mapping Source

RemoveVariableValueAction

### Mapping Target

FeatureTyping

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureTyping::type () : Type [1]

```
SysML2::ActionDefinition.allInstances()  
->any(m | m.qualifiedName = 'SysMLv1Library::RemoveVariableValueAction')
```

#### 7.7.2.3.9.26 RVVAVariable\_Mapping

##### Description

The mapping class creates a reference usage element for the UML4SysML::RemoveVariableValueAction mapping.

##### General Mappings

ToReferenceUsage\_Init  
Mapping

##### Mapping Source

RemoveVariableValueAction

##### Mapping Target

ReferenceUsage

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReferenceUsage::ownedRelationship () : Relationship [0..\*]

```
Set{RVVAVariableRedefinition_Mapping.getMapped(from),  
RVVAVariableFeatureValue_Mapping.getMapped(from),  
AssignmentActionUsageOwningMembership_Factory.create() }
```

#### 7.7.2.3.9.27 RVVAVariableExpressionMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

##### General Mappings

ToMembership\_Init  
Mapping

##### Mapping Source



RemoveVariableValueAction

### Mapping Target

Membership

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Membership::memberElement () : Element [1]  
`from.variable`

#### 7.7.2.3.9.28 RVVAVariableFeatureMembership\_Mapping

### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

### General Mappings

ToFeatureMembership\_Init  
Mapping

### Mapping Source

RemoveVariableValueAction

### Mapping Target

FeatureMembership

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureMembership::ownedMemberFeature () : Feature [1]

```
RVVAVariable_Mapping.getMapped(from)
```

#### 7.7.2.3.9.29 RVVAVariableFeatureReferenceExpression\_Mapping

##### Description

The mapping class creates the feature reference expression element for the UML4SysML::RemoveVariableValueAction mapping.

##### General Mappings

ToFeatureReferenceExpression\_Init  
Mapping

##### Mapping Source

RemoveVariableValueAction

##### Mapping Target

FeatureReferenceExpression

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureReferenceExpression::ownedRelationship () : Relationship [0..\*]

```
Set{RVVAVariableExpressionMembership_Mapping.getMapped(from),  
ReturnParameterFeatureMembership_Factory.create() }
```

#### 7.7.2.3.9.30 RVVAVariableFeatureValue\_Mapping

##### Description

Creates a feature value relationship.

##### General Mappings

ToFeatureValue\_Init  
Mapping

##### Mapping Source

RemoveVariableValueAction

##### Mapping Target

FeatureValue

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureValue::value () : Expression [1]`  
`RVVAVariableFeatureReferenceExpression_Mapping.getMapped (from)`

### 7.7.2.3.9.31 RVVAVariableRedefinition\_Mapping

#### Description

Creates a redefinition relationship for the *redefiningFeature()* and the *redefinedFeature()*.

#### General Mappings

ToRedefinition\_Init  
Mapping

#### Mapping Source

RemoveVariableValueAction

#### Mapping Target

Redefinition

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Redefinition::redefinedFeature () : Feature [1]`  
`SysML2::ReferenceUsage.allInstances ()`  
`->any (m | m.qualifiedName = 'SysMLv1Library::RemoveVariableValueAction::variable')`

## 7.7.3 Activities

### 7.7.3.1 Overview

**Table 3. List of all mappings**

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
Activity	ActionDefinition
ActivityFinalNode	TerminateActionUsage
ActivityParameterNode	not mapped; see next section
ActivityPartition	not mapped; see next section
CentralBufferNode	ActionUsage
ControlFlow	TransitionUsage SuccessionAsUsage
DataStoreNode	ActionUsage
DecisionNode	DecisionNode
ExceptionHandler	not mapped; see next section
FlowFinalNode	not mapped; see next section
ForkNode	ForkNode
InitialNode	not mapped; see next section
InterruptibleActivityRegion	not mapped; see next section
JoinNode	JoinNode
MergeNode	MergeNode
ObjectFlow	TransitionUsage SuccessionFlowUsage
Variable	ItemUsage AttributeUsage

### 7.7.3.2 UML4SysML::Activities elements not mapped

**Table 4. List of SysML v1 elements not mapped of this section**

SysML v1 Concept	Rationale
ActivityFinalNode	Mapping is not specified yet.
ActivityParameterNode	The parameter of the activity is mapped from SysML v1 to SysML v2. The additional concept of the activity parameter node is necessary for the token semantic of SysML v1 activities, which is not part of SysML v2. Therefore, the additional concept of the activity parameter node is not mapped to SysML v2.
ActivityPartition	Mapping is not specified yet.
ExceptionHandler	Mapping is not specified yet.
InterruptibleActivityRegion	Mapping is not specified yet.

### 7.7.3.3 Mapping Specifications

#### 7.7.3.3.1 ActivityAsDefinition\_Mapping

##### Description

A UML4SysML::Activity is mapped to a SysMLv2 ActionDefinition.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
action def SysMLv1Activity {
  in parIn : SysMLv1Block;
  out parOut;
  out parReturn;
}
part def SysMLv1Block;
```

##### General Mappings

Behavior\_Mapping

##### Mapping Source

Activity

##### Mapping Target

ActionDefinition

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ActionDefinition::ownedRelationship () : Relationship [0..\*]  

```
let relationships : Set(KerML::Relationship) =
  Helper.activityOwnedRelationship(from) in
let parameters : Set(UML::Paramter) =
  from.ownedElement->select(e | e.ocIsKindOf(UML::Parameter)) in
relationships->union(parameters
  ->collect(p | ParameterMembership_Mapping.getMapped(p))
)
```

#### 7.7.3.3.2 ActivityEdgeInitialNodeFeatureMembership\_Mapping

##### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

## General Mappings

ToEndFeatureMembership\_Init  
Mapping

## Mapping Source

InitialNode

## Mapping Target

EndFeatureMembership

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- EndFeatureMembership::ownedMemberFeature () : Feature [1]  
`ActivityEdgeSourceInitialNode_Mapping.getMapped(from)`

### 7.7.3.3.3 ActivityEdgeMetadata\_Mapping

#### Description

Adds metadata to the transformation target elements of UML4SysML::ControlFlow and UML::ObjectFlow to map the UML4SysML::ActivityEdge::weight property which has no direct target in SysML v2.

## General Mappings

ToMetadataUsage\_Init  
Mapping

## Mapping Source

ActivityEdge

## Mapping Target

MetadataUsage

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `MetadataUsage::ownedRelationship () : Relationship [0..*]`  
`Set {ActivityEdgeMetadataFeatureTyping_Mapping.getMapped (from) ,  
ActivityEdgeMetadataFeatureMembership_Mapping.getMapped (from) }`
- `MetadataUsage::declaredName () : String [0..1]`  
`'weight'`

### 7.7.3.3.4 ActivityEdgeMetadataFeatureMembership\_Mapping

#### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

#### General Mappings

ToFeatureMembership\_Init  
Mapping

#### Mapping Source

ActivityEdge

#### Mapping Target

FeatureMembership

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureMembership::ownedMemberFeature () : Feature [1]`  
`ActivityEdgeMetadataReferenceUsage_Mapping.getMapped (from)`

### 7.7.3.3.5 ActivityEdgeMetadataFeatureTyping\_Mapping

#### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

#### General Mappings

ToFeatureTyping\_Init  
Mapping

#### **Mapping Source**

ActivityEdge

#### **Mapping Target**

FeatureTyping

#### **Owned Mappings**

(none)

#### **Applicable filters**

(none)

#### **Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureTyping::type () : Type [1]

```
SysML2::MetadataDefinition.allInstances()  
->any(m | m.qualifiedName = 'SysMLv1Library::ActivityEdgeData')
```

### **7.7.3.3.6 ActivityEdgeMetadataFeatureValue\_Mapping**

#### **Description**

Creates a feature value relationship.

#### **General Mappings**

ToFeatureValue\_Init  
Mapping

#### **Mapping Source**

ActivityEdge

#### **Mapping Target**

FeatureValue

#### **Owned Mappings**

(none)

#### **Applicable filters**

(none)

#### **Mapping rules**



In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureValue::value () : Expression [1]`  
`from.weight`

#### 7.7.3.3.7 ActivityEdgeMetadataOwningMembership\_Mapping

##### Description

Creates a owning membership relationship for *ownedMemberElement()*.

##### General Mappings

ToOwningMembership\_Init  
Mapping

##### Mapping Source

ActivityEdge

##### Mapping Target

OwningMembership

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `OwningMembership::ownedMemberElement () : Element [1]`  
`ActivityEdgeMetadata_Mapping.getMapped(from)`

#### 7.7.3.3.8 ActivityEdgeMetadataRedefinition\_Mapping

##### Description

Creates a redefinition relationship for the *redefiningFeature()* and the *redefinedFeature()*.

##### General Mappings

ToRedefinition\_Init  
Mapping

##### Mapping Source

ActivityEdge

### Mapping Target

Redefinition

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Redefinition::redefinedFeature () : Feature [1]`  
`SYSML2::AttributeUsage.allInstances()`  
`->any(m | m.qualifiedName = 'SysMLv1Library::ActivityEdgeData::weight')`

#### 7.7.3.3.9 ActivityEdgeMetadataReferenceUsage\_Mapping

### Description

Creates a reference usage.

### General Mappings

ToReferenceUsage\_Init  
Mapping

### Mapping Source

ActivityEdge

### Mapping Target

ReferenceUsage

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceUsage::ownedRelationship () : Relationship [0..*]`

```
Set{ActivityEdgeMetadataRedefinition_Mapping.getMapped(from),
ActivityEdgeMetadataFeatureValue_Mapping.getMapped(from)}
```

#### 7.7.3.3.10 ActivityEdgeSourceEndFeature\_Mapping

##### Description

Creates a SysML v2 feature for the source activity node of the SysML v1 activity edge which subsets the SysML v2 target element of the source activity node.

##### General Mappings

ToFeature\_Init  
Mapping

##### Mapping Source

Element

##### Mapping Target

Feature

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Feature::isEnd () : Boolean [1]  
  
true
- Feature::ownedRelationship () : Relationship [0..\*]  
  
Set{ActivityEdgeSourceEndSubsetting\_Mapping.getMapped(from)}

#### 7.7.3.3.11 ActivityEdgeSourceInitialNode\_Mapping

##### Description

The UML4SysML::InitialNode is mapped to a subsetted feature of the SysML v2 library element Actions::start.

##### General Mappings

ToFeature\_Init  
Mapping

##### Mapping Source

InitialNode

### Mapping Target

Feature

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Feature::isEnd () : Boolean [1]

true

- Feature::ownedRelationship () : Relationship [0..\*]

Set{ActivityEdgeSourceInitialNodeSubsetting\_Mapping.getMapped(from) }

### 7.7.3.3.12 ActivityEdgeSourceEndFeatureMembership\_Mapping

#### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

#### General Mappings

ToEndFeatureMembership\_Init  
Mapping

#### Mapping Source

Element

#### Mapping Target

EndFeatureMembership

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `EndFeatureMembership::ownedMemberFeature () : Feature [1]`  
`ActivityEdgeSourceEndFeature_Mapping.getMapped(from)`

#### 7.7.3.3.13 ActivityEdgeSourceInitialNodeSubsetting\_Mapping

##### Description

Creates a subsetting relationship.

##### General Mappings

ToReferenceSubsetting\_Init  
Mapping

##### Mapping Source

InitialNode

##### Mapping Target

ReferenceSubsetting

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceSubsetting::referencedFeature () : Feature [1]`  
`SYSML2::ActionUsage.allInstances()`  
`->any(m | m.qualifiedName = 'Actions::Action::start')`

#### 7.7.3.3.14 ActivityEdgeSourceEndSubsetting\_Mapping

##### Description

Creates a subsetting relationship.

##### General Mappings

ToReferenceSubsetting\_Init  
Mapping

##### Mapping Source

Element

### Mapping Target

ReferenceSubsetting

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReferenceSubsetting::referencedFeature () : Feature [1]

from

### 7.7.3.3.15 ActivityEdgeTransitionUsageSourceMembership\_Mapping

#### Description

Creates a membership relationship for *memberElement()*.

#### General Mappings

ToMembership\_Init  
Mapping

#### Mapping Source

ActivityNode

#### Mapping Target

Membership

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Membership::memberElement () : Element [1]

if from.ocIsTypeOf(UML::ActivityParameterNode) then  
from.parameter

```

else
    from
endif

```

### 7.7.3.3.16 ActivityFinalNode\_Mapping

#### Description

A UML4SysML::ActivityFinalNode is mapped to SysML v2 TerminateAction.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```

action def SysMLv1Activity {
    first start;
    then action action1;
    then termine;
}

```

#### General Mappings

NamedElementMain\_Mapping  
ToActionUsage\_Init

#### Mapping Source

ActivityFinalNode

#### Mapping Target

TerminateActionUsage

#### Owned Mappings

(none)

#### Applicable filters

(none)

### 7.7.3.3.17 CentralBufferNode\_Mapping

#### Description

The mapping of the UML4SysML::CentralBufferNode is not defined in detail yet. It will be an action usage which contains the behavior of a central buffer node.

#### General Mappings

ToActionUsage\_Init  
NamedElementMain\_Mapping

#### Mapping Source

CentralBufferNode

#### Mapping Target

ActionUsage

### Owned Mappings

(none)

### Applicable filters

(none)

## 7.7.3.3.18 CommonActivityEdgeSuccessionAsUsage\_Mapping

### Description

The mapping class provides a common mapping of a UML4SysML::ActivityEdge to a SysMLv2 SuccessionAsUsage. The mapping is used for UML4SysML::ControlFlows and UML4SysML::ObjectFlows.

### General Mappings

ToConnector\_Init  
Mapping

### Mapping Source

ActivityEdge

### Mapping Target

SuccessionAsUsage

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- SuccessionAsUsage::ownedRelationship () : Relationship [0..\*]  

```
let relationships : Set(KerML::Relationship) = Set{
if from.source.ocIsKindOf(UML::InitialNode) then
    ActivityEdgeInitialNodeFeatureMembership_Mapping.getMapped(from.source)
else if from.source.ocIsKindOf(UML::ActivityParameterNode) then
    ActivityEdgeSourceEndFeatureMembership_Mapping.getMapped(
        from.source.parameter)
else
    ActivityEdgeSourceEndFeatureMembership_Mapping.getMapped(from.source)
endif
endif,
if from.ocIsKindOf(UML::ObjectFlow) then
    ObjectFlowGuardSuccessionTargetEndFeatureMembership_Mapping.getMapped(from)
else if from.target.ocIsKindOf(UML::FinalNode) then
```



```

        ControlFlowFinalNodeFeatureMembership_Mapping.getMapped(from.target)
    else
        ControlFlowTargetFeatureMembership_Mapping.getMapped(from.target)
    endif
endif} in
if from.guard.oclIsUndefined() then
    relationships
else
    relationships
    ->including(ElementFeatureMembership_Mapping.getMapped(from.guard))
endif

```

### 7.7.3.3.19 CommonVariable\_Mapping

#### Description

Abstract mapping class for UML4SysML::Variable which is defined in the context of UML4SysML::Activity. A UML4SysML::Variable is mapped to a SysMLv2 AttributeUsage or SysMLv2 ItemUsage. See specialized mapping classes for the specific mapping rules.

#### General Mappings

PropertyCommon\_Mapping

#### Mapping Source

Variable

#### Mapping Target

Feature

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Feature::isComposite () : Boolean [1]  
false
- Feature::isEnd () : Boolean [1]  
false
- Feature::isDerived () : Boolean [1]  
false
- Feature::ownedRelationship () : Relationship [0..\*]

```

let typing: KerML::FeatureTyping =
  VariableFeatureTyping_Mapping.getMapped(from) in
if typing.ocllsUndefined() then
  Set{MultiplicityMembership_Mapping.getMapped(from)}
else
  Set{MultiplicityMembership_Mapping.getMapped(from), typing}
endif

```

### 7.7.3.3.20 ControlFlowTransitionUsage\_Mapping

#### Description

A UML4SysML::ControlFlow with a guard condition is mapped to a SysMLv2 TransitionUsage.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```

action def SysMLv1Activity {
  action sysMLv1Action1;
  succession sysMLv1ControlFlow first sysMLv1Action1
    if guardCondition.result then sysMLv1Action2 {
      calc guardCondition {
        return : ScalarValues::Boolean;
        language "English"
      }
      /*
       * thisIsAGuard
       */
    }
  }
  action sysMLv1Action2;
}

```

#### General Mappings

ToTransitionUsage\_Init  
NamedElementMain\_Mapping

#### Mapping Source

ControlFlow

#### Mapping Target

TransitionUsage

#### Owned Mappings

(none)

#### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
not src.guard.ocllsUndefined()
```

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `TransitionUsage::ownedRelationship () : Relationship [0..*]`

```
let relationships : Set(KerML::Relationship)=
  self.oclasType(ElementMain_Mapping).ownedRelationship()
  ->union(Set{
    ActivityEdgeTransitionUsageSourceMembership_Mapping.getMapped(
      from.source),
    CommonParameterReferenceUsageInMembership_Mapping.getMapped(
      from.source),
    ControlFlowTransitionUsageFeatureMembership_Mapping.getMapped(
      from),
    CommonActivityEdgeSuccessionAsUsage_Mapping.getMapped(from),
    CommonReturnParameterReferenceUsageMembership_Mapping.getMapped(
      from)}) in
let relationshipsWithGuard : Set(KerML::Relationship) =
if from.guard.oclasTypeOf(UML::OpaqueExpression) then
  relationships
  ->including(ElementFeatureMembership_Mapping.getMapped(from.guard))
else
  relationships
endif in
let relationshipsConsideringWeight : Set(KerML::Relationship) =
if from.weight.oclasTypeOf(UML::OpaqueExpression) then
  relationshipsWithGuard
else
  relationshipsWithGuard
  ->including(ActivityEdgeMetadataOwningMembership_Mapping.getMapped(from))
endif in
if Helper.hasStereotypeApplied(from, 'SysML::Activities::Probability') then
  relationshipsConsideringWeight
  ->including(ProbabilityOwningMembership_Mapping.getMapped(from))
else
  relationshipsConsideringWeight
endif
```

### 7.7.3.3.21 ControlFlowFinalNodeFeatureMembership\_Mapping

#### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

#### General Mappings

ToEndFeatureMembership\_Init  
Mapping

#### Mapping Source

ActivityNode

#### Mapping Target

EndFeatureMembership

#### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `EndFeatureMembership::ownedMemberFeature () : Feature [1]`  
`ControlFlowTargetFinalNode_Mapping.getMapped(from)`

#### 7.7.3.3.22 ControlFlowTargetFinalNodeSubsetting\_Mapping

##### Description

Creates a subsetting relationship.

##### General Mappings

ToReferenceSubsetting\_Init  
Mapping

##### Mapping Source

FinalNode

##### Mapping Target

ReferenceSubsetting

##### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceSubsetting::referencedFeature () : Feature [1]`  
`SysML2::ActionUsage.allInstances()`  
`->any(m | m.qualifiedName = 'Actions::Action::done')`

#### 7.7.3.3.23 ControlFlowSuccessionAsUsage\_Mapping

##### Description

A `UML4SysML::ControlFlow` without a guard condition is mapped to a `SysMLv2 SuccessionAsUsage`.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```

action def SysMLv1Activity {
    action sysMLv1Action1;
    succession sysMLv1ControlFlow
        first sysMLv1Action1 then sysMLv1Action2;
    action sysMLv1Action2;
}

```

## General Mappings

NamedElementMain\_Mapping  
CommonActivityEdgeSuccessionAsUsage\_Mapping

## Mapping Source

ControlFlow

## Mapping Target

SuccessionAsUsage

## Owned Mappings

(none)

## Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
src.guard.oclIsUndefined()
```

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- SuccessionAsUsage::ownedRelationship () : Relationship [0..\*]  

```

let relationships : Set(KerML::Relationship) = Set{
if from.source.oclIsKindOf(UML::InitialNode) then
    ActivityEdgeInitialNodeFeatureMembership_Mapping.getMapped(from.source)
else
    ActivityEdgeSourceEndFeatureMembership_Mapping.getMapped(from.source)
endif,
if from.oclIsKindOf(UML::ObjectFlow) then
    ObjectFlowGuardSuccessionTargetEndFeatureMembership_Mapping.getMapped(from)
else if from.target.oclIsKindOf(UML::FinalNode) then
    ControlFlowFinalNodeFeatureMembership_Mapping.getMapped(from.target)
else
    ControlFlowTargetFeatureMembership_Mapping.getMapped(from.target)
endif
endif} in
let relationshipsWithGuard : Set(KerML::Relationship) =
if from.guard.oclIsUndefined() then
    relationships
else
    relationships
->including(ElementFeatureMembership_Mapping.getMapped(from.guard))
endif in

```

```

let relationshipsConsideringWeight : Set(KerML::Relationship) =
if from.weight.oclIsUndefined() then
    relationshipsWithGuard
else
    relationshipsWithGuard
    ->including(ActivityEdgeMetadataOwningMembership_Mapping.getMapped(from))
endif in

(if Helper.hasStereotypeApplied(from, 'SysML::Activities::Probability') then
    relationshipsConsideringWeight
    ->including(ProbabilityOwningMembership_Mapping.getMapped(from))
else
    relationshipsConsideringWeight
endif)->union(self.oclAsType(ElementMain_Mapping).ownedRelationship())

```

### 7.7.3.3.24 ControlFlowTargetFinalNode\_Mapping

#### Description

The mapping class maps a UML4SysML::FinalNode to a Feature which will be subsetted by Actions::Action::done. The subsetting is created by the mapping class ControlFlowTargetFinalNodeSubsetting\_Mapping.

#### General Mappings

ToFeature\_Init  
Mapping

#### Mapping Source

FinalNode

#### Mapping Target

Feature

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Feature::ownedRelationship () : Relationship [0..\*]  
Set{ControlFlowTargetFinalNodeSubsetting\_Mapping.getMapped(from) }
- Feature::isEnd () : Boolean [1]  
true

### 7.7.3.3.25 ControlFlowTargetEndFeature\_Mapping

#### Description

The mapping class maps the UML4SysML::ActivityNode to a Feature which is subsetted by the mapping target of the UML4SysML::ActivityNode. The subsetting is created by the mapping class ControlFlowTargetEndSubsetting\_Mapping.

### General Mappings

ToFeature\_Init  
Mapping

### Mapping Source

ActivityNode

### Mapping Target

Feature

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Feature::isEnd () : Boolean [1]  
`true`
- Feature::ownedRelationship () : Relationship [0..\*]  
`Set{ControlFlowTargetEndSubsetting_Mapping.getMapped(from) }`

### 7.7.3.3.26 ControlFlowTargetFeatureMembership\_Mapping

#### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

### General Mappings

ToEndFeatureMembership\_Init  
Mapping

### Mapping Source

ActivityNode

### Mapping Target

EndFeatureMembership

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `EndFeatureMembership::ownedMemberFeature () : Feature [1]`  
`ControlFlowTargetEndFeature_Mapping.getMapped (from)`

#### 7.7.3.3.27 ControlFlowTargetEndSubsetting\_Mapping

##### Description

Creates a subsetting relationship.

##### General Mappings

ToReferenceSubsetting\_Init  
Mapping

##### Mapping Source

ActivityNode

##### Mapping Target

ReferenceSubsetting

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceSubsetting::referencedFeature () : Feature [1]`  
`from`

#### 7.7.3.3.28 ControlFlowTransitionUsageFeatureMembership\_Mapping

##### Description

Creates a feature membership relationship for *ownedMemberFeature()*.



## General Mappings

ToFeatureMembership\_Init  
Mapping

## Mapping Source

ControlFlow

## Mapping Target

TransitionFeatureMembership

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- TransitionFeatureMembership::ownedMemberFeature () : Feature [1]  

```
if from.guard.oclIsKindOf (UML::OpaqueExpression) then
    OpaqueExpressionAsValue_Mapping.getMapped (from.guard)
else
    from.guard
endif
```
- TransitionFeatureMembership::kind () : TransitionFeatureKind [1]  

```
KerML::TransitionFeatureKind::guard
```

### 7.7.3.3.29 ControlNodeObjectFlowFeatureMembership\_Mapping

#### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

## General Mappings

ToFeatureMembership\_Init  
UniqueMapping

## Mapping Source

ObjectFlow

## Mapping Target

FeatureMembership

## Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureMembership::ownedMemberFeature () : Feature [1]`  
`ControlNodeObjectFlowReferenceUsage_Mapping.getMapped(from)`

### 7.7.3.3.30 ControlNodeObjectFlowFeatureValue\_Mapping

#### Description

Creates a feature value relationship.

#### General Mappings

ToFeatureValue\_Init  
UniqueMapping

#### Mapping Source

ObjectFlow

#### Mapping Target

FeatureValue

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureValue::value () : Expression [1]`  

```
if from.source.oclIsTypeOf(UML::ForkNode) then
  ForkNodeObjectFlowFeatureReferenceExpression_Mapping.getMapped(from)
else if from.source.oclIsTypeOf(UML::JoinNode)
  or from.source.oclIsTypeOf(UML::MergeNode) then
  JoinMergeNodeObjectFlowOperatorExpression_Mapping.getMapped(from)
else
  OclUndefined
endif endif
```

### 7.7.3.3.31 ControlNodeObjectFlowReferenceUsage\_Mapping

#### Description

Creates a reference usage.

#### General Mappings

ToReferenceUsage\_Init  
UniqueMapping

#### Mapping Source

ObjectFlow

#### Mapping Target

ReferenceUsage

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReferenceUsage::direction () : FeatureDirectionKind [0..1]  

```
if from.target.oclIsTypeOf(UML::ForkNode)
  or from.target.oclIsTypeOf(UML::JoinNode)
  or from.target.oclIsTypeOf(UML::MergeNode) then
  KerML::FeatureDirectionKind::_in'
else if from.source.oclIsTypeOf(UML::ForkNode)
  or from.target.oclIsTypeOf(UML::JoinNode)
  or from.target.oclIsTypeOf(UML::MergeNode) then
  KerML::FeatureDirectionKind::_out'
else
  OclUndefined
endif endif
```
- ReferenceUsage::isUnique () : Boolean [1]  

```
if from.source.oclIsTypeOf(UML::JoinNode) then
  if from.source.oclAsType(UML::JoinNode).isCombineDuplicate then
    true
  else
    false
  endif
else
  true
endif
```
- ReferenceUsage::ownedRelationship () : Relationship [0..\*]

```

if from.source.ocIsTypeOf(UML::ForkNode)
  or from.source.ocIsTypeOf(UML::JoinNode)
  or from.source.ocIsTypeOf(UML::MergeNode) then
    Set{ControlNodeObjectFlowFeatureValue_Mapping.getMapped(from)}
else
  Set{}
endif

```

- ReferenceUsage::declaredName () : String [0..1]

```

if from.target.ocIsTypeOf(UML::ForkNode)
  or from.target.ocIsTypeOf(UML::JoinNode)
  or from.target.ocIsTypeOf(UML::MergeNode) then
  'inputObject' + from.target.incoming->indexOf(from).toString()
else if from.source.ocIsTypeOf(UML::ForkNode)
  or from.source.ocIsTypeOf(UML::JoinNode)
  or from.target.ocIsTypeOf(UML::MergeNode) then
  'outputObject' + from.source.outgoing->indexOf(from).toString()
else
  OclUndefined
endif endif

```

### 7.7.3.3.32 DataStoreNode\_Mapping

#### Description

The mapping of the UML4SysML::DataStoreNode is not defined in detail yet. It will be an action usage which contains the behavior of a data store node.

#### General Mappings

CentralBufferNode\_Mapping

#### Mapping Source

DataStoreNode

#### Mapping Target

ActionUsage

#### Owned Mappings

(none)

#### Applicable filters

(none)

### 7.7.3.3.33 DecisionNode\_Mapping

#### Description

The UML4SysML::DecisionNode is mapped to a SysMLv2 DecisionNode.

There is no suitable element in SysML v2 for the else condition of an outgoing UML4SysML::ActivityEdge. Therefore, it is mapped to a TextualRepresentation with language "SysML v1" and body "else" (see ExpressionElse\_Mapping class).

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```

action def SysMLv1Activity {
  action sysMLv1Action1;
  succession sysMLv1ControlFlow1 first sysMLv1Action1 then sysMLv1DecisionNode;
  decide sysMLv1DecisionNode;
  succession sysMLv1ControlFlow2 first sysMLv1DecisionNode if {
    return : ScalarValues::Boolean;
    // guard expression, for example, opaque expression
  }.result then sysMLv1Action2;
  succession flow2 first sysMLv1DecisionNode if {
    return : ScalarValues::Boolean;
    language "SysMLv1"
    /*
     * else
     */
  }.result then sysMLv1Action2;
  action sysMLv1Action2;
}

```

## General Mappings

ToUsage\_Init  
NamedElementMain\_Mapping

## Mapping Source

DecisionNode

## Mapping Target

DecisionNode

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- DecisionNode::isComposite () : Boolean [1]  
true

### 7.7.3.34 FlowFinalNodeMembership\_Mapping

#### Description

The mapping class creates a membership relationship to the action usage library element Actions::Action::done.

## General Mappings

ToMembership\_Init  
Mapping

### Mapping Source

FlowFinalNode

### Mapping Target

Membership

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Membership::memberElement () : Element [1]

```
SysMLv2::ActionUsage.allInstances()  
->any(e | e.qualifiedName = 'Actions::Action::done')
```

### 7.7.3.35 ForkNode\_Mapping

#### Description

A UML4SysML::ForkNode is mapped to a SysMLv2 ForkNode. If object flows are connected with the UML4SysML::ForkNode, corresponding input and output parameters are created to transfer the objects through the ForkNode.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
action def SysMLv1Activity {  
  succession cf1 first sysMLv1Action1 then sysMLv1ForkNodeA;  
  succession cf2 first sysMLv1Action2 then sysMLv1ForkNodeA;  
  succession cf3 first sysMLv1ForkNodeA then sysMLv1Action4;  
  
  succession flow of1 from sysMLv1Action1.result to sysMLv1ForkNodeB.inputObject1;  
  succession flow of2 from sysMLv1ForkNodeB.outputObject1 to sysMLv1Action2.inputValue;  
  succession flow of3 from sysMLv1ForkNodeB.outputObject2 to sysMLv1Action3.inputValue;  
  
  fork sysMLv1ForkNodeA;  
  fork sysMLv1ForkNodeB {  
    in ref inputObject1;  
    out ref outputObject1 = inputObject1;  
    out ref outputObject2 = inputObject1;  
  }  
  action sysMLv1Action1 {  
    out item result;  
  }  
  action sysMLv1Action2 {
```

```

        in item inputValue;
    }
    action sysMLv1Action3 {
        in item inputValue;
    }
    action sysMLv1Action4;
}

```

## General Mappings

CommonAction\_Mapping

## Mapping Source

ForkNode

## Mapping Target

ForkNode

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ForkNode::ownedRelationship () : Relationship [0..\*]

```

if not (src.incoming->forAll(e | e.oclIsTypeOf(UML::ControlFlow))
and src.outgoing->forAll(e | e.oclIsTypeOf(UML::ControlFlow))) then
    from.ownedElement->collect(e |
        ElementOwningMembership_Mapping.getMapped(e))->asSet()
->union(from.incoming
    ->collect(i |
        ControlNodeObjectFlowFeatureMembership_Mapping.getMapped(i))->asSet())
->union(from.outgoing
    ->collect(i |
        ControlNodeObjectFlowFeatureMembership_Mapping.getMapped(i))->asSet())
else
    Set{}
endif

```

### 7.7.3.36 ForkNodeObjectFlowFeatureReferenceExpression\_Mapping

#### Description

Creates a feature reference expression.

#### General Mappings

UniqueMapping  
ToFeatureReferenceExpression\_Init

#### Mapping Source

ObjectFlow

#### Mapping Target

FeatureReferenceExpression

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureReferenceExpression::ownedRelationship () : Relationship [0..\*]  

```
Set { ForkNodeObjectFlowMembership_Mapping.getMapped(from) }  
->including (ReturnParameterFeatureMembership_Factory.create())
```

### 7.7.3.37 ForkNodeObjectFlowMembership\_Mapping

#### Description

Creates a membership relationship for *memberElement()*.

#### General Mappings

ToMembership\_Init  
UniqueMapping

#### Mapping Source

ObjectFlow

#### Mapping Target

Membership

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules



In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Membership::memberElement () : Element [1]

```
ControlNodeObjectFlowReferenceUsage_Mapping.getMapped(  
    from.source.oclaSType(UML::ForkNode).incoming  
    ->asOrderedSet()->first())
```

#### 7.7.3.3.38 JoinMergeNodeObjectFlowFeature\_Mapping

##### Description

Creates a feature for the operator expression created by JoinMergeNodeObjectFlowOperatorExpression\_Mapping.

##### General Mappings

ToFeature\_Init  
UniqueMapping

##### Mapping Source

ObjectFlow

##### Mapping Target

Feature

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Feature::ownedRelationship () : Relationship [0..\*]

```
Set{JoinMergeNodeObjectFlowFeatureValue_Mapping.getMapped(from)}
```

#### 7.7.3.3.39 JoinMergeNodeObjectFlowFeatureReferenceExpression\_Mapping

##### Description

Creates a feature reference expression.

##### General Mappings

ToFeatureReferenceExpression\_Init  
UniqueMapping

##### Mapping Source

ObjectFlow

### Mapping Target

FeatureReferenceExpression

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureReferenceExpression::ownedRelationship () : Relationship [0..\*]  

```
Set{JoinMergeNodeObjectFlowMembership_Mapping.getMapped(from) }  
->including(ReturnParameterFeatureMembership_Factory.create())
```

## 7.7.3.3.40 JoinMergeNodeObjectFlowFeatureValue\_Mapping

### Description

Creates a feature value relationship.

### General Mappings

UniqueMapping  
ToFeatureValue\_Init

### Mapping Source

ObjectFlow

### Mapping Target

FeatureValue

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureValue::value () : Expression [1]

`JoinMergeNodeObjectFlowFeatureReferenceExpression_Mapping.getMapped(from)`

#### 7.7.3.3.41 JoinMergeNodeObjectFlowMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

##### General Mappings

ToMembership\_Init  
UniqueMapping

##### Mapping Source

ObjectFlow

##### Mapping Target

Membership

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Membership::memberElement () : Element [1]

`ControlNodeObjectFlowReferenceUsage_Mapping.getMapped(from)`

#### 7.7.3.3.42 JoinMergeNodeObjectFlowOperatorExpression\_Mapping

##### Description

Creates an operator expression to combine the input objects.

##### General Mappings

ToOperatorExpression\_Init  
UniqueMapping

##### Mapping Source

ObjectFlow

##### Mapping Target

OperatorExpression

##### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `OperatorExpression::ownedRelationship () : Relationship [0..*]`  

```
if from.source.ocIsKindOf(UML::ControlNode) then
  from.source.ocAsType(UML::ControlNode).incoming
  ->collect(o | JoinMergeNodeObjectFlowParameterMembership_Mapping.getMapped(o))
  ->including(ReturnParameterFeatureMembership_Factory.create())
else
  Set{}
endif
```
- `OperatorExpression::operator () : String [1]`  

```
' , '
```

#### 7.7.3.3.43 JoinMergeNodeObjectFlowParameterMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

##### General Mappings

ToParameterMembership\_Init  
UniqueMapping

##### Mapping Source

ObjectFlow

##### Mapping Target

ParameterMembership

##### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ParameterMembership::ownedMemberParameter () : Feature [1]`

```
JoinMergeNodeObjectFlowFeature_Mapping.getMapped(from)
```

#### 7.7.3.3.44 InitialNodeMembership\_Mapping

##### Description

The mapping class creates a membership relationship to the action usage library element `Actions::Action::start`.

##### General Mappings

ToMembership\_Init  
Mapping

##### Mapping Source

InitialNode

##### Mapping Target

Membership

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Membership::memberName () : String [0..1]`  

```
if from.name = '' then null else from.name endif
```
- `Membership::memberElement () : Element [1]`  

```
SysMLv2::ActionUsage.allInstances()  
->any(e | e.qualifiedName = 'Actions::Action::start')
```

#### 7.7.3.3.45 JoinNode\_Mapping

##### Description

A `UML4SysML::JoinNode` is mapped to a `SysMLv2 JoinNode`. If object flows are connected with the `UML4SysML::JoinNode`, corresponding input and output parameters are created to transfer the objects through the `JoinNode`.

The output object is specified as follows if `UML4SysML::JoinNode::isCombineDuplicate` is false:

```
out ref outputObject1 nonunique = (inputObject1, inputObject2)
```

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```

action def SysMLv1Activity {
  succession cf1 first sysMLv1Action1 then sysMLv1JoinNodeA;
  succession cf2 first sysMLv1Action2 then sysMLv1JoinNodeA;

  succession flow of1 from sysMLv1Action2.result to sysMLv1JoinNodeB.inputObject1;
  succession flow of2 from sysMLv1Action3.result to sysMLv1JoinNodeB.inputObject2;
  succession flow of3 from sysMLv1JoinNodeB.outputObject1 to sysMLv1Action4.inputValue;

  join sysMLv1JoinNodeA;
  join sysMLv1JoinNodeB {
    in ref inputObject1;
    in ref inputObject2;
    out ref outputObject1 = (inputObject1, inputObject2);
  }
  action sysMLv1Action1;
  action sysMLv1Action2 {
    out item result;
  }
  action sysMLv1Action3 {
    out item result;
  }
  action sysMLv1Action4 {
    in item inputValue;
  }
}

```

## General Mappings

CommonAction\_Mapping

### Mapping Source

JoinNode

### Mapping Target

JoinNode

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- JoinNode::ownedRelationship () : Relationship [0..\*]

```

if not (src.incoming->forAll(e | e.ocIsTypeOf(UML::ControlFlow))
and src.outgoing->forAll(e | e.ocIsTypeOf(UML::ControlFlow))) then
  from.ownedElement
    ->collect(e | ElementOwningMembership_Mapping.getMapped(e))->asSet()
    ->union(from.incoming

```

```

        ->collect(i |
            ControlNodeObjectFlowFeatureMembership_Mapping.getMapped(i)) ->asSet()
    ->union(from.outgoing
        ->collect(i |
            ControlNodeObjectFlowFeatureMembership_Mapping.getMapped(i)) ->asSet()
    else
        Set{}
    endif

```

### 7.7.3.3.46 MergeNode\_Mapping

#### Description

A UML4SysML::MergeNode is mapped to a SysMLv2 MergeNode. If object flows are connected with the UML4SYsML::MergeNode, corresponding input and output parameters are created to transfer the objects through the MergeNode.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```

action def SysMLv1Activity {
    succession cf1 first sysMLv1Action1 then sysMLv1MergeNodeA;
    succession cf2 first sysMLv1Action2 then sysMLv1MergeNodeA;

    succession flow of1 from sysMLv1Action2.result to sysMLv1MergeNodeB.inputObject1;
    succession flow of2 from sysMLv1Action3.result to sysMLv1MergeNodeB.inputObject2;
    succession flow of3 from sysMLv1MergeNodeB.outputObject1 to sysMLv1Action4.inputValue;

    merge sysMLv1MergeNodeA;
    merge sysMLv1MergeNodeB {
        in ref inputObject1;
        in ref inputObject2;
        out ref outputObject1 = (inputObject1, inputObject2);
    }
    action sysMLv1Action1;
    action sysMLv1Action2 {
        out item result;
    }
    action sysMLv1Action3 {
        out item result;
    }
    action sysMLv1Action4 {
        in item inputValue;
    }
}

```

#### General Mappings

CommonAction\_Mapping

#### Mapping Source

MergeNode

#### Mapping Target

MergeNode

#### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- MergeNode::ownedRelationship () : Relationship [0..\*]

```
if not (src.incoming->forAll(e | e.ocIsTypeOf(UML::ControlFlow))
and src.outgoing->forAll(e | e.ocIsTypeOf(UML::ControlFlow))) then
  from.ownedElement
    ->collect(e | ElementOwningMembership_Mapping.getMapped(e)) ->asSet()
    ->union(from.incoming
      ->collect(i |
        ControlNodeObjectFlowFeatureMembership_Mapping.getMapped(i)) ->asSet())
    ->union(from.outgoing
      ->collect(i |
        ControlNodeObjectFlowFeatureMembership_Mapping.getMapped(i)) ->asSet())
else
  Set{}
endif
```

#### 7.7.3.3.47 ObjectFlow\_Mapping

##### Description

A UML4SysML::ObjectFlowFlow without a guard condition is mapped to a SysMLv2 SuccessionFlowUsage.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
action def SysMLv1Acticity {
  action sysMLv1Action1 {
    out outputValue;
  }
  succession flow sysMLv1ObjectFlow of ScalarValues::String
    from sysMLv1Action1.outputValue to sysMLv1Action1.inputValue;
  action sysMLv1Action2 {
    out inputValue;
  }
}
```

##### General Mappings

ToConnector\_Init  
NamedElementMain\_Mapping  
ToFlowUsage\_Init

##### Mapping Source

ObjectFlow

##### Mapping Target



SuccessionFlowUsage

## Owned Mappings

(none)

## Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
src.guard.oclIsUndefined()  
and (not src.target.oclIsTypeOf(UML::ActivityFinalNode))
```

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- SuccessionFlowUsage::ownedRelationship () : Relationship [0..\*]

```
let relationships : Set(KerML::Relationship) =  
  let sourceFeatureMembership : KerML::FeatureMembership  
    = ObjectFlowEndFeatureMembership_Mapping.getMapped(from.source) in  
  let targetFeatureMembership : KerML::FeatureMembership  
    = ObjectFlowEndFeatureMembership_Mapping.getMapped(from.target) in  
  if from.source.oclIsKindOf(UML::ObjectNode) then  
    Set{ObjectFlowItemFeatureMembership_Mapping.getMapped(from),  
      sourceFeatureMembership, targetFeatureMembership}  
  else  
    Set{sourceFeatureMembership, targetFeatureMembership}  
  endif in  
  
let relationshipsConsideringWeight : Set(KerML::Relationship) =  
if from.weight.oclIsUndefined() then  
  relationships  
else  
  relationships  
  ->including (ActivityEdgeMetadataOwningMembership_Mapping.getMapped(from))  
endif in  
  
let relationshipsConsideringRate : Set(KerML::Relationship) =  
if (Helper.hasStereotypeApplied(from, 'SysML::Activities::Rate') or  
  Helper.hasStereotypeApplied(from, 'SysML::Activities::Discrete') or  
  Helper.hasStereotypeApplied(from, 'SysML::Activities::Continuous')) then  
  
  relationshipsConsideringWeight  
  ->including (RateOwningMembership_Mapping.getMapped(from))  
else  
  relationshipsConsideringWeight  
endif in  
  
self.oclAsType(ElementMain_Mapping).ownedRelationship()->union(  
  if Helper.hasStereotypeApplied(from, 'SysML::Activities::Probability') then  
    relationshipsConsideringRate  
    ->including (ProbabilityOwningMembership_Mapping.getMapped(from))  
  else  
    relationshipsConsideringRate  
  endif  
)
```

#### 7.7.3.3.48 ObjectFlowFeatureMembership\_Mapping

##### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

##### General Mappings

ToFeatureMembership\_Init  
Mapping

##### Mapping Source

ObjectFlow

##### Mapping Target

FeatureMembership

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureMembership::ownedMemberFeature () : Feature [1]  
`ObjectFlow_Mapping.getMapped(from)`

#### 7.7.3.3.49 ObjectFlowGuardFeatureMembership\_Mapping

##### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

##### General Mappings

ToFeatureMembership\_Init  
Mapping

##### Mapping Source

ObjectFlow

##### Mapping Target

FeatureMembership

##### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureMembership::ownedMemberFeature () : Feature [1]

```
ObjectFlowGuard_Mapping.getMapped(from)
```

#### 7.7.3.3.50 ObjectFlowGuard\_Mapping

##### Description

A UML4SysML::ObjectFlowFlow with a guard condition is mapped to a combined SysMLv2 TransitionUsage and SysMLv2 SuccessionFlowUsage.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
action def SysMLv1Activity {
  action sysMLv1Action1 {
    out outputValue;
  }

  first sysMLv1Action1 if guardCondition.result then sysMLv1ObjectFlow {
    calc guardCondition {
      return : ScalarValues::Boolean;
      language "English"
    /*
     * guard says ok
     */
  }
}
succession flow sysMLv1ObjectFlow of SysMLv1Block from
  sysMLv1Action1.outputValue to sysMLv1Action2.inputValue;

action sysMLv1Action2 {
  out inputValue;
}
}
```

##### General Mappings

ToTransitionUsage\_Init  
NamedElementMain\_Mapping

##### Mapping Source

ObjectFlow

##### Mapping Target

TransitionUsage

### Owned Mappings

(none)

### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
(not src.guard.oclIsUndefined())  
and (not src.target.oclIsTypeOf(UML::ActivityFinalNode))
```

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- TransitionUsage::ownedRelationship () : Relationship [0..\*]

```
Set{  
  ActivityEdgeTransitionUsageSourceMembership_Mapping.getMapped(from.source),  
  CommonParameterReferenceUsageInMembership_Mapping.getMapped(from.source),  
  ObjectFlowTransitionUsageFeatureMembership_Mapping.getMapped(from),  
  ObjectFlowGuardSuccessionTargetEndFeatureMembership_Mapping.getMapped(from),  
  CommonActivityEdgeSuccessionAsUsage_Mapping.getMapped(from),  
  CommonReturnParameterReferenceUsageMembership_Mapping.getMapped(from)  
}->union(self.oclAsType(ElementMain_Mapping).ownedRelationship())
```

#### 7.7.3.3.51 ObjectFlowGuardSuccessionTargetEndFeature\_Mapping

##### Description

Creates a feature element for the UML4SysML::ObjectFlow mapping.

##### General Mappings

ToFeature\_Init  
Mapping

##### Mapping Source

ObjectFlow

##### Mapping Target

Feature

##### Owned Mappings

- objectFlowGuardSuccessionTargetEndSubsetting :  
ObjectFlowGuardSuccessionTargetEndSubsetting\_Mapping

##### Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Feature::isEnd () : Boolean [1]`  
`true`
- `Feature::ownedRelationship () : Relationship [0..*]`  
`Set{objectFlowGuardSuccessionTargetEndSubsetting.to}`

### 7.7.3.3.52 ObjectFlowGuardSuccessionTargetEndFeatureMembership\_Mapping

#### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

#### General Mappings

ToEndFeatureMembership\_Init  
Mapping

#### Mapping Source

ObjectFlow

#### Mapping Target

EndFeatureMembership

#### Owned Mappings

(none)

#### Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `EndFeatureMembership::ownedMemberFeature () : Feature [1]`  
`ObjectFlowGuardSuccessionTargetEndFeature_Mapping.getMapped(from)`

### 7.7.3.3.53 ObjectFlowGuardSuccessionTargetEndSubsetting\_Mapping

#### Description

Creates a subsetting relationship.

#### General Mappings

ToSubsetting\_Init  
Mapping

#### **Mapping Source**

ObjectFlow

#### **Mapping Target**

Subsetting

#### **Owned Mappings**

- objectFlowGuardSuccessionTargetEndFeature : ObjectFlowGuardSuccessionTargetEndFeature\_Mapping

#### **Applicable filters**

(none)

#### **Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Subsetting::subsettingFeature () : Feature [1]  
`ObjectFlow_Mapping.getMapped(from)`
- Subsetting::subsettingFeature () : Feature [1]  
`objectFlowGuardSuccessionTargetEndFeature.to`

### **7.7.3.3.54 ObjectFlowItemFeature\_Mapping**

#### **Description**

The mapping class maps the source UML4SysML::ObjectNode to a ItemFeature which is typed by the UML4SysML::ObjectNode type.

