An OMG® Systems Modeling Publication

OMG Systems Modeling Language™ (SysML®)

Version 2.0 Beta

Part 2: SysML v1 to SysML v2 Transformation

OMG Document Number: tbd

Date: tbd

Standard document URL: https://www.omg.org/spec/SysML/2.0

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 20
7.4 Initializers 23
7.4.1 Overview 23
7.4.2 Mapping Specifications 23
7.4.2.1 KerML Initializers 23
7.4.2.1.1 ToAnnotatingElement_Init 23
7.4.2.1.2 ToAnnotation_Init 23
7.4.2.1.3 ToAssociation_Init 24
7.4.2.1.4 ToBehavior_Init 24
7.4.2.1.5 ToBindingConnector_Init 24
7.4.2.1.6 ToClassifier_Init 25
7.4.2.1.7 ToComment_Init 25
7.4.2.1.8 ToConjugation_Init 25
7.4.2.1.9 ToConnector_Init 26
7.4.2.1.10 ToDocumentation_Init 26
7.4.2.1.11 ToElement_Init 26
7.4.2.1.12 ToEndFeatureMembership_Init 27
7.4.2.1.13 ToExpression_Init 27
7.4.2.1.14 ToFeature_Init 28
7.4.2.1.15 ToFeatureChainExpression_Init 28
7.4.2.1.16 ToFeatureChaining_Init 29
7.4.2.1.17 ToFeatureMembership_Init 29
7.4.2.1.18 ToFeatureReferenceExpression_Init 29
7.4.2.1.19 ToFeatureTyping_Init 30
7.4.2.1.20 ToFeatureValue_Init 30
7.4.2.1.21 ToFunction_Init 31
7.4.2.1.22 ToImport_Init 31
7.4.2.1.23 ToInteraction_Init 31
7.4.2.1.24 ToInvocationExpression_Init 32
7.4.2.1.25 ToItemFlow_Init 32
7.4.2.1.26 ToMembership_Init 32
7.4.2.1.27 ToMembershipImport_Init 33
7.4.2.1.28 ToNamespace_Init 33
7.4.2.1.29 ToNamespaceImport_Init 33

7.4.2.1.30 ToOperatorExpression_Init 34
7.4.2.1.31 ToOwningMembership_Init 34
7.4.2.1.32 ToPackage_Init 34
7.4.2.1.33 ToParameterMembership_Init 35
7.4.2.1.34 ToPredicate_Init 35
7.4.2.1.35 ToRedefinition_Init 35
7.4.2.1.36 ToReferenceSubsetting_Init 36
7.4.2.1.37 ToRelationship_Init 36
7.4.2.1.38 ToReturnParameterMembership_Init 36
7.4.2.1.39 ToSpecialization_Init 37
7.4.2.1.40 ToStep_Init 37
7.4.2.1.41 ToSubclassification_Init 37
7.4.2.1.42 ToSubsetting_Init 38
7.4.2.1.43 ToSuccession_Init 38
7.4.2.1.44 ToSuccessionItemFlow_Init 39
7.4.2.1.45 ToTextualRepresentation_Init 39
7.4.2.1.46 ToType_Init 39
7.4.2.1.47 ToTypeFeaturing_Init 40
7.4.2.2 System Initializers 40
7.4.2.2.1 ToActionUsage_Init 40
7.4.2.2.2 ToActorMembership_Init 40
7.4.2.2.3 ToAssignmentActionUsage_Init 41
7.4.2.2.4 ToBindingConnectorAsUsage_Init 41
7.4.2.2.5 ToCalculationUsage_Init 41
7.4.2.2.6 ToConjugatedPortDefinition_Init 42
7.4.2.2.7 ToConjugatedPortTyping_Init 42
7.4.2.2.8 ToConnectionUsage_Init 42
7.4.2.2.9 ToConstraintDefinition_Init 42
7.4.2.2.10 ToConstraintUsage_Init 43
7.4.2.2.11 ToDefinition_Init 43
7.4.2.2.12 ToEventOccurerenceUsage_Init 43
7.4.2.2.13 ToFlowUsage_Init 44
7.4.2.2.14 ToItemDefinition_Init 44
7.4.2.2.15 ToItemFeature_Init 44
7.4.2.2.16 ToItemUsage_Init 45
7.4.2.2.17 ToMetadataUsage_Init 45
7.4.2.2.18 ToObjectiveMembership_Init 45
7.4.2.2.19 ToOccurenceDefinition_Init 45
7.4.2.2.20 ToOccurrenceUsage_Init 46
7.4.2.2.21 ToPartUsage_Init 46
7.4.2.2.22 ToPerformActionUsage_Init 46
7.4.2.2.23 ToPortConjugation_Init 47
7.4.2.2.24 ToPortDefinition_Init 47
7.4.2.2.25 ToReferenceUsage_Init 47
7.4.2.2.26 ToRequirementUsage_Init 48
7.4.2.2.27 ToStateSubactionMembership_Init 48
7.4.2.2.28 ToStateUsage_Init 48
7.4.2.2.29 ToSubjectMembership_Init 48
7.4.2.2.30 ToTransitionUsage_Init 49
7.4.2.2.31 ToTriggerInvocationExpression_Init 49
7.4.2.2.32 ToUsage_Init 49
7.5 Factories 50
7.5.1 Overview 50
7.5.2 Mapping Specifications 50
7.5.2.1 BindingConnector_Factory 50
7.5.2.2 BindingMembership_Factory 50