#### **General Mappings**

ObjectFlowItemFeatureUntyped\_Mapping

#### **Mapping Source**

ObjectNode

#### **Mapping Target**

PayloadFeature

#### **Owned Mappings**

(none)

#### **Applicable filters**

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `PayloadFeature::ownedRelationship () : Relationship [0..*]`  
`Set{ObjectFlowItemFeatureTyping_Mapping.getMapped(from) }`

#### 7.7.3.3.55 ObjectFlowItemFeatureMembership\_Mapping

##### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

##### General Mappings

ToFeatureMembership\_Init  
Mapping

##### Mapping Source

ObjectFlow

##### Mapping Target

FeatureMembership

##### Owned Mappings

(none)

##### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureMembership::ownedMemberFeature () : Feature [1]`  
`if from.source.type.ocIsUndefined() then`  
`ObjectFlowItemFeatureUntyped_Mapping.getMapped(from.source)`  
`else`  
`ObjectFlowItemFeature_Mapping.getMapped(from.source)`  
`endif`

#### 7.7.3.3.56 ObjectFlowItemFeatureTyping\_Mapping

##### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

##### General Mappings

TypedElementFeatureTyping\_Mapping

**Mapping Source**

ObjectNode

**Mapping Target**

FeatureTyping

**Owned Mappings**

(none)

**Applicable filters**

(none)

**7.7.3.3.57 ObjectFlowItemFeatureUntyped\_Mapping**

**Description**

The mapping class maps the source UML4SysML::ObjectNode to a ItemFeature without a type.

**General Mappings**

ToFeature\_Init

**Mapping Source**

ObjectNode

**Mapping Target**

PayloadFeature

**Owned Mappings**

(none)

**Applicable filters**

(none)

**7.7.3.3.58 ObjectFlowEndFeatureMembership\_Mapping**

**Description**

Creates a feature membership relationship for *ownedMemberFeature()*.

**General Mappings**

ToEndFeatureMembership\_Init  
Mapping

**Mapping Source**



ActivityNode

### Mapping Target

EndFeatureMembership

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- EndFeatureMembership::ownedMemberFeature () : Feature [1]

`ObjectFlowItemFlowEnd_Mapping.getMapped(from)`

### 7.7.3.3.59 ObjectFlowItemFlowEnd\_Mapping

#### Description

The mapping class maps a UML4SysML::ActivityNode to a ItemFlowEnd which is subsetting by the transformation target of the UML4SysML::ActivityNode.

#### General Mappings

ToFeature\_Init  
Mapping

#### Mapping Source

ActivityNode

#### Mapping Target

FlowEnd

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FlowEnd::ownedRelationship () : Relationship [0..\*]

```
Set{ObjectFlowItemFlowEndSubsetting_Mapping.getMapped(from),
ObjectFlowItemFlowEndFeatureMembership_Mapping.getMapped(from)}
```

- FlowEnd::isEnd () : Boolean [1]

```
true
```

### 7.7.3.3.60 ObjectFlowItemFlowEndReferenceUsage\_Mapping

#### Description

Creates a feature element for the UML4SysML::ObjectFlow mapping.

#### General Mappings

ToReferenceUsage\_Init  
Mapping

#### Mapping Source

ActivityNode

#### Mapping Target

ReferenceUsage

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReferenceUsage::ownedRelationship () : Relationship [0..\*]

```
let redefinition : KerML::Redefinition =
  if from.owner.oclIsTypeOf(UML::AddVariableValueAction) or
  from.owner.oclIsTypeOf(UML::AddStructuralFeatureValueAction) then
    if from.name = 'value' then
      ObjectFlowItemFlowEndRedefinition_Factory.create(
        SYSML2::ReferenceUsage.allInstances()
        ->any(m | m.qualifiedName = 'SysMLv1Library::AddValueAction::value'))
    else if from.name = 'insertAt' then
      ObjectFlowItemFlowEndRedefinition_Factory.create(
        SYSML2::ReferenceUsage.allInstances()
        ->any(m | m.qualifiedName = 'SysMLv1Library::AddValueAction::insertAt'))
    else if from.owner.oclIsTypeOf(UML::AddStructuralFeatureValueAction)
    and (from.name = 'object') then
      ObjectFlowItemFlowEndRedefinition_Factory.create(
        SYSML2::ReferenceUsage.allInstances()
        ->any(m | m.qualifiedName =
          'SysMLv1Library::AddStructuralFeatureValueAction::object'))
```

```

        else
            ObjectFlowItemFlowEndRedefinition_Factory.create(
                ElementMain_Mapping.getMapped(from))
        endif endif endif
    else
        if from.ocIsTypeOf(UML::ActivityParameterNode) then
            ObjectFlowItemFlowEndRedefinition_Factory.create(
                ElementMain_Mapping.getMapped(from.ocIsTypeOf(
                    UML::ActivityParameterNode).parameter))
        else if from.ocIsTypeOf(UML::FlowFinalNode) then
            ObjectFlowItemFlowEndRedefinition_Factory.create(
                ElementMain_Mapping.getMapped(
                    SysMLv2::ActionUsage.allInstances()
                    ->any(e | e.qualifiedName = 'Actions::Action::done'))
            )
        else
            ObjectFlowItemFlowEndRedefinition_Factory.create(
                ElementMain_Mapping.getMapped(from))
        endif endif
    endif in
    Set{redefinition}

```

### 7.7.3.3.61 ObjectFlowItemFlowEndFeatureMembership\_Mapping

#### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

#### General Mappings

ToFeatureMembership\_Init  
Mapping

#### Mapping Source

ActivityNode

#### Mapping Target

FeatureMembership

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureMembership::ownedMemberFeature () : Feature [1]  
 ObjectFlowItemFlowEndReferenceUsage\_Mapping.getMapped(from)

#### 7.7.3.3.62 ObjectFlowItemFlowEndRedefinition\_Mapping

##### Description

Creates a redefinition relationship for the *redefiningFeature()* and the *redefinedFeature()*.

##### General Mappings

ToRedefinition\_Init  
Mapping

##### Mapping Source

ActivityNode

##### Mapping Target

Redefinition

##### Owned Mappings

(none)

##### Applicable filters

(none)

#### 7.7.3.3.63 ObjectFlowItemFlowEndSubsetting\_Mapping

##### Description

Creates a subsetting relationship.

##### General Mappings

ToReferenceSubsetting\_Init  
Mapping

##### Mapping Source

ActivityNode

##### Mapping Target

ReferenceSubsetting

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReferenceSubsetting::referencedFeature () : Feature [1]

```

if from.ocIsKindOf(UML::ActivityParameterNode) then
    Parameter_Mapping.getMapped(from.parameter)
else if from.ocIsKindOf(UML::Pin) then
    CommonAction_Mapping.getMapped(from.owner)
else if from.ocIsKindOf(UML::InitialNode) then
    SysMLv2::ActionUsage.allInstances()
    ->any(e | e.qualifiedName = 'Actions::Action::start')
else if from.ocIsKindOf(UML::FinalNode) then
    SysMLv2::ActionUsage.allInstances()
    ->any(e | e.qualifiedName = 'Actions::Action::done')
else
    from
endif
endif
endif
endif

```

### 7.7.3.3.64 ObjectFlowTransitionUsageFeatureMembership\_Mapping

#### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

#### General Mappings

ToFeatureMembership\_Init  
Mapping

#### Mapping Source

ObjectFlow

#### Mapping Target

TransitionFeatureMembership

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- TransitionFeatureMembership::kind () : TransitionFeatureKind [1]

```
KerML::TransitionFeatureKind::guard
```

- TransitionFeatureMembership::ownedMemberFeature () : Feature [1]

```

    if from.guard.oclIsKindOf(UML::OpaqueExpression) then
        OpaqueExpressionAsValue_Mapping.getMapped(from.guard)
    else
        from.guard
    endif

```

### 7.7.3.3.65 VariableAttribute\_Mapping

#### Description

A UML4SysML::Variable is mapped to a SysML v2 AttributeUsage if the type of the variable is of kind UML4SysML::DataType.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```

action def SysMLv1Activity {
    private attribute sysmlv1Variable : ScalarValues::Integer;
}

```

#### General Mappings

NamedElementMain\_Mapping  
CommonVariable\_Mapping

#### Mapping Source

Variable

#### Mapping Target

AttributeUsage

#### Owned Mappings

(none)

#### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
src.type.oclIsKindOf(UML::DataType)
```

#### Mapping rules

The mapping class only has inherited rules. See the mapping classes in the general mapping section for details.

### 7.7.3.3.66 VariableFeatureTyping\_Mapping

#### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

#### General Mappings

TypedElementFeatureTyping\_Mapping

## Mapping Source

Variable

## Mapping Target

FeatureTyping

## Owned Mappings

(none)

## Applicable filters

(none)

### 7.7.3.3.67 VariableItem\_Mapping

#### Description

A UML4SysML::Variable is mapped to a SysML v2 ItemUsage if the type of the variable is not of kind UML4SysML::DataType.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
action def SysMLv1Activity {  
  private item sysmlv1Variable : SysMLv1Block;  
}  
part def SysMLv1Block;
```

## General Mappings

NamedElementMain\_Mapping  
CommonVariable\_Mapping

## Mapping Source

Variable

## Mapping Target

ItemUsage

## Owned Mappings

(none)

## Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
not src.type.ocIsKindOf(UML::DataType)
```

## Mapping rules

The mapping class only has inherited rules. See the mapping classes in the general mapping section for details.

### 7.7.3.3.68 VariableMembership\_Mapping

#### Description

Creates a membership relationship for *memberElement()*.

#### General Mappings

ElementFeatureMembership\_Mapping

#### Mapping Source

Variable

#### Mapping Target

FeatureMembership

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureMembership::visibility () : VisibilityKind [1]

`KerML::VisibilityKind::private`

## 7.7.4 Classification

### 7.7.4.1 Overview

Table 5. List of all mappings

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
Generalization	Subclassification
GeneralizationSet	not mapped; see next section
InstanceSpecification	PartUsage ConnectionUsage
InstanceValue	FeatureReferenceExpression
Operation	PerformActionUsage
Parameter	ReferenceUsage
ParameterSet	not mapped; see next section



SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
Property	OccurrenceUsage Feature ReferenceUsage AttributeUsage
Slot	Feature
Substitution	Dependency

## 7.7.4.2 Mapping Specifications

### 7.7.4.2.1 BehavioralFeature\_Mapping

#### Description

The mapping class is the abstract base class for UML4SysML::BehavioralFeature mappings.

#### General Mappings

ToUsage\_Init  
Namespace\_Mapping

#### Mapping Source

BehavioralFeature

#### Mapping Target

Usage

#### Owned Mappings

(none)

#### Applicable filters

(none)

### 7.7.4.2.2 Classifier\_Mapping

#### Description

The mapping class is the abstract base class for all mapping classes that map specializations of UML4SysML::Classifier elements.

#### General Mappings

ToClassifier\_Init  
Namespace\_Mapping

#### Mapping Source

Classifier

#### Mapping Target

Classifier

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Classifier::ownedRelationship () : Relationship [0..\*]

```
let generalizations : Set(UML::Generalization) =
    from.ownedElement->select(e | e.ocIsKindOf(UML::Generalization))->asSet() in
let toElementFMS: Set(UML::Element) =
    from.ownedElement->select(e | e.ocIsKindOf(UML::Feature))->asSet() in
let toElementOMS: Set(UML::Element) =
    ((from.ownedElement - toElementFMS) - generalizations) - from.ownedComment in
toElementOMS->collect(e | ElementOwningMembership_Mapping.getMapped(e))
->asSet()
->union(toElementFMS->collect(e | ElementFeatureMembership_Mapping.getMapped(e))
->asSet())
->union(generalizations->collect(e | Generalization_Mapping.getMapped(e))
->asSet())
->union(self.ocAsType(ElementMain_Mapping).ownedRelationship())
```

- Classifier::isAbstract () : Boolean [1]

```
from.isAbstract
```

#### 7.7.4.2.3 DefaultLowerBound\_Mapping

##### Description

The mapping class creates the default lower bound of a multiplicity element.

##### General Mappings

ToExpression\_Init  
Mapping

##### Mapping Source

Element

##### Mapping Target

LiteralInteger

##### Owned Mappings

(none)

##### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `LiteralInteger::ownedRelationship () : Relationship [0..*]`  
`Set { CommonReturnParameterFeatureMembership_Mapping.getMapped (from) }`
- `LiteralInteger::value () : Integer [1]`  
`1`

#### 7.7.4.2.4 DefaultMultiplicityBoundFeatureMembership\_Mapping

##### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

##### General Mappings

ToFeatureMembership\_Init  
Mapping

##### Mapping Source

Element

##### Mapping Target

FeatureMembership

##### Owned Mappings

(none)

##### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureMembership::isComposite () : Boolean [1]`  
`true`

#### 7.7.4.2.5 DefaultMultiplicityElement\_Mapping

##### Description

The mapping class creates a feature element representing the default multiplicity.

##### General Mappings

ToFeature\_Init  
Mapping

### Mapping Source

Element

### Mapping Target

MultiplicityRange

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- MultiplicityRange::isUnique () : Boolean [1]  
`true`
- MultiplicityRange::ownedRelationship () : Relationship [0..\*]  
`OrderedSet { DefaultMultiplicityLowerBoundFeatureMembership_Mapping.getMapped (from) ,  
DefaultMultiplicityUpperBoundFeatureMembership_Mapping.getMapped (from) }`
- MultiplicityRange::declaredName () : String [0..1]  
`'defaultMultiplicity'`

#### 7.7.4.2.6 DefaultMultiplicityLowerBoundFeatureMembership\_Mapping

##### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

##### General Mappings

DefaultMultiplicityBoundFeatureMembership\_Mapping

##### Mapping Source

Element

##### Mapping Target

FeatureMembership

##### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureMembership::ownedMemberFeature () : MultiplicityRange [1]`

`DefaultLowerBound_Mapping.getMapped(from)`

#### 7.7.4.2.7 DefaultMultiplicityMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

##### General Mappings

Mapping  
ToOwningMembership\_Init

##### Mapping Source

Element

##### Mapping Target

OwningMembership

##### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `OwningMembership::ownedMemberElement () : Element [1]`

`DefaultMultiplicityElement_Mapping.getMapped(from)`

#### 7.7.4.2.8 DefaultMultiplicityUpperBoundFeatureMembership\_Mapping

##### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

##### General Mappings

DefaultMultiplicityBoundFeatureMembership\_Mapping

### Mapping Source

Element

### Mapping Target

FeatureMembership

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureMembership::ownedMemberFeature () : MultiplicityRange [1]`  
`DefaultUpperBound_Mapping.getMapped(from)`

## 7.7.4.2.9 DefaultUpperBound\_Mapping

### Description

The mapping class creates the default upper bound of a multiplicity element.

### General Mappings

ToExpression\_Init  
Mapping

### Mapping Source

Element

### Mapping Target

LiterallInteger

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `LiteralInteger::ownedRelationship () : Relationship [0..*]`  
`Set { CommonReturnParameterFeatureMembership_Mapping.getMapped (from) }`
- `LiteralInteger::value () : Integer [1]`  
`1`

#### 7.7.4.2.10 DefaultValue\_Mapping

##### Description

The expected SysML v2 textual syntax of a mapped SysML v2 default value is as follows:

```
attribute sysMLv1Property : ScalarValues::String default := "default value";
```

##### General Mappings

ToFeatureValue\_Init  
Mapping

##### Mapping Source

Property

##### Mapping Target

FeatureValue

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureValue::isDefault () : Boolean [1]`  
`true`
- `FeatureValue::value () : Expression [1]`  
`from.defaultValue`

#### 7.7.4.2.11 ElementFeatureMembership\_Mapping

##### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

### General Mappings

ToFeatureMembership\_Init  
Mapping

### Mapping Source

Element

### Mapping Target

FeatureMembership

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureMembership::ownedMemberFeature () : Feature [1]  
`NamedElementMain_Mapping.getMapped(from)`
- FeatureMembership::visibility () : VisibilityKind [1]  

```
if from.ocIsKindOf(UML::NamedElement) then
  Helper.getKerMLVisibilityKind(from.ocAsType(UML::NamedElement).visibility)
else KerML::VisibilityKind::public endif
```

#### 7.7.4.2.12 Generalization\_Mapping

##### Description

A UML4SysML::Generalization relationship is mapped to a SysML v2 Subclassification.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
part def SysMLv1BlockGeneral;
part def SysMLv1BlockSpecial :> SysMLv1BlockGeneral;
```

### General Mappings

ToSpecialization\_Init  
ElementMain\_Mapping

### Mapping Source



Generalization

## Mapping Target

Subclassification

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Subclassification::superclassifier () : Classifier [1]  

```
if from.general.oclIsTypeOf(UML::PrimitiveType)
    and not (Helper.getScalarValueType(from.general)
            = invalid) then
    Helper.getScalarValueType(from.general)
else
    Classifier_Mapping.getMapped(from.general)
endif
```
- Subclassification::subclassifier () : Classifier [1]  

```
Classifier_Mapping.getMapped(from.specific)
```

### 7.7.4.2.13 InstanceSpecificationLink\_Mapping

#### Description

The UML4SysML::InstanceSpecification that is a link is mapped to a SysMLv2 ConnectionUsage.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
part def SysMLv1Block1;
part def SysMLv1Block2;
connection def SysMLv1Association {
    end : SysMLv1Block1[1];
    end : SysMLv1Block2[1];
}
part sysMLv1InstanceSpecification1 : SysMLv1Block1;
part sysMLv1InstanceSpecification2 : SysMLv1Block2;
connection sysMLv1Link : SysMLv1Association
    connect sysMLv1InstanceSpecification1 to sysMLv1InstanceSpecification2;
```

## General Mappings

NamedElementMain\_Mapping  
ToConnectionUsage\_Init

## Mapping Source

InstanceSpecification

## Mapping Target

ConnectionUsage

## Owned Mappings

(none)

## Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
src.classifier->select( c | c.ocIsTypeOf(UML::Association))->size() > 0
```

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ConnectionUsage::ownedRelationship () : Relationship [0..\*]

```
self.ocIsTypeOf(ElementMain_Mapping).ownedRelationship()  
->union(SlotMembership_Mapping.getMappedColl(from.slot)->asSet())  
->union(from.classifier  
  ->collect(g |  
    InstanceSpecificationFeatureTyping_Mapping.getMapped(from, g))  
  ->asSet())  
->asSet()
```

### 7.7.4.2.14 InstanceSpecification\_Mapping

#### Description

The UML4SysML::InstanceSpecification that is not a link is mapped to a SysMLv2 PartDefinition.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
part def SysMLv1Block {  
  attribute sysMLv1ValueProperty : ScalarValues::String;  
}  
  
part sysMLv1InstanceSpecification : SysMLv1Block {  
  redefines sysMLv1ValueProperty = "Hello InstanceSpecification";  
}
```

## General Mappings

NamedElementMain\_Mapping  
ToPartUsage\_Init

## Mapping Source

InstanceSpecification

### Mapping Target

PartUsage

### Owned Mappings

(none)

### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
src.classifier->select( c | c.ocIsTypeOf(UML::Association))->size() = 0
```

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- PartUsage::ownedRelationship () : Relationship [0..\*]

```
SlotMembership_Mapping.getMappedColl(from.slot)->asSet()  
->union(from.classifier  
  ->collect(g |  
    InstanceSpecificationFeatureTyping_Mapping.getMapped(from, g))  
  ->asSet())  
->union(self.ocAsType(ElementMain_Mapping).ownedRelationship())  
->asSet()
```

- PartUsage::ownedFeatureMembership () : FeatureMembership [0..\*]

```
from.classifier  
->collect(c | InstanceSpecificationToGeneralization_Mapping.getMapped(from, c))
```

#### 7.7.4.2.15 InstanceSpecificationFeatureTyping\_Mapping

##### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

##### General Mappings

ToFeatureTyping\_Init  
Mapping

##### Mapping Source

InstanceSpecification

##### Mapping Target

FeatureTyping with qualifier: classifier:Classifier

##### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureTyping::type (in classifier : Classifier) : Type [1]  
`Classifier_Mapping.getMapped(classifier)`

### 7.7.4.2.16 InstanceValue\_Mapping

#### Description

The UML4SysML::InstanceValue is mapped to a SysMLv2 FeatureReferenceExpression.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
part def SysMLv1Block1;  
part sysMLv1InstanceSpecification : SysMLv1Block1;  
part def SysMLv1Block2 {  
    part sysMLv1PartProperty : SysMLv1Block1  
        = sysMLv1InstanceSpecification;  
}
```

#### General Mappings

ValueSpecification\_Mapping

#### Mapping Source

InstanceValue

#### Mapping Target

FeatureReferenceExpression

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureReferenceExpression::ownedRelationship () : Relationship [0..\*]

```

self.oclAsType (ElementMain_Mapping) .ownedRelationship ()
->including (InstanceValueMembership_Mapping.getMapped (from.instance))
->including (ReturnParameterFeatureMembership_Factory.create ())

```

#### 7.7.4.2.17 InstanceValueMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

##### General Mappings

ToMembership\_Init  
Mapping

##### Mapping Source

InstanceSpecification

##### Mapping Target

Membership

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Membership::memberElement () : Element [1]  
from

#### 7.7.4.2.18 LowerBoundValueFeatureMembership\_Mapping

##### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

##### General Mappings

ToFeatureMembership\_Init

##### Mapping Source

MultiplicityElement

##### Mapping Target

FeatureMembership

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureMembership::ownedMemberFeature () : Feature [0..1]`  
`LiteralInteger_Mapping.getMapped(from.lowerValue)`

### 7.7.4.2.19 MultiplicityElement\_Mapping

#### Description

A `UML4SysML::MultiplicityElement` is mapped to a SysML v2 `MultiplicityRange`.

#### General Mappings

`ToFeature_Init`  
`Mapping`

#### Mapping Source

`MultiplicityElement`

#### Mapping Target

`MultiplicityRange`

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `MultiplicityRange::isUnique () : Boolean [1]`  
`from.isUnique`
- `MultiplicityRange::ownedRelationship () : Relationship [0..*]`  
`OrderedSet {MultiplicityLowerBoundOwningMembership_Mapping.getMapped(from),`  
`MultiplicityUpperBoundOwningMembership_Mapping.getMapped(from) }`

- MultiplicityRange::declaredName () : String [0..1]

```
'multiplicity'
```

#### 7.7.4.2.20 MultiplicityLowerBoundOwningMembership\_Mapping

##### Description

Creates a owning membership relationship for *ownedMemberElement()*.

##### General Mappings

ToOwningMembership\_Init  
Mapping

##### Mapping Source

MultiplicityElement

##### Mapping Target

OwningMembership

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- OwningMembership::ownedMemberElement () : Element [1]

```
if from.lowerValue.oclIsUndefined() then
    DefaultLowerBound_Mapping.getMapped(from)
else
    from.lowerValue
endif
```

- OwningMembership::memberName () : String [0..1]

```
'lowerBound'
```

#### 7.7.4.2.21 MultiplicityMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

##### General Mappings

ToOwningMembership\_Init  
Mapping

**Mapping Source**

MultiplicityElement

**Mapping Target**

OwningMembership

**Owned Mappings**

(none)

**Applicable filters**

(none)

**Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- OwningMembership::ownedMemberElement () : Element [1]  
`MultiplicityElement_Mapping.getMapped(from)`

**7.7.4.2.22 MultiplicityUpperBoundOwningMembership\_Mapping****Description**

Creates a owning membership relationship for *ownedMemberElement()*.

**General Mappings**

ToOwningMembership\_Init  
Mapping

**Mapping Source**

MultiplicityElement

**Mapping Target**

OwningMembership

**Owned Mappings**

(none)

**Applicable filters**

(none)

**Mapping rules**



In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `OwningMembership::memberName () : String [0..1]`  
`'upperBound'`
- `OwningMembership::ownedMemberElement () : Element [1]`  

```

if from.upperValue.oclIsUndefined() then
    DefaultUpperBound_Mapping.getMapped(from)
else
    from.upperValue
endif

```

#### 7.7.4.2.23 Operation\_Mapping

##### Description

A `UML4SysML::Operation` is mapped to a SysML v2 `PerformActionUsage`.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```

part def SysMLv1Block {
    perform action sysMLv1Operation {
        in parIn : ScalarValues::Boolean;
        out result : ScalarValues::Integer;
    }
}

```

##### General Mappings

BehavioralFeature\_Mapping  
 ToPerformActionUsage\_Init

##### Mapping Source

Operation

##### Mapping Target

PerformActionUsage

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `PerformActionUsage::ownedRelationship () : Relationship [0..*]`

```
let parameters: Set(UML::Element) =
  from.ownedElement->select(e | e.ocIsKindOf(UML::Parameter)) in
let parameterSets: Set(UML::Element) =
  from.ownedElement->select(e | e.ocIsKindOf(UML::ParameterSet)) in
self.ocAsType(ElementMain_Mapping).ownedRelationship()
->union(parameters->collect(e |
  ParameterMembership_Mapping.getMapped(e))->asSet())
->union(parameterSets->collect(e |
  ParameterSetMembership_Mapping.getMapped(e))->asSet())
```

#### 7.7.4.2.24 Parameter\_Mapping

##### Description

A `UML4SysML::Parameter` is mapped to a SysML v2 `ReferenceUsage`.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
action def SysMLv1Activity {
  in parIn : ScalarValues::Boolean;
}
```

##### General Mappings

ToReferenceUsage\_Init  
NamedElementMain\_Mapping

##### Mapping Source

Parameter

##### Mapping Target

ReferenceUsage

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceUsage::declaredName () : String [0..1]`

```

if from.direction = UML::ParameterDirectionKind::return then
    'result'
else
    from.name
endif

```

- ReferenceUsage::direction () : FeatureDirectionKind [0..1]

```

Helper.getKerMLParameterDirectionKind(from.direction)

```

- ReferenceUsage::ownedRelationship () : Relationship [0..\*]

```

let typings: Set(KerML::FeatureTyping) =
    if from.type.ocIsUndefined() then
        Set{}
    else
        Set{ParameterToFeatureTyping_Mapping.getMapped(from)}
    endif in
let multiplicities: Set(KerML::Relationship) =
    Set{MultiplicityMembership_Mapping.getMapped(from)} in
let defaultValues: Set(KerML::Relationship) =
    if from.defaultValue.ocIsUndefined() then
        Set{}
    else
        Set{ParameterDefaultValue_Mapping.getMapped(from)}
    endif in
self.ocAsType(ElementMain_Mapping).ownedRelationship()
->union(typings)
->union(multiplicities)
->union(defaultValues)

```

#### 7.7.4.2.25 ParameterDefaultValue\_Mapping

##### Description

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```

attribute value : ScalarValues::String default := "default value";

```

##### General Mappings

ToFeatureValue\_Init  
Mapping

##### Mapping Source

Parameter

##### Mapping Target

FeatureValue

##### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureValue::isDefault () : Boolean [1]`  
`true`
- `FeatureValue::value () : Expression [1]`  
`from.defaultValue`

### 7.7.4.2.26 ParameterMembership\_Mapping

#### Description

Creates a membership relationship for *memberElement()*.

#### General Mappings

ToParameterMembership\_Init  
Mapping

#### Mapping Source

Parameter

#### Mapping Target

ParameterMembership

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ParameterMembership::ownedMemberParameter () : Feature [1]`  
`Parameter_Mapping.getMapped(from)`

### 7.7.4.2.27 ParameterSet\_Mapping

#### Description

A UML4SysML::ParameterSet is mapped to a SysML v2 ReferenceUsage.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
action def SysMLv1Activity {
  in parIn [0..1];
  inout parInOut [0..1];
  out parOut [0..1];
  out parReturn [0..1];

  sysMLv1ParameterSet1 [1] {
    ref parIn = SysMLv1Activity::parIn;
    assert constraint sysMLv1ParameterSet1Condition {
      language "English"
    }
    /*
     * opaque expression parameter set 1
     */
  }
  sysMLv1ParameterSet2 [1] {
    ref parInOut = SysMLv1Activity::parInOut;
    ref parOut = SysMLv1Activity::parOut;
    ref parReturn = SysMLv1Activity::parReturn;
  }
}
```

## General Mappings

ToReferenceUsage\_Init  
Mapping

### Mapping Source

ParameterSet

### Mapping Target

ReferenceUsage

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReferenceUsage::ownedRelationship () : Relationship [0..\*]

```
from.parameter
->collect(p | ParameterSetParameterFeatureMembership_Mapping.getMapped(from, p))
->asSet()
```

- `ReferenceUsage::declaredName () : String [0..1]`

`from.name`

#### 7.7.4.2.28 ParameterSetMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

##### General Mappings

ToFeatureMembership\_Init  
Mapping

##### Mapping Source

ParameterSet

##### Mapping Target

FeatureMembership

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureMembership::ownedMemberFeature () : Feature [1]`

`ParameterSet_Mapping.getMapped(from)`

#### 7.7.4.2.29 ParameterSetParameterFeatureMembership\_Mapping

##### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

##### General Mappings

ToFeatureMembership\_Init  
Mapping

##### Mapping Source

ParameterSet

##### Mapping Target

FeatureMembership with qualifier: parameter:Parameter

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureMembership::ownedMemberFeature (in parameter : Parameter) : Feature [1]`  
`ParameterSetParameterReferenceUsage_Mapping.getMapped(parameter)`

### 7.7.4.2.30 ParameterSetParameterReferenceUsage\_Mapping

#### Description

The mapping class creates the reference usage element for the UML4SysML::ParameterSet mapping.

#### General Mappings

ToReferenceUsage\_Init  
Mapping

#### Mapping Source

Parameter

#### Mapping Target

ReferenceUsage

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceUsage::ownedRelationship () : Relationship [0..*]`  
`Set{ParameterSetParameterReferenceUsageFeatureValue_Mapping.getMapped(from),`  
`MultiplicityMembership_Mapping.getMapped(from) }`

### 7.7.4.2.31 ParameterSetParameterReferenceUsageFeatureValue\_Mapping

#### Description

The mapping class creates the feature reference expression for the reference usage element of the UML4SysML::ParameterSet mapping.

### General Mappings

ToFeatureValue\_Init  
Mapping

### Mapping Source

Parameter

### Mapping Target

FeatureValue

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureValue::value () : Expression [1]

```
ParameterSetParameterReferenceUsageFeatureValueExpression_Mapping.getMapped(from)
```

## 7.7.4.2.32 ParameterSetParameterReferenceUsageFeatureValueExpression\_Mapping

### Description

The mapping class creates the feature reference expression for the UML4SysML::ParameterSet mapping.

### General Mappings

ToFeatureReferenceExpression\_Init  
Mapping

### Mapping Source

Parameter

### Mapping Target

FeatureReferenceExpression

### Owned Mappings

(none)

### Applicable filters



(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureReferenceExpression::ownedRelationship () : Relationship [0..*]`

```
Set{ParameterSetParameterReferenceUsageMembership_Mapping.getMapped(from) ,  
CommonReturnParameterFeatureMembership_Mapping.getMapped(from) }
```

#### 7.7.4.2.33 ParameterSetParameterReferenceUsageMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

##### General Mappings

ToMembership\_Init  
Mapping

##### Mapping Source

Parameter

##### Mapping Target

Membership

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Membership::memberElement () : Element [1]`

```
from
```

#### 7.7.4.2.34 ParameterToFeatureTyping\_Mapping

##### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

##### General Mappings

TypedElementFeatureTyping\_Mapping  
Mapping

### **Mapping Source**

Parameter

### **Mapping Target**

FeatureTyping

### **Owned Mappings**

(none)

### **Applicable filters**

(none)

### **Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureTyping::typedFeature () : Feature [1]  
`parameter.to`

## **7.7.4.2.35 PropertyCommon\_Mapping**

### **Description**

The mapping class is the abstract base class for UML4SysML::Property mappings.

### **General Mappings**

StructuralFeature\_Mapping  
Mapping

### **Mapping Source**

Property

### **Mapping Target**

Feature

### **Owned Mappings**

(none)

### **Applicable filters**

(none)

### **Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Feature::isEnd () : Boolean [1]

```
if from.association.oclIsUndefined() then
    false
else
    from.association.ownedEnd->includes (from)
endif
```

- Feature::isComposite () : Boolean [1]

```
from.isComposite
```

- Feature::isDerived () : Boolean [1]

```
from.isDerived
```

- Feature::ownedRelationship () : Relationship [0..\*]

```
let typings: Set (KerML::FeatureTyping) = if from.type.oclIsUndefined() then
    Set{}
else
    Set{StructuralFeatureToFeatureTyping_Mapping.getMapped(from)}
endif in
let subsettings: Set (KerML::Subsetting) = from.subsettedProperty
    ->collect(p | PropertySubsetting_Mapping.getMapped(from, p))->asSet() in
let defaultValue: Set (KerML::OwningMembership) =
    if from.defaultValue.oclIsUndefined() then
        Set{}
    else
        Set{DefaultValue_Mapping.getMapped(from)}
    endif in
typings->union(subsettings)->union(defaultValue)
->including(MultiplicityMembership_Mapping.getMapped(from))->asSet()
```

#### 7.7.4.2.36 PropertySubsetting\_Mapping

##### Description

Creates a subsetting relationship.

##### General Mappings

ToSubsetting\_Init  
Mapping

##### Mapping Source

Property

##### Mapping Target

Subsetting with qualifier: subsettedProperty:Property

##### Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Subsetting::subsettingFeature (in subsettingProperty : Property) : Feature [1]  
`Property_Mapping.getMapped(subsettingProperty)`
- Subsetting::subsettingFeature () : Feature [1]  
`Property_Mapping.getMapped(from)`

### 7.7.4.2.37 PropertyTypedByClassInterface\_Mapping

#### Description

A UML4SysML::Property typed by a UML4SysML::Class or UML4SysML::Interface is mapped to a SysML v2 OccurrenceUsage.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
part def SysMLv1Block {  
    occurrence sysMLv1Property1 [0..1] : SysMLv1Class;  
    ref occurrence sysMLv1ReferencedProperty [0..1] : SysMLv1Class;  
    occurrence sysMLv1Property2 [0..1] : SysMLv1Interface;  
}
```

#### General Mappings

PropertyCommon\_Mapping  
NamedElementMain\_Mapping

#### Mapping Source

Property

#### Mapping Target

OccurrenceUsage

#### Owned Mappings

(none)

## Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
if src.oclIsTypeOf(UML::Property) then  
    let p: UML::Property = src.oclAsType(UML::Property) in
```

```

    if p.type.ocllsUndefined() then
        false
    else
        (p.type.ocllsTypeOf(UML::Class) or
         p.type.ocllsTypeOf(UML::Interface)) and
        not (p.name.indexOf('base_') > 0) and
        (p.association.ocllsUndefined() or p.association.ownedEnd->excludes(p))
    endif
else
    false
endif

```

## Mapping rules

The mapping class only has inherited rules. See the mapping classes in the general mapping section for details.

### 7.7.4.2.38 PropertyUntyped\_Mapping

#### Description

A UML4SysML::Property is mapped to a SysML v2 Feature. The mapping class maps properties without a type.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```

part def SysMLv1Block {
    attribute sysMLv1Property;
}

```

#### General Mappings

PropertyCommon\_Mapping  
 ToReferenceUsage\_Init  
 NamedElementMain\_Mapping

#### Mapping Source

Property

#### Mapping Target

ReferenceUsage

#### Owned Mappings

(none)

#### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```

src.type.ocllsUndefined() and not
Helper.hasStereotypeApplied(src.owner, 'SysML::ConstraintBlocks::ConstraintBlock')

```

## Mapping rules

The mapping class only has inherited rules. See the mapping classes in the general mapping section for details.

#### **7.7.4.2.39 Realization\_Mapping**

##### **Description**

A UML4SysML::Realization relationship is mapped to a SysML v2 Dependency.

##### **General Mappings**

Abstraction\_Mapping

##### **Mapping Source**

Realization

##### **Mapping Target**

Dependency

##### **Owned Mappings**

(none)

##### **Applicable filters**

(none)

#### **7.7.4.2.40 Slot\_Mapping**

##### **Description**

A UML4SysML::Slot is mapped to a SysML v2 Feature.

##### **General Mappings**

ToFeature\_Init  
ElementMain\_Mapping

##### **Mapping Source**

Slot

##### **Mapping Target**

Feature

##### **Owned Mappings**

(none)

##### **Applicable filters**

(none)

#### **7.7.4.2.41 SlotMembership\_Mapping**

##### **Description**

Creates a membership relationship for *memberElement()*.

### General Mappings

ToFeatureMembership\_Init  
Mapping

### Mapping Source

Slot

### Mapping Target

FeatureMembership

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureMembership::ownedMemberFeature () : Feature [1]  
`from`
- FeatureMembership::isReadOnly () : Boolean [1]  
`from.isReadOnly`
- FeatureMembership::memberName () : String [0..1]  
`from.definingFeature.name`

#### 7.7.4.2.42 SlotFeatureTyping\_Mapping

### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

### General Mappings

ToFeatureTyping\_Init  
Mapping

### Mapping Source

Slot

### Mapping Target

FeatureTyping

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureTyping::type () : Type [1]`  
`ElementMain_Mapping.getMapped(from)`

### 7.7.4.2.43 SlotValue\_Mapping

#### Description

Issue here since a KerML feature cannot have more than one FeatureValue while a UML4SysML::Slot can. How to manage collection of values?

#### General Mappings

ToFeatureValue\_Init  
Mapping

#### Mapping Source

ValueSpecification

#### Mapping Target

FeatureValue

## Owned Mappings

(none)

## Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
src.owner.oclIsKindOf(UML::Slot)
```

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureValue::value () : Expression [1]`  
`from`
- `FeatureValue::featureWithValue () : Feature [1]`



```
Slot_Mapping.getMapped(from.owner)
```

#### 7.7.4.2.44 StructuralFeature\_Mapping

##### Description

The mapping class is the abstract base class for all UML4SysML::StructuralFeature mappings.

##### General Mappings

ToFeature\_Init  
Mapping

##### Mapping Source

StructuralFeature

##### Mapping Target

Feature

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Feature::isUnique () : Boolean [1]  

```
from.isUnique
```
- Feature::isOrdered () : Boolean [1]  

```
from.isOrdered
```
- Feature::isReadOnly () : Boolean [1]  
*abstract rule*
- Feature::ownedRelationship () : Relationship [0..\*]  

```
let typing: KerML::FeatureTyping =  
    StructuralFeatureToFeatureTyping_Mapping.getMapped(from) in  
if typing.oclIsUndefined() then  
    Set{MultiplicityMembership_Mapping.getMapped(from)}  
else  
    Set{MultiplicityMembership_Mapping.getMapped(from), typing}  
endif
```
- Feature::isAbstract () : Boolean [1]  

```
false
```

#### 7.7.4.2.45 StructuralFeatureMembership\_Mapping

##### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

##### General Mappings

ToFeatureMembership\_Init  
Mapping

##### Mapping Source

StructuralFeature

##### Mapping Target

FeatureMembership

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureMembership::visibility () : VisibilityKind [1]  

```
if (from.ocIsKindOf(UML::NamedElement)) then
    Helper.getKerMLVisibilityKind(from.ocAsType(UML::NamedElement).visibility)
else
    KerML::VisibilityKind::public
endif
```
- FeatureMembership::ownedMemberFeature () : Feature [0..1]  

```
NamedElementMain_Mapping.getMapped(from)
```

#### 7.7.4.2.46 StructuralFeatureToFeatureTyping\_Mapping

##### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

##### General Mappings

TypedElementFeatureTyping\_Mapping

##### Mapping Source

StructuralFeature

### Mapping Target

FeatureTyping

### Owned Mappings

(none)

### Applicable filters

(none)

## 7.7.4.2.47 TypedElementFeatureTyping\_Mapping

### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

### General Mappings

ToFeatureTyping\_Init  
Mapping

### Mapping Source

TypedElement

### Mapping Target

FeatureTyping

### Owned Mappings

(none)

### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
not src.type.ocIsUndefined()
```

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureTyping::type () : Type [1]  

```
if from.type.ocIsKindOf(UML::PrimitiveType) then
    Helper.getScalarValueType(from.type)
else if from.type.ocIsKindOf(UML::Enumeration) then
    Helper.getEnumerationType(from.type)
else
    Classifier_Mapping.getMapped(from.type)
endif endif
```

#### 7.7.4.2.48 UpperBoundValueFeatureMembership\_Mapping

##### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

##### General Mappings

ToFeatureMembership\_Init

##### Mapping Source

MultiplicityElement

##### Mapping Target

FeatureMembership

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureMembership::ownedMemberFeature () : Feature [0..1]  

```
if from.upper <> -1 then
    LiteralUnlimitedToInteger_Mapping.getMapped(from.upperValue)
else
    LiteralUnlimitedToUnbounded_Mapping.getMapped(from.upperValue)
endif
```

### 7.7.5 CommonBehavior

#### 7.7.5.1 Overview

Table 6. List of all mappings

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
AnyReceiveEvent	not mapped; see next section
CallEvent	not mapped; see next section
ChangeEvent	CalculationUsage
FunctionBehavior	ActionDefinition
OpaqueBehavior	ActionDefinition
SignalEvent	not mapped; see next section
TimeEvent	CalculationUsage

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
Trigger	AcceptActionUsage

### 7.7.5.2 UML4SysML::CommonBehavior elements not mapped

**Table 7. List of SysML v1 elements not mapped of this section**

SysML v1 Concept	Rationale
CallEvent	The concept of a CallEvent is not supported by SysML v2.

### 7.7.5.3 Mapping Specifications

#### 7.7.5.3.1 Behavior\_Mapping

##### Description

The mapping class is the abstract base class for all UML4SysML::Behavior mappings.

##### General Mappings

ToBehavior\_Init  
Class\_Mapping

##### Mapping Source

Behavior

##### Mapping Target

Behavior

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Behavior::ownedRelationship () : Relationship [0..\*]

```
let parameters: Set(UML::Element) =
    from.ownedElement->select(e | e.ocIsKindOf(UML::Parameter)) in
let parameterSets: Set(UML::Element) =
    from.ownedElement->select(e | e.ocIsKindOf(UML::ParameterSet)) in
let features: Set(UML::Element) =
    from.ownedElement->select(e | e.ocIsKindOf(UML::Property)) in
let elementsOMS: Set(UML::Element) =
    (((from.ownedElement - parameters) parameterSets) - features) in
elementsOMS->collect(e | ElementOwningMembership_Mapping.getMapped(e))
->union(features->collect(e | PropertyMembership_Mapping.getMapped(e)))
```

```
->union(parameters->collect(e | ParameterMembership_Mapping.getMapped(e)))
->union(parameterSets->collect(e | ParameterSetMembership_Mapping.getMapped(e)))
```

### 7.7.5.3.2 ChangeEvent\_Mapping

#### Description

Main mapping class for the mapping of UML4SysML::ChangeEvent.

```
calc sysMLv1ChangeEvent1 {
  language "language"
  /* change expression */
}
```

#### General Mappings

NamedElementMain\_Mapping  
ToCalculationUsage\_Init

#### Mapping Source

ChangeEvent

#### Mapping Target

CalculationUsage

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- CalculationUsage::ownedRelationship () : Relationship [0..\*]

```
from.ownedComment->reject(c | c.annotatedElement->includes(from))
->collect(c| CommentOwnership_Mapping.getMapped(c))->asSet()
->including(ElementOwningMembership_Mapping.getMapped(from.changeExpression))
```

### 7.7.5.3.3 ChangeEventReturnParameter\_Mapping

#### Description

Creates the reference usage for the return parameter of the calculation usage which is the target of the UML4SysML::ChangeEvent mapping.

#### General Mappings

UniqueMapping  
ToReferenceUsage\_Init

#### Mapping Source

ChangeEvent

#### Mapping Target

ReferenceUsage

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReferenceUsage::direction () : FeatureDirectionKind [0..1]  
`KerML::FeatureDirectionKind::_out'`

### 7.7.5.3.4 ChangeEventReturnParameterMembership\_Mapping

#### Description

Creates a membership relationship for *memberElement()*.

#### General Mappings

UniqueMapping  
ToReturnParameterMembership\_Init

#### Mapping Source

ChangeEvent

#### Mapping Target

ReturnParameterMembership

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReturnParameterMembership::ownedMemberParameter () : Feature [1]`

```
ChangeEventReturnParameter_Mapping.getMapped (from)
```

#### 7.7.5.3.5 ChangeTriggerBindingConnector\_Mapping

##### Description

Creates the binding connector between the result of the trigger calculation usage and the result of the time event calculation usage.

##### General Mappings

ToBindingConnectorAsUsage\_Init  
UniqueMapping

##### Mapping Source

Trigger

##### Mapping Target

BindingConnectorAsUsage

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `BindingConnectorAsUsage::ownedRelationship () : Relationship [0..*]`

```
Set { ChangeTriggerReturnEndFeatureMembership_Mapping.getMapped (from) }
```

```
->including (ChangeTriggerEndFeatureMembership_Mapping.getMapped (from) )
```

#### 7.7.5.3.6 ChangeTriggerConstraintUsage\_Mapping

##### Description

Creates the constraint usage of the target of the mapping of a `UML4SysML::Trigger` referencing a `UML4SysML::ChangeEvent`.

##### General Mappings

ToConstraintUsage\_Init  
UniqueMapping



### Mapping Source

Trigger

### Mapping Target

ConstraintUsage

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ConstraintUsage::ownedRelationship () : Relationship [0..*]`  
`Set{ChangeTriggerFeatureMembership_Mapping.getMapped(from) }`  
`->including (ChangeTriggerReturnParameterMembership_Mapping.getMapped(from) )`

#### 7.7.5.3.7 ChangeTriggerEndFeatureMembership\_Mapping

##### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

##### General Mappings

ToEndFeatureMembership\_Init  
UniqueMapping

### Mapping Source

Trigger

### Mapping Target

EndFeatureMembership

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- EndFeatureMembership::ownedMemberFeature () : Feature [1]  
`ChangeTriggerReferenceUsage_Mapping.getMapped(from)`

#### 7.7.5.3.8 ChangeTriggerEventChainingFeature\_Mapping

##### Description

Creates the chaining feature for the event for the mapping of a UML4SysML::Trigger referencing a UML4SysML::ChangeEvent.

##### General Mappings

UniqueMapping  
ToFeatureChaining\_Init

##### Mapping Source

Trigger

##### Mapping Target

FeatureChaining

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureChaining::chainingFeature () : Feature [1]  
`from.event`

#### 7.7.5.3.9 ChangeTriggerEventReturnParameterChainingFeature\_Mapping

##### Description

Creates the chaining feature for the return parameter for the mapping of a UML4SysML::Trigger referencing a UML4SysML::ChangeEvent.

##### General Mappings

ToFeatureChaining\_Init  
UniqueMapping

##### Mapping Source

Trigger

### Mapping Target

FeatureChaining

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureChaining::chainingFeature () : Feature [1]  
`ChangeEventReturnParameter_Mapping.getMapped(from.event)`

## 7.7.5.3.10 ChangeTriggerExpressionFeature\_Mapping

### Description

Creates the feature for the trigger invocation expression of the target of the mapping of a UML4SysML::Trigger referencing a UML4SysML::ChangeEvent.

### General Mappings

ToFeature\_Init  
UniqueMapping

### Mapping Source

Trigger

### Mapping Target

Feature

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Feature::ownedRelationship () : Relationship [0..\*]

```
Set{ChangeTriggerExpressionFeatureValue_Mapping.getMapped(from) }
```

#### 7.7.5.3.11 ChangeTriggerExpressionFeatureMembership\_Mapping

##### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

##### General Mappings

UniqueMapping  
ToFeatureMembership\_Init

##### Mapping Source

Trigger

##### Mapping Target

FeatureMembership

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureMembership::ownedMemberFeature () : Feature [1]

```
ChangeTriggerExpressionInvocationExpression_Mapping.getMapped(from)
```

#### 7.7.5.3.12 ChangeTriggerExpressionFeatureReferenceExpression\_Mapping

##### Description

Creates the feature reference expression for the feature value in the trigger invocation expression of the target of the mapping of a UML4SysML::Trigger referencing a UML4SysML::ChangeEvent.

##### General Mappings

UniqueMapping  
ToFeatureReferenceExpression\_Init

##### Mapping Source

Trigger

##### Mapping Target

FeatureReferenceExpression

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureReferenceExpression::ownedRelationship () : Relationship [0..*]`  
`Set{ChangeTriggerExpressionFeatureMembership_Mapping.getMapped(from) }`  
`->including(ReturnParameterFeatureMembership_Factory.create())`

### 7.7.5.3.13 ChangeTriggerExpressionFeatureTyping\_Mapping

#### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

#### General Mappings

ToFeatureTyping\_Init  
UniqueMapping

#### Mapping Source

Trigger

#### Mapping Target

FeatureTyping

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureTyping::type () : Type [1]`  
`TransitionChangeTriggerConstraintUsage_Mapping.getMapped(from)`

### 7.7.5.3.14 ChangeTriggerExpressionFeatureValue\_Mapping

#### Description

Creates a feature value relationship.

### General Mappings

UniqueMapping  
ToFeatureValue\_Init

### Mapping Source

Trigger

### Mapping Target

FeatureValue

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureValue::value () : Expression [1]

```
ChangeTriggerExpressionFeatureReferenceExpression_Mapping.getMapped(from)
```

## 7.7.5.3.15 ChangeTriggerExpressionInvocationExpression\_Mapping

### Description

Creates the invocation expression for the trigger invocation expression of the target of the mapping of a UML4SysML::Trigger referencing a UML4SysML::ChangeEvent.

### General Mappings

ToInvocationExpression\_Init  
UniqueMapping

### Mapping Source

Trigger

### Mapping Target

InvocationExpression

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `InvocationExpression::ownedRelationship () : Relationship [0..*]`  
`Set { ChangeTriggerExpressionFeatureTyping_Mapping.getMapped (from) }`  
`->including (ReturnParameterFeatureMembership_Factory.create ())`

#### 7.7.5.3.16 ChangeTriggerExpressionParameterMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

##### General Mappings

ToParameterMembership\_Init  
UniqueMapping

##### Mapping Source

Trigger

##### Mapping Target

ParameterMembership

##### Owned Mappings

(none)

##### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ParameterMembership::ownedMemberParameter () : Feature [1]`  
`ChangeTriggerExpressionFeature_Mapping.getMapped (from)`

#### 7.7.5.3.17 ChangeTriggerFeature\_Mapping

##### Description

Creates the feature for the mapping of a `UML4SysML::Trigger` referencing a `UML4SysML::ChangeEvent`.

##### General Mappings

ToFeature\_Init  
UniqueMapping

#### Mapping Source

Trigger

#### Mapping Target

Feature

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Feature::ownedRelationship () : Relationship [0..\*]  
`Set { ChangeTriggerEventChainingFeature_Mapping.getMapped (from) }`  
`->including (ChangeTriggerEventReturnParameterChainingFeature_Mapping.getMapped (from) )`

### 7.7.5.3.18 ChangeTriggerFeatureMembership\_Mapping

#### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

#### General Mappings

UniqueMapping  
ToFeatureMembership\_Init

#### Mapping Source

Trigger

#### Mapping Target

FeatureMembership

#### Owned Mappings

(none)

#### Applicable filters

(none)



## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureMembership::ownedMemberFeature () : Feature [1]`  
`ChangeTriggerBindingConnector_Mapping.getMapped (from)`

### 7.7.5.3.19 ChangeTriggerFeatureValue\_Mapping

#### Description

Creates a feature value relationship.

#### General Mappings

ToFeatureValue\_Init  
UniqueMapping

#### Mapping Source

Trigger

#### Mapping Target

FeatureValue

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureValue::value () : Expression [1]`  
`ChangeTriggerInvocationExpression_Mapping.getMapped (from)`

### 7.7.5.3.20 ChangeTriggerInvocationExpression\_Mapping

#### Description

Creates the trigger invocation expression of the target of the mapping of a `UML4SysML::Trigger` referencing a `UML4SysML::ChangeEvent`.

#### General Mappings

ToTriggerInvocationExpression\_Init  
UniqueMapping

#### Mapping Source

Trigger

### Mapping Target

TriggerInvocationExpression

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- TriggerInvocationExpression::ownedRelationship () : Relationship [0..\*]  

```
Set { ChangeTriggerExpressionParameterMembership_Mapping.getMapped (from) }  
->including (ReturnParameterFeatureMembership_Factory.create () )
```
- TriggerInvocationExpression::kind () : TriggerKind [0..1]  

```
SysML::Systems::Actions::TriggerKind::when
```

#### 7.7.5.3.21 ChangeTriggerReferenceSubsetting\_Mapping

### Description

Creates a subsetting relationship.

### General Mappings

UniqueMapping  
ToReferenceSubsetting\_Init

### Mapping Source

Trigger

### Mapping Target

ReferenceSubsetting

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceSubsetting::referencedFeature () : Feature [1]`  
`ChangeTriggerFeature_Mapping.getMapped(from)`
- `ReferenceSubsetting::ownedRelatedElement () : Element [0..*]`  
`Set{ChangeTriggerFeature_Mapping.getMapped(from)}`

#### 7.7.5.3.22 ChangeTriggerReferenceUsage\_Mapping

##### Description

Creates a reference usage.

##### General Mappings

UniqueMapping  
ToReferenceUsage\_Init

##### Mapping Source

Trigger

##### Mapping Target

ReferenceUsage

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceUsage::direction () : FeatureDirectionKind [0..1]`  
`KerML::FeatureDirectionKind::_in'`
- `ReferenceUsage::ownedRelationship () : Relationship [0..*]`  
`Set{ChangeTriggerFeatureValue_Mapping.getMapped(from)}`

#### 7.7.5.3.23 ChangeTriggerReturnEndFeatureMembership\_Mapping

##### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

##### General Mappings

UniqueMapping  
ToEndFeatureMembership\_Init

#### **Mapping Source**

Trigger

#### **Mapping Target**

EndFeatureMembership

#### **Owned Mappings**

(none)

#### **Applicable filters**

(none)

#### **Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- EndFeatureMembership::ownedMemberFeature () : Feature [1]  
`ChangeTriggerReturnReferenceUsage_Mapping.getMapped(from)`

### **7.7.5.3.24 ChangeTriggerReturnParameter\_Mapping**

#### **Description**

Creates the return parameter feature for the mapping of a UML4SysML::Trigger referencing a UML4SysML::ChangeEvent.

#### **General Mappings**

UniqueMapping  
ToReferenceUsage\_Init

#### **Mapping Source**

Trigger

#### **Mapping Target**

ReferenceUsage

#### **Owned Mappings**

(none)

#### **Applicable filters**

(none)

#### **Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceUsage::direction () : FeatureDirectionKind [0..1]`

`KerML::FeatureDirectionKind::_'out'`

#### 7.7.5.3.25 ChangeTriggerReturnParameterMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

##### General Mappings

ToReturnParameterMembership\_Init  
UniqueMapping

##### Mapping Source

Trigger

##### Mapping Target

ReturnParameterMembership

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReturnParameterMembership::ownedMemberParameter () : Feature [1]`

`ChangeTriggerReturnParameter_Mapping.getMapped(from)`

#### 7.7.5.3.26 ChangeTriggerReturnReferenceSubsetting\_Mapping

##### Description

Creates a subsetting relationship.

##### General Mappings

ToReferenceSubsetting\_Init  
UniqueMapping

##### Mapping Source

Trigger

### Mapping Target

ReferenceSubsetting

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReferenceSubsetting::referencedFeature () : Feature [1]  
`ChangeTriggerReturnParameter_Mapping.getMapped(from)`

### 7.7.5.3.27 ChangeTriggerReturnReferenceUsage\_Mapping

#### Description

Creates a reference usage.

#### General Mappings

UniqueMapping  
ToReferenceUsage\_Init

#### Mapping Source

Trigger

#### Mapping Target

ReferenceUsage

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReferenceUsage::ownedRelationship () : Relationship [0..\*]  
`Set{ChangeTriggerReturnReferenceSubsetting_Mapping.getMapped(from) }`

- ReferenceUsage::isEnd () : Boolean [1]

true

### 7.7.5.3.28 OpaqueBehavior\_Mapping

#### Description

A UML4SysML::OpaqueBehavior is mapped to a SysML v2 ActionDefinition.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
action def SysMLv1OpaqueBehavior {
    language "Built-in Math"
    /*
     * result = 42 + 23;
     */
}
```

#### General Mappings

Behavior\_Mapping

#### Mapping Source

OpaqueBehavior

#### Mapping Target

ActionDefinition

#### Owned Mappings

(none)

#### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
src.owner.ocIsKindOf(UML::Package)
```

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ActionDefinition::ownedRelationship () : Relationship [0..\*]

```
let parameters : Set(UML::Parameter) =
    from.ownedElement->select(e | e.ocIsKindOf(UML::Parameter)) in
let parameterSets : Set(UML::ParameterSet) =
    from.ownedElement->select(e | e.ocIsKindOf(UML::ParameterSet)) in
let features : Set(UML::Property) =
    from.ownedElement->select(e | e.ocIsKindOf(UML::Property)) in
let elementsOMS : Set(UML::Element) =
```

```

        (((from.ownedElement - parameters) - parameterSets) - features) in
elementsOMS->collect(e | ElementOwningMembership_Mapping.getMapped(e))
->union(features->collect(e | PropertyMembership_Mapping.getMapped(e)))
->union(parameters->collect(e | ParameterMembership_Mapping.getMapped(e)))
->union(parameterSets->collect(e | ParameterSetMembership_Mapping.getMapped(e)))
->union(from.language
->collect(1 | OpaqueBehaviorMembership_Mapping.getMapped(from, 1)))

```

### 7.7.5.3.29 OpaqueBehaviorMembership\_Mapping

#### Description

Creates a membership relationship for *memberElement()*.

#### General Mappings

ToOwningMembership\_Init  
Mapping

#### Mapping Source

OpaqueBehavior

#### Mapping Target

OwningMembership with qualifier: language:String

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- OwningMembership::ownedMemberElement (in language : String) : Element [1]

```
OpaqueBehaviorSpecification_Mapping.getMapped(from, language)
```

### 7.7.5.3.30 OpaqueBehaviorSpecification\_Mapping

#### Description

The mapping class creates the SysML v2 TextualRepresentation elements from the languages and bodies properties of the given UML4SysML::OpaqueBehavior.

#### General Mappings

ToTextualRepresentation\_Init  
Mapping

#### Mapping Source



OpaqueBehavior

### Mapping Target

TextualRepresentation with qualifier: language:String

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- TextualRepresentation::body () : String [1]  

```
let index:Integer = from.language->indexOf(language) in  
from._'body'->at(index)
```
- TextualRepresentation::language () : String [1]  

```
language
```

## 7.7.5.3.31 SignalTriggerReferenceUsage\_Mapping

### Description

Creates a reference usage.

### General Mappings

UniqueMapping  
ToReferenceUsage\_Init

### Mapping Source

Trigger

### Mapping Target

ReferenceUsage

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceUsage::direction () : FeatureDirectionKind [0..1]`  
`KerML::FeatureDirectionKind::_in'`
- `ReferenceUsage::ownedRelationship () : Relationship [0..*]`  
`Set{SignalTriggerReferenceUsageFeatureTyping_Mapping.getMapped(from) }`

### 7.7.5.3.32 SignalTriggerReferenceUsageFeatureTyping\_Mapping

#### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

#### General Mappings

UniqueMapping  
 ToFeatureTyping\_Init

#### Mapping Source

Trigger

#### Mapping Target

FeatureTyping

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureTyping::type () : Type [1]`  
`from.event.oclAsType(UML::SignalEvent).signal`

### 7.7.5.3.33 TimeEvent\_Mapping

#### Description

Main mapping class for the mapping of UML4SysML::TimeEvent.

```
calc sysMLv1TimeEvent1 {
  language "language"
  /* duration */
}
```

## General Mappings

NamedElementMain\_Mapping  
ToCalculationUsage\_Init

## Mapping Source

TimeEvent

## Mapping Target

CalculationUsage

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- CalculationUsage::ownedRelationship () : Relationship [0..\*]  

```
from.ownedComment  
  
->reject(c | c.annotatedElement->includes(from))  
  
->collect(c| CommentOwnership_Mapping.getMapped(c))->asSet()  
  
->including(OpaqueExpressionMembership_Mapping.getMapped(from.when.expr))
```

### 7.7.5.3.34 TimeTriggerBindingConnector\_Mapping

#### Description

Creates the binding connector between the result of the trigger calculation usage and the result of the time event calculation usage.

## General Mappings

UniqueMapping  
ToBindingConnectorAsUsage\_Init

## Mapping Source

Trigger

## Mapping Target

BindingConnectorAsUsage

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `BindingConnectorAsUsage::ownedRelationship () : Relationship [0..*]`  
`Set{TimeTriggerReturnEndFeatureMembership_Mapping.getMapped(from) }`  
`->including (TimeTriggerReturnEndFeatureMembership_Mapping.getMapped (from) )`

### 7.7.5.3.35 TimeTriggerCalculationUsage\_Mapping

#### Description

Creates the calculation usage of the target of the mapping of a `UML4SysML::Trigger` referencing a `UML4SysML::TimeEvent`.

#### General Mappings

UniqueMapping  
ToCalculationUsage\_Init

#### Mapping Source

Trigger

#### Mapping Target

CalculationUsage

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `CalculationUsage::declaredName () : String [0..1]`  
`from.name`
- `CalculationUsage::ownedRelationship () : Relationship [0..*]`

```

Set{TimeTriggerReturnParameterMembership_Mapping.getMapped(from)}
->including(TimeTriggerFeatureMembership_Mapping.getMapped(from))

```

### 7.7.5.3.36 TimeTriggerEndFeatureMembership\_Mapping

#### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

#### General Mappings

ToEndFeatureMembership\_Init  
UniqueMapping

#### Mapping Source

Trigger

#### Mapping Target

EndFeatureMembership

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- EndFeatureMembership::ownedMemberFeature () : Feature [1]  
TimeTriggerReferenceUsage\_Mapping.getMapped(from)

### 7.7.5.3.37 TimeTriggerEventChainingFeature\_Mapping

#### Description

Creates the chaining feature for the event for the mapping of a UML4SysML::Trigger referencing a UML4SysML::TimeEvent.

#### General Mappings

ToFeatureChaining\_Init  
UniqueMapping

#### Mapping Source

Trigger

#### Mapping Target

FeatureChaining

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureChaining::chainingFeature () : Feature [1]  
`from.event`

### 7.7.5.3.38 TimeTriggerEventReturnParameterChainingFeature\_Mapping

#### Description

Creates the chaining feature for the return parameter for the mapping of a UML4SysML::Trigger referencing a UML4SysML::TimeEvent.

#### General Mappings

UniqueMapping  
ToFeatureChaining\_Init

#### Mapping Source

Trigger

#### Mapping Target

FeatureChaining

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureChaining::chainingFeature () : Feature [1]  
`TimeTriggerReturnParameter_Mapping.getMapped(from)`

### 7.7.5.3.39 TimeTriggerExpressionFeature\_Mapping

#### Description

Creates the feature for the trigger invocation expression of the target of the mapping of a UML4SysML::Trigger referencing a UML4SysML::TimeEvent.

#### General Mappings

UniqueMapping  
ToFeature\_Init

#### Mapping Source

Trigger

#### Mapping Target

Feature

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Feature::ownedRelationship () : Relationship [0..\*]  

```
Set{TimeTriggerExpressionFeatureValue_Mapping.getMapped(from) }
```

### 7.7.5.3.40 TimeTriggerExpressionFeatureTyping\_Mapping

#### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

#### General Mappings

ToFeatureTyping\_Init  
UniqueMapping

#### Mapping Source

Trigger

#### Mapping Target

FeatureTyping

#### Owned Mappings

(none)

#### **Applicable filters**

(none)

#### **Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureTyping::type () : Type [1]`

`TransitionTimeTriggerCalculationUsage_Mapping.getMapped (from)`

### **7.7.5.3.41 TimeTriggerExpressionFeatureValue\_Mapping**

#### **Description**

Creates a feature value relationship.

#### **General Mappings**

ToFeatureValue\_Init  
UniqueMapping

#### **Mapping Source**

Trigger

#### **Mapping Target**

FeatureValue

#### **Owned Mappings**

(none)

#### **Applicable filters**

(none)

#### **Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureValue::value () : Expression [1]`

`TimeTriggerExpressionInvocationExpression_Mapping.getMapped (from)`

### **7.7.5.3.42 TimeTriggerExpressionInvocationExpression\_Mapping**

#### **Description**

Creates the invocation expression for the trigger invocation expression of the target of the mapping of a `UML4SysML::Trigger` referencing a `UML4SysML::TimeEvent`.



## General Mappings

UniqueMapping  
ToInvocationExpression\_Init

## Mapping Source

Trigger

## Mapping Target

InvocationExpression

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- InvocationExpression::ownedRelationship () : Relationship [0..\*]  

```
Set{TimeTriggerExpressionFeatureTyping_Mapping.getMapped(from) }  
->including(ReturnParameterFeatureMembership_Factory.create())
```

### 7.7.5.3.43 TimeTriggerExpressionParameterMembership\_Mapping

#### Description

Creates a membership relationship for *memberElement()*.

#### General Mappings

UniqueMapping  
ToParameterMembership\_Init

#### Mapping Source

Trigger

#### Mapping Target

ParameterMembership

#### Owned Mappings

(none)

#### Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ParameterMembership::ownedMemberParameter () : Feature [1]`  
`TimeTriggerExpressionFeature_Mapping.getMapped(from)`

### 7.7.5.3.44 TimeTriggerFeature\_Mapping

#### Description

Creates the feature for the mapping of a `UML4SysML::Trigger` referencing a `UML4SysML::TimeEvent`.

#### General Mappings

UniqueMapping  
ToFeature\_Init

#### Mapping Source

Trigger

#### Mapping Target

Feature

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Feature::ownedRelationship () : Relationship [0..*]`  
`Set{TimeTriggerEventChainingFeature_Mapping.getMapped(from) }`  
`->including(TimeTriggerEventReturnParameterChainingFeature_Mapping.getMapped(from) )`

### 7.7.5.3.45 TimeTriggerFeatureMembership\_Mapping

#### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

#### General Mappings

ToFeatureMembership\_Init  
UniqueMapping

### Mapping Source

Trigger

### Mapping Target

FeatureMembership

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureMembership::ownedMemberFeature () : Feature [1]`  
`TimeTriggerBindingConnector_Mapping.getMapped(from)`

## 7.7.5.3.46 TimeTriggerFeatureTyping\_Mapping

### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

### General Mappings

UniqueMapping  
ToFeatureTyping\_Init

### Mapping Source

Trigger

### Mapping Target

FeatureTyping

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureTyping::type () : Type [1]`  

```

if from.event.oclAsType(UML::TimeEvent).isRelative then
    SYSML2::AttributeDefinition.allInstances()
    ->any(m | m.qualifiedName = 'ISQ::DurationValue')
else
    SYSML2::AttributeDefinition.allInstances()
    ->any(m | m.qualifiedName = 'Time::TimeInstantValue')
endif

```

#### 7.7.5.3.47 TimeTriggerFeatureValue\_Mapping

##### Description

Creates a feature value relationship.

##### General Mappings

ToFeatureValue\_Init  
UniqueMapping

##### Mapping Source

Trigger

##### Mapping Target

FeatureValue

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureValue::value () : Expression [1]`  

```

TimeTriggerInvocationExpression_Mapping.getMapped(from)

```

#### 7.7.5.3.48 TimeTriggerInvocationExpression\_Mapping

##### Description

Creates the trigger invocation expression of the target of the mapping of a `UML4SysML::Trigger` referencing a `UML4SysML::TimeEvent`.

##### General Mappings

ToTriggerInvocationExpression\_Init  
UniqueMapping

### Mapping Source

Trigger

### Mapping Target

TriggerInvocationExpression

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- TriggerInvocationExpression::ownedRelationship () : Relationship [0..\*]  

```
Set{TimeTriggerExpressionParameterMembership_Mapping.getMapped(from) }
```
- TriggerInvocationExpression::kind () : TriggerKind [0..1]  

```
if from.event.oclAsType(UML::TimeEvent).isRelative then  
  SysML::Systems::Actions::TriggerKind::after  
else  
  SysML::Systems::Actions::TriggerKind::at  
endif
```

## 7.7.5.3.49 TimeTriggerReferenceSubsetting\_Mapping

### Description

Creates a subsetting relationship.

### General Mappings

ToReferenceSubsetting\_Init  
UniqueMapping

### Mapping Source

Trigger

### Mapping Target

ReferenceSubsetting

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceSubsetting::referencedFeature () : Feature [1]`  
`TimeTriggerFeature_Mapping.getMapped(from)`
- `ReferenceSubsetting::ownedRelatedElement () : Element [0..*]`  
`Set{TimeTriggerFeature_Mapping.getMapped(from)}`

#### 7.7.5.3.50 TimeTriggerReferenceUsage\_Mapping

##### Description

Creates a reference usage.

##### General Mappings

ToReferenceUsage\_Init  
UniqueMapping

##### Mapping Source

Trigger

##### Mapping Target

ReferenceUsage

##### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceUsage::direction () : FeatureDirectionKind [0..1]`  
`KerML::FeatureDirectionKind::_in'`
- `ReferenceUsage::ownedRelationship () : Relationship [0..*]`  
`Set{TimeTriggerFeatureValue_Mapping.getMapped(from)}`

### 7.7.5.3.51 TimeTriggerReturnEndFeatureMembership\_Mapping

#### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

#### General Mappings

ToEndFeatureMembership\_Init  
UniqueMapping

#### Mapping Source

Trigger

#### Mapping Target

EndFeatureMembership

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- EndFeatureMembership::ownedMemberFeature () : Feature [1]  
`TimeTriggerReturnReferenceUsage_Mapping.getMapped(from)`

### 7.7.5.3.52 TimeTriggerReturnParameter\_Mapping

#### Description

Creates the return parameter feature for the mapping of a UML4SysML::Trigger referencing a UML4SysML::TimeEvent.

#### General Mappings

UniqueMapping  
ToReferenceUsage\_Init

#### Mapping Source

Trigger

#### Mapping Target

ReferenceUsage

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceUsage::ownedRelationship () : Relationship [0..*]`  
`Set{TimeTriggerFeatureTyping_Mapping.getMapped(from) }`

### 7.7.5.3.53 TimeTriggerReturnParameterMembership\_Mapping

#### Description

Creates a membership relationship for *memberElement()*.

#### General Mappings

UniqueMapping  
ToReturnParameterMembership\_Init

#### Mapping Source

Trigger

#### Mapping Target

ReturnParameterMembership

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReturnParameterMembership::ownedMemberParameter () : Feature [1]`  
`TimeTriggerReturnParameter_Mapping.getMapped(from)`

### 7.7.5.3.54 TimeTriggerReturnReferenceSubsetting\_Mapping

#### Description

Creates a subsetting relationship.

#### General Mappings



UniqueMapping  
ToReferenceSubsetting\_Init

#### **Mapping Source**

Trigger

#### **Mapping Target**

ReferenceSubsetting

#### **Owned Mappings**

(none)

#### **Applicable filters**

(none)

#### **Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReferenceSubsetting::referencedFeature () : Feature [1]  
`TimeTriggerReturnParameter_Mapping.getMapped (from)`

### **7.7.5.3.55 TimeTriggerReturnReferenceUsage\_Mapping**

#### **Description**

Creates a reference usage.

#### **General Mappings**

UniqueMapping  
ToReferenceUsage\_Init

#### **Mapping Source**

Trigger

#### **Mapping Target**

ReferenceUsage

#### **Owned Mappings**

(none)

#### **Applicable filters**

(none)

#### **Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReferenceUsage::isEnd () : Boolean [1]  
true
- ReferenceUsage::ownedRelationship () : Relationship [0..\*]  
Set { TimeTriggerReturnReferenceSubsetting\_Mapping.getMapped (from) }

### 7.7.5.3.56 Trigger\_Mapping

#### Description

A UML4SysML::Trigger is mapped to a SysML v2 AcceptActionUsage.

#### General Mappings

NamedElementMain\_Mapping  
ToActionUsage\_Init

#### Mapping Source

Trigger

#### Mapping Target

AcceptActionUsage

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- AcceptActionUsage::ownedRelationship () : Relationship [0..\*]  
from.ownedComment  
->reject (c | c.annotatedElement->includes (from))  
->collect (c | CommentOwnership\_Mapping.getMapped (c)) ->asSet ()  
->including (TriggerParameterMembership\_Mapping.getMapped (from))  
->including (ParameterMembership\_Factory.create ())

### 7.7.5.3.57 TriggerParameterMembership\_Mapping

#### Description

Creates a membership relationship for *memberElement()*.

## General Mappings

UniqueMapping  
ToParameterMembership\_Init

## Mapping Source

Trigger

## Mapping Target

ParameterMembership

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ParameterMembership::ownedMemberParameter () : Feature [1]  

```
if from.event.ocIsKindOf(UML::SignalEvent) then
    SignalTriggerReferenceUsage_Mapping.getMapped(from)
else if from.event.ocIsKindOf(UML::TimeEvent) then
    TimeTriggerReferenceUsage_Mapping.getMapped(from)
else if from.event.ocIsKindOf(UML::ChangeEvent) then
    ChangeTriggerReferenceUsage_Mapping.getMapped(from)
else
    OclUndefined
endif endif endif
```

## 7.7.6 CommonStructure

### 7.7.6.1 Overview

Table 8. List of all mappings

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
Abstraction	Dependency
Comment	Comment
Constraint	ConstraintDefinition
Dependency	Dependency
ElementImport	MembershipImport
PackageImport	NamespaceImport
Realization	Dependency

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
Usage	Dependency

## 7.7.6.2 Mapping Specifications

### 7.7.6.2.1 Abstraction\_Mapping

#### Description

A UML4SysML::Abstraction relationship is mapped to a SysML v2 Dependency relationship.

#### General Mappings

Dependency\_Mapping

#### Mapping Source

Abstraction

#### Mapping Target

Dependency

#### Owned Mappings

(none)

#### Applicable filters

(none)

### 7.7.6.2.2 Comment\_Mapping

#### Description

A UML4SysML::Comment is mapped to a SysML v2 Comment.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
part def SysMLv1Block1;
part def SysMLv1Block2;
action def SysMLv1Activity {
    comment about SysMLv1Activity, SysMLv1Block1
    /* comment body */
}
comment about SysMLv1Block1, SysMLv1Block /* comment body */
```

#### General Mappings

ElementMain\_Mapping  
ToAnnotatingElement\_Init

#### Mapping Source

Comment

### Mapping Target

Comment

### Owned Mappings

(none)

### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
not Helper.hasStereotypeApplied(src, 'SysML::ModelElements::ElementGroup')
```

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Comment::ownedRelationship () : Relationship [0..\*]

```
self.oclAsType(ElementMain_Mapping).ownedRelationship()  
->union(self.annotation()->asSet())
```

- Comment::annotation () : Annotation [0..\*]

```
from.annotatedElement  
->collect(e | CommentAnnotation_Mapping.getMapped(from, e))
```

- Comment::body () : String [1]

```
if from.body->isEmpty() then '' else from.body endif
```

## 7.7.6.2.3 CommentAnnotation\_Mapping

### Description

The mapping class creates the annotation relationship for the UML4SysML::Comment mapping.

### General Mappings

ToAnnotation\_Init  
Mapping

### Mapping Source

Comment

### Mapping Target

Annotation with qualifier: annotatedElement:Element

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Annotation::owningAnnotatedElement () : Element [0..1]`  
`null`
- `Annotation::annotatedElement (in annotatedElement : Element) : Element [1]`  
`ElementMain_Mapping.getMapped(annotatedElement)`
- `Annotation::annotatingElement () : AnnotatingElement [1]`  
`Comment_Mapping.getMapped(from)`

#### 7.7.6.2.4 CommentOwnership\_Mapping

### Description

That mapping class creates an ownership relation that is convenient for a Comment. In SysMLv1/UML can be owned by any kind of element, including some that are not translated to SysMLv2 Namespaces.

### General Mappings

ToAnnotation\_Init  
UniqueMapping

### Mapping Source

Comment

### Mapping Target

Annotation

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Annotation::annotatedElement () : Element [1]`  
`ElementMain_Mapping.getMapped(from.owner)`

- `Annotation::ownedRelatedElement () : Element [0..*]`  
`Set{self.annotatingElement () }`
- `Annotation::annotatingElement () : AnnotatingElement [1]`  
`Comment_Mapping.getMapped (from)`

#### 7.7.6.2.5 Constraint\_Mapping

##### Description

A `UML4SysML::Constraint` is mapped to a SysML v2 `ConstraintDefinition` and `AssertConstraintUsages` for the constrained elements.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
part def SysMLv1Block {
    constraint def SysMLv1Constraint {
        calc sysMLv1Constraint {
            language "English"
            /*
             * constraint specification
             */
        }
    }
    assert constraint assert_sysMLv1Constraint : SysMLv1Constraint;
}
```

##### General Mappings

`ToConstraintDefinition_Init`  
`NamedElementMain_Mapping`

##### Mapping Source

`Constraint`

##### Mapping Target

`ConstraintDefinition`

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ConstraintDefinition::ownedRelationship () : Relationship [0..*]`

```

self.oclAsType (ElementMain_Mapping).ownedRelationship()
->union (Set {ElementFeatureMembership_Mapping.getMapped (from.specification),
CommonReturnParameterReferenceUsageMembership_Mapping.getMapped (from.specification)})

```

#### 7.7.6.2.6 ConstrainedElementFeatureMembership\_Mapping

##### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

##### General Mappings

ToFeatureMembership\_Init  
Mapping

##### Mapping Source

Constraint

##### Mapping Target

FeatureMembership

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureMembership::ownedMemberFeature () : Feature [1]  
ConstraintUsage\_Mapping.getMapped (from)

#### 7.7.6.2.7 ConstraintUsageFeatureTyping\_Mapping

##### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

##### General Mappings

ToFeatureTyping\_Init  
Mapping

##### Mapping Source

Constraint

##### Mapping Target

FeatureTyping



## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureTyping::type () : Type [1]`

`from`

### 7.7.6.2.8 ConstraintUsage\_Mapping

#### Description

The mapping class creates the SysML v2 `AssertConstraintUsage` elements for the constrained elements of the `UML4SysML::Constraint` mapping.

#### General Mappings

`ToUsage_Init`  
`Mapping`

#### Mapping Source

`Constraint`

#### Mapping Target

`AssertConstraintUsage`

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `AssertConstraintUsage::declaredName () : String [0..1]`

`'assert_' + from.name`

- `AssertConstraintUsage::ownedRelationship () : Relationship [0..*]`

`from.ownedComment->reject (c | c.annotatedElement->includes (from))`

```
->collect(c| CommentOwnership_Mapping.getMapped(c))->asSet()
->union(Set{ConstraintUsageFeatureTyping_Mapping.getMapped(from),
CommonReturnParameterReferenceUsageMembership_Mapping.getMapped(from)})
```

### 7.7.6.2.9 Dependency\_Mapping

#### Description

A UML4SysML::Dependency relationship is mapped to a SysML v2 Dependency relationship.

#### General Mappings

DirectedRelationship\_Mapping

#### Mapping Source

Dependency

#### Mapping Target

Dependency

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Dependency::client () : Element [0..\*]  

```
from.source->collect(e | ElementMain_Mapping.getMapped(e))
```
- Dependency::declaredName () : String [0..1]  

```
from.name
```
- Dependency::supplier () : Element [0..\*]  

```
from.target->collect(e | ElementMain_Mapping.getMapped(e))
```

### 7.7.6.2.10 DirectedRelationship\_Mapping

#### Description

The mapping class is the abstract base class for all UML4SysML::DirectedRelationship mappings.

#### General Mappings

Relationship\_Mapping

### Mapping Source

DirectedRelationship

### Mapping Target

Relationship

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Relationship::target () : Element [0..\*]  
`from.target->collect (e | ElementMain_Mapping.getMapped (e))`
- Relationship::source () : Element [0..\*]  
`from.source->collect (e | ElementMain_Mapping.getMapped (e))`

#### 7.7.6.2.11 ElementMain\_Mapping

### Description

This is the general abstract class to be used as an ancestor for any class mapping specification.

### General Mappings

ToElement\_Init  
MainMapping

### Mapping Source

Element

### Mapping Target

Element

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Element::ownedRelationship () : Relationship [0..*]`

```
from.ownedComment->reject(c | c.annotatedElement->includes(from))
->collect(c| CommentOwnership_Mapping.getMapped(c))->asSet()
```

- `Element::elementId () : String [1]`

```
Helper.getID(from)
```

#### 7.7.6.2.12 ElementMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

##### General Mappings

ToMembership\_Init  
Mapping

##### Mapping Source

Element

##### Mapping Target

Membership

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Membership::membershipOwningNamespace () : Element [0..*]`

```
Set{ElementMain_Mapping(from)}
-- will not be used since corresponding attribute is derived,
-- but required for redefinition
```

- `Membership::memberElement () : Element [1]`

```
ElementMain_Mapping.getMapped(from)
```

- `Membership::visibility () : VisibilityKind [1]`

```

if (from.ocIsKindOf(UML::NamedElement)) then
    from.ocAsType(UML::NamedElement).visibility
else
    KerML::VisibilityKind::public
endif

```

### 7.7.6.2.13 ElementOwnership\_Mapping

#### Description

The mapping class is the abstract base class for mappings that target ownership relationships.

#### General Mappings

ToRelationship\_Init  
UniqueMapping

#### Mapping Source

Element

#### Mapping Target

Relationship

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Relationship::source () : Element [0..\*]  
`OrderedSet{ElementMain_Mapping.getMapped(from.owner) }`
- Relationship::ownedRelatedElement () : Element [0..\*]  
`self.target ()`
- Relationship::target () : Element [0..\*]  
`OrderedSet{ElementMain_Mapping.getMapped(from) }`

### 7.7.6.2.14 ElementOwningMembership\_Mapping

#### Description

Creates a owning membership relationship for *ownedMemberElement()*.

#### General Mappings

ElementMembership\_Mapping  
ElementOwnership\_Mapping

### Mapping Source

Element

### Mapping Target

OwningMembership

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- OwningMembership::ownedRelatedElement () : Element [0..\*]  

```
Set{self.ownedMemberElement () }
```
- OwningMembership::membershipOwningNamespace () : Element [0..\*]  

```
Set{ElementMain_Mapping(from) }  
-- will not be used since corresponding attribute is derived,  
-- but required for redefinition
```
- OwningMembership::ownedMemberElement () : Element [1]  

```
ElementMain_Mapping.getMapped(from)
```

## 7.7.6.2.15 NamedElementMain\_Mapping

### Description

The mapping class is the abstract base class for mappings of UML4SysML::NamedElements.

### General Mappings

ElementMain\_Mapping

### Mapping Source

NamedElement

### Mapping Target

Element

### Owned Mappings

(none)

#### **Applicable filters**

(none)

#### **Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Element::declaredName () : String [0..1]

from.name

### **7.7.6.2.16 Namespace\_Mapping**

#### **Description**

The mapping class is the abstract base class for UML4SysML::Namespace mappings.

#### **General Mappings**

ToNamespace\_Init

NamedElementMain\_Mapping

#### **Mapping Source**

Namespace

#### **Mapping Target**

Namespace

#### **Owned Mappings**

(none)

#### **Applicable filters**

(none)

#### **Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Namespace::ownedImport () : Import [0..\*]

Set { }

### **7.7.6.2.17 Relationship\_Mapping**

#### **Description**

Th mapping class is the abstract base class for UML4SysML::Relationship mappings.

#### **General Mappings**

ToRelationship\_Init  
ElementMain\_Mapping

### Mapping Source

Relationship

### Mapping Target

Relationship

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Relationship::ownedRelatedElement () : Element [0..\*]  

```
from.relatedElement->select(e | from.ownedElement->includes(e))  
->collect(e | ElementMain_Mapping.getMapped(e))
```
- Relationship::owningRelatedElement () : Element [0..1]  

```
ElementMain_Mapping.getMapped(from.owner)
```

## 7.7.6.2.18 Usage\_Mapping

### Description

A UML4SysML::Usage relationship is mapped to a SysML v2 Dependency relationship.

### General Mappings

Dependency\_Mapping

### Mapping Source

Usage

### Mapping Target

Dependency

### Owned Mappings

(none)

### Applicable filters



(none)

## 7.7.7 InformationFlows

### 7.7.7.1 Overview

**Table 9. List of all mappings**

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
InformationFlow	not mapped; see next section
InformationItem	ItemDefinition

### 7.7.7.2 Mapping Specifications

#### 7.7.7.2.1 InformationFlow\_Mapping

##### Description

A UML4SysML::InformationFlow is mapped to a FlowDefinition, if the UML4SysML::InformationFlow has defined realizing connectors or if it is realized by an association. If the information flow has more than one realizing connector, a FlowDefinition element is created for each of them.

##### General Mappings

ToConnectionUsage\_Init  
UniqueMapping

##### Mapping Source

InformationFlow

##### Mapping Target

FlowDefinition with qualifier: realization:NamedElement

##### Owned Mappings

(none)

##### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
src.ocIsKindOf(UML::InformationFlow) and  
(src.ocAsType(UML::InformationFlow).realizingConnector->notEmpty()  
or src.ocAsType(UML::InformationFlow).realization->notEmpty())
```

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FlowDefinition::ownedRelationship () : Relationship [0..\*]  

```
from.informationSource  
->collect(s | InformationFlowEndFeatureMembership_Mapping.getMapped(from, s))  
->asSet()
```

```

->union(from.informationTarget
  ->collect(t | InformationFlowEndFeatureMembership_Mapping.getMapped(from, t))
  ->asSet())
->union(from.conveyed
  ->collect(i | InformationFlowConveyedFeatureMembership_Mapping.getMapped(i))
  ->asSet())
->union(from.realization->select(a | a.ocIsKindOf(UML::Association))
  ->collect(r | InformationFlowSubclassification_Mapping.getMapped(from, r))
  ->asSet())
->asOrderedSet()

```

### 7.7.7.2.2 InformationFlowConveyedFeatureMembership\_Mapping

#### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

#### General Mappings

ToFeatureMembership\_Init  
Mapping

#### Mapping Source

Classifier

#### Mapping Target

FeatureMembership

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureMembership::ownedMemberFeature () : Feature [1]  

```
InformationItemFlowConveyedItemUsage_Mapping.getMapped(from)
```

### 7.7.7.2.3 InformationFlowEnd\_Mapping

#### Description

The mapping class creates the source feature of the FlowDefinition for the mapping of UML4SysML::InformationFlow.

#### General Mappings

ToFeature\_Init  
UniqueMapping

### Mapping Source

InformationFlow

### Mapping Target

Feature with qualifier: end:NamedElement

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Feature::ownedRelationship () : Relationship [0..\*]  
`Set{InformationFlowFeatureTyping_Mapping.getMapped(from, end) }`
- Feature::isEnd () : Boolean [1]  
`true`

#### 7.7.7.2.4 InformationFlowEndFeatureMembership\_Mapping

##### Description

The mapping class creates the source and the target membership relationships of the FlowDefinition for the UML4SysML::InformationFlow mapping.

##### General Mappings

ToFeatureMembership\_Init  
UniqueMapping

### Mapping Source

InformationFlow

### Mapping Target

FeatureMembership with qualifier: end:NamedElement

### Owned Mappings

(none)

### Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureMembership::ownedMemberFeature (in end : NamedElement) : Feature [1]

```
InformationFlowEnd_Mapping.getMapped(from, end)
```

### 7.7.7.2.5 InformationFlowFeatureTyping\_Mapping

#### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

#### General Mappings

ToFeatureTyping\_Init  
UniqueMapping

#### Mapping Source

InformationFlow

#### Mapping Target

FeatureTyping with qualifier: element:NamedElement

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureTyping::type (in source : NamedElement) : Type [1]

```
ElementMain_Mapping.getMapped(element)
```

### 7.7.7.2.6 InformationFlowSubclassification\_Mapping

#### Description

Creates a Subclassification relationship between the target element of the UML4SysML::InformationFlow mapping and the target element of the UML4SysML::Association which realizes the flow.

#### General Mappings

ToSubclassification\_Init  
Mapping

#### Mapping Source

InformationFlow

### Mapping Target

Subclassification with qualifier: element:Relationship

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Subclassification::superclassifier () : Classifier [1]  
    element
- Subclassification::subclassifier () : Classifier [1]  
    from

#### 7.7.7.2.7 InformationItem\_Mapping

##### Description

A UML4SysML::InformationItem is mapped to a SysML v2 ItemDefinition.

##### General Mappings

Classifier\_Mapping

##### Mapping Source

InformationItem

##### Mapping Target

ItemDefinition

##### Owned Mappings

(none)

##### Applicable filters

(none)

#### 7.7.7.2.8 InformationItemFlowConveyedItemUsage\_Mapping

##### Description

Creates an ItemUsage element representing the conveyed classifier of an UML4SysML::InformationFlow.

## General Mappings

ToItemUsage\_Init  
Mapping

## Mapping Source

Classifier

## Mapping Target

ItemUsage

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ItemUsage::ownedRelationship () : Relationship [0..\*]

```
Set{ InformationItemFlowConveyedItemUsageFeatureTyping_Mapping.getMapped(from) }
```

### 7.7.7.2.9 InformationItemFlowConveyedItemUsageFeatureTyping\_Mapping

#### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

## General Mappings

ToFeatureTyping\_Init  
Mapping

## Mapping Source

Classifier

## Mapping Target

FeatureTyping

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureTyping::type () : Type [1]`

from

## 7.7.8 Interactions

### 7.7.8.1 Overview

**Table 10. List of all mappings**

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
ActionExecutionSpecification	ActionUsage
BehaviorExecutionSpecification	ActionUsage
CombinedFragment	Interaction
ConsiderIgnoreFragment	Interaction
Continuation	not mapped; see next section
DestructionOccurrenceSpecification	not mapped; see next section
ExecutionOccurrenceSpecification	not mapped; see next section
Gate	not mapped; see next section
GeneralOrdering	not mapped; see next section
Interaction	Interaction Behavior
InteractionConstraint	ConstraintDefinition
InteractionOperand	Namespace Interaction
InteractionUse	Step
Lifeline	PartUsage
Message	Flow
MessageOccurrenceSpecification	not mapped; see next section
OccurrenceSpecification	not mapped; see next section
PartDecomposition	Step
StateInvariant	Invariant

### 7.7.8.2 UML4SysML::Interactions elements not mapped

**Table 11. List of SysML v1 elements not mapped of this section**

SysML v1 Concept	Rationale
ConsiderIgnoreFragment	Mapping is not specified yet.
Continuation	Mapping is not specified yet.

SysML v1 Concept	Rationale
DestructionOccurrenceSpecification	Mapping is not specified yet.
ExecutionOccurrenceSpecification	Mapping is not specified yet.
Gate	Mapping is not specified yet.
GeneralOrdering	Mapping is not specified yet.
InteractionConstraint	Mapping is not specified yet.
MessageOccurrenceSpecification	Mapping is not specified yet.
OccurrenceSpecification	Mapping is not specified yet.
PartDecomposition	Mapping is not specified yet.

### 7.7.8.3 Mapping Specifications

#### 7.7.8.3.1 ActionExecutionSpecification\_Mapping

##### Description

A UML4SysML::ActionExecutionSpecification is mapped to a SysML v2 ActionUsage.

##### General Mappings

ToActionUsage\_Init  
NamedElementMain\_Mapping

##### Mapping Source

ActionExecutionSpecification

##### Mapping Target

ActionUsage

##### Owned Mappings

(none)

##### Applicable filters

(none)

#### 7.7.8.3.2 BehaviorExecutionSpecification\_Mapping

##### Description

A UML4SysML::BehaviorExecutionSpecification is mapped to a SysML v2 ActionUsage.

##### General Mappings

ToActionUsage\_Init  
NamedElementMain\_Mapping

##### Mapping Source



BehaviorExecutionSpecification

### Mapping Target

ActionUsage

### Owned Mappings

(none)

### Applicable filters

(none)

## 7.7.8.3.3 CombinedFragment\_Mapping

### Description

A UML4SysML::CombinedFragment is mapped to a SysMLv2 Interaction.

### General Mappings

NamedElementMain\_Mapping  
ToInteraction\_Init

### Mapping Source

CombinedFragment

### Mapping Target

Interaction

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Interaction::ownedRelationship () : Relationship [0..\*]

```
let operands: Set(UML::Element) =  
    from.ownedElement->select(e | e.ocIsKindOf(UML::InteractionOperand)) in  
let occurrencesSpecs: Set(UML::Element) =  
    from.ownedElement->select(e | e.ocIsKindOf(UML::OccurrenceSpecification)) in  
let elements: Set(UML::Element) =  
    (from.ownedElement - operands) - occurrencesSpecs in  
    elements->collect(e |  
        ElementOwningMembership_Mapping.getMapped(e) ->asSet()  
    ->union(operands->collect(e |
```

```

InteractionOperandMembership_Mapping.getMapped(e)) ->asSet())
->union(self.oclaSType(ElementMain_Mapping).ownedRelationship())

```

### 7.7.8.3.4 CombinedFragmentMembership\_Mapping

#### Description

Creates a membership relationship for *memberElement()*.

#### General Mappings

ToFeatureMembership\_Init  
Mapping

#### Mapping Source

CombinedFragment

#### Mapping Target

FeatureMembership

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureMembership::memberFeature () : Feature [1]  

```

ElementMain_Mapping.getMapped(from)

```
- FeatureMembership::ownedMemberFeature () : Feature [0..1]  

```

self.memberFeature()

```

### 7.7.8.3.5 ExecutionSpecificationMembership\_Mapping

#### Description

Creates a membership relationship for *memberElement()*.

#### General Mappings

ToEndFeatureMembership\_Init  
Mapping

#### Mapping Source

ExecutionSpecification

### Mapping Target

EndFeatureMembership

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- EndFeatureMembership::ownedMemberFeature () : Feature [0..1]  
`self.memberFeature()`
- EndFeatureMembership::memberFeature () : Feature [1]  
`ElementMain_Mapping.getMapped(from)`

### 7.7.8.3.6 Interaction\_Mapping

#### Description

A UML4SysML::Interaction is mapped to a SysMLv2 Interaction.

#### General Mappings

Namespace\_Mapping  
ToInteraction\_Init

#### Mapping Source

Interaction

#### Mapping Target

Interaction

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Interaction::ownedRelationship () : Relationship [0..*]`

```

let lifelines: Set(UML::Element) = from.lifeline in
let messageOccurrences: Set(UML::Element) =
  from.ownedElement->select(e |
    e.ocIsKindOf(UML::MessageOccurrenceSpecification)) in
let executionOccurrences: Set(UML::Element) =
  from.fragment->select(e | e.ocIsKindOf(UML::ExecutionSpecification)) in
let occurrencesSpecs: Set(UML::Element) =
  from.fragment->select(e | e.ocIsKindOf(UML::OccurrenceSpecification)) in
let messages: Set(UML::Element) = from.message in
let invariants: Set(UML::Element) =
  from.fragment->select(e | e.ocIsKindOf(UML::StateInvariant)) in
let interactionUsages: Set(UML::Element) =
  from.fragment->select(e | e.ocIsKindOf(UML::InteractionUse)) in
let combinedFragments: Set(UML::Element) =
  from.ownedElement->select(e | e.ocIsKindOf(UML::CombinedFragment)) in
let continuations: Set(UML::Element) =
  from.ownedElement->select(e | e.ocIsKindOf(UML::Continuation)) in
let elements: Set(UML::Element) =
  (((((((from.ownedElement - lifelines) - messageOccurrences)
    - executionOccurrences) - occurrencesSpecs) - messages) -
    combinedFragments) - invariants) -
    interactionUsages) - continuations) - from.ownedComment in

elements->collect(e | ElementOwningMembership_Mapping.getMapped(e))->asSet()
->union(lifelines->collect(e | LifelineMembership_Mapping.getMapped(e))->asSet())
->union(executionOccurrences
  ->collect(e | ExecutionSpecificationMembership_Mapping.getMapped(e))->asSet())
->union(messages->collect(e | MessageMembership_Mapping.getMapped(e))->asSet())
->union(combinedFragments
  ->collect(e | CombinedFragmentMembership_Mapping.getMapped(e))->asSet())
->union(invariants
  ->collect(e | StateInvariantMembership_Mapping.getMapped(e))->asSet())
->union(interactionUsages
  ->collect(e | InteractionUseMembership_Mapping.getMapped(e))->asSet())
->union(self.ocAsType(ElementMain_Mapping).ownedRelationship())

```

### 7.7.8.3.7 InteractionOperand\_Mapping

#### Description

A `UML4SysML::InteractionOperand` is mapped to a SysML v2 Interaction.