7.5.2.3 Documentation_Factory 50
7.5.2.4 DocumentationMembership_Factory 51
7.5.2.5 EmptyObjectiveMembership_Factory 51
7.5.2.6 EmptyRequirementUsage_Factory 52
7.5.2.7 EndFeature_Factory 52
7.5.2.8 EndFeatureMembership_Factory 52
7.5.2.9 EmptySubject_Factory 52
7.5.2.10 EmptySubjectMembership_Factory 53
7.5.2.11 FeatureBooleanValue_Factory 53
7.5.2.12 FeatureStringValue_Factory 53
7.5.2.13 FeatureValue_Factory 53
7.5.2.14 LiteralBoolean_Factory 54
7.5.2.15 LiteralInteger_Factory 54
7.5.2.16 LiteralNull_Factory 55
7.5.2.17 LiteralRational_Factory 55
7.5.2.18 LiteralString_Factory 56
7.5.2.19 LowerBound_Factory 56
7.5.2.20 MetaclassQNameToMetadataFeature_Factory 57
7.5.2.21 MetadataFeature_Factory 58
7.5.2.22 MetadataOwningMembership_Factory 58
7.5.2.23 MultiplicityElement_Factory 58
7.5.2.24 MultiplicityLowerBoundMembership_Factory 59
7.5.2.25 MultiplicityMembership_Factory 59
7.5.2.26 MultiplicityUpperBoundMembership_Factory 60
7.5.2.27 ParameterMembership_Factory 61
7.5.2.28 QNameToFeatureTyping_Factory 61
7.5.2.29 ReferenceUsage_Factory 62
7.5.2.30 ReferenceUsageAsParameter_Factory 62
7.5.2.31 ReferenceUsageMembership_Factory 63
7.5.2.32 ReturnParameterMembership_Factory 63
7.5.2.33 Subsetting_Factory 64
7.5.2.34 TypeToFeatureTyping_Factory 64
7.5.2.35 UntypedReturnParameterFeature_Factory 64
7.5.2.36 UntypedReturnParameterMembership_Factory 65
7.5.2.37 UpperBound_Factory 65
7.6 Mappings from UML4SysML metaclasses 65
7.6.1 Overview 66
7.6.2 Actions 66
7.6.2.1 Overview 66
7.6.2.2 Mapping Specifications 67
7.6.2.2.1 Accept Event Actions 68
7.6.2.2.1.1 AcceptActionUsageMembership_Mapping 68
7.6.2.2.1.2 AcceptCallAction_Mapping 68
7.6.2.2.1.3 AcceptEventActionCommon_Mapping 69
7.6.2.2.1.4 AcceptEventActionToPLParameter_Mapping 70
7.6.2.2.1.5 PayloadArgMembership_Mapping 71
7.6.2.2.1.6 PayloadParameterMembership_Mapping 71
7.6.2.2.1.7 ReplyAction_Mapping 72
7.6.2.2.1.8 UnmarshallAction_Mapping 72
7.6.2.2.1.9 AcceptEventAction_Mapping 73
7.6.2.2.2 Actions 73
7.6.2.2.2.1 ActionFeatureMembership_Mapping 73
7.6.2.2.2.2 ActionParameterMembership_Mapping 74
7.6.2.2.2.3 CommonAction_Mapping 74
7.6.2.2.2.4 OpaqueAction_Mapping 75
7.6.2.2.2.5 OpaqueActionSpecification_Mapping 76