#### General Mappings

NamedElementMain\_Mapping  
ToInteraction\_Init

#### Mapping Source

InteractionOperand

#### Mapping Target

Interaction

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Interaction::ownedRelationship () : Relationship [0..*]`

```
let executionOccurrences: Set(UML::Element) =
    from.ownedElement->select(e | e.ocIsKindOf(UML::ExecutionSpecification)) in
let occurrencesSpecs: Set(UML::Element) =
    from.ownedElement->select(e | e.ocIsKindOf(UML::OccurrenceSpecification)) in
let continuations: Set(UML::Element) =
    from.ownedElement->select(e | e.ocIsKindOf(UML::Continuation)) in
let elements: Set(UML::Element) =
    (((from.ownedElement - executionOccurrences) - occurrencesSpecs) -
    continuations) - from.ownedComment in
elements->collect(e | ElementOwningMembership_Mapping.getMapped(e)) ->asSet()
->union(self.ocAsType(ElementMain_Mapping).ownedRelationship())
->union(executionOccurrences
->collect(e | ExecutionSpecificationMembership_Mapping.getMapped(e)) ->asSet())
```

### 7.7.8.3.8 InteractionOperandMembership\_Mapping

#### Description

Creates a membership relationship for *memberElement()*.

#### General Mappings

ToFeatureMembership\_Init  
Mapping

#### Mapping Source

InteractionOperand

#### Mapping Target

FeatureMembership

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureMembership::ownedMemberFeature () : Feature [0..1]`

```
self.memberFeature()
```

- `FeatureMembership::memberFeature () : Feature [1]`

```
ElementMain_Mapping.getMapped(from)
```

### 7.7.8.3.9 InteractionUse\_Mapping

#### Description

A `UML4SysML::InteractionUse` is mapped to a SysML v2 Step.

#### General Mappings

ToStep\_Init  
Namespace\_Mapping

#### Mapping Source

InteractionUse

#### Mapping Target

Step

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Step::ownedRelationship () : Relationship [0..*]`

```
self.oclAsType(ElementMain_Mapping).ownedRelationship()
```

```
->including(InteractionUseFeatureTyping_Mapping.getMapped(from))
```

### 7.7.8.3.10 InteractionUseMembership\_Mapping

#### Description

Creates a membership relationship for *memberElement()*.

#### General Mappings

ToFeatureMembership\_Init  
Mapping

#### Mapping Source

InteractionUse

#### Mapping Target

FeatureMembership

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureMembership::ownedMemberFeature () : Feature [0..1]  
`self.memberFeature()`
- FeatureMembership::memberFeature () : Feature [1]  
`ElementMain_Mapping.getMapped(from)`

### 7.7.8.3.11 InteractionUseFeatureTyping\_Mapping

#### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

#### General Mappings

ToFeatureTyping\_Init  
Mapping

#### Mapping Source

InteractionUse

#### Mapping Target

FeatureTyping

#### Owned Mappings

(none)

#### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureTyping::type () : Type [1]`  
`ElementMain_Mapping.getMapped(from.refersTo)`

#### 7.7.8.3.12 LifelineMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

##### General Mappings

ToFeatureMembership\_Init  
Mapping

##### Mapping Source

Lifeline

##### Mapping Target

FeatureMembership

##### Owned Mappings

(none)

##### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureMembership::ownedMemberFeature () : Feature [0..1]`  
`self.memberFeature()`
- `FeatureMembership::memberFeature () : Feature [1]`  
`ElementMain_Mapping.getMapped(from)`

#### 7.7.8.3.13 LifelinePartUsage\_Mapping

##### Description

A UML4SysML::Lifeline is mapped to a SysML v2 PartUsage.



## General Mappings

ToPartUsage\_Init  
NamedElementMain\_Mapping

## Mapping Source

Lifeline

## Mapping Target

PartUsage

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- PartUsage::ownedRelationship () : Relationship [0..\*]

```
self.oclAsType(ElementMain_Mapping).ownedRelationship()  
->including(LifelineFeatureTyping_Mapping.getMapped(from))
```

### 7.7.8.3.14 LifelineFeatureTyping\_Mapping

#### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

#### General Mappings

ToFeatureTyping\_Init  
Mapping

#### Mapping Source

Lifeline

#### Mapping Target

FeatureTyping

#### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureTyping::type () : Type [1]`  
`ElementMain_Mapping.getMapped(from.represents.type)`

### 7.7.8.3.15 Message\_Mapping

#### Description

A `UML4SysML::Message` is mapped to a SysML v2 `ItemFlow`.

#### General Mappings

`ToItemFlow_Init`  
`NamedElementMain_Mapping`

#### Mapping Source

`Message`

#### Mapping Target

`Flow`

#### Owned Mappings

(none)

### Applicable filters

(none)

### 7.7.8.3.16 MessageMembership\_Mapping

#### Description

Creates a membership relationship for *memberElement()*.

#### General Mappings

`ToFeatureMembership_Init`

#### Mapping Source

`Message`

#### Mapping Target

`FeatureMembership`

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureMembership::memberFeature () : Feature [1]`  
`ElementMain_Mapping.getMapped(from)`
- `FeatureMembership::ownedMemberFeature () : Feature [0..1]`  
`self.memberFeature()`

### 7.7.8.3.17 StateInvariant\_Mapping

#### Description

A `UML4SysML::StateInvariant` is mapped to a SysML v2 Invariant.

#### General Mappings

`ToExpression_Init`  
`Namespace_Mapping`

#### Mapping Source

`StateInvariant`

#### Mapping Target

`Invariant`

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Invariant::ownedRelationship () : Relationship [0..*]`  
`self.oclAsType(ElementMain_Mapping).ownedRelationship()`  
`->including(StateInvariantFeatureTyping_Mapping.getMapped(from))`

### 7.7.8.3.18 StateInvariantMembership\_Mapping

#### Description

Creates a membership relationship for *memberElement()*.

#### General Mappings

ToFeatureMembership\_Init  
Mapping

#### Mapping Source

StateInvariant

#### Mapping Target

FeatureMembership

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureMembership::memberFeature () : Feature [1]  
`ElementMain_Mapping.getMapped (from)`
- FeatureMembership::ownedMemberFeature () : Feature [0..1]  
`self.memberFeature ()`

### 7.7.8.3.19 StateInvariantFeatureTyping\_Mapping

#### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

#### General Mappings

ToFeatureTyping\_Init  
Mapping

#### Mapping Source

StateInvariant

#### Mapping Target

FeatureTyping

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureTyping::type () : Type [1]`  
`ElementMain_Mapping.getMapped(from.invariant)`

## 7.7.9 Packages

### 7.7.9.1 Overview

**Table 12. List of all mappings**

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
Extension	ConnectionDefinition
ExtensionEnd	OccurrenceUsage Feature ReferenceUsage AttributeUsage
Image	not mapped; see next section
Model	Package
Package	Package
PackageMerge	not mapped; see next section
Profile	Package
ProfileApplication	not mapped; see next section
Stereotype	MetadataDefinition

### 7.7.9.2 UML4SysML::Packages elements not mapped

**Table 13. List of SysML v1 elements not mapped of this section**

SysML v1 Concept	Rationale
Extension	The mapping of the extension relationship is performed in the context of Stereotype_Mapping.
ExtensionEnd	The mapping of the extension end property is performed in the context of Stereotype_Mapping.
Image	Mapping is not specified yet.
PackageMerge	The concept of the PackageMerge relationship is not supported by SysML v2.

### 7.7.9.3 Mapping Specifications

#### 7.7.9.3.1 ElementImport\_Mapping

##### Description

A UML4SysML::ElementImport is mapped to a SysMLv2 MembershipImport. The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
package SysMLv1Package1 {
    import SysMLv1Package2::SysMLv1Block;
    import SysMLv1Package2::SysMLv1ValueType;
}
package SysMLv1Package2 {
    part def SysMLv1Block;
    attribute def SysMLv1ValueType;
}
```

##### General Mappings

ToMembershipImport\_Init  
NamedElementMain\_Mapping

##### Mapping Source

ElementImport

##### Mapping Target

MembershipImport

##### Owned Mappings

(none)

##### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
if src.oclIsKindOf(UML::ElementImport) then
    Helper.hasMainMapping(src.oclAsType(UML::ElementImport).importedElement)
else
    false
endif
```

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- MembershipImport::importedMembership () : Namespace [1]  
`ElementOwningMembership_Mapping.getMapped(from.importedElement)`
- MembershipImport::importedMemberName () : String [0..1]  
`from.alias`

- MembershipImport::visibility () : VisibilityKind [1]

```
Helper.getKerMLVisibilityKind(from.visibility)
```

### 7.7.9.3.2 Model\_Mapping

#### Description

SysMLv2 has no explicit model element for a model. The UML4SysML::Model element is mapped to a SysMLv2 Package. The property "viewpoint" is mapped to a metadata defined in the SysML v1 library. The expected SysML v2 textual notation of a UML4SysML::Model with URI and viewpoint is as follows. If URI or viewpoint are not set in the source model, the metadata is not generated.

```
package SysMLv1Model {
  @SysMLv1Library::PackageData {URI="https://omg.org";}
  @SysMLv1Library::ModelData {'viewpoint'="The viewpoint of the model element.";}
}
```

#### General Mappings

Package\_Mapping

#### Mapping Source

Model

#### Mapping Target

Package

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Package::ownedRelationship () : Relationship [0..\*]

```
let relationships : Set(KerML::Relationship) =
  self.oclAsType(Package_Mapping).ownedRelationship() in
if from.viewpoint.oclIsUndefined() or from.viewpoint = '' then
  relationships
else
  relationships
  ->including (ModelViewpointMetadataMembership_Mapping.getMapped(from))
endif
```

### 7.7.9.3.3 ModelViewpointMetadataUsage\_Mapping

#### Description

The mapping class creates the annotating feature to annotate the generated Package element with metadata to store the UML4SysML::Model::viewpoint property.

### General Mappings

ToMetadataUsage\_Init  
Mapping

### Mapping Source

Model

### Mapping Target

MetadataUsage

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- MetadataUsage::declaredName () : String [0..1]  
`'viewpoint'`
- MetadataUsage::ownedRelationship () : Relationship [0..\*]  
`Set{ModelViewpointMetadataFeatureTyping_Mapping.getMapped(from), ModelViewpointMetadataFeatur`

#### 7.7.9.3.4 ModelViewpointMetadataFeatureMembership\_Mapping

##### Description

The mapping class creates the feature membership relationship for the metadata feature to store the UML4SysML::Model::viewpoint property.

### General Mappings

ToFeatureMembership\_Init  
Mapping

### Mapping Source

Model

### Mapping Target

FeatureMembership



## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureMembership::ownedMemberFeature () : Feature [0..1]`  
`ModelViewpointMetadataReferenceUsage_Mapping.getMapped (from)`

### 7.7.9.3.5 ModelViewpointMetadataReferenceUsage\_Mapping

#### Description

The mapping class creates the `MetadataFeature` for the mapping of the property `UML4SysML::Model::viewpoint`.

#### General Mappings

`ToReferenceUsage_Init`  
`Mapping`

#### Mapping Source

`Model`

#### Mapping Target

`ReferenceUsage`

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceUsage::ownedRelationship () : Relationship [0..*]`  
`Set {ModelViewpointMetadataRedefinition_Mapping.getMapped (from) ,`  
`ModelViewpointMetadataFeatureValue_Mapping.getMapped (from) }`

### 7.7.9.3.6 ModelViewpointMetadataFeatureTyping\_Mapping

#### Description

The mapping class creates the FeatureTyping relationship for the AnnotatingFeature for the metadata to store the UML4SysML::Model::viewpoint property.

### General Mappings

ToFeatureTyping\_Init  
Mapping

### Mapping Source

Model

### Mapping Target

FeatureTyping

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureTyping::type () : Type [1]  

```
SysMLv2::MetadataDefinition.allInstances()  
->any(m | m.qualifiedName = 'SysMLv1Library::ModelData')
```

### 7.7.9.3.7 ModelViewpointMetadataMembership\_Mapping

#### Description

The mapping class creates a membership relationship for the metadata feature value for the UML4SysML::Model::viewpoint property.

### General Mappings

ToOwningMembership\_Init  
Mapping

### Mapping Source

Model

### Mapping Target

OwningMembership

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `OwningMembership::ownedMemberElement () : Element [1]`  
`ModelViewpointMetadataUsage_Mapping.getMapped(from)`

#### 7.7.9.3.8 ModelViewpointMetadataFeatureValue\_Mapping

##### Description

The mapping class maps the value of the property `UML4SysML::Model::viewpoint`.

##### General Mappings

`ToFeatureValue_Init`  
`Mapping`

##### Mapping Source

`Model`

##### Mapping Target

`FeatureValue`

##### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureValue::value () : Expression [1]`  
`ModelViewpointValue_Mapping.getMapped(from)`

#### 7.7.9.3.9 ModelViewpointMetadataRedefinition\_Mapping

##### Description

The mapping class creates the redefinition of the attribute for the metadata `UML4SysML::Model::viewpoint`.

##### General Mappings

ToRedefinition\_Init  
Mapping

### Mapping Source

Model

### Mapping Target

Redefinition

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Redefinition::redefinedFeature () : Feature [1]`  

```
let m : SYSML2::Membership =  
  SYSML2::AttributeUsage.allInstances()  
  ->collect(dt | dt.owningRelationship)  
  ->select(r | r.ocIsKindOf(SYSML2::Membership))  
  ->any(m | m.memberName = 'viewpoint') in  
if (m.ocIsUndefined()) then  
  invalid  
else  
  m.memberElement  
endif
```

#### 7.7.9.3.10 ModelViewpointValue\_Mapping

##### Description

The mapping class maps the value expression of the property `UML4SysML::Model::viewpoint`.

##### General Mappings

ToExpression\_Init  
Mapping

### Mapping Source

Model

### Mapping Target

LiteralString

### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `LiteralString::value () : String [1]`  
`LiteralString_Factory.create(from.viewpoint)`

### 7.7.9.3.11 Package\_Mapping

#### Description

A `UML4SysML::Package` is mapped to a SysML v2 Package. The property "URI" is mapped to a metadata if it has a value. The expected SysML v2 textual notation of a `UML4SysML::Package` is as follows:

```
package ThisIsAPackageWithURI {  
    metadata SysMLv1Library::PackageData {URI="https://omg.org";} }  
}
```

#### General Mappings

Namespace\_Mapping

#### Mapping Source

Package

#### Mapping Target

Package

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Package::ownedRelationship () : Relationship [0..*]`  
`Helper.packageOwnedRelationship(from)`

### 7.7.9.3.12 PackageImport\_Mapping

#### Description

A UML4SysML::PackageImport is mapped to a SysML v2 NamespaceImport. The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
import SysMLv1Package::*;
```

#### General Mappings

ToNamespaceImport\_Init  
ElementMain\_Mapping

#### Mapping Source

PackageImport

#### Mapping Target

NamespaceImport

#### Owned Mappings

(none)

#### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
if src.ocIsKindOf(UML::PackageImport) then
    Helper.isInScope(src.ocAsType(UML::PackageImport).importedPackage)
else
    false
endif
```

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- NamespaceImport::visibility () : VisibilityKind [0..1]  
`Helper.getKerMLVisibilityKind(from.visibility)`
- NamespaceImport::importedNamespace () : Namespace [1]  
`Namespace_Mapping.getMapped(from.importedPackage)`

### 7.7.9.3.13 PackageURIMetadataUsage\_Mapping

#### Description

The mapping class creates the annotating feature to annotate the generated Package element with metadata to store the UML4SysML::Package::URI property.

#### General Mappings

ToMetadataUsage\_Init  
Mapping

#### Mapping Source

Package

#### Mapping Target

MetadataUsage

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- MetadataUsage::ownedRelationship () : Relationship [0..\*]  

```
Set { PackageURIFeatureTyping_Mapping.getMapped (from) ,  
      PackageURIFeatureMembership_Mapping.getMapped (from) }
```
- MetadataUsage::declaredName () : String [0..1]  

```
'URI '
```

#### 7.7.9.3.14 PackageURIFeatureMembership\_Mapping

##### Description

The mapping class creates the feature membership relationship for the metadata feature to store the UML4SysML::Package::URI property.

##### General Mappings

ToFeatureMembership\_Init  
Mapping

#### Mapping Source

Package

#### Mapping Target

FeatureMembership

#### Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureMembership::ownedMemberFeature () : Feature [1]`

```
PackageURIMetadataReferenceUsage_Mapping.getMapped(from)
```

### 7.7.9.3.15 PackageURIFeatureTyping\_Mapping

#### Description

The mapping class creates the FeatureTyping relationship for the AnnotatingFeature for the metadata to store the UML4SysML::Package::URI property.

#### General Mappings

ToFeatureTyping\_Init  
Mapping

#### Mapping Source

Package

#### Mapping Target

FeatureTyping

#### Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureTyping::type () : Type [1]`

```
let m: SysMLv2::Membership = SysMLv2::AttributeDefinition.allInstances()  
->collect(dt | dt.owningRelationship)  
->select(r | r.ocIsKindOf(SysMLv2::Membership))  
->any(m | m.memberName = 'PackageData' ) in  
  
if (m.ocIsUndefined()) then  
    invalid  
else  
    m.memberElement  
endif
```



### 7.7.9.3.16 PackageURIMetadataReferenceUsage\_Mapping

#### Description

The mapping class creates the MetadataFeature for the mapping of the property UML4SysML::Package::URI.

#### General Mappings

ToReferenceUsage\_Init  
Mapping

#### Mapping Source

Package

#### Mapping Target

ReferenceUsage

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReferenceUsage::ownedRelationship () : Relationship [0..\*]  

```
Set{PackageURIRedefinition_Mapping.getMapped(from),  
PackageURIMetadataFeatureValue_Mapping.getMapped(from)}
```

### 7.7.9.3.17 PackageURIMetadataFeatureValue\_Mapping

#### Description

The mapping class maps the value of the property UML4SysML::Package::URI.

#### General Mappings

ToFeatureValue\_Init  
Mapping

#### Mapping Source

Package

#### Mapping Target

FeatureValue

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureValue::value () : Expression [1]`  
`PackageURIValue_Mapping.getMapped (from)`
- `FeatureValue::featureWithValue () : Feature [1]`  
`packageURIMetadataReferenceUsage.to`

### 7.7.9.3.18 PackageURIMetadataMembership\_Mapping

#### Description

The mapping class creates a membership relationship for the metadata feature value for the UML4SysML::Package::URI property.

#### General Mappings

ToOwningMembership\_Init  
Mapping

#### Mapping Source

Package

#### Mapping Target

OwningMembership

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `OwningMembership::ownedMemberElement () : Element [1]`  
`PackageURIMetadataUsage_Mapping.getMapped (from)`

### 7.7.9.3.19 PackageURIRedefinition\_Mapping

#### Description

The mapping class creates the redefinition of the attribute for the metadata UML4SysML::Package::URI.

#### General Mappings

ToRedefinition\_Init  
Mapping

#### Mapping Source

Package

#### Mapping Target

Redefinition

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Redefinition::redefinedFeature () : Feature [1]  

```
let m : SysMLv2::Membership =  
  SysMLv2::AttributeUsage.allInstances()  
  ->collect(dt | dt.owningRelationship)  
  ->select(r | r.ocIsKindOf(SYSML2::Membership))  
  ->any(m | m.memberName = 'URI') in  
if (m.ocIsUndefined()) then  
  invalid  
else  
  m.memberElement  
endif
```

### 7.7.9.3.20 PackageURIValue\_Mapping

#### Description

The mapping class maps the value expression of the property UML4SysML::Package::URI.

#### General Mappings

ToExpression\_Init  
Mapping

#### Mapping Source

Package

### Mapping Target

LiteralString

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- LiteralString::value () : String [1]

from.URI

## 7.7.9.3.21 Profile\_Mapping

### Description

A UML4SysML::Profile is mapped to a SysML v2 Package.

### General Mappings

Package\_Mapping

### Mapping Source

Profile

### Mapping Target

Package

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Package::ownedRelationship () : Relationship [0..\*]

```
self.oclAsType(Package_Mapping).ownedRelationship()  
->including(ProfileMetadataMembership_Mapping.getMapped(from))
```

### 7.7.9.3.22 ProfileMetadataMembership\_Mapping

#### Description

The mapping class creates a membership relationship for the metadata feature value for the UML4SysML::Model::viewpoint property.

#### General Mappings

ToOwningMembership\_Init  
Mapping

#### Mapping Source

Profile

#### Mapping Target

OwningMembership

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- OwningMembership::ownedMemberElement () : Element [1]

```
ProfileMetadataUsage_Mapping.getMapped(from)
```

### 7.7.9.3.23 ProfileMetadataUsage\_Mapping

#### Description

The mapping class creates the annotating feature to annotate the generated Package element with metadata to store the UML4SysML::Model::viewpoint property.

#### General Mappings

ToMetadataUsage\_Init  
Mapping

#### Mapping Source

Profile

#### Mapping Target

MetadataUsage

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- MetadataUsage::declaredName () : String [0..1]  
'Profile'

#### 7.7.9.3.24 StereotypeMetadataDefinition\_Mapping

##### Description

A UML4SysML::Stereotype is mapped to a SysML v2 MetadataDefinition.

##### General Mappings

Class\_Mapping

##### Mapping Source

Stereotype

##### Mapping Target

MetadataDefinition

##### Owned Mappings

(none)

##### Applicable filters

(none)

#### 7.7.9.3.25 StereotypeMetadataDefinitionMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

##### General Mappings

ElementOwningMembership\_Mapping

##### Mapping Source

Stereotype

### Mapping Target

OwningMembership

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- OwningMembership::ownedMemberElement () : Element [0..1]

`ElementMain_Mapping.getMapped (from)`

## 7.7.9.3.26 StereotypeOccurrenceUsage\_Mapping

### Description

The mapping class maps the usage of a stereotype to a SysML v2 OccurrenceUsage.

### General Mappings

ToOccurrenceUsage\_Init  
Mapping

### Mapping Source

Stereotype

### Mapping Target

OccurrenceUsage

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- OccurrenceUsage::ownedRelationship () : Relationship [0..\*]

```
Set{StereotypeOccurrenceUsageFeatureTyping_Mapping.getMapped(from),  
StereotypeOccurrenceUsageMultiplicityMembership_Mapping.getMapped(from)}
```

#### 7.7.9.3.27 StereotypeOccurrenceUsageFeatureTyping\_Mapping

##### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

##### General Mappings

ToFeatureTyping\_Init  
Mapping

##### Mapping Source

Stereotype

##### Mapping Target

FeatureTyping

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureTyping::type () : Type [1]

```
StereotypeOccurrenceDefinition_Mapping.getMapped(from)
```

#### 7.7.9.3.28 StereotypeOccurrenceUsageMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

##### General Mappings

ToMembership\_Init  
Mapping

##### Mapping Source

Stereotype

##### Mapping Target

Membership



## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Membership::memberElement () : Element [1]  
`StereotypeOccurrenceUsage_Mapping.getMapped(from)`

### 7.7.9.3.29 StereotypeOccurrenceUsageMultiplicityMembership\_Mapping

#### Description

Creates a membership relationship for *memberElement()*.

#### General Mappings

ToMembership\_Init  
Mapping

#### Mapping Source

Stereotype

#### Mapping Target

Membership

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Membership::memberElement () : Element [1]  
`self.ownedMemberElement ()`
- Membership::ownedMemberElement () : Element [0..1]  
`StereotypeOccurrenceUsageMultiplicityRange_Mapping.getMapped(from)`

### 7.7.9.3.30 StereotypeOccurenceUsageMultiplicityRange\_Mapping

#### Description

The mapping class creates the multiplicity range element for the UML4SysML::Stereotype mapping.

#### General Mappings

ToFeature\_Init  
Mapping

#### Mapping Source

Stereotype

#### Mapping Target

MultiplicityRange

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- MultiplicityRange::ownedRelationship () : Relationship [0..\*]  
`Set{StereotypeOccurenceUsageMultiplicityRangeMembership_Mapping.getMapped(from) }`

### 7.7.9.3.31 StereotypeOccurenceUsageMultiplicityRangeInfinity\_Mapping

#### Description

The mapping class creates the literal infinity element for the multiplicity range element for the UML4SysML::Stereotype mapping.

#### General Mappings

ToExpression\_Init  
Mapping

#### Mapping Source

Stereotype

#### Mapping Target

LiteralInfinity

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `LiteralInfinity::ownedRelationship () : Relationship [0..*]`

```
Set{StereotypeOccurenceUsageInfinityReturnParameterMembership_Mapping.getMapped(from)}
```

### 7.7.9.3.32 StereotypeOccurenceUsageInfinityReturnParameter\_Mapping

#### Description

The mapping class creates the return parameter relationship for the literal infinity element for the multiplicity range element for the `UML4SysML::Stereotype` mapping.

#### General Mappings

ToFeature\_Init  
Mapping

#### Mapping Source

Stereotype

#### Mapping Target

Feature

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Feature::direction () : FeatureDirectionKind [0..1]`

```
SysMLv2::FeatureDirectionKind::out
```

### 7.7.9.3.33 StereotypeOccurenceUsageInfinityReturnParameterMembership\_Mapping

#### Description

## General Mappings

ToReturnParameterMembership\_Init  
Mapping

## Mapping Source

Stereotype

## Mapping Target

ReturnParameterMembership

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReturnParameterMembership::ownedMemberParameter () : Feature [0..1]  
`StereotypeOccurenceUsageInfinityReturnParameter_Mapping.getMapped(from)`
- ReturnParameterMembership::memberParameter () : Feature [1]  
`self.ownedMemberParameter()`
- ReturnParameterMembership::ownedRelatedElement () : Element [0..\*]  

```
let member: KerML::Element = self.ownedMemberParameter() in
if member.oclIsUndefined() then
  Set{}
else
  Set{self.ownedMemberParameter()}
endif
```

### 7.7.9.3.34 StereotypeOccurenceUsageMultiplicityRangeMembership\_Mapping

#### Description

Creates a membership relationship for *memberElement()*.

## General Mappings

ToMembership\_Init  
Mapping

## Mapping Source

Stereotype

## Mapping Target

Membership

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Membership::memberElement () : Element [1]  
`self.ownedMemberElement ()`
- Membership::ownedMemberElement () : Element [0..1]  
`StereotypeOccurenceUsageMultiplicityRangeInfinity_Mapping.getMapped (from)`

## 7.7.10 SimpleClassifiers

### 7.7.10.1 Overview

Table 14. List of all mappings

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
DataType	AttributeDefinition
Enumeration	EnumerationDefinition
EnumerationLiteral	ConnectionUsage EnumerationUsage
Interface	PortDefinition
InterfaceRealization	Dependency
PrimitiveType	AttributeDefinition
Reception	ItemUsage
Signal	ItemDefinition

### 7.7.10.2 Mapping Specifications

#### 7.7.10.2.1 Attribute\_Mapping

##### Description

An UML4SysML::Property is mapped to a SysMLv2 AttributeUsage.

##### General Mappings

PropertyCommon\_Mapping  
NamedElementMain\_Mapping

### Mapping Source

Property

### Mapping Target

AttributeUsage

### Owned Mappings

(none)

### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
if src.ocIsKindOf(UML::Property) and not
    Helper.hasStereotypeApplied(src.owner,
        'SysML::ConstraintBlocks::ConstraintBlock') then
    let p: UML::Property = src.ocAsType(UML::Property) in
    if p.type.ocIsUndefined() then
        false
    else
        p.type.ocIsKindOf(UML::DataType) and
        (p.association.ocIsUndefined() or p.association.ownedEnd->excludes(p))
    endif
else
    false
endif
```

### Mapping rules

The mapping class only has inherited rules. See the mapping classes in the general mapping section for details.

#### 7.7.10.2.2 AttributeRedefined\_Mapping

##### Description

An UML4SysML::SimpleClassifiers::Property is mapped to a SysML v2 AttributeUsage.

##### General Mappings

PropertyCommon\_Mapping

### Mapping Source

Property

### Mapping Target

ReferenceUsage

### Owned Mappings

(none)

### Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceUsage::ownedRelationship () : Relationship [0..*]`

```
let typing: KerML::FeatureTyping =
  AssociationToFeatureTyping_Mapping.getMapped(from) in
let subsetting: Set(KerML::Subsetting) =
  from.subsettedProperty
  ->collect(p | PropertySubsetting_Mapping.getMapped(from, p))->asSet() in
let subsettingMultiplicityTyping: Set(KerML::Relationship) =
  subsetting
  ->union(Set{AttributeRedefinedRedefinition_Mapping.getMapped(from)}->union(
    if typing.ocIsUndefined() then
      Set{MultiplicityMembership_Mapping.getMapped(from)}
    else
      Set{MultiplicityMembership_Mapping.getMapped(from), typing}
    endif)->asSet() in
if from.defaultValue.ocIsUndefined() then
  subsettingMultiplicityTyping
else
  subsettingMultiplicityTyping
  ->including(PropertyDefaultValue_Mapping.getMapped(from))
endif
```

### 7.7.10.2.3 AttributeRedefinedRedefinition\_Mapping

#### Description

Creates a redefinition relationship for the *redefiningFeature()* and the *redefinedFeature()*.

#### General Mappings

ToRedefinition\_Init  
Mapping

#### Mapping Source

Property

#### Mapping Target

Redefinition

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Redefinition::redefinedFeature () : Feature [1]`  
`from.redefinedProperty.get(0)`

#### 7.7.10.2.4 AttributeRedefinedMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

##### General Mappings

ElementFeatureMembership\_Mapping

##### Mapping Source

Element

##### Mapping Target

FeatureMembership

##### Owned Mappings

(none)

##### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
src.oclIsKindOf(UML::Property)
and (src.oclAsType(UML::Property).redefinedElement->size() > 0)
```

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureMembership::ownedMemberFeature () : Feature [0..1]`  
`AttributeRedefined_Mapping.getMapped(from)`

#### 7.7.10.2.5 AttributeRedefinedFeatureTyping\_Mapping

##### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

##### General Mappings

StructuralFeatureToFeatureTyping\_Mapping

##### Mapping Source



StructuralFeature

### Mapping Target

FeatureTyping

### Owned Mappings

(none)

### Applicable filters

(none)

## 7.7.10.2.6 BehavoredClassifier\_Mapping

### Description

The abstract mapping class maps the abstract metaclass UML4SysML::BehavoredClassifiers to a SysMLv2 Classifier. The mapping class is used by concrete mapping classes, for example, Block\_Mapping.

### General Mappings

Classifier\_Mapping

### Mapping Source

BehavoredClassifier

### Mapping Target

Classifier

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Classifier::ownedRelationship () : Relationship [0..\*]

```
let toElementFMS: Set(UML::Element) =
  from.ownedElement->select(e | (e.ocIsKindOf(UML::Property) and
    (e.ocIsType(UML::Property).redefinedProperty->size() = 0)) or
    e.ocIsKindOf(UML::Operation) or e.ocIsKindOf(UML::Connector)) in
let redefinedAttributes: Set(UML::Element) =
  from.ownedElement->select(e | from.ocIsKindOf(UML::DataType) and
    (e.ocIsType(UML::Property).redefinedProperty->size() > 0)) in
let generalizations : Set(UML::Generalization) =
  from.ownedElement
```

```

->select(e | e.ocIsKindOf(UML::Generalization)) in
let constraints : Set(UML::Constraint) =
  UML::Constraint.allInstances()
->select( c | c.constrainedElement->includes(from)) in
let toElementOMS: Set(UML::Element) =
  (((from.ownedElement - toElementFMS) - redefinedAttributes) -
  generalizations) - from.ownedComment in
let relationships: Sequence(KerML::Relationship) =
toElementOMS->collect(e | ElementOwningMembership_Mapping.getMapped(e))->asSet()
->union(toElementFMS->collect(e |
  ElementFeatureMembership_Mapping.getMapped(e))->asSet())
->union(constraints->collect(e |
  ConstrainedElementFeatureMembership_Mapping.getMapped(e))->asSet())
->union(redefinedAttributes->collect(e |
  AttributeRedefinedMembership_Mapping.getMapped(e))->asSet())
->union(generalizations->collect(e |
  Generalization_Mapping.getMapped(e))->asSet())
->union(self.oclAsType(ElementMain_Mapping).ownedRelationship()) in
if from.classifierBehavior.ocIsUndefined() then
  relationships
else
  relationships
->including(BehavioredClassifierFeatureMembership_Mapping.getMapped(from))
endif

```

#### 7.7.10.2.7 BehavioredClassifierFeatureMembership\_Mapping

##### Description

##### General Mappings

ToFeatureMembership\_Init  
Mapping

##### Mapping Source

BehavioredClassifier

##### Mapping Target

FeatureMembership

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureMembership::ownedMemberFeature () : Feature [0..1]  
BehavioredClassifierActionUsage\_Mapping.getMapped(from)

#### 7.7.10.2.8 BehavoredClassifierFeatureTyping\_Mapping

##### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

##### General Mappings

ToFeatureTyping\_Init  
Mapping

##### Mapping Source

BehavoredClassifier

##### Mapping Target

FeatureTyping

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureTyping::type () : Type [1]  
from

#### 7.7.10.2.9 BehavoredClassifierActionUsage\_Mapping

##### Description

The BehavoredClassifierToPerformActionUsage\_Mapping class creates a PerformActionUsage element to call the transformed SysML v1 classifier behavior.

##### General Mappings

ToActionUsage\_Init  
Mapping

##### Mapping Source

BehavoredClassifier

##### Mapping Target

ActionUsage

##### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ActionUsage::declaredName () : String [0..1]`  
`'classifierBehavior'`
- `ActionUsage::ownedRelationship () : Relationship [0..*]`  
`Set{BehavioredClassifierFeatureTyping_Mapping.getMapped(from) }`

### 7.7.10.2.10 DataType\_Mapping

#### Description

A `UML4SysML::SimpleClassifiers::DataType` is mapped to a `SysML v2 AttributeDefinition`. The mapping also cover the transformation of `UML4SysML::PrimitiveType` elements.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
part def SysMLv1Block {  
    attribute sysMLv1Property : ScalarValues::Integer;  
}
```

#### General Mappings

Classifier\_Mapping

#### Mapping Source

DataType

#### Mapping Target

AttributeDefinition

#### Owned Mappings

(none)

#### Applicable filters

(none)

### 7.7.10.2.11 Enumeration\_Mapping

#### Description

A UML4SysML::Enumeration is mapped to a SysML v2 EnumerationDefinition.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
enum def SysMLv1Enumeration {  
    enum sysMLv1Literal1;  
    enum sysMLv1Literal2;  
}
```

## General Mappings

DataType\_Mapping

## Mapping Source

Enumeration

## Mapping Target

EnumerationDefinition

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- EnumerationDefinition::ownedRelationship () : Relationship [0..\*]

```
self.oclAsType(Classifier_Mapping).ownedRelationship()  
->union(from.ownedLiteral->collect(e |  
    EnumerationVariantMembership_Mapping.getMapped(e)) ->asSet())
```

- EnumerationDefinition::isVariation () : Boolean [1]

```
true
```

### 7.7.10.2.12 EnumerationLiteral\_Mapping

## Description

A UML4SysML::EnumerationLiteral is mapped to a SysML v2 EnumerationUsage.

## General Mappings

ToFeature\_Init

InstanceSpecification\_Mapping

**Mapping Source**

EnumerationLiteral

**Mapping Target**

EnumerationUsage

**Owned Mappings**

(none)

**Applicable filters**

(none)

**7.7.10.2.13 EnumerationVariantMembership\_Mapping****Description**

The EnumerationVariantMembership\_Mapping class creates the variant membership relationship between the enumeration definition and a enumeration usage.

**General Mappings**

ToOwningMembership\_Init  
Mapping

**Mapping Source**

EnumerationLiteral

**Mapping Target**

VariantMembership

**Owned Mappings**

(none)

**Applicable filters**

(none)

**Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- VariantMembership::ownedMemberElement () : Element [1]

from

**7.7.10.2.14 Interface\_Mapping****Description**

A UML4SysML::Interface is mapped to a SysMLv2 PortDefinition. The mapping also includes the generation of an appropriate ConjugatedPortDefinition. That mappings is performed by the mapping classes InterfaceConjugatedPortDefinitionMembership\_Mapping, InterfacePortConjugation\_Mapping, and InterfaceConjugatedPortDefinition\_Mapping.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
port def SysMLv1Interface {  
    attribute sysMLv1Property;  
}
```

### General Mappings

ToPortDefinition\_Init  
Classifier\_Mapping

### Mapping Source

Interface

### Mapping Target

PortDefinition

### Owned Mappings

- conjugatedPortDefinitionMembership : InterfaceConjugatedPortDefinitionMembership\_Mapping

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- PortDefinition::ownedRelationship () : Relationship [0..\*]  
  
self.oclAsType(Classifier\_Mapping).ownedRelationship()  
->including(conjugatedPortDefinitionMembership)

#### 7.7.10.2.15 InterfaceConjugatedPortDefinition\_Mapping

### Description

As part of the mapping from a UML4SysML::Interface to a SysMLv2 PortDefinition, this mapping class is used to create the appropriate ConjugatedPortDefinition.

### General Mappings

ToPortDefinition\_Init  
Mapping

### Mapping Source

Interface

### Mapping Target

ConjugatedPortDefinition

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ConjugatedPortDefinition::declaredName () : String [0..1]  
`'~'+from.name`
- ConjugatedPortDefinition::ownedRelationship () : Relationship [0..\*]  
`Set{InterfacePortConjugation_Mapping.getMapped(from)}`

## 7.7.10.2.16 InterfaceConjugatedPortDefinitionMembership\_Mapping

### Description

As part of the mapping from a UML4SysML::Interface to a SysML v2 PortDefinition, this mapping class is used to create the membership relationship for the ConjugatedPortDefinition.

### General Mappings

ToOwningMembership\_Init  
Mapping

### Mapping Source

Interface

### Mapping Target

OwningMembership

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules



In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `OwningMembership::ownedMemberElement () : Element [1]`  
`InterfaceConjugatedPortDefinition_Mapping.getMapped (from)`

#### 7.7.10.2.17 InterfacePortConjugation\_Mapping

##### Description

As part of the mapping from a `UML4SysML::Interface` to a `SysML v2 PortDefinition`, this mapping class is used to create the appropriate `PortConjugation` relationship.

##### General Mappings

`ToRelationship_Init`  
`Mapping`

##### Mapping Source

`Interface`

##### Mapping Target

`PortConjugation`

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `PortConjugation::conjugatedType () : Type [1]`  
`SysMLv2::ConjugatedPortDefinition.allInstances ()`  
`->collect (cpd | cpd.owningRelationship)`  
`->select (r | r.oclIsKindOf (SysMLv2::Membership))`  
`->any (m | m.memberName = from.name)`
- `PortConjugation::originalPortDefinition () : PortDefinition [1]`  
`from`

#### 7.7.10.2.18 InterfaceRealization\_Mapping

##### Description

A `UML4SysML::InterfaceRealization` is mapped to a `SysMLv2 Subclassification` relationship.

##### General Mappings

ToSpecialization\_Init  
Mapping

#### **Mapping Source**

InterfaceRealization

#### **Mapping Target**

Subclassification

#### **Owned Mappings**

(none)

#### **Applicable filters**

(none)

#### **Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Subclassification::subclassifier () : Type [1]  
`Classifier_Mapping.getMapped(from.specific)`
- Subclassification::superclassifier () : Type [1]  
`Classifier_Mapping.getMapped(from.general)`

### **7.7.10.2.19 PrimitiveType\_Mapping**

#### **Description**

The PrimitiveType\_Mapping class maps a UML4SysML::PrimitiveType to a SysML v2 AttributeDefinition.

#### **General Mappings**

DataType\_Mapping

#### **Mapping Source**

PrimitiveType

#### **Mapping Target**

AttributeDefinition

#### **Owned Mappings**

(none)

#### **Applicable filters**

(none)

#### 7.7.10.2.20 Reception\_Mapping

##### Description

A UML4SysML::Reception is mapped to a SysML v2 AttributeUsage with feature direction "in".

##### General Mappings

BehavioralFeature\_Mapping

##### Mapping Source

Reception

##### Mapping Target

ItemUsage

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ItemUsage::direction () : FeatureDirectionKind [0..1]  
`SysMLv2::FeatureDirectionKind::in`
- ItemUsage::ownedRelationship () : Relationship [0..\*]  
`self.oclAsType (ElementMain_Mapping).ownedRelationship ()`  
`->including (ReceptionFeatureTyping_Mapping.getMapped (from) )`

#### 7.7.10.2.21 ReceptionFeatureTyping\_Mapping

##### Description

A UML4SysML::Reception is mapped to SysML v2 AttributeUsage. The ReceptionToFeatureTyping\_Mapping class creates the type of the AttributeUsage which is the Signal of the Reception.

##### General Mappings

TypedElementFeatureTyping\_Mapping

##### Mapping Source

Reception

##### Mapping Target

FeatureTyping

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureTyping::type () : Type [1]`  
`Classifier_Mapping.getMapped(from.signal)`

## 7.7.10.2.22 Signal\_Mapping

### Description

A `UML4SysML::Signal` is mapped to a SysML v2 `ItemDefinition`.

### General Mappings

`Classifier_Mapping`

### Mapping Source

`Signal`

### Mapping Target

`ItemDefinition`

### Owned Mappings

(none)

### Applicable filters

(none)

## 7.7.11 StateMachines

### 7.7.11.1 Overview

**Table 15. List of all mappings**

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
ConnectionPointReference	StateUsage
FinalState	StateUsage
Pseudostate	StateUsage ActionUsage

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
Region	StateUsage
State	StateUsage
StateMachine	StateDefinition
Transition	TransitionUsage

## 7.7.11.2 Mapping Specifications

### 7.7.11.2.1 ChangeTriggerReferenceUsage\_Mapping

#### Description

\*\*\* not specified yet \*\*\*

#### General Mappings

UniqueMapping  
ToReferenceUsage\_Init

#### Mapping Source

Trigger

#### Mapping Target

ReferenceUsage

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReferenceUsage::isEnd () : Boolean [1]  
`true`
- ReferenceUsage::ownedRelationship () : Relationship [0..\*]  
`Set{ChangeTriggerReferenceSubsetting_Mapping.getMapped(from) }`

### 7.7.11.2.2 CommonPseudostate\_Mapping

#### Description

Abstract mapping class for common rules for pseudostates mappings.

## General Mappings

Namespace\_Mapping

## Mapping Source

Pseudostate

## Mapping Target

Namespace

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Namespace::ownedRelationship () : Relationship [0..\*]  

```
let toFeatureMS : Set(UML::Element) =  
    from.ownedElement->select(e | e.ocIsKindOf(UML::Region))->asSet() in  
let toElementOMS : Set(UML::Element) =  
    from.ownedElement - toFeatureMS in  
toElementOMS  
->collect(e | ElementOwningMembership_Mapping.getMapped(e))->asSet()  
->union(toFeatureMS  
->collect(e | ElementFeatureMembership_Mapping.getMapped(e))->asSet())  
->union(self.ocAsType(ElementMain_Mapping).ownedRelationship())
```

### 7.7.11.2.3 ConnectionPointReference\_Mapping

#### Description

A UML4SysML::ConnectionPointReference element is mapped to a SysML v2 StateUsage.

## General Mappings

Namespace\_Mapping  
ToStateUsage\_Init

## Mapping Source

ConnectionPointReference

## Mapping Target

StateUsage

## Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- StateUsage::isComposite () : Boolean [1]

false

- StateUsage::ownedRelationship () : Relationship [0..\*]

```
let toFeatureMS : Set(UML::Element) =
  from.ownedElement->select(e | e.ocIsKindOf(UML::Region)) in
let toElementOMS : Set(UML::Element) =
  (from.ownedElement - toFeatureMS) - from.ownedComment in
toElementOMS->collect(e | ElementOwningMembership_Mapping.getMapped(e))
->asSet()
->union(toFeatureMS
  ->collect(e | ElementFeatureMembership_Mapping.getMapped(e))->asSet())
->union(self.oclAsType(ElementMain_Mapping).ownedRelationship())
```

#### 7.7.11.2.4 DoBehaviorStateSubactionMembership\_Mapping

##### Description

Creates a state subaction membership relationship for *memberFeature()*.

##### General Mappings

StateBehaviorStateSubactionMembership\_Mapping

##### Mapping Source

Behavior

##### Mapping Target

StateSubactionMembership

##### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- StateSubactionMembership::kind () : StateSubactionKind [1]

`SysMLv2::SubactionKind::do`

#### 7.7.11.2.5 EntryBehaviorStateSubactionMembership\_Mapping

##### Description

Creates a state subaction membership relationship for *memberFeature()*.

##### General Mappings

StateBehaviorStateSubactionMembership\_Mapping

##### Mapping Source

Behavior

##### Mapping Target

StateSubactionMembership

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- StateSubactionMembership::kind () : StateSubactionKind [1]

`SysMLv2::SubactionKind::entry`

#### 7.7.11.2.6 ExitBehaviorStateSubactionMembership\_Mapping

##### Description

Creates a state subaction membership relationship for *memberFeature()*.

##### General Mappings

StateBehaviorStateSubactionMembership\_Mapping

##### Mapping Source

Behavior

##### Mapping Target



StateSubactionMembership

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- StateSubactionMembership::kind () : StateSubactionKind [1]  
`SysMLv2::SubactionKind::exit`

#### 7.7.11.2.7 FinalState\_Mapping

##### Description

A UML4SysML::FinalState is mapped to a SysML v2 StateUsage. The details of the mapping are not defined yet.

##### General Mappings

State\_Mapping

##### Mapping Source

FinalState

##### Mapping Target

StateUsage

##### Owned Mappings

(none)

##### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
src.ocIsTypeOf(UML::FinalState)
```

##### Mapping rules

The mapping class only has inherited rules. See the mapping classes in the general mapping section for details.

#### 7.7.11.2.8 InitialState\_Mapping

##### Description

The mapping class maps a Pseudostate with kind = initial to a SysML v2 ActionUsage.

## General Mappings

CommonPseudostate\_Mapping

## Mapping Source

Pseudostate

## Mapping Target

ActionUsage

## Owned Mappings

(none)

## Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
(src.kind = PseudostateKind::initial)
```

## Mapping rules

The mapping class only has inherited rules. See the mapping classes in the general mapping section for details.

### 7.7.11.2.9 InitialStateSubactionMembership\_Mapping

## Description

Creates a StateSubactionMembership relationship.

## General Mappings

ToStateSubactionMembership\_Init  
Mapping

## Mapping Source

Pseudostate

## Mapping Target

StateSubactionMembership

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- StateSubactionMembership::kind () : StateSubactionKind [1]  
`SysMLv2::SubactionKind::entry`
- StateSubactionMembership::ownedMemberFeature () : Feature [1]  
`InitialState_Mapping.getMapped(from)`

#### 7.7.11.2.10 PseudoState\_Mapping

##### Description

A UML4SysML::PseudoState is mapped to a SysML v2 StateUsage.

##### General Mappings

CommonPseudostate\_Mapping  
ToStateUsage\_Init

##### Mapping Source

Pseudostate

##### Mapping Target

StateUsage

##### Owned Mappings

(none)

##### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
(src.kind <> PseudostateKind::initial)
```

##### Mapping rules

The mapping class only has inherited rules. See the mapping classes in the general mapping section for details.

#### 7.7.11.2.11 Region\_Mapping

##### Description

A UML4SysML::Region is mapped to SysML v2 StateUsage.

##### General Mappings

Namespace\_Mapping  
ToStateUsage\_Init

##### Mapping Source

Region

## Mapping Target

StateUsage

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- StateUsage::ownedRelationship () : Relationship [0..\*]

```
let initialState : Set(UML::Pseudostate) =
    from.ownedElement->select(e | e.ocIsKindOf(UML::Pseudostate)
        and e.ocIsType(UML::Pseudostate).kind = PseudostateKind::initial)->asSet() in
let toFeatureMS : Set(UML::Element) =
    from.ownedElement->select(e | e.ocIsKindOf(UML::Region))->asSet() in
let toElementOMS : Set(UML::Element) =
    ((from.ownedElement - initialState) - toFeatureMS) - from.ownedComment in
toElementOMS->collect(e | ElementOwningMembership_Mapping.getMapped(e))->asSet()
->union(toFeatureMS->collect(e |
    ElementFeatureMembership_Mapping.getMapped(e))->asSet())
->union(initialState->collect(e |
    InitialStateMembership_Mapping.getMapped(e))->asSet())
->union(self.ocIsType(ElementMain_Mapping).ownedRelationship())
```

### 7.7.11.2.12 State\_Mapping

#### Description

A UML4SysML::State is mapped to a SysMLv2 StateUsage. If it is a composite state, it is mapped to a parallel state.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
state SysMLv1State parallel {
    entry; then SysMLv1StateA;
    state SysMLv1StateA;
}
```

## General Mappings

Namespace\_Mapping  
ToStateUsage\_Init

## Mapping Source

State

## Mapping Target

StateUsage

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- StateUsage::isParallel () : Boolean [1]

```
from.isComposite
```

- StateUsage::ownedRelationship () : Relationship [0..\*]

```
let toFeatureMS : Set(UML::Element) =
  from.ownedElement->select(e | e.ocIsKindOf(UML::Region))->asSet() in
let toElementOMS : Set(UML::Element) =
  (from.ownedElement - toFeatureMS) - from.ownedComment in
let relationships : Set(KerML::Relationship) =
  toElementOMS
  ->collect(e | ElementOwningMembership_Mapping.getMapped(e))->asSet()
->union(toFeatureMS
  ->collect(e | ElementFeatureMembership_Mapping.getMapped(e))->asSet())
->union(self.ocAsType(ElementMain_Mapping).ownedRelationship()) in
```

```
let consideredEntry : Set(KerML::Relationship) =
if (from.entry.ocIsUndefined()) then
  relationships
else
  relationships
  ->including(
    EntryBehaviorStateSubactionMembership_Mapping.getMapped(
      from.entry))
endif in
```

```
let consideredDo : Set(KerML::Relationship) =
if (from.doActivity.ocIsUndefined()) then
  consideredEntry
else
  consideredEntry
  ->including(DoBehaviorStateSubactionMembership_Mapping.getMapped(
    from.doActivity))
endif in
if (from.exit.ocIsUndefined()) then
  consideredDo
else
  consideredDo
  ->including(ExitBehaviorStateSubactionMembership_Mapping.getMapped(
```

```

        from.exit())
    endif

```

#### 7.7.11.2.13 StateBehaviorPerformActionUsage\_Mapping

##### Description

The mapping class creates a perform action usage typed by the target element of the mapping of the source behavior element.

##### General Mappings

ToPerformActionUsage\_Init  
Mapping

##### Mapping Source

Behavior

##### Mapping Target

PerformActionUsage

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- PerformActionUsage::ownedRelationship () : Relationship [0..\*]

```

Set{StateBehaviorPerformActionUsageFeatureTyping_Mapping.getMapped(from) }

```

#### 7.7.11.2.14 StateBehaviorPerformActionUsageFeatureTyping\_Mapping

##### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

##### General Mappings

ToFeatureTyping\_Init  
Mapping

##### Mapping Source

Behavior

##### Mapping Target

FeatureTyping

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureTyping::type () : Type [1]`

`from`

### 7.7.11.2.15 StateBehaviorStateSubactionMembership\_Mapping

#### Description

Abstract mapping class for mapping classes for state behavior mappings (enty, do and exit).

#### General Mappings

`ToStateSubactionMembership_Init  
Mapping`

#### Mapping Source

`Behavior`

#### Mapping Target

`StateSubactionMembership`

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `StateSubactionMembership::ownedMemberFeature () : Feature [1]`

`StateBehaviorPerformActionUsage_Mapping.getMapped (from)`

### 7.7.11.2.16 StateDefinition\_Mapping

#### Description

A `UML4SysML::StateMachine` is mapped to a SysML v2 `StateDefinition`.

## General Mappings

Behavior\_Mapping

## Mapping Source

StateMachine

## Mapping Target

StateDefinition

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- StateDefinition::ownedRelationship () : Relationship [0..\*]

```
let initialState : Set(UML::Element) =
  from.ownedElement
  ->select(e | e.ocIsKindOf(UML::Pseudostate) and
    e.ocIsType(UML::Pseudostate).kind = UML::PseudostateKind::initial) in
let toParameterMS : Set(UML::Element) =
  from.ownedElement->select(e | e.ocIsKindOf(UML::Parameter)) in
let parameterSets : Set(UML::Element) =
  from.ownedElement->select(e | e.ocIsKindOf(UML::ParameterSet)) in
let toFeatureMS : Set(UML::Element) =
  from.ownedElement->select(e | e.ocIsKindOf(UML::Region)
    or e.ocIsKindOf(UML::ChangeEvent) or e.ocIsKindOf(UML::TimeEvent)) in
let rejectedElements : Set(UML::Element) =
  from.ownedElement->select(e | e.ocIsKindOf(UML::SignalEvent)) in
let toElementOMS : Set(UML::Element) =
  ((from.ownedElement - toFeatureMS) - toParameterMS) - initialState in
toElementOMS->collect(e | ElementOwningMembership_Mapping.getMapped(e))
->union(toFeatureMS->collect(e | ElementFeatureMembership_Mapping.getMapped(e)))
->union(toParameterMS->collect(e | ParameterMembership_Mapping.getMapped(e)))
->union(parameterSets->collect(e | ParameterSetMembership_Mapping.getMapped(e)))
->union(initialState->collect(e | InitialStateMembership_Mapping.getMapped(e)))
```

- StateDefinition::isParallel () : Boolean [1]

```
from.region->size() > 1
```

### 7.7.11.2.17 TimeTriggerReferenceUsage\_Mapping

#### Description

\*\*\* not specified yet \*\*\*



## General Mappings

ToReferenceUsage\_Init  
UniqueMapping

## Mapping Source

Trigger

## Mapping Target

ReferenceUsage

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReferenceUsage::ownedRelationship () : Relationship [0..\*]  
`Set{TimeTriggerReferenceSubsetting_Mapping.getMapped(from) }`
- ReferenceUsage::isEnd () : Boolean [1]  
`true`

### 7.7.11.2.18 Transition\_Mapping

#### Description

A UML4SysML::Transition is mapped to a SysML v2 TransitionUsage.

## General Mappings

Namespace\_Mapping  
ToTransitionUsage\_Init

## Mapping Source

Transition

## Mapping Target

TransitionUsage

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `TransitionUsage::ownedRelationship () : Relationship [0..*]`

```
self.oclAsType(ElementMain_Mapping).ownedRelationship()  
->union((from.ownedElement - from.ownedComment)  
->collect(e | ElementOwningMembership_Mapping.getMapped(e)) ->asSet())  
->union(from.trigger->select(t | t.event.oclIsKindOf(UML::ChangeEvent)  
or t.event.oclIsKindOf(UML::TimeEvent))  
->collect(e | TransitionTriggerFeatureMembership_Mapping.getMapped(e))  
->asSet())  
->including(TransitionSuccession_Mapping.getMapped(from))
```

- `TransitionUsage::target () : ActionUsage [1]`

`from.target`

- `TransitionUsage::source () : ActionUsage [1]`

`from.source`

### 7.7.11.2.19 TransitionSuccession\_Mapping

#### Description

The mapping class creates the source Feature element of the Succession that is part of the TransitionUsage that is the target element of the UML4SysML::Transition mapping.

#### General Mappings

ToConnector\_Init  
ToMembership\_Init  
Mapping

#### Mapping Source

Transition

#### Mapping Target

Succession

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Succession::ownedRelationship () : Relationship [0..\*]  
`OrderedSet { TransitionSuccessionSourceMembership_Mapping.getMapped (from) ,  
TransitionSuccessionTargetMembership_Mapping.getMapped (from) }`

#### 7.7.11.2.20 TransitionSourceToSubsetting\_Mapping

##### Description

Creates a subsetting relationship.

##### General Mappings

ToSubsetting\_Init  
Mapping

##### Mapping Source

Transition

##### Mapping Target

Subsetting

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Subsetting::subsettingFeature () : Feature [1]  
`ElementMain_Mapping.getMapped (from.source)`
- Subsetting::subsettingFeature () : Feature [1]  
`TransitionSuccessionSource_Mapping.getMapped (from)`

#### 7.7.11.2.21 TransitionSuccessionSource\_Mapping

##### Description

The mapping class creates the Succession element that is part of the TransitionUsage that is the target element of the UML4SysML::Transition mapping.

##### General Mappings

ToFeature\_Init  
Mapping

#### Mapping Source

Transition

#### Mapping Target

Feature

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Feature::ownedRelationship () : Relationship [0..\*]  
`Set{TransitionSourceToSubsetting_Mapping.getMapped(from) }`
- Feature::isEnd () : Boolean [1]  
`true`
- Feature::declaredName () : String [0..1]  
`'source'`

### 7.7.11.2.22 TransitionSuccessionSourceMembership\_Mapping

#### Description

Creates a membership relationship for *memberElement()*.

#### General Mappings

ToEndFeatureMembership\_Init  
Mapping

#### Mapping Source

Transition

#### Mapping Target

EndFeatureMembership

#### Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- EndFeatureMembership::ownedMemberFeature () : Feature [1]  
`TransitionSuccessionSource_Mapping.getMapped(from)`

### 7.7.11.2.23 TransitionSuccessionTarget\_Mapping

#### Description

The mapping class creates the target Feature element of the Succession that is part of the TransitionUsage that is the target element of the UML4SysML::Transition mapping.

#### General Mappings

ToFeature\_Init  
Mapping

#### Mapping Source

Transition

#### Mapping Target

Feature

#### Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Feature::isEnd () : Boolean [1]  
`true`
- Feature::declaredName () : String [0..1]  
`'target'`
- Feature::ownedRelationship () : Relationship [0..\*]  
`Set{TransitionTargetToSubsetting_Mapping.getMapped(from) }`

#### 7.7.11.2.24 TransitionSuccessionTargetMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

##### General Mappings

ToEndFeatureMembership\_Init  
Mapping

##### Mapping Source

Transition

##### Mapping Target

EndFeatureMembership

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- EndFeatureMembership::ownedMemberFeature () : Feature [1]  
`TransitionSuccessionTarget_Mapping.getMapped(from)`

#### 7.7.11.2.25 TransitionTargetToSubsetting\_Mapping

##### Description

Creates a subsetting relationship.

##### General Mappings

ToSubsetting\_Init  
Mapping

##### Mapping Source

Transition

##### Mapping Target

Subsetting

##### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Subsetting::subsettingFeature () : Feature [1]  
`ElementMain_Mapping.getMapped(from.target)`
- Subsetting::subsettingFeature () : Feature [1]  
`TransitionSuccessionTarget_Mapping.getMapped(from)`

### 7.7.11.2.26 TransitionTriggerFeatureMembership\_Mapping

#### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

#### General Mappings

ToFeatureMembership\_Init  
UniqueMapping

#### Mapping Source

Trigger

#### Mapping Target

FeatureMembership

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureMembership::ownedMemberFeature () : Feature [1]  
  
`if from.event.ocliIsKindOf(UML::TimeEvent) then`  
`TimeTriggerCalculationUsage_Mapping.getMapped(from)`

```

else if from.event.oclIsKindOf(UML::ChangeEvent) then
    ChangeTriggerConstraintUsage_Mapping.getMapped(from)
else
    OclUndefined
endif endif

```

## 7.7.12 StructuredClassifiers

### 7.7.12.1 Overview

**Table 16. List of all mappings**

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
Association	ConnectionDefinition
AssociationClass	OccurrenceDefinition ConnectionDefinition
Class	OccurrenceDefinition
Connector	ConnectionUsage
ConnectorEnd	Feature
Port	PortUsage OccurrenceUsage Feature AttributeUsage

### 7.7.12.2 Mapping Specifications

#### 7.7.12.2.1 AssociationClass\_Mapping

##### Description

A UML4SysML::AssociationClass is mapped to a SysML v2 ConnectionDefinition. The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```

part def SysMLv1Block1;
part def SysMLv1Block2;
connection def SysMLv1AssociationBlock {
    end : SysMLv1Block1;
    end : SysMLv1Block2;
}

```

##### General Mappings

AssociationCommon\_Mapping

##### Mapping Source

AssociationClass

##### Mapping Target

ConnectionDefinition



## Owned Mappings

(none)

## Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
not Helper.hasStereotypeApplied(src, 'SysML::Blocks::Block')
```

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ConnectionDefinition::ownedRelationship () : Relationship [0..*]`

```
let nonOwnedEnds: OrderedSet(UML::Property) =  
    (from.memberEnd-from.ownedEnd)->asOrderedSet() in  
let generalizations : Set(UML::Generalization) =  
    from.ownedElement->select(e | e.ocIsKindOf(UML::Generalization)) in  
let others: OrderedSet(UML::Element) =  
    ((from.ownedElement-from.memberEnd)-generalizations)->asOrderedSet() in  
nonOwnedEnds->collect(e | NonOwnedEndMembership_Mapping.getMapped(e))  
->union(from.ownedEnd->collect(e | OwnedEndMembership_Mapping.getMapped(e)))  
->union(generalizations->collect(e | Generalization_Mapping.getMapped(e)))  
->union(others->collect(e | ElementOwningMembership_Mapping.getMapped(e)))  
->asOrderedSet()
```

### 7.7.12.2.2 AssociationCommon\_Mapping

#### Description

A `UML4SysML::Association` is mapped to a `SysML v2 ConnectionDefinition`. This is the abstract base class of all concrete association mapping classes.

#### General Mappings

Classifier\_Mapping  
Relationship\_Mapping

#### Mapping Source

Association

#### Mapping Target

Association

#### Owned Mappings

(none)

#### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
src.memberEnd->select( m | m.type.ocIsKindOf(UML::UseCase))->isEmpty()
```

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Association::ownedRelationship () : Relationship [0..\*]

```
let nonOwnedEnds: OrderedSet(UML::Property) =
    (from.memberEnd-from.ownedEnd)->asOrderedSet() in
nonOwnedEnds->collect(e | NonOwnedEndMembership_Mapping.getMapped(e))->asOrderedSet()
->union(self.ocIsType(Classifier_Mapping).ownedRelationship()->asOrderedSet())
->asOrderedSet()
```

### 7.7.12.2.3 AssociationMetadataUsage\_Mapping

#### Description

The mapping class creates the MetadataUsage element to annotate a ConnectionDefinition that its mapping source element is a derived association.

#### General Mappings

ToMetadataUsage\_Init  
Mapping

#### Mapping Source

Association

#### Mapping Target

MetadataUsage

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- MetadataUsage::ownedRelationship () : Relationship [0..\*]

```
Set{AssociationToFeatureTyping_Mapping.getMapped(from),
AssociationMetadataUsageFeatureMembership_Mapping.getMapped(from)}
```

#### 7.7.12.2.4 AssociationMetadataUsageFeatureMembership\_Mapping

##### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

##### General Mappings

ToFeatureMembership\_Init  
Mapping

##### Mapping Source

Association

##### Mapping Target

FeatureMembership

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureMembership::ownedMemberFeature () : Feature [1]  
`AssociationMetadataUsageFeature_Mapping.getMapped(from)`

#### 7.7.12.2.5 AssociationMetadataUsageFeatureTyping\_Mapping

##### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

##### General Mappings

ToFeatureTyping\_Init  
Mapping

##### Mapping Source

Association

##### Mapping Target

FeatureTyping

##### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureTyping::type () : Type [1]`  

```
SYSML2::MetadataDefinition.allInstances()  
->any(m | m.qualifiedName = 'SysMLv1Library::AssociationData')
```

### 7.7.12.2.6 AssociationMetadataUsageFeature\_Mapping

#### Description

The mapping class creates the feature of the MetadataUsage.

#### General Mappings

ToFeature\_Init  
Mapping

#### Mapping Source

Association

#### Mapping Target

Feature

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Feature::ownedRelationship () : Relationship [0..*]`  

```
Set{AssociationMetadataUsageRedefinition_Mapping.getMapped(from),  
AssociationMetadataUsageFeatureValue_Mapping.getMapped(from) }
```

### 7.7.12.2.7 AssociationMetadataUsageFeatureValue\_Mapping

#### Description

Creates a feature value relationship.

## General Mappings

ToFeatureValue\_Init  
Mapping

## Mapping Source

Association

## Mapping Target

FeatureValue

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureValue::value () : Expression [1]  
`LiteralBoolean_Factory.create(from.isDerived)`

### 7.7.12.2.8 AssociationMetadataUsageMembership\_Mapping

#### Description

Creates a membership relationship for *memberElement()*.

#### General Mappings

ToOwningMembership\_Init  
Mapping

#### Mapping Source

Association

#### Mapping Target

OwningMembership

#### Owned Mappings

(none)

#### Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `OwningMembership::ownedMemberElement () : Element [1]`  
`AssociationMetadataUsage_Mapping.getMapped(from)`

### 7.7.12.2.9 AssociationMetadataUsageRedefinition\_Mapping

#### Description

Creates a redefinition relationship for the *redefiningFeature()* and the *redefinedFeature()*.

#### General Mappings

ToRedefinition\_Init  
Mapping

#### Mapping Source

Association

#### Mapping Target

Redefinition

#### Owned Mappings

(none)

#### Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Redefinition::redefinedFeature () : Feature [1]`  
`SysML2::AttributeUsage.allInstances ()`  
`->any(m | m.qualifiedName = 'SysMLv1Library::AssociationData::isDerived')`

### 7.7.12.2.10 Class\_Mapping

#### Description

A `UML4SysML::Class` is mapped to a SysML v2 `OccurrenceDefinition`. The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
occurrence def UML4SysMLClass;
```

#### General Mappings

BehavioeredClassifier\_Mapping

**Mapping Source**

Class

**Mapping Target**

OccurrenceDefinition

**Owned Mappings**

(none)

**Applicable filters**

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
not Helper.isRequirement(src) and not src.oclIsTypeOf(UML::AssociationClass)
```

**Mapping rules**

The mapping class only has inherited rules. See the mapping classes in the general mapping section for details.

**7.7.12.2.11 ConnectionDefEnd\_Mapping**

**Description**

\*\*\* not specified yet \*\*\*

**General Mappings**

UniqueMapping  
End\_Mapping

**Mapping Source**

Property

**Mapping Target**

Feature

**Owned Mappings**

(none)

**Applicable filters**

(none)

**Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Feature::ownedRelationship () : Relationship [0..*]`

```
let crossFMultiplicity: Set(SysML2::ReferenceUsage) =
  if from.association.ownedEnd->includes(from) and
    not ((from.opposite.isComposite and from.lower = 0) or
      (from.lower = 0 and from.upper = -1)) then
    Set {MultiplicityReferenceUsage_Mapping.getMapped(from)}
  else
    Set{}
  endif in
let typings: Set(KerML::FeatureTyping) = if from.type.ocIsUndefined() then
  Set{}
else
  Set{StructuralFeatureToFeatureTyping_Mapping.getMapped(from)}
endif in
let subsettings: Set(KerML::CrossSubsetting) =
  if from.association.ownedEnd->excludes(from) and from.opposite.lower = 0 and
    not (from.isComposite or from.opposite.upper = -1) then
    Set{CrossSubsetting_Mapping.getMapped(from)}
  else
    Set{}
  endif in
let defaultValue: Set(KerML::OwningMembership) =
  if from.defaultValue.ocIsUndefined() then
    Set{}
  else
    Set{DefaultValue_Mapping.getMapped(from)}
  endif in
crossFMultiplicity->union (typings)
->union(subsettings)->union(defaultValue)
->including (MultiplicityMembership_Factory.create(1,1))->asSet()
```

#### 7.7.12.2.12 ConnectionDefEndMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

##### General Mappings

UniqueMapping  
ToFeatureMembership\_Init

##### Mapping Source

Property

##### Mapping Target

FeatureMembership

##### Owned Mappings

(none)

##### Applicable filters



(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureMembership::ownedMemberFeature () : Feature [1]`  
`ConnectionDefEnd_Mapping.getMapped(from)`

#### 7.7.12.2.13 ConnectionEndToSubsetting\_Mapping

### Description

Creates a subsetting relationship.

### General Mappings

ToSubsetting\_Init  
Mapping

### Mapping Source

ConnectorEnd

### Mapping Target

Subsetting

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Subsetting::subsettingFeature () : Feature [1]`  
`ConnectorEndToOwnedFeature_Mapping.getMapped(from)`
- `Subsetting::subsettingFeature () : Feature [1]`  

```
let propertyPath: OrderedSet(UML::Property) =
    Helper.getTagValueAsElementColl
    (src, 'SysML::Blocks::NestedConnectorEnd', 'propertyPath')
    ->asOrderedSet() in
if propertyPath->isEmpty() then
    ElementMain_Mapping.getMapped(from.role)
else
    ConnectorEndToSubsettingFeature_Mapping.getMapped(from)
endif
```

- Subsetting::ownedRelationship () : Relationship [0..\*]

```
let propertyPath: OrderedSet(UML::Property) =
  Helper.getTagValueAsElementColl
    (from, 'SysML::Blocks::NestedConnectorEnd', 'propertyPath')
  ->asOrderedSet() in
if propertyPath->notEmpty() then
  OrderedSet{ConnectorEndToSubsettingFeatureMembership_Mapping.getMapped(from)}
else
  OrderedSet{}
endif
```

#### 7.7.12.2.14 Connector\_Mapping

##### Description

A UML4SysML::Connector is mapped to a SysMLv2 ConnectionUsage. The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
part def SysMLv1Block3 {
  part sysMLv1PartProperty1 : SysMLv1Block1;
  part sysMLv1PartProperty2 : SysMLv1Block2;
  connection sysMLv1Connector connect sysMLv1PartProperty1 to sysMLv1PartProperty2;
}
part def SysMLv1Block1;
part def SysMLv1Block2;
```

##### General Mappings

NamedElementMain\_Mapping  
ToConnector\_Init

##### Mapping Source

Connector

##### Mapping Target

ConnectionUsage

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ConnectionUsage::ownedRelationship () : Relationship [0..\*]

```
from.end->collect(e | ConnectorEndToMembership_Mapping.getMapped(e))->asSet()
->union(self.oclAsType(ElementMain_Mapping).ownedRelationship())
```

#### 7.7.12.2.15 ConnectorEndToFeatureCommon\_Mapping

##### Description

The mapping class is the abstract base class for UML4SysML::ConnectorEnd mapping classes.

##### General Mappings

ToFeature\_Init  
Mapping

##### Mapping Source

ConnectorEnd

##### Mapping Target

Feature

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Feature::isOrdered () : Boolean [1]  
`from.isOrdered`

#### 7.7.12.2.16 ConnectorEndToMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

##### General Mappings

ToFeatureMembership\_Init  
Mapping

##### Mapping Source

ConnectorEnd

##### Mapping Target

EndFeatureMembership

##### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- EndFeatureMembership::ownedMemberFeature () : Feature [1]  
`ConnectorEndToOwnedFeature_Mapping.getMapped(from)`

### 7.7.12.2.17 ConnectorEndToOwnedFeature\_Mapping

#### Description

The mapping class creates the SysML v2 Feature element for the UML4SysML::ConnectorEnd mapping.

#### General Mappings

ConnectorEndToFeatureCommon\_Mapping  
ElementMain\_Mapping

#### Mapping Source

ConnectorEnd

#### Mapping Target

Feature

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Feature::ownedRelationship () : Relationship [0..\*]  

```
let subsetting: KerML::Subsetting =  
    ConnectionEndToSubsetting_Mapping.getMapped(from) in  
if subsetting.oclIsUndefined() then  
    OrderedSet{MultiplicityMembership_Mapping.getMapped(from)}  
else  
    OrderedSet{MultiplicityMembership_Mapping.getMapped(from), subsetting}  
endif
```

### 7.7.12.2.18 ConnectorEndToSubsettedFeature\_Mapping

#### Description

The mapping class maps UML4SysML::ConnectorEnd that are part of a SysML::Ports&Flows::NestedConnectorEnd.

#### General Mappings

ConnectorEndToFeatureCommon\_Mapping

#### Mapping Source

ConnectorEnd

#### Mapping Target

Feature

#### Owned Mappings

(none)

#### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
let propertyPath: OrderedSet(UML::Property) =
  Helper.getTagValueAsElementColl(src, 'SysML::Blocks::NestedConnectorEnd', 'propertyPath')
->asOrderedSet() in
propertyPath->notEmpty()
```

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Feature::declaredName () : String [0..1]

'featureChain'

- Feature::ownedRelationship () : Relationship [0..\*]

```
let propertyPath: OrderedSet(UML::Property) =
  Helper.getTagValueAsElementColl
    (from, 'SysML::Blocks::NestedConnectorEnd', 'propertyPath')
->asOrderedSet() in
let chain: OrderedSet(KerML::FeatureChaining) =
  propertyPath->collect(p | PropertyToFeatureChaining_Mapping.getMapped(p))
->asOrderedSet()
->including(PropertyToFeatureChaining_Mapping.getMapped(from.role)) in
chain->union(OrderedSet{MultiplicityMembership_Mapping.getMapped(from)})
```

### 7.7.12.2.19 ConnectorEndToSubsettedFeatureMembership\_Mapping

#### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

## General Mappings

ToFeatureMembership\_Init  
Mapping

## Mapping Source

ConnectorEnd

## Mapping Target

EndFeatureMembership

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- EndFeatureMembership::ownedMemberFeature () : Feature [1]  
`ConnectorEndToSubsettedFeature_Mapping.getMapped(from)`

### 7.7.12.2.20 ConnectorType\_Mapping

#### Description

A UML4SysML::Association is mapped to a SysML v2 ConnectionDefinition.

## General Mappings

AssociationCommon\_Mapping

## Mapping Source

Association

## Mapping Target

ConnectionDefinition

## Owned Mappings

(none)

## Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```

let this: UML::Association = src.oclAsType(UML::Association) in
if this.oclIsUndefined() then
    false
else
    not src.memberEnd->exists( m | m.type.oclIsKindOf(UML::UseCase)) and
    not src.isDerived and
    not src.oclIsTypeOf(UML::AssociationClass) and
    Helper.isConnectionDef(src)
endif

```

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ConnectionDefinition::ownedRelationship () : Relationship [0..\*]

```

from.memberEnd
->collect(e | ConnectionDefEndMembership_Mapping.getMapped(e))
->asOrderedSet()

```

### 7.7.12.2.21 ConnectorTypeDerived\_Mapping

#### Description

The mapping class is a concrete mapping class of the abstract AssociationCommon\_Mapping class for mappings of derived associations. The UML4SysML::Association::isDerived property is not supported in SysML v2. To preserve the information, it is stored in a metadata annotation.

#### General Mappings

AssociationCommon\_Mapping

#### Mapping Source

Association

#### Mapping Target

ConnectionDefinition

#### Owned Mappings

(none)

#### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```

(src.memberEnd->select( m | m.type.oclIsKindOf(UML::UseCase))->isEmpty()) and
(let this: UML::Association = src.oclAsType(UML::Association) in
if this.oclIsUndefined() then
    false
else
    this.isDerived and
    not this.oclIsTypeOf(UML::AssociationClass) and

```

```

    Helper.isConnectionDef(this)
endif)

```

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ConnectionDefinition::ownedRelationship () : Relationship [0..\*]

```

self.oclAsType(AssociationCommon_Mapping).ownedRelationship()
->including(AssociationMetadataUsageMembership_Mapping.getMapped(from))

```

### 7.7.12.2.22 CrossSubsetting\_Mapping

#### Description

Creates a subsetting relationship.

#### General Mappings

UniqueMapping  
ToSubsetting\_Init

#### Mapping Source

Property

#### Mapping Target

CrossSubsetting

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- CrossSubsetting::subsettingFeature () : Feature [1]

```

NonOwnedEnd_Mapping.getMapped(from)

```

### 7.7.12.2.23 End\_Mapping

#### Description

The mapping class is the abstract base class of mapping classes for properties that are defined by association ends.

#### General Mappings

PropertyCommon\_Mapping



**Mapping Source**

Property

**Mapping Target**

Feature

**Owned Mappings**

(none)

**Applicable filters**

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
src.ocIsKindOf(UML::Property) and  
not src.ocIsType(UML::Property).association.ocIsUndefined()
```

**Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Feature::isEnd () : Boolean [1]  
  
true

**7.7.12.2.24 EndMembership\_Mapping****Description**

Creates a membership relationship for *memberElement()*.

**General Mappings**

StructuralFeatureMembership\_Mapping

**Mapping Source**

Property

**Mapping Target**

EndFeatureMembership

**Owned Mappings**

(none)

**Applicable filters**

(none)

**7.7.12.2.25 EndToSubsettedFeature\_Mapping****Description**

The mapping class creates a feature element for the UML4SysML::ConnectorEnd mapping.

### General Mappings

PropertyCommon\_Mapping

### Mapping Source

Property

### Mapping Target

Feature

### Owned Mappings

(none)

### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
let property: UML::Property = src.oclAsType(UML::Property) in
not property.association.oclIsUndefined()
and property.association.ownedEnd->excludes(property)
```

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Feature::ownedRelationship () : Relationship [0..\*]

```
let chain: OrderedSet(KerML::FeatureChaining) =
    OrderedSet{EndToSubsettedFeatureChaining_Mapping.getMapped(from)} in
chain->including(MultiplicityMembership_Mapping.getMapped(from))
```

#### 7.7.12.2.26 EndToSubsettedFeatureChaining\_Mapping

### Description

The mapping class creates a feature chaining element for the UML4SysML::ConnectorEnd mapping.

### General Mappings

ToRelationship\_Init  
Mapping

### Mapping Source

Property

### Mapping Target

FeatureChaining

### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureChaining::declaredName () : String [0..1]  
`'featureChain'`
- FeatureChaining::chainingFeature () : Feature [1]  
`from`

### 7.7.12.2.27 MultiplicityReferenceUsage\_Mapping

#### Description

Creates a reference usage.

#### General Mappings

UniqueMapping  
ToReferenceUsage\_Init

#### Mapping Source

Property

#### Mapping Target

ReferenceUsage

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReferenceUsage::ownedRelationship () : Relationship [0..\*]  
`Set{MultiplicityMembership_Factory.create(from.lower,from.upper)}`

### 7.7.12.2.28 NonOwnedEndSubsetting\_Mapping

#### Description

Creates a subsetting relationship.

### General Mappings

ToSubsetting\_Init  
Mapping

### Mapping Source

Property

### Mapping Target

Subsetting

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Subsetting::subsettingFeature () : Feature [1]

from

## 7.7.12.2.29 NonOwnedEndToSubsettingFeatureMembership\_Mapping

### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

### General Mappings

ToFeatureMembership\_Init  
Mapping

### Mapping Source

Property

### Mapping Target

FeatureMembership

### Owned Mappings

(none)

### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
src.ocIsKindOf(UML::Property) and  
not src.ocIsType(UML::Property).association.ocIsUndefined()
```

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureMembership::ownedMemberFeature () : Feature [1]  
`EndToSubsettedFeature_Mapping.getMapped(from)`

#### 7.7.12.2.30 NonOwnedEnd\_Mapping

##### Description

The mapping class maps UML4SysML::Property elements that are not owned by an association to a SysML v2 Feature element.

##### General Mappings

End\_Mapping  
UniqueMapping

##### Mapping Source

Property

##### Mapping Target

Feature

##### Owned Mappings

- nonOwnedEndTyping : NonOwnedEndFeatureTyping\_Mapping

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Feature::ownedRelationship () : Relationship [0..\*]  
`Set{MultiplicityMembership_Mapping.getMapped(from),  
nonOwnedEndTyping.to,  
NonOwnedEndSubsettingMembership_Mapping.getMapped(from),  
NonOwnedEndToSubsettedFeatureMembership_Mapping.getMapped(from) }  
->union(from.qualifier  
->collect(q | ElementFeatureMembership_Mapping.getMapped(q)) ->asSet())`
- Feature::declaredName () : String [0..1]

'nonOwnedEnd'

### 7.7.12.2.31 NonOwnedEndMembership\_Mapping

#### Description

Creates a membership relationship for *memberElement()*.

#### General Mappings

EndMembership\_Mapping

#### Mapping Source

Property

#### Mapping Target

EndFeatureMembership

#### Owned Mappings

(none)

#### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
src.ocIsKindOf(UML::Property)
and not src.ocIsType(UML::Property).association.ocIsUndefined()
and src.ocIsType(UML::Property).association.ownedEnd->excludes(src)
```

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- EndFeatureMembership::ownedMemberFeature () : Feature [1]

```
NonOwnedEnd_Mapping.getMapped(from)
```

### 7.7.12.2.32 NonOwnedEndSubsettingMembership\_Mapping

#### Description

Creates a membership relationship for *memberElement()*.

#### General Mappings

ToOwningMembership\_Init  
Mapping

#### Mapping Source

Property

#### Mapping Target

OwningMembership

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- OwningMembership::ownedMemberElement () : Element [1]  
`NonOwnedEndSubsetting_Mapping.getMapped(from)`

#### 7.7.12.2.33 NonOwnedEndFeatureTyping\_Mapping

##### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

##### General Mappings

StructuralFeatureToFeatureTyping\_Mapping  
UniqueMapping

##### Mapping Source

Property

##### Mapping Target

FeatureTyping

##### Owned Mappings

- nonOwnedEnd : NonOwnedEnd\_Mapping

##### Applicable filters

(none)

#### 7.7.12.2.34 OwnedEnd\_Mapping

##### Description

The mapping class maps UML4SysML::Property elements that are owned by an association to a SysML v2 Feature element.

##### General Mappings

End\_Mapping  
NamedElementMain\_Mapping

## Mapping Source

Property

## Mapping Target

Feature

## Owned Mappings

(none)

## Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
let p: UML::Property = src.oclAsType(UML::Property) in
not p.oclIsUndefined() and
(not p.association.oclIsUndefined()
 and p.association.ownedEnd->includes(p)) and
(not p.association.memberEnd
->select( m | (not m.type.oclIsUndefined())
 and m.type.oclIsTypeOf(UML::UseCase)) ->notEmpty())
```

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Feature::ownedRelationship () : Relationship [0..\*]

```
let qualifiers: Set(KerML::FeatureMembership) =
  from.qualifier
  ->collect(q | ElementFeatureMembership_Mapping.getMapped(q)) ->asSet() in
let typing: KerML::FeatureTyping =
  StructuralFeatureToFeatureTyping_Mapping.getMapped(from) in
let subsetting: Set(KerML::Subsetting) =
  from.subsettedProperty
  ->collect(p | PropertySubsetting_Mapping.getMapped(from, p)) ->asSet() in
let subsettingMultiplicityTyping: Set(KerML::Relationship) =
  subsetting->union(if typing.oclIsUndefined() then
    Set{MultiplicityMembership_Mapping.getMapped(from)}
  else
    Set{MultiplicityMembership_Mapping.getMapped(from), typing}
  endif) ->asSet() in
let relationships: Set(KerML::Relationship) = qualifiers->union(
  if from.defaultValue.oclIsTypeOf(UML::OpaqueExpression) then
    subsettingMultiplicityTyping
    ->including(ElementOwningMembership_Mapping.getMapped(from.defaultValue))
  else
    subsettingMultiplicityTyping
  endif) in

if from.defaultValue.oclIsUndefined() then
  relationships
else
  relationships->including(
    if from.defaultValue.oclIsTypeOf(UML::OpaqueExpression) then
```



```

        DefaultValueOpaqueExpression_Mapping.getMapped(from.defaultValue)
    else
        DefaultValue_Mapping.getMapped(from.defaultValue)
    endif)
endif

```

### 7.7.12.2.35 OwnedEndMembership\_Mapping

#### Description

Creates a membership relationship for *memberElement()*.

#### General Mappings

EndMembership\_Mapping

#### Mapping Source

Property

#### Mapping Target

EndFeatureMembership

#### Owned Mappings

(none)

#### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```

src.ocIsKindOf(UML::Property)
and not src.ocIsType(UML::Property).association.ocIsUndefined()
and src.ocIsType(UML::Property).association.ownedEnd->includes(src)

```

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- EndFeatureMembership::ownedMemberFeature () : Feature [1]

```
OwnedEnd_Mapping.getMapped(from)
```

### 7.7.12.2.36 Port\_Mapping

#### Description

A UML4SysML::Port that is typed by an interface block is mapped to a SysML v2 PortUsage.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```

port sysMLv1Port : SysMLv1InterfaceBlock;
port def SysMLv1InterfaceBlock

```

## General Mappings

PropertyCommon\_Mapping  
NamedElementMain\_Mapping

## Mapping Source

Port

## Mapping Target

PortUsage

## Owned Mappings

(none)

## Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
if src.oclIsTypeOf(UML::Port) and
not Helper.hasStereotypeApplied(src.owner,
'SysML::ConstraintBlocks::ConstraintBlock' ) then
    let p: UML::Port = src.oclAsType(UML::Port) in
        if p.type.oclIsUndefined() then
            false
        else
            true
        endif
    else
        false
    endif
endif
```

## Mapping rules

The mapping class only has inherited rules. See the mapping classes in the general mapping section for details.

### 7.7.12.2.37 PortUntyped\_Mapping

#### Description

A UML4SysML::Port that is untyped is mapped to a SysML v2 PortUsage.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
port sysMLv1Port;
```

## General Mappings

PropertyUntyped\_Mapping

## Mapping Source

Port

#### Mapping Target

PortUsage

#### Owned Mappings

(none)

#### Applicable filters

(none)

### 7.7.12.2.38 PropertyToFeatureChaining\_Mapping

#### Description

The mapping class creates the SysML v2 FeatureChaining for the UML4SysML::Property mapping.

#### General Mappings

ToRelationship\_Init  
Mapping

#### Mapping Source

Property

#### Mapping Target

FeatureChaining

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureChaining::chainingFeature () : Feature [1]  
`ElementMain_Mapping.getMapped(from)`

### 7.7.12.2.39 QualifierMembership\_Mapping

#### Description

Creates a membership relationship for *memberElement()*.

#### General Mappings

StructuralFeatureMembership\_Mapping

#### Mapping Source

StructuralFeature

#### Mapping Target

FeatureMembership

#### Owned Mappings

(none)

#### Applicable filters

(none)

### 7.7.13 UseCases

#### 7.7.13.1 Overview

Table 17. List of all mappings

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
Actor	PartDefinition
Extend	not mapped; see next section
ExtensionPoint	not mapped; see next section
Include	IncludeUseCaseUsage
UseCase	UseCaseDefinition

#### 7.7.13.2 UML4SysML::UseCases elements not mapped

Table 18. List of SysML v1 elements not mapped of this section

SysML v1 Concept	Rationale
Extend	The semantics of the UML4SysML::Extend relationship is not supported by SysML v2.
ExtensionPoint	The semantics of the UML4SysML::Extend relationship is not supported by SysML v2 Therefore, UML4SysML::ExtensionPoint is also not covered by the transformation.

#### 7.7.13.3 Mapping Specifications

##### 7.7.13.3.1 Actor\_Mapping

###### Description

A UML4SysML::Actor is mapped to a SysML v2 PartDefinition. The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
part def SysMLv1Actor;
```

## General Mappings

ElementMain\_Mapping  
BehavioredClassifier\_Mapping

## Mapping Source

Actor

## Mapping Target

PartDefinition

## Owned Mappings

(none)

## Applicable filters

(none)

### 7.7.13.3.2 Include\_Mapping

#### Description

A UML4SysML::Include is mapped to a SysML v2 IncludeUseCaseUsage. The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
use case def SysMLv1UseCase1 {  
    include use case : SysMLv1UseCase2;  
}  
use case def SysMLv1UseCase2;
```

## General Mappings

ToOccurrenceUsage\_Init  
NamedElementMain\_Mapping

## Mapping Source

Include

## Mapping Target

IncludeUseCaseUsage

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- IncludeUseCaseUsage::ownedRelationship () : Relationship [0..\*]  

```
Set{IncludeFeatureTyping_Mapping.getMapped(from),
ReturnParameterFeatureMembership_Factory.create(),
EmptySubjectMembership_Factory.create()}
->union(self.oclAsType(ElementMain_Mapping).ownedRelationship())
```

### 7.7.13.3.3 IncludeFeatureTyping\_Mapping

#### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

#### General Mappings

ToFeatureTyping\_Init  
Mapping

#### Mapping Source

Include

#### Mapping Target

FeatureTyping

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureTyping::type () : Type [1]  

```
from.addition
```

### 7.7.13.3.4 UseCase\_Mapping

#### Description

A UML4SysML::UseCase is mapped to a SysML v2 UseCaseDefinition. The expected SysML v2 textual syntax of a mapped UML4SysML::UseCase with a defined subject is as follows.

```
use case def SysMLv1UseCase {
  subject subject_SysMLv1Block : SysMLv1Block;
}
part def SysMLv1Block;
```

Currently, only one use case subject is supported by the mapping class. Since the UML4SysML::Extend relationship is not considered by the SysML v1 to SysML v2 transformation, the extension points of a use case are also not mapped.

## General Mappings

BehavioredClassifier\_Mapping  
NamedElementMain\_Mapping

## Mapping Source

UseCase

## Mapping Target

UseCaseDefinition

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- UseCaseDefinition::ownedRelationship () : Relationship [0..\*]

```
let properties : Set(UML::Element) =
  from.ownedElement->select(e | e.ocIsKindOf(UML::Property) and
    e.ocIsType(UML::Property).association.ocIsUndefined()) in
let actors : Set(UML::Property) =
  UML::Association.allInstances()
  ->collect(m | m.memberEnd)
  ->flatten()
  ->select(m | m.type = from)->collect(a | a.owningAssociation)
  ->collect(p | p.memberEnd->select(m | not (m.type = from)))->flatten() in
let extensionPoints : Sequence(UML::Element) =
  from.ownedElement->select(e | e.ocIsKindOf(UML::ExtensionPoint)) in
let extend : Sequence(UML::Element) =
  from.ownedElement->select(e | e.ocIsKindOf(UML::Extend)) in
let include : Sequence(UML::Element) =
  from.ownedElement->select(e | e.ocIsKindOf(UML::Include)) in
let elements : Set(UML::Element) =
  (((from.ownedElement-properties) - extensionPoints) - extend) - include) in
let relationships : Sequence(KerML::Relationship) =
  elements->collect(e | ElementOwningMembership_Mapping.getMapped(e))
  ->union(properties->collect(e | PropertyMembership_Mapping.getMapped(e)))
  ->including(UseCaseSubjectMembership_Mapping.getMapped(from))
  ->including(UseCaseObjectiveMembership_Mapping.getMapped(from))
  ->including(CommonReturnParameterReferenceUsageMembership_Mapping.getMapped(from))
  ->union(actors->collect(e | UseCaseActorMembership_Mapping.getMapped(e))) in
if from.classifierBehavior.ocIsUndefined() then
  relationships
else
  relationships
```

```

->including (BehavioredClassifierFeatureMembership_Mapping.getMapped(from))
endif

```

### 7.7.13.3.5 UseCaseActor\_Mapping

#### Description

The mapping class creates the PartUsage representing an actor of the use case.

#### General Mappings

ToPartUsage\_Init  
Mapping

#### Mapping Source

Property

#### Mapping Target

PartUsage

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- PartUsage::declaredName () : String [0..1]  
`from.name`
- PartUsage::ownedRelationship () : Relationship [0..\*]  
`Set{UseCaseActorFeatureTyping_Mapping.getMapped(from) }`

### 7.7.13.3.6 UseCaseActorFeatureTyping\_Mapping

#### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

#### General Mappings

ToFeatureTyping\_Init  
Mapping

#### Mapping Source

Property



### Mapping Target

FeatureTyping

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureTyping::type () : Type [1]  
`from.type`

### 7.7.13.3.7 UseCaseActorMembership\_Mapping

#### Description

Creates a membership relationship for *memberElement()*.

#### General Mappings

ToActorMembership\_Init  
Mapping

#### Mapping Source

Property

#### Mapping Target

ActorMembership

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ActorMembership::ownedMemberParameter () : Feature [1]  
`UseCaseActor_Mapping.getMapped(from)`

#### 7.7.13.3.8 UseCaseEmptySubjectReferenceUsage\_Mapping

##### Description

The mapping class creates an "empty" ReferenceUsage for the subject, if the subject is not given at the SysML v1 UseCase element.

##### General Mappings

ToReferenceUsage\_Init  
Mapping

##### Mapping Source

UseCase

##### Mapping Target

ReferenceUsage

##### Owned Mappings

(none)

##### Applicable filters

(none)

#### 7.7.13.3.9 UseCaseObjectiveMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

##### General Mappings

ToObjectiveMembership\_Init  
Mapping

##### Mapping Source

UseCase

##### Mapping Target

ObjectiveMembership

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ObjectiveMembership::ownedMemberFeature () : Feature [1]`  
`UseCaseObjectiveRequirementUsage_Mapping.getMapped(from)`

#### 7.7.13.3.10 UseCaseObjectiveRequirementUsage\_Mapping

##### Description

The mapping class creates the RequirementUsage element for the use case objective. The element is not set by an element from the SysML v1 UseCase.

##### General Mappings

ToRequirementUsage\_Init  
Mapping

##### Mapping Source

UseCase

##### Mapping Target

RequirementUsage

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `RequirementUsage::ownedRelationship () : Relationship [0..*]`  
`Set { UseCaseObjectiveSubjectMembership_Mapping.getMapped(from) ,`  
`CommonReturnParameterReferenceUsageMembership_Mapping.getMapped(from) }`

#### 7.7.13.3.11 UseCaseObjectiveSubjectMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

##### General Mappings

ToSubjectMembership\_Init  
Mapping

##### Mapping Source

UseCase

### Mapping Target

SubjectMembership

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- SubjectMembership::ownedMemberParameter () : Feature [1]  
`UseCaseEmptySubjectReferenceUsage_Mapping.getMapped (from)`

## 7.7.13.3.12 UseCaseSubjectFeatureTyping\_Mapping

### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

### General Mappings

ToFeatureTyping\_Init  
Mapping

### Mapping Source

UseCase

### Mapping Target

FeatureTyping

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureTyping::type () : Type [1]

```
if from.subject->size() > 0 then from.subject->get(0) else invalid endif
```

### 7.7.13.3.13 UseCaseSubjectMembership\_Mapping

#### Description

Creates a membership relationship for *memberElement()*.

#### General Mappings

ToSubjectMembership\_Init  
Mapping

#### Mapping Source

UseCase

#### Mapping Target

SubjectMembership

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- SubjectMembership::ownedMemberParameter () : Feature [1]

```
if from.subject->size() > 0 then
    UseCaseSubjectReferenceUsage_Mapping.getMapped(from)
else
    UseCaseEmptySubjectReferenceUsage_Mapping.getMapped(from)
endif
```

### 7.7.13.3.14 UseCaseSubjectReferenceUsage\_Mapping

#### Description

The mapping class creates the ReferenceUsage element for the subject.

#### General Mappings

UseCaseEmptySubjectReferenceUsage\_Mapping

#### Mapping Source

UseCase

#### Mapping Target

ReferenceUsage

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReferenceUsage::ownedRelationship () : Relationship [0..\*]  
`Set{UseCaseSubjectFeatureTyping_Mapping.getMapped(from) }`
- ReferenceUsage::declaredName () : String [0..1]  
`'subject_' + from.subject->get(0).name`

## 7.7.14 Values

### 7.7.14.1 Overview

**Table 19. List of all mappings**

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
Duration	Expression
DurationConstraint	ConstraintDefinition
DurationInterval	Expression
DurationObservation	not mapped; see next section
Expression	Expression OperatorExpression
Interval	Expression
IntervalConstraint	ConstraintDefinition
LiteralBoolean	LiteralBoolean
LiteralInteger	LiteralInteger
LiteralNull	NullExpression
LiteralReal	LiteralRational
LiteralString	LiteralString
LiteralUnlimitedNatural	LiteralInfinity
OpaqueExpression	CalculationUsage
StringExpression	Expression OperatorExpression
TimeConstraint	ConstraintDefinition

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
TimeExpression	Expression
TimeInterval	Expression
TimeObservation	not mapped; see next section

### 7.7.14.2 UML4SysML::Values elements not mapped

**Table 20.** List of SysML v1 elements not mapped of this section

SysML v1 Concept	Rationale
Duration	Mapping is not specified yet.
DurationConstraint	Mapping is not specified yet.
DurationInterval	Mapping is not specified yet.
DurationObservation	Mapping is not specified yet.
Interval	Mapping is not specified yet.
IntervalConstraint	Mapping is not specified yet.
StringExpression	Mapping is not specified yet.
TimeConstraint	Mapping is not specified yet.
TimeInterval	Mapping is not specified yet.
TimeObservation	Mapping is not specified yet.

### 7.7.14.3 Mapping Specifications

#### 7.7.14.3.1 EqualOperatorExpressionFeature\_Mapping

##### Description

The mapping class creates the feature element for the equal operator.

##### General Mappings

ToFeature\_Init  
Mapping

##### Mapping Source

TypedElement

##### Mapping Target

Feature

##### Owned Mappings

(none)

##### Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Feature::ownedRelationship () : Relationship [0..*]`

```
Set { EqualOperatorExpressionFeatureValue_Mapping.getMapped (from) }
```

### 7.7.14.3.2 EqualOperatorExpressionFeatureValue\_Mapping

#### Description

Creates a feature value relationship.

#### General Mappings

ToFeatureValue\_Init  
Mapping

#### Mapping Source

TypedElement

#### Mapping Target

FeatureValue

#### Owned Mappings

(none)

#### Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureValue::value () : Expression [1]`

```
CommonFeatureReferenceExpression_Mapping.getMapped (from)
```

### 7.7.14.3.3 EqualOperatorExpressionOperandParameterMembership\_Mapping

#### Description

Creates a membership relationship for *memberElement()*.

#### General Mappings

ToParameterMembership\_Init  
Mapping

#### Mapping Source



TypedElement

### Mapping Target

ParameterMembership

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ParameterMembership::visibility () : VisibilityKind [1]  
`KerML::VisibilityKind::private`
- ParameterMembership::ownedMemberParameter () : Feature [1]  
`EqualOperatorExpressionFeature_Mapping.getMapped(from)`

## 7.7.14.3.4 Expression\_Mapping

### Description

A UML4SysML::Expression element is mapped to a SysML v2 OperatorExpression element.

### General Mappings

ToExpression\_Init  
NamedElementMain\_Mapping

### Mapping Source

Expression

### Mapping Target

OperatorExpression

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `OperatorExpression::operator () : String [1]`

```
from.symbol
```

#### 7.7.14.3.5 ExpressionElse\_Mapping

##### Description

A `UML4SysML::Expression` element with operator "else" is mapped to a SysML v2 `TextualRepresentation` element with language set to "SysMLv1" and body set to "else".

##### General Mappings

`Expression_Mapping`

##### Mapping Source

`Expression`

##### Mapping Target

`OperatorExpression`

##### Owned Mappings

(none)

##### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
src.symbol = 'else'
```

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `OperatorExpression::ownedRelationship () : Relationship [0..*]`

```
self.oclAsType(ElementMain_Mapping).ownedRelationship()  
->including(ExpressionElseMembership_Mapping.getMapped(from))
```

#### 7.7.14.3.6 ExpressionElseMembership\_Mapping

##### Description

Creates the membership relationship for the textual representation for the else guard condition specification.