7.6.2.2.2.6 OpaqueActionSpecificationMembership_Mapping 76
7.6.2.2.2.7 Pin_Mapping 77
7.6.2.2.2.8 PinToParameter_Mapping 78
7.6.2.2.2.9 PinToParameterMembership_Mapping 79
7.6.2.2.2.10 PinToReturnParameterMembership_Mapping 79
7.6.2.2.2.11 ValuePin_Mapping 80
7.6.2.2.2.12 ValuePinUntyped_Mapping 81
7.6.2.2.3 Invocation Actions 81
7.6.2.2.3.1 BroadcastSignalAction_Mapping 82
7.6.2.2.3.2 CallAction_Mapping 82
7.6.2.2.3.3 CallBehaviorAction_Mapping 82
7.6.2.2.3.4 CallOperationAction_Mapping 83
7.6.2.2.3.5 InvocationAction_Mapping 84
7.6.2.2.3.6 SendObjectAction_Mapping 84
7.6.2.2.3.7 SendSignalAction_Mapping 85
7.6.2.2.4 Link Actions 85
7.6.2.2.4.1 ClearAssociationAction_Mapping 85
7.6.2.2.4.2 CreateLinkAction_Mapping 86
7.6.2.2.4.3 CreateLinkObjectAction_Mapping 87
7.6.2.2.4.4 DestroyLinkAction_Mapping 87
7.6.2.2.4.5 ReadLinkAction_Mapping 88
7.6.2.2.4.6 ReadLinkObjectEndAction_Mapping 88
7.6.2.2.5 Object Actions 89
7.6.2.2.5.1 CreateObjectAction_Mapping 89
7.6.2.2.5.2 DestroyObjectAction_Mapping 89
7.6.2.2.5.3 ReadIsClassifiedObjectAction_Mapping 90
7.6.2.2.5.4 ReadExtentAction_Mapping 91
7.6.2.2.5.5 ReadSelfAction_Mapping 91
7.6.2.2.5.6 ReclassifyObjectAction_Mapping 92
7.6.2.2.5.7 StartClassifierBehaviorAction_Mapping 92
7.6.2.2.5.8 StartObjectBehaviorAction_Mapping 93
7.6.2.2.5.9 TestIdentityAction_Mapping 93
7.6.2.2.5.10 TIAOperatorExpression_Mapping 94
7.6.2.2.5.11 TIAResultExpressionMembership_Mapping 95
7.6.2.2.5.12 ValueSpecificationAction_Mapping 95
7.6.2.2.6 Other Actions 96
7.6.2.2.6.1 RaiseExceptionAction_Mapping 96
7.6.2.2.6.2 ReduceAction_Mapping 96
7.6.2.2.7 Structural Feature Actions 97
7.6.2.2.7.1 AddStructuralFeatureValueAction_Mapping 97
7.6.2.2.7.2 ClearStructuralFeatureAction_Mapping 97
7.6.2.2.7.3 ReadStructuralFeatureAction_Mapping 98
7.6.2.2.7.4 RemoveStructuralFeatureValueAction_Mapping 98
7.6.2.2.7.5 StructuralFeatureAction_Mapping 99
7.6.2.2.7.6 WriteStructuralFeatureValueAction_Mapping 99
7.6.2.2.8 Structured Actions 100
7.6.2.2.8.1 LoopNode_Mapping 100
7.6.2.2.8.2 SequenceNode_Mapping 100
7.6.2.2.8.3 StructuredActivityNode_Mapping 101
7.6.2.2.9 Variable Actions 101
7.6.2.2.9.1 AddVariableValueAction_Mapping 101
7.6.2.2.9.2 ClearVariableAction_Mapping 102
7.6.2.2.9.3 ReadVariableAction_Mapping 103
7.6.2.2.9.4 RemoveVariableValueAction_Mapping 104
7.6.3 Activities 104
7.6.3.1 Overview 104