##### General Mappings

ToOwningMembership\_Init  
Mapping

**Mapping Source**

Expression

**Mapping Target**

OwningMembership

**Owned Mappings**

(none)

**Applicable filters**

(none)

**Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- OwningMembership::ownedMemberElement () : Element [1]  
`ExpressionElseSpecification_Mapping.getMapped(from)`

**7.7.14.3.7 ExpressionElseSpecification\_Mapping**

**Description**

Creates the textual representation for the else guard condition specification.

**General Mappings**

ToTextualRepresentation\_Init  
Mapping

**Mapping Source**

Expression

**Mapping Target**

TextualRepresentation

**Owned Mappings**

(none)

**Applicable filters**

(none)

**Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `TextualRepresentation::body () : String [1]`  
`'else'`
- `TextualRepresentation::language () : String [1]`  
`'SysMLv1'`

#### 7.7.14.3.8 LiteralBoolean\_Mapping

##### Description

The mapping class maps UML4SysML::LiteralBoolean to SysML v2 LiteralBoolean.

##### General Mappings

LiteralSpecificationCommon\_Mapping

##### Mapping Source

LiteralBoolean

##### Mapping Target

LiteralBoolean

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `LiteralBoolean::value () : Boolean [1]`  
`from.value`

#### 7.7.14.3.9 LiteralInteger\_Mapping

##### Description

The mapping class maps UML4SysML::LiteralInteger to SysML v2 LiteralInteger.

##### General Mappings

LiteralSpecificationCommon\_Mapping

##### Mapping Source

LiteralInteger

#### **Mapping Target**

LiteralInteger

#### **Owned Mappings**

(none)

#### **Applicable filters**

(none)

#### **Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- LiteralInteger::value () : Integer [1]  
from.value

### **7.7.14.3.10 LiteralNull\_Mapping**

#### **Description**

The mapping class maps UML4SysML::LiteralNull to SysML v2 NullExpression.

#### **General Mappings**

LiteralSpecificationCommon\_Mapping

#### **Mapping Source**

LiteralNull

#### **Mapping Target**

NullExpression

#### **Owned Mappings**

(none)

#### **Applicable filters**

(none)

### **7.7.14.3.11 LiteralReal\_Mapping**

#### **Description**

The mapping class maps UML4SysML::LiteralReal to SysML v2 LiteralRational.

#### **General Mappings**

LiteralSpecificationCommon\_Mapping

#### **Mapping Source**

LiteralReal

#### **Mapping Target**

LiteralRational

#### **Owned Mappings**

(none)

#### **Applicable filters**

(none)

#### **Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- LiteralRational::value () : Real [1]

`from.value`

### **7.7.14.3.12 LiteralSpecificationCommon\_Mapping**

#### **Description**

The mapping class is the abstract base class for all concrete UML4SysML::LiteralSpecification mappings.

#### **General Mappings**

ValueSpecification\_Mapping

#### **Mapping Source**

LiteralSpecification

#### **Mapping Target**

LiteralExpression

#### **Owned Mappings**

(none)

#### **Applicable filters**

(none)

#### **Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `LiteralExpression::ownedRelationship () : Relationship [0..*]`

```
let ownerships: Set(SYSML2::Relationship) =
    self.oclAsType(ElementMain_Mapping).ownedRelationship()
    ->including(CommonReturnParameterFeatureMembership_Mapping.getMapped(from)) in
if from.type.oclIsUndefined() then
    ownerships
else
    ownerships->including(LiteralSpecificationTyping_Mapping.getMapped(from))
endif
```

#### 7.7.14.3.13 LiteralSpecificationFeatureTyping\_Mapping

##### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

##### General Mappings

TypedElementFeatureTyping\_Mapping

##### Mapping Source

LiteralSpecification

##### Mapping Target

FeatureTyping

##### Owned Mappings

(none)

##### Applicable filters

(none)

#### 7.7.14.3.14 LiteralString\_Mapping

##### Description

The mapping class maps UML4SysML::LiteralString to the SysML v2 LiteralString.

##### General Mappings

LiteralSpecificationCommon\_Mapping

##### Mapping Source

LiteralString

##### Mapping Target

LiteralString

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `LiteralString::value () : String [1]`  
`if from.value.oclIsUndefined() then '' else from.value endif`

### 7.7.14.3.15 LiteralUnlimitedUnbounded\_Mapping

#### Description

The mapping class maps UML4SysML::LiteralUnlimited to SysML v2 LiteralInfinity if it is the unlimited value.

#### General Mappings

LiteralUnlimitedInteger\_Mapping

#### Mapping Source

LiteralUnlimitedNatural

#### Mapping Target

LiteralInfinity

## Owned Mappings

(none)

## Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
(from.value = -1)
```

## Mapping rules

The mapping class only has inherited rules. See the mapping classes in the general mapping section for details.

### 7.7.14.3.16 LiteralUnlimitedInteger\_Mapping

#### Description

The mapping class maps UML4SysML::LiteralUnlimited to SysML v2 LiteralInteger if it is not the unlimited value.

#### General Mappings



LiteralSpecificationCommon\_Mapping

#### **Mapping Source**

LiteralUnlimitedNatural

#### **Mapping Target**

LiteralInteger

#### **Owned Mappings**

(none)

#### **Applicable filters**

(none)

#### **Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- LiteralInteger::value () : Integer [1]

`from.value`

### **7.7.14.3.17 OpaqueExpressionAsValue\_Mapping**

#### **Description**

The mapping class maps a UML4SysML::OpaqueExpression if it is used as a value to a SysML v2 FeatureChainExpression.

#### **General Mappings**

ToExpression\_Init  
Mapping

#### **Mapping Source**

OpaqueExpression

#### **Mapping Target**

FeatureChainExpression

#### **Owned Mappings**

(none)

#### **Applicable filters**

(none)

#### **Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureChainExpression::ownedRelationship () : Relationship [0..*]`

```
Set{OpaqueExpressionParameterMembership_Mapping.getMapped(from),
CommonReturnParameterFeatureMembership_Mapping.getMapped(from)}
```

### 7.7.14.3.18 OpaqueExpression\_Mapping

#### Description

A `UML4SysML::OpaqueExpression` element is mapped to a `SysMLv2 CalculationUsage` element.. The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
calc sysMLv1OpaqueExpression {
    return result : ScalarValues::Integer;
    language "Built-in Math"
    /*
     * result = 42 + 23;
     */
}
```

#### General Mappings

`CommonAction_Mapping`  
`ValueSpecification_Mapping`

#### Mapping Source

`OpaqueExpression`

#### Mapping Target

`CalculationUsage`

#### Owned Mappings

(none)

#### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
if src.owner.ocIsKindOf(UML::TimeExpression) then
    not src.owner.owner.ocIsKindOf(UML::TimeEvent)
else
    true
endif
```

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `CalculationUsage::ownedRelationship () : Relationship [0..*]`

```

Set{OpaqueExpressionMembership_Mapping.getMapped(from),
OpaqueExpressionReferenceUsageReturnParameterMembership_Mapping.getMapped(from)}
->union(self.oclAsType(ElementMain_Mapping).ownedRelationship())

```

#### 7.7.14.3.19 OpaqueExpressionFeature\_Mapping

##### Description

The mapping class creates the feature of the FeatureChainExpression.

##### General Mappings

ToFeature\_Init  
Mapping

##### Mapping Source

OpaqueExpression

##### Mapping Target

Feature

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Feature::ownedRelationship () : Relationship [0..\*]

```

Set{OpaqueExpressionFeatureValue_Mapping.getMapped(from),
OpaqueExpressionFeatureFeatureMembership_Mapping.getMapped(from)}

```

#### 7.7.14.3.20 OpaqueExpressionFeatureFeature\_Mapping

##### Description

The mapping class creates the Feature of the FeatureReferenceExpression.

##### General Mappings

ToFeature\_Init  
Mapping

##### Mapping Source

OpaqueExpression

##### Mapping Target

Feature

#### **Owned Mappings**

(none)

#### **Applicable filters**

(none)

### **7.7.14.3.21 OpaqueExpressionFeatureFeatureMembership\_Mapping**

#### **Description**

Creates a feature membership relationship for *ownedMemberFeature()*.

#### **General Mappings**

ToFeatureMembership\_Init  
Mapping

#### **Mapping Source**

OpaqueExpression

#### **Mapping Target**

FeatureMembership

#### **Owned Mappings**

(none)

#### **Applicable filters**

(none)

#### **Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureMembership::ownedMemberFeature () : Feature [1]  
`OpaqueExpressionFeatureFeature_Mapping.getMapped(from)`

### **7.7.14.3.22 OpaqueExpressionFeatureValue\_Mapping**

#### **Description**

Creates a feature value relationship.

#### **General Mappings**

ToFeatureValue\_Init  
Mapping

**Mapping Source**

OpaqueExpression

**Mapping Target**

FeatureValue

**Owned Mappings**

(none)

**Applicable filters**

(none)

**Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureValue::value () : Expression [1]

`OpaqueExpressionFeatureValueExpression_Mapping.getMapped (from)`

**7.7.14.3.23 OpaqueExpressionFeatureValueExpression\_Mapping****Description**

The mapping class creates the value of the FeatureChainExpression that is a FeatureReferenceExpression.

**General Mappings**

ToExpression\_Init  
Mapping

**Mapping Source**

OpaqueExpression

**Mapping Target**

FeatureReferenceExpression

**Owned Mappings**

(none)

**Applicable filters**

(none)

**Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureReferenceExpression::ownedRelationship () : Relationship [0..*]`

```
Set{OpaqueExpressionFeatureValueExpressionMembership_Mapping.getMapped(from),  
ReturnParameterFeatureMembership_Factory.create() }
```

#### 7.7.14.3.24 OpaqueExpressionFeatureValueExpressionMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

##### General Mappings

ToMembership\_Init  
Mapping

##### Mapping Source

OpaqueExpression

##### Mapping Target

Membership

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Membership::memberElement () : Element [1]`

*from*

#### 7.7.14.3.25 OpaqueExpressionMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

##### General Mappings

ToOwningMembership\_Init  
Mapping

##### Mapping Source

OpaqueExpression

### Mapping Target

OwningMembership

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- OwningMembership::ownedMemberElement () : Element [1]  
`OpaqueExpressionSpecification_Mapping.getMapped(from)`

### 7.7.14.3.26 OpaqueExpressionParameterMembership\_Mapping

#### Description

Creates a membership relationship for *memberElement()*.

#### General Mappings

ToParameterMembership\_Init  
Mapping

#### Mapping Source

OpaqueExpression

#### Mapping Target

ParameterMembership

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ParameterMembership::ownedMemberParameter () : Feature [1]  
`OpaqueExpressionFeature_Mapping.getMapped(from)`

#### 7.7.14.3.27 OpaqueExpressionReferenceUsageReturnParameterMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

##### General Mappings

ToReturnParameterMembership\_Init  
Mapping

##### Mapping Source

OpaqueExpression

##### Mapping Target

ReturnParameterMembership

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReturnParameterMembership::ownedMemberParameter () : Feature [1]  

```
if from.type.oclIsUndefined() then
    OpaqueExpressionReferenceUsageUntyped_Mapping.getMapped(from)
else
    OpaqueExpressionReferenceUsage_Mapping.getMapped(from)
endif
```

#### 7.7.14.3.28 OpaqueExpressionReferenceUsage\_Mapping

##### Description

The mapping class creates the return parameter reference usage of the calculation usage.

##### General Mappings

ToReferenceUsage\_Init  
Mapping

##### Mapping Source

OpaqueExpression

##### Mapping Target

ReferenceUsage



## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceUsage::ownedRelationship () : Relationship [0..*]`  
`Set{OpaqueExpressionReferenceUsageFeatureTyping_Mapping.getMapped(from) }`
- `ReferenceUsage::direction () : FeatureDirectionKind [0..1]`  
`KerML::FeatureDirectionKind::_'out'`

### 7.7.14.3.29 OpaqueExpressionReferenceUsageFeatureTyping\_Mapping

#### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

#### General Mappings

TypedElementFeatureTyping\_Mapping

#### Mapping Source

OpaqueExpression

#### Mapping Target

FeatureTyping

#### Owned Mappings

(none)

#### Applicable filters

(none)

### 7.7.14.3.30 OpaqueExpressionReferenceUsageUntyped\_Mapping

#### Description

The mapping class creates the return parameter reference usage of the calculation usage, if the UML4SysML::OpaqueExpression is untyped.

#### General Mappings

ToReferenceUsage\_Init  
Mapping

### Mapping Source

OpaqueExpression

### Mapping Target

ReferenceUsage

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReferenceUsage::direction () : FeatureDirectionKind [0..1]

`KerML::FeatureDirectionKind::_out'`

## 7.7.14.3.31 OpaqueExpressionSpecification\_Mapping

### Description

The mapping class creates the specification of the calculation usage based on the language and body of the UML4SysML::OpaqueExpression.

### General Mappings

ToTextualRepresentation\_Init  
Mapping

### Mapping Source

OpaqueExpression

### Mapping Target

TextualRepresentation

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `TextualRepresentation::language () : String [1]`  
`if from.language->size() = 0 then invalid else from.language.get(0) endif`
- `TextualRepresentation::body () : String [1]`  
`if from.body->size() = 0 then invalid else from.body.get(0) endif`

### 7.7.14.3.32 TimeExpression\_Mapping

#### Description

A `UML4SysML::TimeExpression` is mapped to a SysML v2 Expression. The details of the mapping are not specified yet.

#### General Mappings

ValueSpecification\_Mapping

#### Mapping Source

TimeExpression

#### Mapping Target

Expression

#### Owned Mappings

(none)

#### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
not src.owner.ocIsKindOf(UML::TimeEvent)
```

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Expression::ownedRelationship () : Relationship [0..*]`  

```
let ownedComments : Set(KerML::Relationship) =
  from.ownedComment->reject(c | c.annotatedElement->includes(from))
  ->collect(c| CommentOwnership_Mapping.getMapped(c))->asSet() in
let expression : Set(KerML::Relationship) = if from.expr.ocIsUndefined() then
  Set{}
else
  Set{ElementOwningMembership_Mapping.getMapped(from.expr)}
endif in
(if from.type.ocIsUndefined() then
  Set{CommonReturnParameterFeatureMembership_Mapping.getMapped(from)}
else
```

```

        Set{LiteralSpecificationTyping_Mapping.getMapped(from),
          CommonReturnParameterFeatureMembership_Mapping.getMapped(from)}
      endif)
    ->union(ownedComments)
    ->union(expression)

```

### 7.7.14.3.33 ValueSpecification\_Mapping

#### Description

The mapping class is the abstract base class of all mapping classes for special value specifications.

#### General Mappings

NamedElementMain\_Mapping  
ToExpression\_Init

#### Mapping Source

ValueSpecification

#### Mapping Target

Expression

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Expression::ownedRelationship () : Relationship [0..\*]  

```

      (if from.type.oclIsUndefined() then
        Set{CommonReturnParameterFeatureMembership_Mapping.getMapped(from)}
      else
        Set{LiteralSpecificationTyping_Mapping.getMapped(from),
          CommonReturnParameterFeatureMembership_Mapping.getMapped(from)}
      endif) ->union(self.oclAsType(ElementMain_Mapping).ownedRelationship())

```

## 7.8 Mappings from SysML v1.7 stereotypes

### 7.8.1 Overview

The following subclauses of Mappings from SysML v1.7 stereotypes are organized according to the main packages of SysML v1.

## 7.8.2 Activities

### 7.8.2.1 Overview

**Table 21. List of all mappings**

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
Continuous	MetadataUsage
ControlOperator	
Discrete	MetadataUsage
NoBuffer	
Optional	
Overwrite	
Probability	MetadataUsage
Rate	MetadataUsage

### 7.8.2.2 SysML::Activities elements not mapped

**Table 22. List of SysML v1 elements not mapped of this section**

SysML v1 Concept	Rationale
ControlOperator	The concept that an action can control other actions is not supported by SysML v2.
NoBuffer	Mapping is not specified yet.
Optional	The stereotype states that the lower multiplicity of the parameter is 0. Since the multiplicity of the parameter is transformed, the additional statement that the parameter is optional is redundant. Therefore, the stereotype is not considered in the transformation.
Overwrite	Mapping is not specified yet.

### 7.8.2.3 Mapping Specifications

#### 7.8.2.3.1 ProbabilityMetadataUsage\_Mapping

##### Description

A SysML::Activities::Probability is mapped to a SysML v2 MetadataUsage owned by the appropriate target element of the UML4SysML::ActivityEdge or UML4SysML::ParameterSet.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
action def SysMLv1Activity {
    action sysMLv1Action1;
    succession sysMLv1ControlFlow1 first sysMLv1Action1 then sysMLv1Action2 {
        @SysMLv1Library::ProbabilityData {probability = 0.42;}
    }
    action sysMLv1Action2;
}
```

##### General Mappings

ToMetadataUsage\_Init  
Mapping

#### Mapping Source

Element

#### Mapping Target

MetadataUsage

#### Owned Mappings

(none)

#### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
Helper.hasStereotypeApplied(src, 'SysML::Activities::Probability')
```

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- MetadataUsage::ownedRelationship () : Relationship [0..\*]

```
Set{ProbabilityMetadataUsageFeatureTyping_Mapping.getMapped(from),  
ProbabilityMetadataUsageFeatureMembership_Mapping.getMapped(from)}
```

#### 7.8.2.3.2 ProbabilityMetadataUsageFeatureMembership\_Mapping

##### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

##### General Mappings

ToFeatureMembership\_Init  
Mapping

#### Mapping Source

Element

#### Mapping Target

FeatureMembership

#### Owned Mappings

(none)

#### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
Helper.hasStereotypeApplied(src, 'SysML::Activities::Probability')
```

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureMembership::ownedMemberFeature () : Feature [1]

```
ProbabilityMetadataUsageReferenceUsage_Mapping.getMapped(from)
```

#### 7.8.2.3.3 ProbabilityMetadataUsageFeatureTyping\_Mapping

##### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

##### General Mappings

ToFeatureTyping\_Init  
Mapping

##### Mapping Source

Element

##### Mapping Target

FeatureTyping

##### Owned Mappings

(none)

##### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
Helper.hasStereotypeApplied(src, 'SysML::Activities::Probability')
```

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureTyping::type () : Type [1]

```
SysML2::MetadataDefinition.allInstances()  
->any(m | m.qualifiedName = 'SysMLv1Library::ProbabilityData')
```

#### 7.8.2.3.4 ProbabilityMetadataUsageReferenceUsage\_Mapping

##### Description

Creates a reference usage.

### General Mappings

ToReferenceUsage\_Init  
Mapping

### Mapping Source

Element

### Mapping Target

ReferenceUsage

### Owned Mappings

(none)

### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
Helper.hasStereotypeApplied(src, 'SysML::Activities::Probability')
```

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReferenceUsage::ownedRelationship () : Relationship [0..\*]

```
Set{ProbabilityMetadataUsageReferenceUsageRedefinition_Mapping.getMapped(from) ,  
ProbabilityMetadataUsageReferenceUsageFeatureValue_Mapping.getMapped(from) }
```

#### 7.8.2.3.5 ProbabilityMetadataUsageReferenceUsageFeatureValue\_Mapping

##### Description

Creates a feature value relationship.

### General Mappings

ToFeatureValue\_Init  
Mapping

### Mapping Source

Element

### Mapping Target

FeatureValue

### Owned Mappings



(none)

#### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
Helper.hasStereotypeApplied(src, 'SysML::Activities::Probability')
```

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureValue::value () : Expression [1]

```
let probability : OclAny =  
  Helper.getTagValue(from, 'SysML::Activities::Probability', 'probability') in  
  LiteralRational_Factory.create(probability)
```

### 7.8.2.3.6 ProbabilityMetadataUsageReferenceUsageRedefinition\_Mapping

#### Description

Creates a redefinition relationship for the *redefiningFeature()* and the *redefinedFeature()*.

#### General Mappings

ToRedefinition\_Init  
Mapping

#### Mapping Source

Element

#### Mapping Target

Redefinition

#### Owned Mappings

(none)

#### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
Helper.hasStereotypeApplied(src, 'SysML::Activities::Probability')
```

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Redefinition::redefinedFeature () : Feature [1]

```
SysML2::AttributeUsage.allInstances()
->any(m | m.qualifiedName = 'SysMLv1Library::ProbabilityData::probability')
```

### 7.8.2.3.7 ProbabilityOwningMembership\_Mapping

#### Description

Creates a owning membership relationship for *ownedMemberElement()*.

#### General Mappings

ToOwningMembership\_Init  
Mapping

#### Mapping Source

Element

#### Mapping Target

OwningMembership

#### Owned Mappings

(none)

#### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
Helper.hasStereotypeApplied(src, 'SysML::Activities::Probability')
```

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- OwningMembership::ownedMemberElement () : Element [1]  
`ProbabilityMetadataUsage_Mapping.getMapped(from)`

### 7.8.2.3.8 RateMetadataUsage\_Mapping

#### Description

A SysML::Activities::Rate and the specializations SysML::Activities::Discrete and SysML::Activities::Continuous are mapped to a SysML v2 MetadataUsage owned by the appropriate target element of the UML4SysML::ActivityEdge or UML4SysML::Parameter.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
succession flow sysMLv1ObjectFlow of SysMLv1Block
  from sysMLv1Action1.outputValue to sysMLv1Action1.inputValue {
    @SysMLv1Library::RateData {isDiscrete = true;}
  }
```

The mapping of the rate instance value is not supported yet.

## General Mappings

ToMetadataUsage\_Init  
Mapping

## Mapping Source

Element

## Mapping Target

MetadataUsage

## Owned Mappings

(none)

## Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
Helper.hasStereotypeApplied(src, 'SysML::Activities::Rate')  
or Helper.hasStereotypeApplied(src, 'SysML::Activities::Continuous')  
or Helper.hasStereotypeApplied(src, 'SysML::Activities::Discrete')
```

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- MetadataUsage::ownedRelationship () : Relationship [0..\*]

```
let relationships : Set(KerML::Relationship) =  
    Set{RateMetadataUsageFeatureTyping_Mapping.getMapped(from)} in  
if Helper.hasStereotypeApplied(from, 'SysML::Activities::Discrete') then  
    relationships  
->including(  
        RateMetadataUsageDiscreteFeatureMembership_Mapping.getMapped(from))  
else if Helper.hasStereotypeApplied(from, 'SysML::Activities::Continuous') then  
    relationships  
->including(  
        RateMetadataUsageContinuousFeatureMembership_Mapping.getMapped(from))  
else  
    relationships  
endif  
endif
```

### 7.8.2.3.9 RateMetadataUsageContinuousFeatureMembership\_Mapping

#### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

#### General Mappings

ToFeatureMembership\_Init  
Mapping

#### Mapping Source

Element

#### Mapping Target

FeatureMembership

#### Owned Mappings

(none)

#### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
Helper.hasStereotypeApplied(src, 'SysML::Activities::Continuous')
```

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureMembership::ownedMemberFeature () : Feature [1]

```
RateMetadataUsageContinuousReferenceUsage_Mapping.getMapped(from)
```

### 7.8.2.3.10 RateMetadataUsageFeatureValue\_Mapping

#### Description

Creates a feature value relationship.

#### General Mappings

ToFeatureValue\_Init  
Mapping

#### Mapping Source

Element

#### Mapping Target

FeatureValue

#### Owned Mappings

(none)

#### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
Helper.hasStereotypeApplied(src, 'SysML::Activities::Rate')
or Helper.hasStereotypeApplied(src, 'SysML::Activities::Continuous')
or Helper.hasStereotypeApplied(src, 'SysML::Activities::Discrete')
```

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureValue::value () : Expression [1]

```
LiteralBoolean_Factory.create(true)
```

### 7.8.2.3.11 RateMetadataUsageContinuousReferenceUsage\_Mapping

#### Description

Creates a reference usage.

#### General Mappings

ToReferenceUsage\_Init  
Mapping

#### Mapping Source

Element

#### Mapping Target

ReferenceUsage

#### Owned Mappings

(none)

#### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
Helper.hasStereotypeApplied(src, 'SysML::Activities::Continuous')
```

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReferenceUsage::ownedRelationship () : Relationship [0..\*]

```
Set{RateMetadataUsageContinuousReferenceUsageRedefinition_Mapping.getMapped(from) ,  
RateMetadataUsageFeatureValue_Mapping.getMapped(from) }
```

### 7.8.2.3.12 RateMetadataUsageContinuousReferenceUsageRedefinition\_Mapping

#### Description

Creates a redefinition relationship for the *redefiningFeature()* and the *redefinedFeature()*.

#### General Mappings

ToRedefinition\_Init  
Mapping

#### Mapping Source

Element

#### Mapping Target

Redefinition

#### Owned Mappings

(none)

#### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
Helper.hasStereotypeApplied(src, 'SysML::Activities::Continuous')
```

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Redefinition::redefinedFeature () : Feature [1]`

```
SysML2::AttributeUsage.allInstances()  
->any(m | m.qualifiedName = 'SysMLv1Library::RateData::isContinuous')
```

### 7.8.2.3.13 RateMetadataUsageDiscreteFeatureMembership\_Mapping

#### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

#### General Mappings

ToFeatureMembership\_Init  
Mapping

#### Mapping Source

Element

#### Mapping Target

FeatureMembership

## Owned Mappings

(none)

## Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
Helper.hasStereotypeApplied(src, 'SysML::Activities::Discrete')
```

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureMembership::ownedMemberFeature () : Feature [1]

```
RateMetadataUsageDiscreteReferenceUsage_Mapping.getMapped(from)
```

### 7.8.2.3.14 RateMetadataUsageDiscreteReferenceUsage\_Mapping

#### Description

Creates a reference usage.

#### General Mappings

ToReferenceUsage\_Init  
Mapping

#### Mapping Source

Element

#### Mapping Target

ReferenceUsage

## Owned Mappings

(none)

## Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
Helper.hasStereotypeApplied(src, 'SysML::Activities::Discrete')
```

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReferenceUsage::ownedRelationship () : Relationship [0..\*]

```
Set{RateMetadataUsageDiscreteReferenceUsageRedefinition_Mapping.getMapped(from),
RateMetadataUsageFeatureValue_Mapping.getMapped(from)}
```

### 7.8.2.3.15 RateMetadataUsageDiscreteReferenceUsageRedefinition\_Mapping

#### Description

Creates a redefinition relationship for the *redefiningFeature()* and the *redefinedFeature()*.

#### General Mappings

ToRedefinition\_Init  
Mapping

#### Mapping Source

Element

#### Mapping Target

Redefinition

#### Owned Mappings

(none)

#### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
Helper.hasStereotypeApplied(src, 'SysML::Activities::Discrete')
```

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Redefinition::redefinedFeature () : Feature [1]

```
SysML2::AttributeUsage.allInstances()
->any(m | m.qualifiedName = 'SysMLv1Library::RateData::isDiscrete')
```

### 7.8.2.3.16 RateMetadataUsageFeatureTyping\_Mapping

#### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

#### General Mappings

ToFeatureTyping\_Init  
Mapping

#### Mapping Source

Element



## Mapping Target

FeatureTyping

## Owned Mappings

(none)

## Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
Helper.hasStereotypeApplied(src, 'SysML::Activities::Rate')  
or Helper.hasStereotypeApplied(src, 'SysML::Activities::Continuous')  
or Helper.hasStereotypeApplied(src, 'SysML::Activities::Discrete')
```

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureTyping::type () : Type [1]  

```
SysML2::MetadataDefinition.allInstances()  
->any(m | m.qualifiedName = 'SysMLv1Library::RateData')
```

### 7.8.2.3.17 RateOwningMembership\_Mapping

#### Description

Creates a owning membership relationship for *ownedMemberElement()*.

#### General Mappings

ToOwningMembership\_Init  
Mapping

#### Mapping Source

Element

#### Mapping Target

OwningMembership

#### Owned Mappings

(none)

#### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
Helper.hasStereotypeApplied(src, 'SysML::Activities::Rate')  
or Helper.hasStereotypeApplied(src, 'SysML::Activities::Continuous')  
or Helper.hasStereotypeApplied(src, 'SysML::Activities::Discrete')
```

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `OwningMembership::ownedMemberElement () : Element [1]`  
`RateMetadataUsage_Mapping.getMapped (from)`

### 7.8.2.3.18 Model Libraries

#### 7.8.2.3.18.1 ControlValues

##### 7.8.2.3.18.1.1 ControlValueKind

The enumeration `ControlValueKind` is mapped to the SysML v2 enumeration definition `SysMLv1Library::Enumerations::ControlValueKind` (see [7.3.2](#)).

## 7.8.3 Allocations

### 7.8.3.1 Overview

**Table 23. List of all mappings**

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
Allocate	AllocationUsage
AllocateActivityPartition	

### 7.8.3.2 SysML::Allocations elements not mapped

**Table 24. List of SysML v1 elements not mapped of this section**

SysML v1 Concept	Rationale
AllocateActivityPartition	Mapping is not specified yet.

### 7.8.3.3 Mapping Specifications

#### 7.8.3.3.1 Allocation\_Mapping

##### Description

A `SysML::Allocations::Allocate` is mapped to a SysML v2 `AllocationDefinition` if it is an allocation between definition elements.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
action def SysMLv1Activity {
    action sysMLv1Action;
}
part def SysMLv1Block {
    part sysMLv1PartProperty : AnotherSysMLv1Block;
}
part def AnotherSysMLv1Block;

// Allocation of definition
allocation def SysMLv1Allocation {
    end :>> source : SysMLv1Activity;
    end :>> target : SysMLv1Block;
```

```

}

// Allocation of usage
allocation def {
    end :>> source : SysMLv1Activity;
    end :>> target : SysMLv1Block;
    allocate source.sysMLv1Action to target.sysMLv1PartProperty;
}

// Allocation of usage to definition
allocation def {
    end :>> source : SysMLv1Activity;
    end :>> target : SysMLv1Block;
    allocate source.sysMLv1Action to target;
}

```

## General Mappings

Abstraction\_Mapping

### Mapping Source

Abstraction

### Mapping Target

AllocationDefinition

### Owned Mappings

(none)

### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
(Helper.hasStereotypeApplied(src, 'SysML::Allocations::Allocate'))
```

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- AllocationDefinition::ownedRelationship () : Relationship [0..\*]

```

let relationships : Set(KerML::Relationship) =
    Set{AllocationSourceFeatureMembership_Mapping.getMapped(from.client.get(0)),
        AllocationTargetFeatureMembership_Mapping.getMapped(from.supplier.get(0))}
    ->union(self.oclAsType(ElementMain_Mapping).ownedRelationship()) in
if from.client.get(0).oclIsKindOf(UML::Type) then
    relationships
else
    relationships->including(AllocationUsageFeatureMembership_Mapping.getMapped(from))
endif

```

#### 7.8.3.3.2 AllocationFeatureMembership\_Mapping

##### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

### General Mappings

ToFeatureMembership\_Init  
Mapping

### Mapping Source

NamedElement

### Mapping Target

FeatureMembership

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureMembership::ownedMemberFeature () : Feature [1]  
`AllocationSourceReferenceUsage_Mapping.getMapped(from)`

## 7.8.3.3.3 AllocationFeatureTyping\_Mapping

### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

### General Mappings

ToFeatureTyping\_Init  
Mapping

### Mapping Source

NamedElement

### Mapping Target

FeatureTyping

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureTyping::type () : Type [1]`  

```
if from.ocIsKindOf(UML::Type) then
    from
else
    from.owner
endif
```

#### 7.8.3.3.4 AllocationReferenceUsage\_Mapping

##### Description

Creates a reference usage.

##### General Mappings

ToReferenceUsage\_Init  
UniqueMapping

##### Mapping Source

NamedElement

##### Mapping Target

ReferenceUsage

##### Owned Mappings

(none)

##### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceUsage::ownedRelationship () : Relationship [0..*]`  

```
Set{AllocationFeatureTyping_Mapping.getMapped(from),
AllocationSourceReferenceUsageRedefinition_Mapping.getMapped(from)}
```
- `ReferenceUsage::isEnd () : Boolean [1]`  

```
true
```

#### 7.8.3.3.5 AllocationSourceReferenceUsageRedefinition\_Mapping

##### Description

Creates a redefinition relationship for the *redefiningFeature()* and the *redefinedFeature()*.

##### General Mappings

ToRedefinition\_Init

##### Mapping Source

NamedElement

##### Mapping Target

Redefinition

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Redefinition::redefinedFeature () : Feature [1]`  

```
SYSML2::ReferenceUsage.allInstances()  
->any(m | m.qualifiedName = 'Allocations::Allocation::source')
```

#### 7.8.3.3.6 AllocationTargetFeatureMembership\_Mapping

##### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

##### General Mappings

ToFeatureMembership\_Init

##### Mapping Source

NamedElement

##### Mapping Target

FeatureMembership

##### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureMembership::ownedMemberFeature () : Feature [1]`  
`AllocationTargetReferenceUsage_Mapping.getMapped(from)`

#### 7.8.3.3.7 AllocationTargetReferenceUsage\_Mapping

##### Description

Creates a reference usage.

##### General Mappings

`ToReferenceUsage_Init`  
`UniqueMapping`

##### Mapping Source

`NamedElement`

##### Mapping Target

`ReferenceUsage`

##### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceUsage::ownedRelationship () : Relationship [0..*]`  
`Set{AllocationFeatureTyping_Mapping.getMapped(from),`  
`AllocationTargetReferenceUsageRedefinition_Mapping.getMapped(from) }`
- `ReferenceUsage::isEnd () : Boolean [1]`  
`true`

#### 7.8.3.3.8 AllocationTargetReferenceUsageRedefinition\_Mapping

##### Description

Creates a redefinition relationship for the *redefiningFeature()* and the *redefinedFeature()*.

### General Mappings

ToRedefinition\_Init

### Mapping Source

NamedElement

### Mapping Target

Redefinition

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Redefinition::redefinedFeature () : Feature [1]`

```
SysML2::ReferenceUsage.allInstances()  
->any(m | m.qualifiedName = 'Allocations::Allocation::target')
```

#### 7.8.3.3.9 AllocationUsage\_Mapping

### Description

A SysML::Allocations::Allocate is mapped to a SysML v2 AllocationUsage owned by a AllocationDefinition if a usage element is source or target of the allocation relationship.

### General Mappings

ToUsage\_Init  
Mapping

### Mapping Source

Abstraction

### Mapping Target

AllocationUsage

### Owned Mappings

(none)

### Applicable filters



(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `AllocationUsage::ownedRelationship () : Relationship [0..*]`  
`Set{AllocationUsageSourceEndFeatureMembership_Mapping.getMapped(from.client.get(0)), AllocationUsageTargetEndFeatureMembership_Mapping.getMapped(from.target.get(0))}`

#### 7.8.3.3.10 AllocationUsageEndFeatureMembership\_Mapping

##### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

##### General Mappings

ToEndFeatureMembership\_Init  
Mapping

##### Mapping Source

NamedElement

##### Mapping Target

EndFeatureMembership

##### Owned Mappings

(none)

##### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `EndFeatureMembership::ownedMemberFeature () : Feature [1]`  
`AllocationUsageSourceFeature_Mapping.getMapped(from)`

#### 7.8.3.3.11 AllocationUsageFeature\_Mapping

##### Description

Creates a feature element as an end of the allocation usage relationship.

##### General Mappings

ToFeature\_Init  
Mapping

### Mapping Source

NamedElement

### Mapping Target

Feature

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Feature::ownedRelationship () : Relationship [0..\*]  
`Set{AllocationUsageSourceFeatureSubsetting_Mapping.getMapped(from) }`

### 7.8.3.3.12 AllocationUsageFeatureChaining\_Mapping

#### Description

Creates the first feature chaining element for the subsetting feature for the feature element which represents an end of the allocation usage relationship.

#### General Mappings

ToFeatureChaining\_Init  
Mapping

### Mapping Source

NamedElement

### Mapping Target

FeatureChaining

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureChaining::chainingFeature () : Feature [1]`

`AllocationSourceReferenceUsage_Mapping.getMapped (from)`

#### **7.8.3.3.13 AllocationUsageFeatureChainingChainedFeature\_Mapping**

##### **Description**

Creates the second feature chaining element for the subsetting feature for the feature element which represents an end of the allocation usage relationship.

##### **General Mappings**

ToFeatureChaining\_Init  
Mapping

##### **Mapping Source**

NamedElement

##### **Mapping Target**

FeatureChaining

##### **Owned Mappings**

(none)

##### **Applicable filters**

(none)

##### **Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureChaining::chainingFeature () : Feature [1]`

`from`

#### **7.8.3.3.14 AllocationUsageFeatureMembership\_Mapping**

##### **Description**

Creates a feature membership relationship for *ownedMemberFeature()*.

##### **General Mappings**

ToFeatureMembership\_Init  
Mapping

##### **Mapping Source**

Abstraction

### Mapping Target

FeatureMembership

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureMembership::ownedMemberFeature () : Feature [1]  
`AllocationUsage_Mapping.getMapped(from)`

### 7.8.3.3.15 AllocationUsageFeatureSubsetting\_Mapping

#### Description

Creates a subsetting relationship.

#### General Mappings

ToReferenceSubsetting\_Init  
Mapping

#### Mapping Source

NamedElement

#### Mapping Target

ReferenceSubsetting

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReferenceSubsetting::ownedRelatedElement () : Element [0..\*]  
`if from.ocIsKindOf(UML::Type) then  
Set{}`

```

else
    Set{AllocationUsageSourceFeatureSubsettingFeature_Mapping.getMapped(from)}
endif

```

### 7.8.3.3.16 AllocationUsageFeatureSubsettingFeature\_Mapping

#### Description

Creates the subsetting feature for the feature element which represents an end of the allocation usage relationship.

#### General Mappings

ToFeature\_Init  
Mapping

#### Mapping Source

NamedElement

#### Mapping Target

Feature

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Feature::ownedRelationship () : Relationship [0..\*]

```

Set{AllocationUsageSourceFeatureChaining_Mapping.getMapped(from),
AllocationUsageFeatureChainingChainedFeature_Mapping.getMapped(from)}

```

### 7.8.3.3.17 AllocationUsageTargetEndFeatureMembership\_Mapping

#### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

#### General Mappings

ToEndFeatureMembership\_Init

#### Mapping Source

NamedElement

#### Mapping Target

EndFeatureMembership

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `EndFeatureMembership::ownedMemberFeature () : Feature [1]`  
`AllocationUsageTargetFeature_Mapping.getMapped(from)`

### 7.8.3.3.18 AllocationUsageTargetFeature\_Mapping

#### Description

Creates a feature element as an end of the allocation usage relationship.

#### General Mappings

ToFeature\_Init

#### Mapping Source

NamedElement

#### Mapping Target

Feature

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Feature::ownedRelationship () : Relationship [0..*]`  
`Set{AllocationUsageTargetFeatureSubsetting_Mapping.getMapped(from) }`

### 7.8.3.3.19 AllocationUsageTargetFeatureChaining\_Mapping

#### Description

Creates the first feature chaining element for the subsetting feature for the feature element which represents an end of the allocation usage relationship.

## General Mappings

ToFeatureChaining\_Init

## Mapping Source

NamedElement

## Mapping Target

FeatureChaining

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureChaining::chainingFeature () : Feature [1]

`AllocationTargetReferenceUsage_Mapping.getMapped (from)`

### 7.8.3.3.20 AllocationUsageTargetFeatureSubsetting\_Mapping

#### Description

Creates a subsetting relationship.

## General Mappings

ToReferenceSubsetting\_Init

## Mapping Source

NamedElement

## Mapping Target

ReferenceSubsetting

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceSubsetting::ownedRelatedElement () : Element [0..*]`

```
if from.ocIsKindOf(UML::Type) then
    Set{}
else
    Set{AllocationUsageTargetFeatureSubsettingFeature_Mapping.getMapped(from)}
endif
```

### 7.8.3.3.21 AllocationUsageTargetFeatureSubsettingFeature\_Mapping

#### Description

Creates the subsetting feature for the feature element which represents an end of the allocation usage relationship.

#### General Mappings

ToFeature\_Init

#### Mapping Source

NamedElement

#### Mapping Target

Feature

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Feature::ownedRelationship () : Relationship [0..*]`

```
Set{AllocationUsageTargetFeatureChaining_Mapping.getMapped(from),
AllocationUsageFeatureChainingChainedFeature_Mapping.getMapped(from)}
```

## 7.8.4 Blocks

### 7.8.4.1 Overview

Table 25. List of all mappings

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
AdjunctProperty	
BindingConnector	BindingConnectorAsUsage



SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
Block	PartDefinition PartDefinition
BoundReference	
ClassifierBehaviorProperty	
ConnectorProperty	
DistributedProperty	
EndPathMultiplicity	
NestedConnectorEnd	
ParticipantProperty	
PropertySpecificType	
ValueType	AttributeDefinition

#### 7.8.4.2 SysML::Blocks elements not mapped

**Table 26. List of SysML v1 elements not mapped of this section**

SysML v1 Concept	Rationale
AdjunctProperty	The concept of adjunct properties is not needed in SysML v2, where the principal of the adjunct property can be used directly in the appropriate place.
BoundReference	Mapping is not specified yet.
ClassifierBehaviorProperty	The classifier behavior is already mapped to a property which also plays the role of the classifier behavior property. Therefore, there is no explicit mapping of a classifier behavior property.
ConnectorProperty	The connector property is a special case of an adjunct property and is not mapped, just like the adjunct property.
DirectedRelationshipPropertyPath	The stereotype is abstract is therefore not mapped. The concept of the DirectedRelationshipPropertyPath is included in the SysML v2 language.
DistributedProperty	Mapping is not specified yet.
ElementPropertyPath	The stereotype is abstract is therefore not mapped. The concept of the ElementPropertyPath is included in the SysML v2 language.
EndPathMultiplicity	Mapping is not specified yet.
NestedConnectorEnd	The concept of NestedConnectorEnd is already included in the SysML v2 language. It is not required to do an explicit mapping.
ParticipantProperty	Mapping is not specified yet.
PropertySpecificType	Mapping is not specified yet.

### 7.8.4.3 Mapping Specifications

#### 7.8.4.3.1 AssociationBlock\_Mapping

##### Description

An AssociationBlock is mapped to a SysML v2 ConnectionDefinition.

The SysML::Blocks::ParticipantProperties transformation is not defined yet. Therefore, the mapping is currently identical with the mapping of UML4SysML::AssociationClass.

##### General Mappings

AssociationClass\_Mapping

##### Mapping Source

AssociationClass

##### Mapping Target

ConnectionDefinition

##### Owned Mappings

(none)

##### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
Helper.hasStereotypeApplied(src, 'SysML::Blocks::Block')
```

##### Mapping rules

The mapping class only has inherited rules. See the mapping classes in the general mapping section for details.

#### 7.8.4.3.2 BindingConnector\_Mapping

##### Description

A SysML::Blocks::BindingConnector is mapped to a SysML v2 BindingConnectorAsUsage.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
part def SysMLv1Block1 {
    part sysMLv1PartProperty1 : SysMLv1Block2;
    part sysMLv1PartProperty2 : SysMLv1Block2;

    binding sysMLv1BindingConnector
        bind sysMLv1PartProperty1 = sysMLv1PartProperty2;
}
part def SysMLv1Block2;
```

##### General Mappings

Connector\_Mapping

#### Mapping Source

Connector

#### Mapping Target

BindingConnectorAsUsage

#### Owned Mappings

(none)

#### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
Helper.hasStereotypeApplied(src, 'SysML::Blocks::BindingConnector')
```

#### Mapping rules

The mapping class only has inherited rules. See the mapping classes in the general mapping section for details.

### 7.8.4.3.3 Block\_Mapping

#### Description

A SysML::Blocks::Block is mapped to a SysML v2 PartDefinition.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
part definition SysMLv1Block;
```

#### General Mappings

Class\_Mapping

#### Mapping Source

Class

#### Mapping Target

PartDefinition

#### Owned Mappings

(none)

#### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```

not src.ocIsTypeOf (UML::AssociationClass)
and Helper.hasStereotypeApplied(src, 'SysML::Blocks::Block')
and not Helper.hasStereotypeApplied(src, 'SysML::ConstraintBlocks::ConstraintBlock')
and not Helper.hasStereotypeApplied(src, 'SysML::Ports&Flows::InterfaceBlock')

```

## Mapping rules

The mapping class only has inherited rules. See the mapping classes in the general mapping section for details.

### 7.8.4.3.4 EncapsulatedBlock\_Mapping

#### Description

A SysML::Block with *isEncapsulated=true* is mapped to a SysML v2 PartDefinition, and, additionally, gets a metadata feature defined by the SysML v1 library which represents the SysML v1 isEncapsulated property.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```

part def SysMLv1EncapsulatedBlock {
  @SysMLv1Library::BlockData {isEncapsulated = true;}
}

```

## General Mappings

Block\_Mapping

### Mapping Source

Class

### Mapping Target

PartDefinition

### Owned Mappings

(none)

### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```

not src.ocIsTypeOf (UML::AssociationClass) and
  Helper.hasStereotypeApplied(src, 'SysML::Blocks::Block') and
  not Helper.hasStereotypeApplied(src, 'SysML::ConstraintBlocks::ConstraintBlock') and
  not Helper.hasStereotypeApplied(src, 'SysML::Ports&Flows::InterfaceBlock') and
  Helper.getTagValue(src, 'SysML::Blocks::Block', 'isEncapsulated')

```

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- PartDefinition::ownedRelationship () : Relationship [0..\*]

```

let toElementFMS: Set(UML::Element) =
    from.ownedElement->select(e | e.ocIsKindOf(UML::Property) and
    (e.ocIsType(UML::Property).redefinedProperty->size() = 0)) in
let redefinedAttributes: Set(UML::Element) =
    from.ownedElement->select(e | from.ocIsKindOf(UML::DataType) and
    (e.ocIsType(UML::Property).redefinedProperty->size() > 0)) in
let generalizations : Set(UML::Generalization) =
    from.ownedElement->select(e | e.ocIsKindOf(UML::Generalization)) in
let toElementOMS: Set(UML::Element) =
    (((from.ownedElement - toElementFMS) - redefinedAttributes) -
    generalizations) in
let relationships: Sequence(UML::Element) =
    toElementOMS->collect(e | ElementOwningMembership_Mapping.getMapped(e))
->union(toElementFMS
    ->collect(e | ElementFeatureMembership_Mapping.getMapped(e)))
->union(redefinedAttributes
    ->collect(e | AttributeRedefinedMembership_Mapping.getMapped(e)))
->union(generalizations->collect(e | Generalization_Mapping.getMapped(e)))
->including(EncapsulatedBlockMetadataMembership_Mapping.getMapped(from)) in
if from.classifierBehavior.ocIsUndefined() then
    relationships
else
    relationships
    ->append(BehavioredClassifierFeatureMembership_Mapping.getMapped(from))
endif

```

#### 7.8.4.3.5 EncapsulatedBlockMetadataMembership\_Mapping

##### Description

Creates a membership relationship for *memberElement()*.

##### General Mappings

ToOwningMembership\_Init  
Mapping

##### Mapping Source

Class

##### Mapping Target

OwningMembership

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- OwningMembership::ownedMemberElement () : Element [1]

```
EncapsulatedBlockMetadata_Mapping.getMapped(from)
```

#### 7.8.4.3.6 EncapsulatedBlockMetadata\_Mapping

##### Description

The mapping class creates the metadata for the property SysML::Blocks::Block::isEncapsulated.

##### General Mappings

ToMetadataUsage\_Init  
Mapping

##### Mapping Source

Class

##### Mapping Target

MetadataUsage

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- MetadataUsage::ownedRelationship () : Relationship [0..\*]

```
Set{EncapsulatedBlockMetadataFeatureTyping_Mapping.getMapped(from) ,  
EncapsulatedBlockMetadataFeatureMembership_Mapping.getMapped(from) }
```

#### 7.8.4.3.7 EncapsulatedBlockMetadataFeatureMembership\_Mapping

##### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

##### General Mappings

ToFeatureMembership\_Init  
Mapping

##### Mapping Source

Class

##### Mapping Target

FeatureMembership

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureMembership::ownedMemberFeature () : Feature [0..1]`  
`EncapsulatedBlockMetadataReferenceUsage_Mapping.getMapped(from)`

### 7.8.4.3.8 EncapsulatedBlockMetadataFeatureTyping\_Mapping

#### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

#### General Mappings

ToFeatureTyping\_Init  
Mapping

#### Mapping Source

Class

#### Mapping Target

FeatureTyping

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureTyping::type () : Type [1]`  
`SysML2::MetadataDefinition.allInstances()`  
`->any(m | m.qualifiedName = 'SysMLv1Library::BlockData')`

### 7.8.4.3.9 EncapsulatedBlockMetadataReferenceUsage\_Mapping

#### Description

Creates a reference usage.

### General Mappings

ToReferenceUsage\_Init  
Mapping

### Mapping Source

Class

### Mapping Target

ReferenceUsage

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReferenceUsage::ownedRelationship () : Relationship [0..\*]

```
Set{EncapsulatedBlockMetadataRedefinition_Mapping.getMapped(from),  
EncapsulatedBlockMetadataFeatureValue_Mapping.getMapped(from) }
```

#### 7.8.4.3.10 EncapsulatedBlockMetadataFeatureValue\_Mapping

### Description

Creates a feature value relationship.

### General Mappings

ToFeatureValue\_Init  
Mapping

### Mapping Source

Class

### Mapping Target

FeatureValue

### Owned Mappings

(none)

### Applicable filters



(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureValue::value () : Expression [1]`

```
LiteralBoolean_Factory.create(true)
```

#### 7.8.4.3.11 EncapsulatedBlockMetadataRedefinition\_Mapping

##### Description

Creates a redefinition relationship for the *redefiningFeature()* and the *redefinedFeature()*.

##### General Mappings

ToRedefinition\_Init  
Mapping

##### Mapping Source

Class

##### Mapping Target

Redefinition

##### Owned Mappings

(none)

##### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Redefinition::redefinedFeature () : Feature [1]`

```
SysML2::AttributeUsage.allInstances()  
->any(m | m.qualifiedName = 'SysMLv1Library::BlockData::isEncapsulated')
```

#### 7.8.4.3.12 FlowPropertyPart\_Mapping

##### Description

A `UML4SysML::Property` which is typed by a block and to which the SysML v1 stereotype `FlowProperty` has been applied is mapped as in the general mapping class `PartProperty_Mapping` but the target feature is always referential and the flow direction specified in the stereotype `FlowProperty` is considered.

##### General Mappings

PartProperty\_Mapping

## Mapping Source

Property

## Mapping Target

PartUsage

## Owned Mappings

(none)

## Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
Helper.hasStereotypeApplied(src, 'SysML::Ports&Flows::FlowProperty')
and not src.type.ocIsUndefined()
and src.type.ocIsKindOf(UML::Class)
and Helper.hasStereotypeApplied(src.type, 'SysML::Blocks::Block')
```

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- PartUsage::isComposite () : Boolean [1]  
  
false
- PartUsage::direction () : FeatureDirectionKind [0..1]  
  
Helper.getFlowDirectionKind(  
  
Helper.getTagValue(from,  
  
'SysML::Ports&Flows::FlowProperty', 'direction'))

### 7.8.4.3.13 PartProperty\_Mapping

#### Description

A UML4SysML::Property which is typed by a block is mapped to a SysML::PartUsage. The derived property Property::isComposite is directly mapped to PartUsage::isComposite.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
part def SysMLv1Block1 {
    part sysMLv1PartProperty1 : SysMLv1Block2;
    ref part sysMLv1ReferencedPartProperty2 : SysMLv1Block2;
}
part def SysMLv1Block2;
```

## General Mappings

PropertyTypedByClassInterface\_Mapping

### Mapping Source

Property

### Mapping Target

PartUsage

### Owned Mappings

(none)

### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
if src.ocIsKindOf(UML::Property) and not src.ocIsKindOf(UML::Port) then
    let p: UML::Property = src.ocAsType(UML::Property) in
        not p.type.ocIsUndefined() and
        Helper.hasStereotypeApplied(p.type, 'SysML::Blocks::Block') and
        (p.association.ocIsUndefined() or p.association.ownedEnd->excludes(p))
    else
        false
endif
```

### Mapping rules

The mapping class only has inherited rules. See the mapping classes in the general mapping section for details.

#### 7.8.4.3.14 Model Libraries

##### 7.8.4.3.14.1 PrimitiveValueTypes

The SysML v1 model library PrimitiveValueTypes contains primitive types that are mapped to the appropriate scalar values in SysML v2.

###### 7.8.4.3.14.1.1 Boolean

The SysML v1 primitive type Boolean is mapped to the SysML v2 ScalarValues::Boolean element.

###### 7.8.4.3.14.1.2 Complex

The SysML v1 primitive type Complex is mapped to the SysML v2 ScalarValues::Complex element.

###### 7.8.4.3.14.1.3 Integer

The SysML v1 primitive type Integer is mapped to the SysML v2 ScalarValues::Integer element.

###### 7.8.4.3.14.1.4 Number

The SysML v1 primitive type Number is abstract. Therefore, no mapping is defined for it.

#### 7.8.4.3.14.1.5 Real

The SysML v1 primitive type Real is mapped to the SysML v2 ScalarValues::Real element.

#### 7.8.4.3.14.1.6 String

The SysML v1 primitive type String is mapped to the SysML v2 ScalarValues::String element.

#### 7.8.4.3.14.2 UnitAndQuantityKind

The SysML v1 model library UnitAndQuantityKind contains the blocks Unit and QuantityKind.

##### 7.8.4.3.14.2.1 QuantityKind

The mapping of the SysML v1 QuantityKind element is not specified yet.

##### 7.8.4.3.14.2.2 Unit

The mapping of the SysML v1 QuantityKind element is not specified yet.

#### 7.8.4.3.15 ValueType\_Mapping

##### Description

A SysML::Blocks::ValueType is mapped to a SysML v2 AttributeDefinition.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
attribute definition SysMLv1ValueType;
```

##### General Mappings

DataType\_Mapping

##### Mapping Source

DataType

##### Mapping Target

AttributeDefinition

##### Owned Mappings

(none)

##### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
Helper.hasStereotypeApplied(from, 'SysML::Blocks::ValueType')
```

##### Mapping rules

The mapping class only has inherited rules. See the mapping classes in the general mapping section for details.

## 7.8.5 ConstraintBlocks

### 7.8.5.1 Overview

Table 27. List of all mappings

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
ConstraintBlock	ConstraintDefinition

### 7.8.5.2 Mapping Specifications

#### 7.8.5.2.1 ConstraintBlock\_Mapping

##### Description

A SysML::ConstraintBlocks::ConstraintBlock is mapped to a SysML v2 ConstraintDefinition.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
onstraint def SysMLv1ConstraintBlock {
    in attribute a : ScalarValues::Integer;
    in attribute b : ScalarValues::Integer;
    in attribute c : ScalarValues::Integer;

    constraint constraintExpression {
        language "OCL2.0"
        /*
         * c == a + b
         */
    }
}
```

##### General Mappings

Class\_Mapping

##### Mapping Source

Class

##### Mapping Target

ConstraintDefinition

##### Owned Mappings

(none)

##### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
Helper.hasStereotypeApplied(src, 'SysML::ConstraintBlocks::ConstraintBlock')
```

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ConstraintDefinition::ownedRelationship () : Relationship [0..*]`

```
let generalizations : Set(UML::Generalization) =
    from.ownedElement->select(e | e.ocIsKindOf(UML::Generalization)) in
let toElementFMS : Set(UML::Element) =
    from.ownedElement
    ->select(e | e.ocIsKindOf(UML::Property) or e.ocIsKindOf(UML::Constraint)) in
let toElementOMS: Set(UML::Element) =
    (from.ownedElement - generalizations) - toElementFMS in
toElementOMS->collect(e | ElementOwningMembership_Mapping.getMapped(e))
->union(toElementFMS->collect(e | ElementFeatureMembership_Mapping.getMapped(e)))
->union(generalizations->collect(e | Generalization_Mapping.getMapped(e)))
->including(CommonReturnParameterReferenceUsageMembership_Mapping.getMapped(from))
```

### 7.8.5.2.2 ConstraintParameter\_Mapping

#### Description

The mapping class maps SysML v1 constraint parameter to SysML v2 attribute usages.

#### General Mappings

PropertyCommon\_Mapping  
NamedElementMain\_Mapping

#### Mapping Source

Property

#### Mapping Target

AttributeUsage

#### Owned Mappings

(none)

#### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
if src.ocIsKindOf(UML::Property) and
Helper.hasStereotypeApplied(src.owner, 'SysML::ConstraintBlocks::ConstraintBlock') then
    let p: UML::Property = src.ocAsType(UML::Property) in
    if p.type.ocIsUndefined() then
        false
    else
        true
    endif
else
    false
endif
```

#### Mapping rules

The mapping class only has inherited rules. See the mapping classes in the general mapping section for details.

## 7.8.6 Model Elements

### 7.8.6.1 Overview

**Table 28.** List of all mappings

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
Conform	
ElementGroup	Package
Expose	
Problem	Comment
Rationale	Comment
Stakeholder	ItemDefinition
View	
Viewpoint	

### 7.8.6.2 SysML::ModelElements elements not mapped

**Table 29.** List of SysML v1 elements not mapped of this section

SysML v1 Concept	Rationale
Conform	Mapping is not specified yet.
Expose	Mapping is not specified yet.
View	Mapping is not specified yet.

### 7.8.6.3 Mapping Specifications

#### 7.8.6.3.1 ProblemRationaleMetadataFeatureMembership\_Mapping

##### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

##### General Mappings

ToFeatureMembership\_Init  
Mapping

##### Mapping Source

Comment

##### Mapping Target

FeatureMembership

##### Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureMembership::ownedMemberFeature () : Feature [0..1]  
`ProblemRationaleMetadataReferenceUsage_Mapping.getMapped(from)`

### 7.8.6.3.2 ProblemRationaleMetadataFeatureTyping\_Mapping

#### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

#### General Mappings

ToFeatureTyping\_Init  
Mapping

#### Mapping Source

Comment

#### Mapping Target

FeatureTyping

#### Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureTyping::type () : Type [1]  

```
if Helper.hasStereotypeApplied(from, 'SysML::ModelElements::Problem') then
  SysML2::MetadataDefinition.allInstances()
  ->any(m | m.qualifiedName = 'ModelingMetadata::Issue')
else if Helper.hasStereotypeApplied(from, 'SysML::ModelElements::Rationale') then
  SysML2::MetadataDefinition.allInstances()
  ->any(m | m.qualifiedName = 'ModelingMetadata::Rationale')
else invalid endif endif
```



### 7.8.6.3.3 ProblemRationaleMetadataReferenceUsage\_Mapping

#### Description

Creates a reference usage.

#### General Mappings

ToReferenceUsage\_Init  
Mapping

#### Mapping Source

Comment

#### Mapping Target

ReferenceUsage

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReferenceUsage::ownedRelationship () : Relationship [0..\*]  

```
Set{ProblemRationaleMetadataRedefinition_Mapping.getMapped(from),  
ProblemRationaleMetadataFeatureValue_Mapping.getMapped(from)}
```

### 7.8.6.3.4 ProblemRationaleMetadataFeatureValue\_Mapping

#### Description

Creates a feature value relationship.

#### General Mappings

ToFeatureValue\_Init  
Mapping

#### Mapping Source

Comment

#### Mapping Target

FeatureValue

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureValue::value () : Expression [1]`  
`LiteralString_Factory.create(from.body)`

### 7.8.6.3.5 ProblemRationaleMetadataMembership\_Mapping

#### Description

Creates a membership relationship for *memberElement()*.

#### General Mappings

ToOwningMembership\_Init  
Mapping

#### Mapping Source

Comment

#### Mapping Target

OwningMembership

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `OwningMembership::ownedMemberElement () : Element [1]`  
`ProblemRationaleMetadataUsage_Mapping.getMapped(from)`

### 7.8.6.3.6 Concern\_Mapping

#### Description

The concern comments of a SysML::ModelElements::Stakeholder or a SysML::ModelElements::Viewpoint are mapped to SysML v2 ConcernUsages. The concern comments of the stakeholder are mapped to ConcernUsages which reference the stakeholder item definition.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
item def SysMLv1Stakeholder {
    @SysMLv1Library::StakeholderData {isStakeholder = true;}
}
concern concernCommentXMI_ID {
    doc /* concern string */
    stakeholder : SysMLv1Stakeholder;
}
```

## General Mappings

Comment\_Mapping

### Mapping Source

Comment

### Mapping Target

ConcernUsage

## Owned Mappings

(none)

## Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
(not Helper.hasStereotypeApplied(src, 'SysML::ModelElements::ElementGroup')) and
((UML::Classifier.allInstances()
->select(s |
    Helper.hasStereotypeApplied(s, 'SysML::ModelElements::Stakeholder'))
->collect(c |
    Helper.getTagValue(c, 'SysML::ModelElements::Stakeholder', 'concernList'))
->flatten()
->includes(src)) or
(UML::Classifier.allInstances()
->select(s |
    Helper.hasStereotypeApplied(s, 'SysML::ModelElements::Viewpoint'))
->collect(c |
    Helper.getTagValue(c, 'SysML::ModelElements::Viewpoint', 'concernList'))
->flatten()->includes(src)))
```

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ConcernUsage::ownedRelationship () : Relationship [0..\*]

```
let toStakeholderMS : Set(UML::Classifier) =
```

```

UML::Classifier.allInstances()
->select(s |
    Helper.hasStereotypeApplied(s, 'SysML::ModelElements::Stakeholder'))
->select(s |
    Helper.getTagValue(s, 'SysML::ModelElements::Stakeholder', 'concernList')
->flatten()->includes(from))->asSet() in
toStakeholderMS
->including(
    CommonReturnParameterReferenceUsageMembership_Mapping.getMapped(from))
->including(EmptySubjectMembership_Factory.create())
->union(self.oclAsType(Comment_Mapping).ownedRelationship())

```

### 7.8.6.3.7 ConcernDocumentation\_Mapping

#### Description

The mapping class creates the documentation element with the body string of the UML4SysML::Comment model element representing a concern.

#### General Mappings

ToDocumentation\_Init  
Mapping

#### Mapping Source

Comment

#### Mapping Target

Documentation

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Documentation::body () : String [1]  
from.body

### 7.8.6.3.8 ConcernOwningMembership\_Mapping

#### Description

Creates a owning membership relationship for *ownedMemberElement()*.

## General Mappings

ToOwningMembership\_Init  
Mapping

## Mapping Source

Comment

## Mapping Target

OwningMembership

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- OwningMembership::ownedMemberElement () : Element [1]  
`ConcernDocumentation_Mapping.getMapped(from)`

### 7.8.6.3.9 ConcernStakeholderMembership\_Mapping

#### Description

Creates a membership relationship for *memberElement()*.

## General Mappings

ToParameterMembership\_Init  
Mapping

## Mapping Source

Classifier

## Mapping Target

StakeholderMembership

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- StakeholderMembership::ownedMemberParameter () : Feature [1]  
`ConcernStakeholderPartUsage_Mapping.getMapped(from)`

### 7.8.6.3.10 ConcernStakeholderPartUsage\_Mapping

#### Description

In SysML v1, the stakeholder element has concerns. In SysML v2, the Concern element has stakeholders. This mapping class creates a PartUsage of the type of the stakeholder for the concern element.

#### General Mappings

ToPartUsage\_Init  
Mapping

#### Mapping Source

Classifier

#### Mapping Target

PartUsage

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- PartUsage::ownedRelationship () : Relationship [0..\*]  
`Set { ConcernStakeholderPartUsageFeatureTyping_Mapping.getMapped(from),  
ConcernStakeholderPartUsageOwningMembership_Mapping.getMapped(from) }`

### 7.8.6.3.11 ConcernStakeholderPartUsageFeatureTyping\_Mapping

#### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

#### General Mappings

ToFeatureTyping\_Init  
Mapping

**Mapping Source**

Classifier

**Mapping Target**

FeatureTyping

**Owned Mappings**

(none)

**Applicable filters**

(none)

**Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureTyping::type () : Type [1]

from

**7.8.6.3.12 ConcernStakeholderPartUsageOwningMembership\_Mapping****Description**

Creates a owning membership relationship for *ownedMemberElement()*.

**General Mappings**

ToOwningMembership\_Init  
Mapping

**Mapping Source**

Classifier

**Mapping Target**

OwningMembership

**Owned Mappings**

(none)

**Applicable filters**

(none)

**Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `OwningMembership::ownedMemberElement () : Element [1]`  
`ConcernStakeholderPartUsageFeature_Mapping.getMapped(from)`

### 7.8.6.3.13 ConcernStakeholderPartUsageFeature\_Mapping

#### Description

The mapping class creates a feature element for the concern stakeholder part usage.

#### General Mappings

ToFeature\_Init  
Mapping

#### Mapping Source

Classifier

#### Mapping Target

Multiplicity

#### Owned Mappings

(none)

#### Applicable filters

(none)

### 7.8.6.3.14 ElementGroup\_Mapping

#### Description

A SysML::ModelElements::ElementGroup element is mapped to a SysML v2 Package with membership import relationships representing the grouping.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
package ElementGroupModel {
  part def SysMLv1Block1;
  attribute def SysMLv1ValueType;
  part def SysMLv1Block2 {
    part sysMLv1PartProperty:SysMLv1Block1;
  }
}

package SysMLv1ElementGroup {
  import ElementGroupModel::SysMLv1Block1;
  import ElementGroupModel::SysMLv1ValueType;
  import ElementGroupModel::SysMLv1Block2::sysMLv1PartProperty;

  @SysMLv1Library::ElementGroupData {criterion = "criterion string";}
}
```



## General Mappings

Comment\_Mapping

## Mapping Source

Comment

## Mapping Target

Package

## Owned Mappings

(none)

## Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
Helper.hasStereotypeApplied(src, 'SysML::ModelElements::ElementGroup')
```

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Package::ownedRelationship () : Relationship [0..\*]

```
let elements : Set(KerML::Relationahip) =  
  Helper.getTagValueAsElementColl(from,  
    'SysML::ModelElements::ElementGroup', 'member')  
  ->collect(e | CommonElementImport_Mapping.getMapped(e)) in  
elements->including(ElementGroupMetadaMembership_Mapping.getMapped(from))  
->union(self.oclAsType(ElementMain_Mapping).ownedRelationship())
```

- Package::declaredName () : String [0..1]

```
Helper.getTagValueAsString(from, 'SysML::ModelElements::ElementGroup', 'name')
```

### 7.8.6.3.15 ElementGroupMetadaMembership\_Mapping

#### Description

Creates a membership relationship for *memberElement()*.

#### General Mappings

ToOwningMembership\_Init  
Mapping

#### Mapping Source

Comment

### Mapping Target

OwningMembership

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- OwningMembership::ownedMemberElement () : Element [1]  
`ElementGroupMetadataUsage_Mapping.getMapped(from)`

### 7.8.6.3.16 ElementGroupMetadataFeatureMembership\_Mapping

#### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

#### General Mappings

ToFeatureMembership\_Init  
Mapping

#### Mapping Source

Comment

#### Mapping Target

FeatureMembership

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureMembership::ownedMemberFeature () : Feature [1]  
`ElementGroupMetadataReferenceUsage_Mapping.getMapped(from)`

### 7.8.6.3.17 ElementGroupMetadataFeatureTyping\_Mapping

#### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

#### General Mappings

ToFeatureTyping\_Init  
Mapping

#### Mapping Source

Comment

#### Mapping Target

FeatureTyping

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureTyping::type () : Type [1]  
  

```
SYSMML2::MetadataDefinition.allInstances()  
->any(m | m.qualifiedName = 'SysMLv1Library::ElementGroupData')
```

### 7.8.6.3.18 ElementGroupMetadataFeatureValue\_Mapping

#### Description

Creates a feature value relationship.

#### General Mappings

ToFeatureValue\_Init  
Mapping

#### Mapping Source

Comment

#### Mapping Target

FeatureValue

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureValue::value () : Expression [1]

```
let criterion: String =  
  Helper.getTagValueAsString(  
    from,  
    'SysML::ModelElements::ElementGroup', 'criterion') in  
  LiteralString_Factory.create(criterion)
```

### 7.8.6.3.19 ElementGroupMetadataRedefinition\_Mapping

#### Description

Creates a redefinition relationship for the *redefiningFeature()* and the *redefinedFeature()*.

#### General Mappings

ToRedefinition\_Init  
Mapping

#### Mapping Source

Comment

#### Mapping Target

Redefinition

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Redefinition::redefinedFeature () : Feature [1]

```
let m : SYSML2::Membership =  
  SYSML2::AttributeUsage.allInstances()  
  ->collect(dt | dt.owningRelationship)  
  ->select(r | r.ocIsKindOf(SYSML2::Membership))
```

```

->any(m | m.memberName = 'criterion') in
if (m.ocliIsUndefined()) then
    invalid
else
    m.memberElement
endif

```

### 7.8.6.3.20 ElementGroupMetadataReferenceUsage\_Mapping

#### Description

Creates a reference usage.

#### General Mappings

ToReferenceUsage\_Init  
Mapping

#### Mapping Source

Comment

#### Mapping Target

ReferenceUsage

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReferenceUsage::ownedRelationship () : Relationship [0..\*]

```

Set{ElementGroupMetadataRedefinition_Mapping.getMapped(from),
ElementGroupMetadataFeatureValue_Mapping.getMapped(from)}

```

### 7.8.6.3.21 ElementGroupMetadataUsage\_Mapping

#### Description

The mapping class creates the metadata usage element for the SysML::ModelElements::ElementGroup mapping.

#### General Mappings

ToMetadataUsage\_Init  
Mapping

#### Mapping Source

Comment

## Mapping Target

MetadataUsage

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- MetadataUsage::ownedRelationship () : Relationship [0..\*]

```
Set{ElementGroupMetadataFeatureTyping_Mapping.getMapped(from),  
ElementGroupMetadataFeatureMembership_Mapping.getMapped(from)}
```

### 7.8.6.3.22 ProblemRationale\_Mapping

#### Description

The mapping class combines the mapping of SysML::ModelElements::Problem and SysML::ModelElements::Rationale. The SysML::ModelElements::Problem is mapped to the library element ModelingMetadata::Issue and the SysML::ModelElements::Rationale is mapped to ModelingMetadata::Rationale.

The expected SysML v2 textual syntax of the mapping is as follows.

```
@ModelingMetadata::Issue {text = "This is a problem statement";}  
  
@ModelingMetadata::Rationale {text = "This is a rationale statement";}
```

#### General Mappings

Comment\_Mapping

#### Mapping Source

Comment

#### Mapping Target

Comment

#### Owned Mappings

(none)

#### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
(not Helper.hasStereotypeApplied(src, 'SysML::ModelElements::ElementGroup')) and
(Helper.hasStereotypeApplied(src, 'SysML::ModelElements::Problem') or
Helper.hasStereotypeApplied(src, 'SysML::ModelElements::Rationale'))
```

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Comment::ownedRelationship () : Relationship [0..*]`

```
self.oclAsType(ElementMain_Mapping).ownedRelationship()
->including(ProblemRationaleMetadataMembership_Mapping.getMapped(from))
```

### 7.8.6.3.23 ProblemRationaleMetadataRedefinition\_Mapping

#### Description

Creates a redefinition relationship for the *redefiningFeature()* and the *redefinedFeature()*.

#### General Mappings

ToRedefinition\_Init  
Mapping

#### Mapping Source

Comment

#### Mapping Target

Redefinition

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Redefinition::redefinedFeature () : Feature [1]`

```
if Helper.hasStereotypeApplied(from, 'SysML::ModelElements::Problem') then
  SYSML2::AttributeUsage.allInstances()
  ->any(m | m.qualifiedName = 'ModelingMetadata::Issue::text')
else if Helper.hasStereotypeApplied(from, 'SysML::ModelElements::Rationale') then
  SYSML2::AttributeUsage.allInstances()
  ->any(m | m.qualifiedName = 'ModelingMetadata::Rationale::text')
else
  invalid
endif
endif
```

#### 7.8.6.3.24 ProblemRationaleMetadataUsage\_Mapping

##### Description

The mapping class creates the metadata usage element for the SysML::ModelElements::Problem and SysML::ModelElements::Rationale transformation target.

##### General Mappings

ToMetadataUsage\_Init  
Mapping

##### Mapping Source

Comment

##### Mapping Target

MetadataUsage

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- MetadataUsage::ownedRelationship () : Relationship [0..\*]

```
Set{ProblemRationaleMetadataFeatureTyping_Mapping.getMapped(from),  
ProblemRationaleMetadataFeatureMembership_Mapping.getMapped(from)}
```

#### 7.8.6.3.25 Stakeholder\_Mapping

##### Description

A SysML::ModelElements::Stakeholder is mapped to a SysML v2 ItemDefinition with metadata to tag it as a stakeholder. The concern comments of the stakeholder are mapped to ConcernUsages which reference the stakeholder item definition.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
item def SysMLv1Stakeholder {@SysMLv1Library::StakeholderData {isStakeholder = true;}}  
concern concernCommentXMI_ID {  
  doc /* concern string */  
  stakeholder : SysMLv1Stakeholder;  
}
```

##### General Mappings



Class\_Mapping

## Mapping Source

Class

## Mapping Target

ItemDefinition

## Owned Mappings

(none)

## Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
Helper.hasStereotypeApplied(src, 'SysML::ModelElements::Stakeholder')
```

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ItemDefinition::ownedRelationship () : Relationship [0..\*]

```
let toElementFMS: Set(UML::Element) =
  from.ownedElement
  ->select(e | (e.ocIsKindOf(UML::Property) and
    (e.ocAsType(UML::Property).redefinedProperty->size()
      = 0)) or e.ocIsKindOf(UML::Operation)) in
let redefinedAttributes: Set(UML::Element) =
  from.ownedElement
  ->select(e | from.ocIsKindOf(UML::DataType) and
    (e.ocAsType(UML::Property).redefinedProperty->size() > 0)) in
let generalizations : Set(UML::Generalization) =
  from.ownedElement
  ->select(e | e.ocIsKindOf(UML::Generalization)) in
let constraints : Set(UML::Constraint) =
  UML::Constraint.allInstances()
  ->select(c | c.constrainedElement->includes(from)) in
let toElementOMS: Set(UML::Element) =
  (((from.ownedElement - toElementFMS) - redefinedAttributes) -
    generalizations) in
let relationships: Sequence(KerML::Relationship) =
toElementOMS->collect(e | ElementOwningMembership_Mapping.getMapped(e))
->union(toElementFMS->collect(e | ElementFeatureMembership_Mapping.getMapped(e)))
->union(constraints
  ->collect(e | ConstrainedElementFeatureMembership_Mapping.getMapped(e)))
->union(redefinedAttributes
  ->collect(e | AttributeRedefinedMembership_Mapping.getMapped(e)))
->union(generalizations->collect(e | Generalization_Mapping.getMapped(e)))
->including(StakeholderMetadataOwningMembership_Mapping.getMapped(from)) in
if from.classifierBehavior.ocIsUndefined() then
  relationships
else
  relationships
```

```

        ->append(BehavioedClassifierFeatureMembership_Mapping.getMapped(from))
    endif

```

### 7.8.6.3.26 StakeholderMetadataUsage\_Mapping

#### Description

The mapping class creates the metadata usage element for the SysML::ModelElements::Stakeholder mapping.

#### General Mappings

ToMetadataUsage\_Init  
Mapping

#### Mapping Source

Classifier

#### Mapping Target

MetadataUsage

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- MetadataUsage::ownedRelationship () : Relationship [0..\*]

```

Set{StakeholderMetadataFeatureTyping_Mapping.getMapped(from) ,
StakeholderMetadataFeatureMembership_Mapping.getMapped(from) }

```

### 7.8.6.3.27 StakeholderMetadataFeatureMembership\_Mapping

#### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

#### General Mappings

ToFeatureMembership\_Init  
Mapping

#### Mapping Source

Classifier

#### Mapping Target

FeatureMembership

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureMembership::ownedMemberFeature () : Feature [1]`  
`StakeholderMetadataReferenceUsage_Mapping.getMapped (from)`

### 7.8.6.3.28 StakeholderMetadataFeatureTyping\_Mapping

#### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

#### General Mappings

ToFeatureTyping\_Init  
Mapping

#### Mapping Source

Classifier

#### Mapping Target

FeatureTyping

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureTyping::type () : Type [1]`  
`SysML2::MetadataDefinition.allInstances ()`  
`->any (m | m.qualifiedName = 'SysMLv1Library::StakeholderData')`

### 7.8.6.3.29 StakeholderMetadataOwningMembership

#### Description

Creates a owning membership relationship for *ownedMemberElement()*.

### General Mappings

ToOwningMembership\_Init  
Mapping

### Mapping Source

Classifier

### Mapping Target

OwningMembership

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- OwningMembership::ownedMemberElement () : Element [1]  
`StakeholderMetadataUsage_Mapping.getMapped(from)`

## 7.8.6.3.30 StakeholderMetadataReferenceUsage\_Mapping

### Description

Creates a reference usage.

### General Mappings

ToReferenceUsage\_Init  
Mapping

### Mapping Source

Classifier

### Mapping Target

ReferenceUsage

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceUsage::ownedRelationship () : Relationship [0..*]`

```
Set{StakeholderMetadataReferenceUsageRedefinition_Mapping.getMapped(from) ,  
StakeholderMetadataReferenceUsageFeatureValue_Mapping.getMapped(from) }
```

#### 7.8.6.3.31 StakeholderMetadataReferenceUsageFeatureValue\_Mapping

##### Description

Creates a feature value relationship.

##### General Mappings

ToFeatureValue\_Init  
Mapping

##### Mapping Source

Classifier

##### Mapping Target

FeatureValue

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureValue::value () : Expression [1]`

```
LiteralBoolean_Factory.create(true)
```

#### 7.8.6.3.32 StakeholderMetadataReferenceUsageRedefinition\_Mapping

##### Description

Creates a redefinition relationship for the *redefiningFeature()* and the *redefinedFeature()*.

##### General Mappings

ToRedefinition\_Init  
Mapping

## Mapping Source

Classifier

## Mapping Target

Redefinition

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Redefinition::redefinedFeature () : Feature [1]`

```
SysML2::AttributeUsage.allInstances()  
->any(m | m.qualifiedName = 'SysMLv1Library::StakeholderData::isStakeholder')
```

### 7.8.6.3.33 Viewpoint\_Mapping

#### Description

A `SysML::ModelElements::Viewpoint` is mapped to a SysML v2 `ViewDefinition` with an owned SysML v2 `ViewpointUsage`. In SysML v1, the viewpoint combines the purpose and stakeholder concerns as well as presentation information. This is covered by a SysML v2 `ViewDefinition` with owned SysML v2 `ViewpointUsage`.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
view def SysMLv1Viewpoint {  
  viewpoint sysMLv1Viewpoint {  
    frame concern1XmiID1;  
    frame concern2XmiID2;  
    metadata SysMLv1Library::ViewpointData {  
      languages = ("language1", "language2");  
      presentations = ("presentation1", "presentation2");  
    }  
    require constraint {  
      doc /* thisIsThePurpose */  
    }  
  }  
  satisfy sysMLv1Viewpoint;  
  rendering {  
    action : SysMLv1ViewpointMethodBehavior1;  
    action : SysMLv1ViewpointMethodBehavior2;  
  }  
}  
action def SysMLv1ViewpointMethodBehavior1;  
action def SysMLv1ViewpointMethodBehavior2;  
  
item def SysMLv1Stakeholder {@SysMLv1Library::StakeholderData {isStakeholder = true;}}
```

```

concern concern1XmiID1 {
  doc /* Concern1 */
  stakeholder : SysMLv1Stakeholder;
}
concern concern2XmiID2 {
  doc /* Concern2 */
  stakeholder : SysMLv1Stakeholder;
}

```

## General Mappings

Class\_Mapping

### Mapping Source

Class

### Mapping Target

ViewDefinition

### Owned Mappings

(none)

### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
Helper.hasStereotypeApplied(src, 'SysML::ModelElements::Viewpoint')
```

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ViewDefinition::ownedRelationship () : Relationship [0..\*]

```

let toElementFMS: Set(UML::Element) =
  from.ownedElement->select(e | (e.ocIsKindOf(UML::Property) and
    (e.ocAsType(UML::Property).redefinedProperty->size() = 0)) or
    e.ocIsKindOf(UML::Comment)) in
let redefinedAttributes: Set(UML::Element) =
  from.ownedElement->select(e | from.ocIsKindOf(UML::DataType) and
    (e.ocAsType(UML::Property).redefinedProperty->size() > 0)) in
let generalizations : Set(UML::Generalization) =
  from.ownedElement->select(e | e.ocIsKindOf(UML::Generalization)) in
let toElementOMS: Set(UML::Element) =
  (((from.ownedElement - toElementFMS) - redefinedAttributes) -
    generalizations) in
let relationships: Sequence(UML::Element) =
  toElementOMS->collect(e | ElementOwningMembership_Mapping.getMapped(e))
->union(toElementFMS->collect(e | ElementFeatureMembership_Mapping.getMapped(e)))
->union(redefinedAttributes
  ->collect(e | AttributeRedefinedMembership_Mapping.getMapped(e)))
->union(generalizations->collect(e | Generalization_Mapping.getMapped(e)))
->including(ViewpointViewpointUsageFeatureMembership_Mapping.getMapped(from))

```

```

->including(ViewpointSatisfyFeatureMembership_Mapping.getMapped(from))
->including(ViewpointRenderingFeatureMembership_Mapping.getMapped(from))
->including(
    CommonReturnParameterReferenceUsageMembership_Mapping.getMapped(from)) in
if from.classifierBehavior.oclIsUndefined() then
    relationships
else
    relationships
->append(BehavioiredClassifierFeatureMembership_Mapping.getMapped(from))
endif

```

### 7.8.6.3.34 ViewpointConcernReferenceSubsetting\_Mapping

#### Description

Creates a subsetting relationship.

#### General Mappings

ToReferenceSubsetting\_Init  
Mapping

#### Mapping Source

Comment

#### Mapping Target

ReferenceSubsetting

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReferenceSubsetting::referencedFeature () : Feature [1]  
from

### 7.8.6.3.35 ViewpointConcernUsage\_Mapping

#### Description

The mapping class creates the concern usage element for the SysML::ModelElements::Viewpoint mapping.

#### General Mappings

ToRequirementUsage\_Init  
Mapping



### Mapping Source

Comment

### Mapping Target

ConcernUsage

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ConcernUsage::ownedRelationship () : Relationship [0..\*]  

```
Set{ViewpointConcernReferenceSubsetting_Mapping.getMapped(from),  
EmptySubjectMembership_Factory.create(),  
CommonReturnParameterReferenceUsageMembership_Mapping.getMapped(from)}
```

### 7.8.6.3.36 ViewpointConstraintUsage\_Mapping

#### Description

The mapping class creates the constraint usage element for the SysML::ModelElements::Viewpoint mapping.

#### General Mappings

ToConstraintUsage\_Init  
Mapping

### Mapping Source

Class

### Mapping Target

ConstraintUsage

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ConstraintUsage::ownedRelationship () : Relationship [0..*]`  

```
Set{ViewpointConstraintUsageOwningMembership_Mapping.getMapped(from),  
ReturnParameterFeatureMembership_Factory.create() }
```

#### 7.8.6.3.37 ViewpointConstraintUsageDocumentation\_Mapping

##### Description

The mapping class creates the documentation element for the SysML::ModelElements::Viewpoint mapping.

##### General Mappings

ToDocumentation\_Init  
Mapping

##### Mapping Source

Class

##### Mapping Target

Documentation

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Documentation::body () : String [1]`  

```
Helper.getTagValueAsString(from, 'SysML::ModelElements::Viewpoint', 'purpose')
```

#### 7.8.6.3.38 ViewpointConstraintUsageOwningMembership\_Mapping

##### Description

Creates a owning membership relationship for *ownedMemberElement()*.

##### General Mappings

ToOwningMembership\_Init  
Mapping

##### Mapping Source

Class

### Mapping Target

OwningMembership

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- OwningMembership::ownedMemberElement () : Element [1]  
`ViewpointConstraintUsageDocumentation_Mapping.getMapped(from)`

## 7.8.6.3.39 ViewpointFramedConcernMembership\_Mapping

### Description

Creates a membership relationship for *memberElement()*.

### General Mappings

ToFeatureMembership\_Init  
Mapping

### Mapping Source

Comment

### Mapping Target

FramedConcernMembership

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FramedConcernMembership::ownedMemberFeature () : Feature [1]  
`ViewpointConcernUsage_Mapping.getMapped(from)`

#### 7.8.6.3.40 ViewpointLanguagesMetadataFeatureMembership\_Mapping

##### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

##### General Mappings

ToFeatureMembership\_Init  
Mapping

##### Mapping Source

Class

##### Mapping Target

FeatureMembership

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureMembership::ownedMemberFeature () : Feature [1]  
`ViewpointLanguagesMetadataReferenceUsage_Mapping.getMapped(from)`

#### 7.8.6.3.41 ViewpointLanguagesMetadataFeatureValue\_Mapping

##### Description

Creates a feature value relationship.

##### General Mappings

ToFeatureValue\_Init  
Mapping

##### Mapping Source

Class

##### Mapping Target

FeatureValue

##### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureValue::value () : Expression [1]`

```
ViewpointLanguagesMetadataOperatorExpression_Mapping.getMapped(from)
```

### 7.8.6.3.42 ViewpointLanguagesMetadataRedefinition\_Mapping

#### Description

Creates a redefinition relationship for the *redefiningFeature()* and the *redefinedFeature()*.

#### General Mappings

ToRedefinition\_Init  
Mapping

#### Mapping Source

Class

#### Mapping Target

Redefinition

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Redefinition::redefinedFeature () : Feature [1]`

```
SYSMML2::AttributeUsage.allInstances()  
->any(m | m.qualifiedName = 'SysMLv1Library::ViewpointData::languages')
```

### 7.8.6.3.43 ViewpointLanguagesMetadataReferenceUsage\_Mapping

#### Description

Creates a reference usage.

## General Mappings

ToReferenceUsage\_Init  
Mapping

## Mapping Source

Class

## Mapping Target

ReferenceUsage

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReferenceUsage::ownedRelationship () : Relationship [0..\*]

```
Set{ViewpointLanguagesMetadataRedefinition_Mapping.getMapped(from) ,  
ViewpointLanguagesMetadataFeatureValue_Mapping.getMapped(from) }
```

### 7.8.6.3.44 ViewpointMetadataFeatureTyping\_Mapping

#### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

#### General Mappings

ToFeatureTyping\_Init  
Mapping

#### Mapping Source

Class

#### Mapping Target

FeatureTyping

#### Owned Mappings

(none)

#### Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureTyping::type () : Type [1]`

```
SysML2::MetadataDefinition.allInstances()  
->any(m | m.qualifiedName = 'SysMLv1Library::ViewpointData')
```

### 7.8.6.3.45 ViewpointLanguagesMetadataOperatorExpression\_Mapping

#### Description

The mapping class creates the operator expression for the list of languages of the `SysML::ModelElements::Viewpoint` mapping.

#### General Mappings

`ToOperatorExpression_Init`  
Mapping

#### Mapping Source

Class

#### Mapping Target

`OperatorExpression`

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `OperatorExpression::ownedRelationship () : Relationship [0..*]`  

```
Helper.getTagValueAsStringColl(from, 'SysML::ModelElements::Viewpoint', 'language')  
->collect(e | StringParameterMembership_Factory.create(e))
```
- `OperatorExpression::operator () : String [1]`  

```
' , '
```

### 7.8.6.3.46 ViewpointMetadataOwningMembership\_Mapping

#### Description

Creates a owning membership relationship for `ownedMemberElement()`.

## General Mappings

ToOwningMembership\_Init  
Mapping

## Mapping Source

Class

## Mapping Target

OwningMembership

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- OwningMembership::ownedMemberElement () : Element [1]  
`ViewpointMetadataUsage_Mapping.getMapped(from)`

### 7.8.6.3.47 ViewpointMetadataUsage\_Mapping

#### Description

The mapping class creates the metadata usage element for the SysML::ModelElements::Viewpoint mapping.

## General Mappings

ToMetadataUsage\_Init  
Mapping

## Mapping Source

Class

## Mapping Target

MetadataUsage

## Owned Mappings

(none)

## Applicable filters

(none)



## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `MetadataUsage::ownedRelationship () : Relationship [0..*]`  

```
Set{ViewpointMetadataFeatureTyping_Mapping.getMapped(from) ,  
ViewpointLanguagesMetadataFeatureMembership_Mapping.getMapped(from) ,  
ViewpointPresentationsMetadataFeatureMembership_Mapping.getMapped(from) }
```

### 7.8.6.3.48 ViewpointPresentationsMetadataFeatureMembership\_Mapping

#### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

#### General Mappings

ToFeatureMembership\_Init  
Mapping

#### Mapping Source

Class

#### Mapping Target

FeatureMembership

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureMembership::ownedMemberFeature () : Feature [1]`  

```
ViewpointPresentationsMetadataReferenceUsage_Mapping.getMapped(from)
```

### 7.8.6.3.49 ViewpointPresentationsMetadataFeatureValue\_Mapping

#### Description

Creates a feature value relationship.

#### General Mappings

ToFeatureValue\_Init  
Mapping

### Mapping Source

Class

### Mapping Target

FeatureValue

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureValue::value () : Expression [1]`  
`ViewpointPresentationsMetadataOperatorExpression_Mapping.getMapped (from)`

## 7.8.6.3.50 ViewpointPresentationsMetadataOperatorExpression\_Mapping

### Description

The mapping class creates the operator expression for the list of presentations of the SysML::ModelElements::Viewpoint mapping.

### General Mappings

ToOperatorExpression\_Init  
Mapping

### Mapping Source

Class

### Mapping Target

OperatorExpression

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `OperatorExpression::operator () : String [1]`  
`' , '`
- `OperatorExpression::ownedRelationship () : Relationship [0..*]`  

```
Helper.getTagValueAsStringColl (from,
    'SysML::ModelElements::Viewpoint', 'presentation')
->collect (e | StringParameterMembership_Factory.create (e))
```

#### 7.8.6.3.51 ViewpointPresentationsMetadataRedefinition\_Mapping

##### Description

Creates a redefinition relationship for the *redefiningFeature()* and the *redefinedFeature()*.

##### General Mappings

ToRedefinition\_Init  
Mapping

##### Mapping Source

Class

##### Mapping Target

Redefinition

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Redefinition::redefinedFeature () : Feature [1]`  

```
SysML2::AttributeUsage.allInstances ()
->any (m | m.qualifiedName = 'SysMLv1Library::ViewpointData::presentations')
```

#### 7.8.6.3.52 ViewpointPresentationsMetadataReferenceUsage\_Mapping

##### Description

Creates a reference usage.

##### General Mappings

ToReferenceUsage\_Init  
Mapping

#### Mapping Source

Class

#### Mapping Target

ReferenceUsage

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReferenceUsage::ownedRelationship () : Relationship [0..\*]

```
Set{ViewpointPresentationsMetadataRedefinition_Mapping.getMapped(from) ,  
ViewpointPresentationsMetadataFeatureValue_Mapping.getMapped(from) }
```

### 7.8.6.3.53 ViewpointRenderingFeatureMembership\_Mapping

#### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

#### General Mappings

ToFeatureMembership\_Init  
Mapping

#### Mapping Source

Class

#### Mapping Target

FeatureMembership

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureMembership::ownedMemberFeature () : Feature [1]`  
`ViewpointRenderingUsage_Mapping.getMapped(from)`

#### 7.8.6.3.54 ViewpointRenderingUsage\_Mapping

##### Description

The mapping class creates the rendering usage element for the SysML::ModelElements::Viewpoint mapping class.

##### General Mappings

ToPartUsage\_Init  
Mapping

##### Mapping Source

Class

##### Mapping Target

RenderingUsage

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `RenderingUsage::ownedRelationship () : Relationship [0..*]`  
`from.ownedOperation`  
`->select( o | Helper.hasStereotypeApplied(o, 'Create') )`  
`->collect( e |`  
`ViewpointRenderingUsageActionUsageFeatureMembership_Mapping.getMapped(e) )`

#### 7.8.6.3.55 ViewpointRenderingUsageActionUsage\_Mapping

##### Description

The mapping class creates the action usage element for the rendering usage element for the SysML::ModelElements::Viewpoint mapping class.

##### General Mappings

ToActionUsage\_Init  
Mapping

### Mapping Source

Class

### Mapping Target

ActionUsage

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ActionUsage::ownedRelationship () : Relationship [0..\*]  
`Set{ViewpointRenderingUsageActionUsageFeatureTyping_Mapping.getMapped(from) }`

### 7.8.6.3.56 ViewpointRenderingUsageActionUsageFeatureMembership\_Mapping

#### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

#### General Mappings

ToFeatureMembership\_Init  
Mapping

### Mapping Source

Class

### Mapping Target

FeatureMembership

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureMembership::ownedMemberFeature () : Feature [1]`

`ViewpointRenderingUsageActionUsage_Mapping.getMapped(from)`

#### **7.8.6.3.57 ViewpointRenderingUsageActionUsageFeatureTyping\_Mapping**

##### **Description**

Creates a feature typing relationship owned by the element *typedFeature()*.

##### **General Mappings**

ToFeatureTyping\_Init  
Mapping

##### **Mapping Source**

Class

##### **Mapping Target**

FeatureTyping

##### **Owned Mappings**

(none)

##### **Applicable filters**

(none)

#### **7.8.6.3.58 ViewpointRequirementConstraintMembership\_Mapping**

##### **Description**

Creates a membership relationship for *memberElement()*.

##### **General Mappings**

ToFeatureMembership\_Init  
Mapping

##### **Mapping Source**

Class

##### **Mapping Target**

RequirementConstraintMembership

##### **Owned Mappings**

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- RequirementConstraintMembership::ownedMemberFeature () : Feature [1]  
`ViewpointConstraintUsage_Mapping.getMapped(from)`

### 7.8.6.3.59 ViewpointSatisfyFeatureMembership\_Mapping

#### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

#### General Mappings

ToFeatureMembership\_Init  
Mapping

#### Mapping Source

Class

#### Mapping Target

FeatureMembership

#### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureMembership::ownedMemberFeature () : Feature [1]  
`ViewpointSatisfyRequirementUsage_Mapping.getMapped(from)`

### 7.8.6.3.60 ViewpointSatisfyRequirementUsage\_Mapping

#### Description

The mapping class creates the satisfy requirement usage element for the SysML::ModelElements::Viewpoint mapping.

#### General Mappings



ToRequirementUsage\_Init  
Mapping

#### Mapping Source

Class

#### Mapping Target

SatisfyRequirementUsage

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- SatisfyRequirementUsage::ownedRelationship () : Relationship [0..\*]

```
Set{ViewpointSatisfyRequirementUsageReferenceSubsetting_Mapping.getMapped(from) ,  
EmptySubjectMembership_Factory.create() ,  
ReturnParameterFeatureMembership_Factory.create() }
```

### 7.8.6.3.61 ViewpointSatisfyRequirementUsageReferenceSubsetting\_Mapping

#### Description

Creates a subsetting relationship.

#### General Mappings

ToReferenceSubsetting\_Init  
Mapping

#### Mapping Source

Class

#### Mapping Target

ReferenceSubsetting

#### Owned Mappings

(none)

#### Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceSubsetting::referencedFeature () : Feature [1]`  
`ViewpointViewpointUsage_Mapping.getMapped(from)`

### 7.8.6.3.62 ViewpointViewpointUsage\_Mapping

#### Description

The mapping class creates the embedded viewpoint usage for the SysML::ModelElements::Viewpoint mapping.

#### General Mappings

ToUsage\_Init  
Mapping

#### Mapping Source

Class

#### Mapping Target

ViewpointUsage

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ViewpointUsage::ownedRelationship () : Relationship [0..*]`  

```
Helper.getTagValueAsElementColl(  
    from, 'SysML::ModelElements::Viewpoint', 'concernList')  
->collect(e | ViewpointFramedConcernMembership_Mapping.getMapped(e))  
->including(ViewpointMetadataOwningMembership_Mapping.getMapped(from))  
->including(EmptySubjectMembership_Factory.create())  
->including(ViewpointRequirementConstraintMembership_Mapping.getMapped(from))
```
- `ViewpointUsage::declaredName () : String [0..1]`  

```
from.name.substring(1,1).toLowerCase() + from.name.substring(2, from.name.size())
```

### 7.8.6.3.63 ViewpointViewpointUsageFeatureMembership\_Mapping

#### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

## General Mappings

ToFeatureMembership\_Init  
Mapping

## Mapping Source

Class

## Mapping Target

FeatureMembership

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureMembership::ownedMemberFeature () : Feature [1]  
`ViewpointViewpointUsage_Mapping.getMapped(from)`

## 7.8.7 PortsAndFlows

### 7.8.7.1 Overview

**Table 30. List of all mappings**

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
AcceptChangeStructuralFeatureEventAction	AcceptActionUsage
AddFlowPropertyValueOnNestedPortAction	
ChangeStructuralFeatureEvent	
DirectedFeature	PerformActionUsage
FlowProperty	AttributeUsage OccurrenceUsage ReferenceUsage PartUsage
FullPort	PartUsage
InterfaceBlock	PortDefinition
InvocationOnNestedPortAction	
ItemFlow	

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
ProxyPort	
TriggerOnNestedPort	
~InterfaceBlock	PortDefinition

### 7.8.7.2 SysML::Ports&Flows elements not mapped

**Table 31. List of SysML v1 elements not mapped of this section**

SysML v1 Concept	Rationale
AddFlowPropertyValueOnNestedPortAction	Mapping is not specified yet.
ChangeStructuralFeatureEvent	Mapping is not specified yet.
InvocationOnNestedPortAction	Mapping is not specified yet.
TriggerOnNestedPort	Mapping is not specified yet.

### 7.8.7.3 Mapping Specifications

#### 7.8.7.3.1 AcceptChangeStructuralFeatureEventAction\_Mapping

##### Description

The SysML::PortsAndFlows::AcceptChangeStructuralFeatureEventAction element is mapped to SysML v2 AcceptActionUsage. The details of the mapping are not defined yet.

##### General Mappings

AcceptEventAction\_Mapping

##### Mapping Source

AcceptEventAction

##### Mapping Target

AcceptActionUsage

##### Owned Mappings

(none)

##### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
Helper.hasStereotypeApplied(src,
'SysML::Ports&Flows::AcceptChangeStructuralFeatureEventAction')
```

##### Mapping rules

The mapping class only has inherited rules. See the mapping classes in the general mapping section for details.