7.6.3.2 Mapping Specifications 105
7.6.3.2.1 ActivityAsDefinition_Mapping 105
7.6.3.2.2 ActivityAsDefinitionCommon_Mapping 106
7.6.3.2.3 ActivityEdge_Mapping 107
7.6.3.2.4 ActivityFinalNode_Mapping 109
7.6.3.2.5 CentralBufferNode_Mapping 109
7.6.3.2.6 CommonNode_Mapping 110
7.6.3.2.7 CommonVariable_Mapping 110
7.6.3.2.8 ControlFlowSuccessionAsUsage_Mapping 111
7.6.3.2.9 DataStoreNode_Mapping 112
7.6.3.2.10 DecisionNode_Mapping 112
7.6.3.2.11 ForkNode_Mapping 113
7.6.3.2.12 JoinNode_Mapping 114
7.6.3.2.13 MergeNode_Mapping 115
7.6.3.2.14 NodeMembership_Mapping 116
7.6.3.2.15 ObjectFlow_Mapping 116
7.6.3.2.16 ObjectFlowGuardSuccessionTargetEndSubsetting_Mapping 117
7.6.3.2.17 VariableAttribute_Mapping 118
7.6.3.2.18 VariableFeatureMembership_Mapping 118
7.6.3.2.19 VariableFeatureTyping_Mapping 119
7.6.3.2.20 VariableItem_Mapping 119
7.6.4 Classification 120
7.6.4.1 Overview 120
7.6.4.2 Mapping Specifications 120
7.6.4.2.1 BehavioralFeature_Mapping 120
7.6.4.2.2 Classifier_Mapping 121
7.6.4.2.3 ClassifierToFeature_Mapping 121
7.6.4.2.4 DefaultValue_Mapping 122
7.6.4.2.5 ElementFeatureMembership_Mapping 123
7.6.4.2.6 FromPropertyBasic_Mapping 124
7.6.4.2.7 Generalization_Mapping 125
7.6.4.2.8 InstanceSpecificationCommon_Mapping 126
7.6.4.2.9 InstanceSpecificationLink_Mapping 127
7.6.4.2.10 InstanceSpecification_Mapping 127
7.6.4.2.11 InstanceValue_Mapping 128
7.6.4.2.12 MultiplicityMembership_Mapping 129
7.6.4.2.13 Operation_Mapping 130
7.6.4.2.14 PackageableElementFeatureMembership_Mapping 130
7.6.4.2.15 Parameter_Mapping 131
7.6.4.2.16 ParameterMembership_Mapping 133
7.6.4.2.17 ParameterReferenceUsage_Mapping 133
7.6.4.2.18 ParameterSet_Mapping 134
7.6.4.2.19 ParameterSetMembership_Mapping 135
7.6.4.2.20 ParameterSetParameterFeatureMembership_Mapping 136
7.6.4.2.21 PropertyCommon_Mapping 137
7.6.4.2.22 PropertyLike_Mapping 138
7.6.4.2.23 PropertyLikeNamed_Mapping 138
7.6.4.2.24 PropertyRedefinition_Mapping 138
7.6.4.2.25 PropertySubsetting_Mapping 139
7.6.4.2.26 PropertyToReferenceUsage_Mapping 139
7.6.4.2.27 PropertyTypedByClassInterface_Mapping 140
7.6.4.2.28 PropertyUntyped_Mapping 141
7.6.4.2.29 RegularPropertyFeatureMembership_Mapping 142
7.6.4.2.30 Slot_Mapping 142
7.6.4.2.31 SlotMembership_Mapping 143
7.6.4.2.32 StructuralFeatureMembership_Mapping 143