### 7.8.7.3.2 CommonFullPort\_Mapping

#### Description

The abstract mapping class is the base class of the mapping classes for the SysML::Ports&Flows::FullPort mappings.

#### General Mappings

PropertyCommon\_Mapping

#### Mapping Source

Port

#### Mapping Target

PartUsage

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- PartUsage::ownedRelationship () : Relationship [0..\*]

```
let typings: Set(KerML::FeatureTyping) = if from.type.ocIsUndefined() then
  Set{}
else
  Set{StructuralFeatureToFeatureTyping_Mapping.getMapped(from)}
endif in
let subsettings: Set(KerML::Subsetting) = from.subsettedProperty
->collect(p | PropertySubsetting_Mapping.getMapped(from, p))->asSet() in
let defaultValue: Set(KerML::OwningMembership) =
if from.defaultValue.ocIsUndefined() then
  Set{}
else
  Set{DefaultValue_Mapping.getMapped(from)}
endif in
typings->union(subsettings)->union(defaultValue)
->including(MultiplicityMembership_Mapping.getMapped(from))->asSet()
->including(FullPortMetadataOwningMembership_Mapping.getMapped(from))
```

### 7.8.7.3.3 ConjugatedPortDefinition\_Mapping

#### Description

A SysML::Ports&Flows::InterfaceBlock element is mapped to a SysML v2 ConjugatedPortDefinition owned by the PortDefinition that is the target element of the main mapping of the SysML::Ports&Flows::InterfaceBlock.

## General Mappings

ToClassifier\_Init  
Mapping

## Mapping Source

Class

## Mapping Target

ConjugatedPortDefinition

## Owned Mappings

- portConjugation : PortConjugation\_Mapping

## Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
Helper.hasStereotypeApplied(src, 'SysML::Ports&Flows::InterfaceBlock')
```

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ConjugatedPortDefinition::ownedRelationship () : Relationship [0..\*]

```
Set{portConjugation.to}
```

### 7.8.7.3.4 FlowProperty\_Mapping

#### Description

A UML4SysML::Property which satisfies the filter condition of PropertyTypedByClassInterface\_Mapping and to which the SysML v1 stereotype FlowProperty has been applied is mapped as in the general mapping class PropertyTypedByClassInterface\_Mapping but the target feature is always referential and the flow direction specified in the stereotype FlowProperty is considered.

## General Mappings

PropertyTypedByClassInterface\_Mapping

## Mapping Source

Property

## Mapping Target

OccurrenceUsage

## Owned Mappings

(none)

### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
Helper.hasStereotypeApplied(src, 'SysML::Ports&Flows::FlowProperty')
  and ((not src.type.ocIsUndefined())
    and (src.type.ocIsKindOf(UML::Class)
      or src.type.ocIsKindOf(UML::Interface)))
```

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- OccurrenceUsage::isComposite () : Boolean [1]  
  
false
- OccurrenceUsage::direction () : FeatureDirectionKind [0..1]

```
Helper.getFlowDirectionKind(Helper.getTagValue(from,
'SysML::Ports&Flows::FlowProperty', 'direction'))
```

#### 7.8.7.3.5 FlowPropertyAttribute\_Mapping

##### Description

A UML4SysML::Property which satisfies the filter condition of Attribute\_Mapping and to which the SysML v1 stereotype FlowProperty has been applied is mapped as in the general mapping class Attribute\_Mapping with consideration of the flow direction specified in the stereotype FlowProperty.

##### General Mappings

Attribute\_Mapping

##### Mapping Source

Property

##### Mapping Target

AttributeUsage

##### Owned Mappings

(none)

### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
Helper.hasStereotypeApplied(src, 'SysML::Ports&Flows::FlowProperty')
  and (not src.type.ocIsUndefined() and src.type.ocIsKindOf(UML::DataType))
```

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `AttributeUsage::direction () : FeatureDirectionKind [0..1]`

```
Helper.getFlowDirectionKind (Helper.getTagValue (from,
'SysML::Ports&Flows::FlowProperty', 'direction'))
```

### 7.8.7.3.6 FlowPropertyUntyped\_Mapping

#### Description

A UML4SysML::Property which satisfies the filter condition of PropertyUntyped\_Mapping and to which the SysML v1 stereotype FlowProperty has been applied is mapped as in the general mapping class PropertyUntyped\_Mapping but the target feature is always referential and the flow direction specified in the stereotype FlowProperty is considered.

#### General Mappings

PropertyUntyped\_Mapping

#### Mapping Source

Property

#### Mapping Target

ReferenceUsage

#### Owned Mappings

(none)

#### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
Helper.hasStereotypeApplied(src, 'SysML::Ports&Flows::FlowProperty')
and src.type.ocIsUndefined()
and not Helper.hasStereotypeApplied(src.owner,
'SysML::ConstraintBlocks::ConstraintBlock')
```

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceUsage::isComposite () : Boolean [1]`

```
false
```

- `ReferenceUsage::direction () : FeatureDirectionKind [0..1]`

```
Helper.getFlowDirectionKind (Helper.getTagValue (from,
'SysML::Ports&Flows::FlowProperty', 'direction'))
```



### 7.8.7.3.7 FullPort\_Mapping

#### Description

A SysML::Ports&Flows::FullPort element is mapped to a part usage in SysML v2 with metadata that marks the part usage as a full port. The metadata is defined in the SysML v1 library for SysML v2.

The mapping class FullPortUntyped\_Mapping does the same for full ports that have no type.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
part sysMLv1FullPort : SysMLv1Block {SysMLv1Library::PortData {isFullPort = true;}}
```

#### General Mappings

Port\_Mapping  
CommonFullPort\_Mapping

#### Mapping Source

Port

#### Mapping Target

PartUsage

#### Owned Mappings

(none)

#### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
(not src.type.ocIsUndefined()) and  
Helper.hasStereotypeApplied(src, 'SysML::Ports&Flows::FullPort')
```

#### Mapping rules

The mapping class only has inherited rules. See the mapping classes in the general mapping section for details.

### 7.8.7.3.8 FullPortMetadata\_Mapping

#### Description

Create the metadata usage element to annotate a port with the information that its SysML v1 mapping source element is a SysML v1 full port element.

#### General Mappings

ToMetadataUsage\_Init  
Mapping

#### Mapping Source

Port

### Mapping Target

MetadataUsage

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- MetadataUsage::ownedRelationship () : Relationship [0..\*]  

```
Set { FullPortMetadataFeatureTyping_Mapping.getMapped (from) ,  
      FullPortMetadataFeatureMembership_Mapping.getMapped (from) }
```

#### 7.8.7.3.9 FullPortMetadataFeatureMembership\_Mapping

### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

### General Mappings

ToFeatureMembership\_Init  
Mapping

### Mapping Source

Port

### Mapping Target

FeatureMembership

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureMembership::ownedMemberFeature () : Feature [1]

```
FullPortMetadataReferenceUsage_Mapping.getMapped(from)
```

### 7.8.7.3.10 FullPortMetadataFeatureTyping\_Mapping

#### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

#### General Mappings

ToFeatureTyping\_Init  
Mapping

#### Mapping Source

Port

#### Mapping Target

FeatureTyping

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureTyping::type () : Type [1]

```
SYSML2::MetadataDefinition.allInstances()  
->any(m | m.qualifiedName = 'SysMLv1Library::PortData')
```

### 7.8.7.3.11 FullPortMetadataOwningMembership\_Mapping

#### Description

Creates a owning membership relationship for *ownedMemberElement()*.

#### General Mappings

ToOwningMembership\_Init  
Mapping

#### Mapping Source

Port

#### Mapping Target

OwningMembership

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `OwningMembership::ownedMemberElement () : Element [1]`  
`FullPortMetadata_Mapping.getMapped (from)`

### 7.8.7.3.12 FullPortMetadataReferenceUsage\_Mapping

#### Description

Creates a reference usage.

#### General Mappings

`ToReferenceUsage_Init`  
`Mapping`

#### Mapping Source

Port

#### Mapping Target

ReferenceUsage

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceUsage::ownedRelationship () : Relationship [0..*]`  
`Set { FullPortMetadataReferenceUsageRedefinition_Mapping.getMapped (from) ,`  
`FullPortMetadataReferenceUsageFeatureValue_Mapping.getMapped (from) }`

### 7.8.7.3.13 FullPortMetadataReferenceUsageFeatureValue\_Mapping

#### Description

Creates a feature value relationship.

### General Mappings

ToFeatureValue\_Init  
Mapping

### Mapping Source

Port

### Mapping Target

FeatureValue

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureValue::value () : Expression [1]  
`LiteralBoolean_Factory.create(true)`

## 7.8.7.3.14 FullPortMetadataReferenceUsageRedefinition\_Mapping

### Description

Creates a redefinition relationship for the *redefiningFeature()* and the *redefinedFeature()*.

### General Mappings

ToRedefinition\_Init  
Mapping

### Mapping Source

Port

### Mapping Target

Redefinition

### Owned Mappings

(none)

### Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Redefinition::redefinedFeature () : Feature [1]`

```
SysML2::AttributeUsage.allInstances()  
->any(m | m.qualifiedName = 'SysMLv1Library::PortData::isFullPort')
```

### 7.8.7.3.15 FullPortUntyped\_Mapping

#### Description

A `SysML::Ports&Flows::FullPort` element is mapped to a part usage in SysML v2 with metadata that marks the part usage as a full port. The metadata is defined in the SysML v1 library for SysML v2.

The mapping class `FullPort_Mapping` does the same for full ports with a type.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
part sysMLv1FullPort {SysMLv1Library::PortData {isFullPort = true;}}
```

#### General Mappings

PortUntyped\_Mapping  
CommonFullPort\_Mapping

#### Mapping Source

Port

#### Mapping Target

PartUsage

#### Owned Mappings

(none)

#### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
src.type.ocliIsUndefined() and  
Helper.hasStereotypeApplied(src, 'SysML::Ports&Flows::FullPort')
```

#### Mapping rules

The mapping class only has inherited rules. See the mapping classes in the general mapping section for details.

### 7.8.7.3.16 InterfaceBlock\_Mapping

#### Description

A SysML::Ports&Flows::InterfaceBlock element is mapped to a SysML v2 PortDefinition.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
port def SysMLv1InterfaceBlock;
```

#### General Mappings

Block\_Mapping

#### Mapping Source

Class

#### Mapping Target

PortDefinition

#### Owned Mappings

(none)

#### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
Helper.hasStereotypeApplied(src, 'SysML::Ports&Flows::InterfaceBlock')
```

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- PortDefinition::ownedRelationship () : Relationship [0..\*]

```
self.oclAsType(Block_Mapping).ownedRelationship()  
->including(InterfaceBlockOwningMembership_Mapping.getMapped(from))
```

### 7.8.7.3.17 InterfaceBlockConjugated\_Mapping

#### Description

A SysML::Ports&Flows::~InterfaceBlock element is mapped to a SysML v2 PortDefinition. The SysML v1 constraints ensure that the port definition is compatible with the appropriate port definition, which is the target of the mapping of the original interface block. Instead of the special tilde symbol, the port definition name gets a "c" symbol as a prefix. The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
port def cSysMLv1InterfaceBlock;
```

## General Mappings

InterfaceBlock\_Mapping

## Mapping Source

Class

## Mapping Target

PortDefinition

## Owned Mappings

(none)

## Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
Helper.hasStereotypeApplied(src, 'SysML::Ports&Flows::~InterfaceBlock')
```

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- PortDefinition::declaredName () : String [0..1]  

```
'c' + from.name.substring(2,from.name.size())
```

### 7.8.7.3.18 InterfaceBlockOwningMembership\_Mapping

#### Description

Creates a owning membership relationship for *ownedMemberElement()*.

#### General Mappings

ToOwningMembership\_Init  
Mapping

#### Mapping Source

Class

#### Mapping Target

OwningMembership

#### Owned Mappings

(none)

#### Applicable filters



(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `OwningMembership::ownedMemberElement () : Element [1]`  
`ConjugatedPortDefinition_Mapping.getMapped(from)`

#### 7.8.7.3.19 OperationDirectedFeature\_Mapping

##### Description

The mapping class sets the direction of the perform action usage if the SysML v1 mapping source operation has the stereotype `SysML::Ports&Flows::DirectedFeature` applied.

##### General Mappings

Operation\_Mapping

##### Mapping Source

Operation

##### Mapping Target

PerformActionUsage

##### Owned Mappings

(none)

##### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
Helper.hasStereotypeApplied(src, 'SysML::Ports&Flows::DirectedFeature')
```

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `PerformActionUsage::direction () : FeatureDirectionKind [0..1]`  

```
Helper.getKerMLFeatureDirectionKind(  
  Helper.getTagValueAsElement(  
    from, 'SysML::Ports&Flows::DirectedFeature', 'featureDirection'  
  )  
)
```

#### 7.8.7.3.20 PortConjugation\_Mapping

##### Description

Creates a PortConjugation between a PortDefinition and a ConjugatedPortDefinition element.

## General Mappings

ToConjugation\_Init  
Mapping

## Mapping Source

Class

## Mapping Target

PortConjugation

## Owned Mappings

- conjugatedPortDefinition : ConjugatedPortDefinition\_Mapping

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- PortConjugation::originalPortDefinition () : Type [1]  
from
- PortConjugation::conjugatedType () : Type [1]  
conjugatedPortDefinition.to

## 7.8.8 Requirements

### 7.8.8.1 Overview

Table 32. List of all mappings

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
Copy	
DeriveReq	ConnectionUsage
Refine	Dependency
Requirement	RequirementUsage
Satisfy	SatisfyRequirementUsage
TestCase	VerificationCaseDefinition
Trace	Dependency
Verify	RequirementVerificationMembership

## 7.8.8.2 SysML::Requirements elements not mapped

Table 33. List of SysML v1 elements not mapped of this section

SysML v1 Concept	Rationale
Copy	The copy relationship is not covered by SysML v2.

## 7.8.8.3 Mapping Specifications

### 7.8.8.3.1 DeriveReq\_Mapping

#### Description

A SysML::Requirements::DeriveReq relationship is mapped to a SysML v2 DerivationConnections::Derivation model library element.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
requirement <'id1'> SysMLv1Requirement {
    doc /*
        * requirement text
    */
}
requirement <'id2'> SysMLv1RequirementDerived {
    doc /*
        * requirement text
    */
}
connection : DerivationConnections::Derivation
    connect SysMLv1RequirementDerived to SysMLv1Requirement;
```

#### General Mappings

Abstraction\_Mapping  
ToConnectionUsage\_Init

#### Mapping Source

Abstraction

#### Mapping Target

ConnectionUsage

#### Owned Mappings

(none)

#### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
Helper.hasStereotypeApplied(src, 'SysML::Requirements::DeriveReq')
```

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ConnectionUsage::ownedRelationship () : Relationship [0..*]`

```
Set{DeriveReqFeatureTyping_Mapping.getMapped(from),
DeriveReqSourceEndFeatureMembership_Mapping.getMapped(from),
DeriveReqTargetEndFeatureMembership_Mapping.getMapped(from)}
->union(self.oclAsType(ElementMain_Mapping).ownedRelationship())
```

### 7.8.8.3.2 DeriveReqFeatureTyping\_Mapping

#### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

#### General Mappings

ToFeatureTyping\_Init  
Mapping

#### Mapping Source

Dependency

#### Mapping Target

FeatureTyping

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureTyping::type () : Type [1]`

```
SYSML2::ConnectionDefinition.allInstances()
->any(m | m.qualifiedName = 'DerivationConnections::Derivation')
```

### 7.8.8.3.3 DeriveReqSourceEndFeatureMembership\_Mapping

#### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

#### General Mappings

ToEndFeatureMembership\_Init  
Mapping

### Mapping Source

Dependency

### Mapping Target

EndFeatureMembership

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- EndFeatureMembership::ownedMemberFeature () : Feature [1]  
`DeriveReqSourceFeature_Mapping.getMapped (from)`

#### 7.8.8.3.4 DeriveReqSourceFeature\_Mapping

##### Description

The mapping class creates the source feature of the ConnectionUsage relationship for the mapping of the SysML v1 deriveReq relationship.

##### General Mappings

ToFeature\_Init  
Mapping

### Mapping Source

Dependency

### Mapping Target

Feature

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Feature::ownedRelationship () : Relationship [0..*]`

```
Set{DeriveReqSourceFeatureReferenceSubsetting_Mapping.getMapped(from) }
```

#### 7.8.8.3.5 DeriveReqSourceFeatureReferenceSubsetting\_Mapping

##### Description

Creates a subsetting relationship.

##### General Mappings

ToReferenceSubsetting\_Init  
Mapping

##### Mapping Source

Dependency

##### Mapping Target

ReferenceSubsetting

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceSubsetting::referencedFeature () : Feature [1]`

```
from.client->any(c | true)
```

#### 7.8.8.3.6 DeriveReqTargetEndFeatureMembership\_Mapping

##### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

##### General Mappings

ToEndFeatureMembership\_Init  
Mapping

##### Mapping Source

Dependency

## Mapping Target

EndFeatureMembership

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- EndFeatureMembership::ownedMemberFeature () : Feature [1]  
`DeriveReqTargetFeature_Mapping.getMapped(from)`

### 7.8.8.3.7 DeriveReqTargetFeature\_Mapping

#### Description

The mapping class creates the target feature of the ConnectionUsage relationship for the mapping of the SysML v1 deriveReq relationship.

#### General Mappings

ToFeature\_Init  
Mapping

#### Mapping Source

Dependency

#### Mapping Target

Feature

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Feature::ownedRelationship () : Relationship [0..\*]  
`Set{DeriveReqTargetFeatureReferenceSubsetting_Mapping.getMapped(from) }`

### 7.8.8.3.8 DeriveReqTargetFeatureReferenceSubsetting\_Mapping

#### Description

Creates a subsetting relationship.

#### General Mappings

ToReferenceSubsetting\_Init  
Mapping

#### Mapping Source

Dependency

#### Mapping Target

ReferenceSubsetting

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReferenceSubsetting::referencedFeature () : Feature [1]  
  
from.supplier->any(c | true)

### 7.8.8.3.9 Refine\_Mapping

#### Description

A SysML::Requirements::Refine relationship is mapped to a SysML v2 Dependency relationship annotated with a metadata usage tagging it as a former SysML v1 refine relationship.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
requirement <'idl'> SysMLv1Requirement {  
  doc /*  
    * requirement text  
  */  
}  
use case def SysMLv1UseCase;  
  
dependency from SysMLv1UseCase to SysMLv1Requirement {  
  @SysMLv1Library::RefineData {isRefine = true;}  
}
```

#### General Mappings



Abstraction\_Mapping

### Mapping Source

Abstraction

### Mapping Target

Dependency

### Owned Mappings

(none)

### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
Helper.hasStereotypeApplied(src, 'SysML::Requirements::Refine')
```

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Dependency::ownedRelationship () : Relationship [0..\*]

```
self.oclAsType(ElementMain_Mapping).ownedRelationship()  
->including(RefineAnnotation_Mapping.getMapped(from))
```

## 7.8.8.3.10 RefineAnnotation\_Mapping

### Description

The mapping class creates the annotation relationship for the SysML::Requirements::Refine mapping.

### General Mappings

ToAnnotation\_Init  
Mapping

### Mapping Source

Abstraction

### Mapping Target

Annotation

### Owned Mappings

(none)

### Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Annotation::annotatingElement () : AnnotatingElement [1]`  
`RefineMetadataUsage_Mapping.getMapped (from)`

### 7.8.8.3.11 RefineMetadataFeatureMembership\_Mapping

#### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

#### General Mappings

ToFeatureMembership\_Init  
Mapping

#### Mapping Source

Abstraction

#### Mapping Target

FeatureMembership

#### Owned Mappings

(none)

#### Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureMembership::ownedMemberFeature () : Feature [1]`  
`RefineMetadataReferenceUsage_Mapping.getMapped (from)`

### 7.8.8.3.12 RefineMetadataReferenceUsage\_Mapping

#### Description

Creates a reference usage.

#### General Mappings

ToReferenceUsage\_Init  
Mapping

#### Mapping Source

Abstraction

### Mapping Target

ReferenceUsage

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReferenceUsage::ownedRelationship () : Relationship [0..\*]

```
Set { RefineMetadataReferenceUsageRedefinition_Mapping.getMapped (from) ,  
      RefineMetadataReferenceUsageFeatureValue_Mapping.getMapped (from) }
```

#### 7.8.8.3.13 RefineMetadataReferenceUsageFeatureValue\_Mapping

### Description

Creates a feature value relationship.

### General Mappings

ToFeatureValue\_Init  
Mapping

### Mapping Source

Abstraction

### Mapping Target

FeatureValue

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureValue::value () : Expression [1]

```
LiteralBoolean_Factory.create(true)
```

#### 7.8.8.3.14 RefineMetadataReferenceUsageRedefinition\_Mapping

##### Description

Creates a redefinition relationship for the *redefiningFeature()* and the *redefinedFeature()*.

##### General Mappings

ToRedefinition\_Init  
Mapping

##### Mapping Source

Abstraction

##### Mapping Target

Redefinition

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Redefinition::redefinedFeature () : Feature [1]

```
SysML2::AttributeUsage.allInstances()  
->any(m | m.qualifiedName = 'SysMLv1Library::RefineData::isRefine')
```

#### 7.8.8.3.15 RefineMetadataUsage\_Mapping

##### Description

Create the metadata usage element to annotate a dependency relationship with the information that its SysML v1 mapping source element is a SysML v1 refine relationship.

##### General Mappings

ToMetadataUsage\_Init  
Mapping

##### Mapping Source

Abstraction

##### Mapping Target

MetadataUsage

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `MetadataUsage::ownedRelationship () : Relationship [0..*]`  

```
Set{RefineMetadataUsageFeatureTyping_Mapping.getMapped(from),  
RefineMetadataFeatureMembership_Mapping.getMapped(from)}
```

### 7.8.8.3.16 RefineMetadataUsageFeatureTyping\_Mapping

#### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

#### General Mappings

ToFeatureTyping\_Init  
Mapping

#### Mapping Source

Abstraction

#### Mapping Target

FeatureTyping

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureTyping::type () : Type [1]`  

```
SysML2::MetadataDefinition.allInstances()  
->any(m | m.qualifiedName = 'SysMLv1Library::RefineData')
```

### 7.8.8.3.17 Requirement\_Mapping

#### Description

A SysML::Requirement is mapped to a SysML v2 RequirementUsage.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
requirement <'id1'> SysMLv1Requirement {
  doc /*
        * requirement text
      */

  requirement <'id2'> SysMLv1NestedRequirement {
    doc /*
          * requirement text
        */
  }
}
```

## General Mappings

NamedElementMain\_Mapping  
ToRequirementUsage\_Init

## Mapping Source

Class

## Mapping Target

RequirementUsage

## Owned Mappings

(none)

## Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
Helper.isRequirement(src)
```

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- RequirementUsage::ownedRelationship () : Relationship [0..\*]

```
self.oclAsType(ElementMain_Mapping).ownedRelationship()
->including(CommonReturnParameterReferenceUsageMembership_Mapping.getMapped(from))
->including(RequirementDocumentationMembership_Mapping.getMapped(from))
->including(RequirementSubjectMembership_Mapping.getMapped(from))
```

- RequirementUsage::reqId () : String [1]

```
let stereotype: UML::Stereotype = Helper.getRequirementStereotype(from) in
Helper.getTagValueAsString(from, stereotype.qualifiedName, 'id')
```

### 7.8.8.3.18 RequirementDocumentation\_Mapping

#### Description

The mapping class creates a Comment contained in a Requirement which contains the SysML::Requirements::AbstractRequirement::text property.

#### General Mappings

ToDocumentation\_Init  
Mapping

#### Mapping Source

Class

#### Mapping Target

Documentation

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Documentation::body () : String [1]

```
let stereotype: UML::Stereotype = Helper.getRequirementStereotype(from) in  
Helper.getTagValueAsString(from, stereotype.qualifiedName, 'text')
```

### 7.8.8.3.19 RequirementDocumentationMembership\_Mapping

#### Description

Creates a membership relationship for *memberElement()*.

#### General Mappings

ToOwningMembership\_Init  
Mapping

#### Mapping Source

Class

#### Mapping Target

OwningMembership

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `OwningMembership::ownedMemberElement () : Element [1]`  
`RequirementDocumentation_Mapping.getMapped (from)`

### 7.8.8.3.20 RequirementSubject\_Mapping

#### Description

The mapping class creates the subject reference usage element of the requirement. It is not used since the concept does not exist SysML v1.

#### General Mappings

ToReferenceUsage\_Init  
Mapping

#### Mapping Source

Class

#### Mapping Target

ReferenceUsage

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceUsage::direction () : FeatureDirectionKind [0..1]`  
`KerML::FeatureDirectionKind::_in'`

### 7.8.8.3.21 RequirementSubjectMembership\_Mapping

#### Description



The subject is not used, because it is not a SysML v1 concept, but must be created for a SysML v2 requirement.

## General Mappings

ToParameterMembership\_Init  
Mapping

## Mapping Source

Class

## Mapping Target

SubjectMembership

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- SubjectMembership::ownedMemberParameter () : Feature [0..1]

```
RequirementSubject_Mapping.getMapped(from)
```

### 7.8.8.3.22 Satisfy\_Mapping

#### Description

A SysML::Requirements::Satisfy relationship is mapped to a SysML v2 SatisfyRequirementUsage.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
// satisfy relationship from a block
part def SysMLv1Block {
    part sysMLv1PartProperty;
}
requirement <'ReqId1'> SysMLv1Requirement { doc /* requirement text */ }

ref :SysMLv1Block = all SysMLv1Block {
    satisfy requirement SysMLv1Requirement by self;
}

// satisfy relationship from a part property
satisfy SysMLv1Requirement by sysMLv1BlockUsage.sysMLv1PartProperty {
    sysMLv1BlockUsage : SysMLv1Block;
}
```

## General Mappings

ToOccurrenceUsage\_Init  
Abstraction\_Mapping

### Mapping Source

Abstraction

### Mapping Target

SatisfyRequirementUsage

### Owned Mappings

(none)

### Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
let satisfy: UML::Abstraction = src.oclAsType(UML::Abstraction) in
  if satisfy.oclIsUndefined() then
    false
  else
    Helper.hasStereotypeApplied(satisfy, 'SysML::Requirements::Satisfy')
  endif
```

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- SatisfyRequirementUsage::ownedRelationship () : Relationship [0..\*]

```
let relationships : Set(KerML::Relationship) =
  self.oclAsType(ElementMain_Mapping).ownedRelationship()
->including(SatisfyFeatureTyping_Mapping.getMapped(from))
->including(SatisfySubjectSubjectMembership_Mapping.getMapped(from))
->including(CommonReturnParameterReferenceUsageMembership_Mapping.getMapped(from)) in
if from.client->any(c | true).oclIsKindOf(UML::Property) then
  relationships
  ->including(SatisfyReferenceUsageFeatureMembership_Mapping.getMapped(from))
else
  relationships
endif
```

#### 7.8.8.3.23 SatisfyReferenceUsage\_Mapping

##### Description

Creates a reference usage.

##### General Mappings

ToReferenceUsage\_Init  
Mapping

##### Mapping Source

Abstraction

### Mapping Target

ReferenceUsage

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReferenceUsage::declaredName () : String [0..1]  

```
from.client  
->any(c | true).owner.name.substring(1,1).toLowerCase()  
+ from.client  
->any(c | true).owner.name.  
substring(2,from.client->any(c | true).owner.name.size())  
+ 'SatisfyClientUsage'
```
- ReferenceUsage::ownedRelationship () : Relationship [0..\*]  

```
Set{SatisfyReferenceUsageFeatureTyping_Mapping.getMapped(from) }
```

#### 7.8.8.3.24 SatisfyReferenceUsageFeatureMembership\_Mapping

##### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

##### General Mappings

ToFeatureMembership\_Init  
Mapping

##### Mapping Source

Abstraction

##### Mapping Target

FeatureMembership

##### Owned Mappings

(none)

##### Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureMembership::ownedMemberFeature () : Feature [1]`  
`SatisfyReferenceUsage_Mapping.getMapped(from)`

### 7.8.8.3.25 SatisfySubjectReferenceUsage\_Mapping

#### Description

Creates a reference usage.

#### General Mappings

ToReferenceUsage\_Init  
Mapping

#### Mapping Source

Abstraction

#### Mapping Target

ReferenceUsage

#### Owned Mappings

(none)

#### Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceUsage::ownedRelationship () : Relationship [0..*]`  
`Set{SatisfySubjectReferenceUsageFeatureValue_Mapping.getMapped(from) }`
- `ReferenceUsage::direction () : FeatureDirectionKind [0..1]`  
`KerML::FeatureDirectionKind::_in'`

### 7.8.8.3.26 SatisfySubjectReferenceUsageValue\_Mapping

#### Description

The mapping class create the feature reference expression for the subject of the SatisfyRequirementUsage element.

#### General Mappings

ToFeatureReferenceExpression\_Init  
Mapping

#### Mapping Source

Abstraction

#### Mapping Target

FeatureReferenceExpression

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureReferenceExpression::ownedRelationship () : Relationship [0..\*]  

```
Set{SatisfySubjectReferenceUsageValueOwningMembership_Mapping.getMapped(from),  
ReturnParameterFeatureMembership_Factory.create() }
```

### 7.8.8.3.27 SatisfySubjectReferenceUsageValueFeature\_Mapping

#### Description

The mapping class creates the feature element for the feature reference expression of the subject of the SatisRequirementUsage element.

#### General Mappings

ToFeature\_Init  
Mapping

#### Mapping Source

Abstraction

#### Mapping Target

Feature

#### Owned Mappings

(none)

#### Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Feature::ownedRelationship () : Relationship [0..*]`

```
Set { SatisfySubjectReferenceUsageFeatureChaining_Mapping.getMapped (from) ,  
      SatisfySubjectReferenceUsageValueFeatureChainingProperty_Mapping.getMapped (from) }
```

### 7.8.8.3.28 SatisfySubjectReferenceUsageFeatureChaining\_Mapping

#### Description

The mapping class creates the feature chaining element from SysML v2 SatisfyRequirementUsage's reference usage element.

#### General Mappings

ToFeatureChaining\_Init  
Mapping

#### Mapping Source

Abstraction

#### Mapping Target

FeatureChaining

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureChaining::chainingFeature () : Feature [1]`

```
SatisfyReferenceUsage_Mapping.getMapped (from)
```

### 7.8.8.3.29 SatisfySubjectReferenceUsageValueFeatureChainingProperty\_Mapping

#### Description

The mapping class creates the feature chaining element from the source element of the SysML v1 satisfy relationship.

#### General Mappings

ToFeatureChaining\_Init  
Mapping

### Mapping Source

Abstraction

### Mapping Target

FeatureChaining

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureChaining::chainingFeature () : Feature [1]`  
`from.client->any(c | true)`

### 7.8.8.3.30 SatisfySubjectReferenceUsageFeatureValue\_Mapping

#### Description

Creates a feature value relationship.

#### General Mappings

ToFeatureValue\_Init  
Mapping

#### Mapping Source

Abstraction

#### Mapping Target

FeatureValue

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureValue::value () : Expression [1]`

`SatisfySubjectReferenceUsageValue_Mapping.getMapped (from)`

#### **7.8.8.3.31 SatisfySubjectReferenceUsageValueOwningMembership\_Mapping**

##### **Description**

Creates a owning membership relationship for *ownedMemberElement()*.

##### **General Mappings**

ToOwningMembership\_Init  
Mapping

##### **Mapping Source**

Abstraction

##### **Mapping Target**

OwningMembership

##### **Owned Mappings**

(none)

##### **Applicable filters**

(none)

##### **Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `OwningMembership::ownedMemberElement () : Element [1]`

`SatisfySubjectReferenceUsageValueFeature_Mapping.getMapped (from)`

#### **7.8.8.3.32 SatisfySubjectSubjectMembership\_Mapping**

##### **Description**

Creates a membership relationship for *memberElement()*.

##### **General Mappings**

ToSubjectMembership\_Init  
Mapping

##### **Mapping Source**

Abstraction



### Mapping Target

SubjectMembership

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- SubjectMembership::ownedMemberParameter () : Feature [1]  
`SatisfySubjectReferenceUsage_Mapping.getMapped(from)`

### 7.8.8.3.33 SatisfyFeatureTyping\_Mapping

#### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

#### General Mappings

ToFeatureTyping\_Init  
Mapping

#### Mapping Source

Abstraction

#### Mapping Target

FeatureTyping

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- FeatureTyping::type () : Type [1]  
`from.supplier->any(s | true)`

#### 7.8.8.3.34 SatisfyReferenceUsageFeatureTyping\_Mapping

##### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

##### General Mappings

ToFeatureTyping\_Init  
Mapping

##### Mapping Source

Abstraction

##### Mapping Target

FeatureTyping

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureTyping::type () : Type [1]`  
`from.client->any(c | true).owner`

#### 7.8.8.3.35 TestCaseActivity\_Mapping

##### Description

A `SysML::Requirements::TestCase` applied to an activity is mapped to a SysML v2 `VerificationCaseDefinition` element.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
verification def SysMLv1ActivityTestCase {  
    return verdict : VerificationCases::VerdictKind;  
}
```

##### General Mappings

ActivityAsDefinition\_Mapping

##### Mapping Source

Activity

## Mapping Target

VerificationCaseDefinition

## Owned Mappings

(none)

## Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
Helper.hasStereotypeApplied(src, 'SysML::Requirements::TestCase')
```

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- VerificationCaseDefinition::ownedRelationship () : Relationship [0..\*]

```
let relationships : Set(KerML::Relationship) =
  Helper.activityOwnedRelationship(from) in
let verdictParameter : Set(UML::Parameter) =
  from.ownedElement->select(e | e.ocIsKindOf(UML::Parameter) and
    (e.ocIsType(UML::Parameter).type.name = 'VerdictKind')) in
let parameters : Set(UML::Parameter) =
  ((from.ownedElement->select(e | e.ocIsKindOf(UML::Parameter))) -
    verdictParameter) in
let verifyRelationships : Set(UML::Abstraction) =
  from.clientDependency
  ->select( v |
    Helper.hasStereotypeApplied(v, 'SysML::Requirements::Verify')) in
relationships
->union(parameters->collect(p | ParameterMembership_Mapping.getMapped(p)))
->union(verdictParameter
  ->collect(vp |
    TestCaseActivityReturnParameterMembership_Mapping.getMapped(vp)))
->including(EmptySubjectMembership_Factory.create())
->including(EmptyObjectiveMembership_Factory.create())
->union(verifyRelationships->collect(v | Verify_Mapping.getMapped(v)))
```

### 7.8.8.3.36 TestCaseActivityReturnParameterMembership\_Mapping

#### Description

Creates a membership relationship for *memberElement()*.

#### General Mappings

ParameterMembership\_Mapping

#### Mapping Source

Parameter

#### Mapping Target

ReturnParameterMembership

#### **Owned Mappings**

(none)

#### **Applicable filters**

(none)

### **7.8.8.3.37 TestCaseVerifyObjectiveMembership\_Mapping**

#### **Description**

Creates a the objective membership relationship.

#### **General Mappings**

UniqueMapping  
ToFeatureMembership\_Init

#### **Mapping Source**

Abstraction

#### **Mapping Target**

ObjectiveMembership

#### **Owned Mappings**

(none)

#### **Applicable filters**

(none)

#### **Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ObjectiveMembership::ownedMemberFeature () : Feature [1]  
`TestCaseVerifyObjectiveRequirementUsage_Mapping.getMapped(from)`

### **7.8.8.3.38 TestCaseVerifyObjectiveRequirementUsage\_Mapping**

#### **Description**

The mapping class creates the objective requirements usage of the SysML v2 verification case.

#### **General Mappings**

ToRequirementUsage\_Init  
UniqueMapping

**Mapping Source**

Abstraction

**Mapping Target**

RequirementUsage

**Owned Mappings**

(none)

**Applicable filters**

(none)

**Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- RequirementUsage::ownedRelationship () : Relationship [0..\*]  
`Set{Verify_Mapping.getMapped(from) }`

**7.8.8.3.39 TestCaseVerifyRequirementUsageReferenceSubsetting\_Mapping****Description**

Creates a subsetting relationship.

**General Mappings**

ToSubsetting\_Init  
Mapping

**Mapping Source**

Abstraction

**Mapping Target**

ReferenceSubsetting

**Owned Mappings**

(none)

**Applicable filters**

(none)

**Mapping rules**

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `ReferenceSubsetting::referencedFeature () : Feature [1]`  
`from.supplier->get (0)`

#### 7.8.8.3.40 TestCaseVerifyRequirementUsage\_Mapping

##### Description

The mapping class creates the requirements usage of the SysML v2 test case for the verify relationship.

##### General Mappings

ToUsage\_Init  
Mapping

##### Mapping Source

Abstraction

##### Mapping Target

RequirementUsage

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `RequirementUsage::ownedRelationship () : Relationship [0..*]`  
`Set { TestCaseVerifyRequirementUsageReferenceSubsetting_Mapping.getMapped (from) ,`  
`EmptySubjectMembership_Factory.create () ,`  
`CommonReturnParameterReferenceUsageMembership_Mapping.getMapped (from) }`

#### 7.8.8.3.41 Trace\_Mapping

##### Description

A SysML::Requirements::Trace relationship is mapped to a SysML v2 Dependency relationship annotated with a metadata usage tagging it as a former SysML v1 trace relationship.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
requirement <'idl'> SysMLv1Requirement1 {
    doc /*
        * requirement text
```

```

        */
    }
    requirement <'id2'> SysMLv1Requirement2 {
        doc /*
            * requirement text
            */
    }
    dependency from SysMLv1Requirement1 to SysMLv1Requirement2 {
        @SysMLv1Library::TraceData {isTrace = true;}
    }

```

## General Mappings

Abstraction\_Mapping

## Mapping Source

Abstraction

## Mapping Target

Dependency

## Owned Mappings

(none)

## Applicable filters

This mapping applies only if the following (OCL) condition implemented by the operation *filter(src : Element) : Boolean* is verified:

```
Helper.hasStereotypeApplied(src, 'SysML::Requirements::Trace')
```

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- Dependency::ownedRelationship () : Relationship [0..\*]

```

self.oclAsType(ElementMain_Mapping).ownedRelationship()
->including(TraceAnnotation_Mapping.getMapped(from))

```

### 7.8.8.3.42 TraceAnnotation\_Mapping

#### Description

The mapping class creates the annotation relationship for the SysML::Requirements::Trace mapping.

## General Mappings

ToAnnotation\_Init  
Mapping

## Mapping Source

Abstraction

### Mapping Target

Annotation

### Owned Mappings

(none)

### Applicable filters

(none)

### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Annotation::annotatingElement () : AnnotatingElement [1]`  
`TraceMetadataUsage_Mapping.getMapped (from)`

### 7.8.8.3.43 TraceMetadataFeatureMembership\_Mapping

#### Description

Creates a feature membership relationship for *ownedMemberFeature()*.

#### General Mappings

ToFeatureMembership\_Init  
Mapping

#### Mapping Source

Abstraction

#### Mapping Target

FeatureMembership

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureMembership::ownedMemberFeature () : Feature [1]`  
`TraceMetadataReferenceUsage_Mapping.getMapped (from)`



#### 7.8.8.3.44 TraceMetadataReferenceUsage\_Mapping

##### Description

Creates a reference usage.

##### General Mappings

ToReferenceUsage\_Init  
Mapping

##### Mapping Source

Abstraction

##### Mapping Target

ReferenceUsage

##### Owned Mappings

(none)

##### Applicable filters

(none)

##### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- ReferenceUsage::ownedRelationship () : Relationship [0..\*]

```
Set { TraceMetadataReferenceUsageRedefinition_Mapping.getMapped (from) ,  
      TraceMetadataReferenceUsageFeatureValue_Mapping.getMapped (from) }
```

#### 7.8.8.3.45 TraceMetadataReferenceUsageFeatureValue\_Mapping

##### Description

Creates a feature value relationship.

##### General Mappings

ToFeatureValue\_Init  
Mapping

##### Mapping Source

Abstraction

##### Mapping Target

FeatureValue

##### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureValue::value () : Expression [1]`  
`LiteralBoolean_Factory.create(true)`

### 7.8.8.3.46 TraceMetadataReferenceUsageRedefinition\_Mapping

#### Description

Creates a redefinition relationship for the *redefiningFeature()* and the *redefinedFeature()*.

#### General Mappings

ToRedefinition\_Init  
Mapping

#### Mapping Source

Abstraction

#### Mapping Target

Redefinition

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `Redefinition::redefinedFeature () : Feature [1]`  
`SysML2::AttributeUsage.allInstances()`  
`->any(m | m.qualifiedName = 'SysMLv1Library::TraceData::isTrace')`

### 7.8.8.3.47 TraceMetadataUsage\_Mapping

#### Description

Create the metadata usage element to annotate a dependency relationship with the information that its SysML v1 mapping source element is a SysML v1 trace relationship.

## General Mappings

ToMetadataUsage\_Init  
Mapping

## Mapping Source

Abstraction

## Mapping Target

MetadataUsage

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- MetadataUsage::ownedRelationship () : Relationship [0..\*]  
  
`Set{TraceMetadataUsageFeatureTyping_Mapping.getMapped(from) ,  
TraceMetadataFeatureMembership_Mapping.getMapped(from) }`

### 7.8.8.3.48 TraceMetadataUsageFeatureTyping\_Mapping

#### Description

Creates a feature typing relationship owned by the element *typedFeature()*.

## General Mappings

ToFeatureTyping\_Init  
Mapping

## Mapping Source

Abstraction

## Mapping Target

FeatureTyping

## Owned Mappings

(none)

## Applicable filters

(none)

## Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- `FeatureTyping::type () : Type [1]`

```
SysML2::MetadataDefinition.allInstances()  
->any(m | m.qualifiedName = 'SysMLv1Library::TraceData')
```

### 7.8.8.3.49 Verify\_Mapping

#### Description

A `SysML::Requirements::Verify` relationship is mapped to a SysML v2 `RequirementVerificationMembership` relationship.

The following shows an example of what the textual SysML v2 syntax of the result of the transformation may look like.

```
requirement <'idl'> SysMLv1Requirement {  
    doc /*  
        * requirement text  
        */  
}  
verification def SysMLv1TestCase {  
    objective objective_SysMLv1TestCase {  
        verify SysMLv1Requirement;  
    }  
    return verdict : VerificationCases::VerdictKind;  
}
```

#### General Mappings

ToRelationship\_Init  
Mapping

#### Mapping Source

Abstraction

#### Mapping Target

RequirementVerificationMembership

#### Owned Mappings

(none)

#### Applicable filters

(none)

#### Mapping rules

In addition to the inherited rules, the following lists the mapping class specific mapping rules for the target element properties.

- RequirementVerificationMembership::ownedRelatedElement () : Element [0..\*]  
`Set{TestCaseVerifyRequirementUsage_Mapping.getMapped(from) }`

### **7.8.8.3.50 Model Libraries**

#### **7.8.8.3.50.1 Verdicts**

##### **7.8.8.3.50.1.1 VerdictKind**

The enumeration VerdictKind is mapped to the SysML v2 VerificationCases::VerdictKind model library element.