7.6.4.2.33 TypeToFeatureTyping_Mapping 143
7.6.5 CommonBehavior 144
7.6.5.1 Overview 144
7.6.5.2 Mapping Specifications 144
7.6.5.2.1 Behavior_Mapping 144
7.6.5.2.2 ChangeEvent_Mapping 145
7.6.5.2.3 Event_Mapping 146
7.6.5.2.4 OpaqueBehavior_Mapping 146
7.6.5.2.5 OpaqueBehaviorMembership_Mapping 147
7.6.5.2.6 OpaqueBehaviorSpecification_Mapping 148
7.6.5.2.7 TimeEvent_Mapping 148
7.6.5.2.8 Trigger_Mapping 149
7.6.6 CommonStructure 149
7.6.6.1 Overview 149
7.6.6.2 Mapping Specifications 150
7.6.6.2.1 Abstraction_Mapping 150
7.6.6.2.2 Comment_Mapping 150
7.6.6.2.3 CommentAnnotation_Mapping 151
7.6.6.2.4 Constraint_Mapping 152
7.6.6.2.5 Dependency_Mapping 153
7.6.6.2.6 DirectedRelationship_Mapping 154
7.6.6.2.7 ElementMain_Mapping 154
7.6.6.2.8 ElementMembership_Mapping 155
7.6.6.2.9 ElementOwningMembership_Mapping 156
7.6.6.2.10 ElementUnique_Mapping 156
7.6.6.2.11 NamedElementMain_Mapping 157
7.6.6.2.12 NamedElementUnique_Mapping 157
7.6.6.2.13 Namespace_Mapping 158
7.6.6.2.14 Relationship_Mapping 158
7.6.7 InformationFlows 159
7.6.7.1 Overview 159
7.6.7.2 Mapping Specifications 159
7.6.7.2.1 InformationItem_Mapping 159
7.6.8 Interactions 160
7.6.8.1 Overview 160
7.6.8.2 Mapping Specifications 160
7.6.8.2.1 ActionExecutionSpecification_Mapping 160
7.6.8.2.2 BehaviorExecutionSpecification_Mapping 161
7.6.8.2.3 CombinedFragment_Mapping 161
7.6.8.2.4 ExecutionSpecificationMembership_Mapping 162
7.6.8.2.5 Interaction_Mapping 162
7.6.8.2.6 InteractionOperand_Mapping 163
7.6.8.2.7 InteractionUse_Mapping 164
7.6.8.2.8 LifelinePartUsage_Mapping 164
7.6.8.2.9 Message_Mapping 165
7.6.8.2.10 StateInvariant_Mapping 165
7.6.9 Packages 166
7.6.9.1 Overview 166
7.6.9.2 Mapping Specifications 166
7.6.9.2.1 ElementImport_Mapping 166
7.6.9.2.2 Model_Mapping 167
7.6.9.2.3 Package_Mapping 168
7.6.9.2.4 PackageImport_Mapping 169
7.6.9.2.5 Profile_Mapping 170
7.6.9.2.6 Stereotype_Mapping 170

7.6.10 SimpleClassifiers....171
7.6.10.1 Overview....171
7.6.10.2 Mapping Specifications....171
7.6.10.2.1 Attribute_Mapping....171
7.6.10.2.2 BehavioredClassifier_Mapping....172
7.6.10.2.3 BehaviorToActionUsage_Mapping....173
7.6.10.2.4 DataType_Mapping....173
7.6.10.2.5 Enumeration_Mapping....174
7.6.10.2.6 EnumerationLiteral_Mapping....175
7.6.10.2.7 EnumerationVariantMembership_Mapping....175
7.6.10.2.8 Interface_Mapping....176
7.6.10.2.9 InterfaceConjugatedPortDefinition_Mapping....176
7.6.10.2.10 InterfaceConjugatedPortDefinitionMembership_Mapping....177
7.6.10.2.11 InterfacePortConjugation_Mapping....178
7.6.10.2.12 InterfaceRealization_Mapping....178
7.6.10.2.13 PrimitiveType_Mapping....179
7.6.10.2.14 PrimitiveType2_Mapping....180
7.6.10.2.15 Reception_Mapping....180
7.6.10.2.16 Signal_Mapping....181
7.6.11 StateMachines....181
7.6.11.1 Overview....181
7.6.11.2 Mapping Specifications....181
7.6.11.2.1 CommonPseudostate_Mapping....182
7.6.11.2.2 ConnectionPointReference_Mapping....182
7.6.11.2.3 DoBehaviorStateSubactionMembership_Mapping....183
7.6.11.2.4 EntryBehaviorStateSubactionMembership_Mapping....183
7.6.11.2.5 ExitBehaviorStateSubactionMembership_Mapping....184
7.6.11.2.6 FinalState_Mapping....185
7.6.11.2.7 InitialState_Mapping....185
7.6.11.2.8 PseudoState_Mapping....186
7.6.11.2.9 Region_Mapping....186
7.6.11.2.10 State_Mapping....187
7.6.11.2.11 StateBehaviorStateSubactionMembership_Mapping....188
7.6.11.2.12 StateDefinition_Mapping....188
7.6.11.2.13 Transition_Mapping....189
7.6.11.2.14 TransitionSuccession_Mapping....190
7.6.12 StructuredClassifiers....190
7.6.12.1 Overview....190
7.6.12.2 Mapping Specifications....191
7.6.12.2.1 Association_Mapping....191
7.6.12.2.2 Class_Mapping....191
7.6.12.2.3 ClassCommon_Mapping....192
7.6.12.2.4 ConnectionDefEnd_Mapping....193
7.6.12.2.5 ConnectionEndToSubsetting_Mapping....194
7.6.12.2.6 Connector_Mapping....194
7.6.12.2.7 ConnectorEndToFeatureCommon_Mapping....195
7.6.12.2.8 ConnectorEndToOwnedFeature_Mapping....196
7.6.12.2.9 ConnectorEndToSubsettedFeature_Mapping....197
7.6.12.2.10 ConnectorType_Mapping....198
7.6.12.2.11 CrossSubsetting_Mapping....198
7.6.12.2.12 End_Mapping....199
7.6.12.2.13 EndFeatureMembership_Mapping....200
7.6.12.2.14 NonOwnedEnd_Mapping....201
7.6.12.2.15 OwnedEnd_Mapping....201
7.6.12.2.16 Port_Mapping....202
7.6.12.2.17 PropertyToFeatureChaining_Mapping....203

7.6.13 UseCases 203
7.6.13.1 Overview 203
7.6.13.2 Mapping Specifications 204
7.6.13.2.1 Actor_Mapping 204
7.6.13.2.2 Include_Mapping 204
7.6.13.2.3 IncludeMembership_Mapping 205
7.6.13.2.4 UseCase_Mapping 205
7.6.13.2.5 UseCaseActorMembership_Mapping 206
7.6.14 Values 207
7.6.14.1 Overview 207
7.6.14.2 Mapping Specifications 208
7.6.14.2.1 Expression_Mapping 208
7.6.14.2.2 LiteralBoolean_Mapping 208
7.6.14.2.3 LiteralInteger_Mapping 209
7.6.14.2.4 LiteralNull_Mapping 209
7.6.14.2.5 LiteralReal_Mapping 210
7.6.14.2.6 LiteralSpecificationCommon_Mapping 210
7.6.14.2.7 LiteralString_Mapping 211
7.6.14.2.8 LiteralUnlimitedUnbounded_Mapping 211
7.6.14.2.9 LiteralUnlimitedInteger_Mapping 212
7.6.14.2.10 OpaqueExpression_Mapping 213
7.6.14.2.11 OpaqueExpressionLike_Mapping 213
7.6.14.2.12 OpaqueExpressionSpecification_Mapping 214
7.6.14.2.13 ReturnParameterMembership_Mapping 214
7.6.14.2.14 TimeExpression_Mapping 215
7.6.14.2.15 TypeToReturnParameterFeature_Mapping 215
7.6.14.2.16 ValueSpecification_Mapping 216
7.6.14.2.17 ValueSpecificationToResult_Mapping 217
7.6.14.2.18 ValueSpecificationToResultMembership_Mapping 218
7.7 Mappings from SysML v1.7 stereotypes 218
7.7.1 Overview 218
7.7.2 Activities 218
7.7.2.1 Overview 218
7.7.2.2 Mapping Specifications 219
7.7.2.2.1 Model Libraries 219
7.7.2.2.1.1 ControlValues 219
7.7.2.2.1.1.1 ControlValueKind 219
7.7.3 Allocations 219
7.7.3.1 Overview 219
7.7.3.2 Mapping Specifications 219
7.7.3.2.1 AllocationUsage_Mapping 219
7.7.4 Blocks 220
7.7.4.1 Overview 220
7.7.4.2 Mapping Specifications 221
7.7.4.2.1 BindingConnector_Mapping 221
7.7.4.2.2 Block_Mapping 221
7.7.5 ConstraintBlocks 222
7.7.5.1 Overview 222
7.7.5.2 Mapping Specifications 222
7.7.5.2.1 ConstraintBlock_Mapping 222
7.7.5.2.2 ConstraintParameter_Mapping 223
7.7.5.2.3 ConstraintParameterMembership_Mapping 224
7.7.6 Model Elements 225
7.7.6.1 Overview 225
7.7.6.2 Mapping Specifications 225
7.7.6.2.1 ElementGroup_Mapping 225

7.7.6.2.2 Stakeholder_Mapping 226
7.7.7 PortsAndFlows 227
7.7.7.1 Overview 227
7.7.7.2 Mapping Specifications 228
7.7.7.2.1 AcceptChangeStructuralFeatureEventAction_Mapping 228
7.7.7.2.2 CommonInterfaceBlock_Mapping 228
7.7.7.2.3 ConjugatedInterfaceBlock_Mapping 229
7.7.7.2.4 FullPort_Mapping 229
7.7.7.2.5 InterfaceBlock_Mapping 230
7.7.7.2.6 PortConjugation_Mapping 231
7.7.7.2.7 TriggerToReceiver_Mapping 231
7.7.8 Requirements 232
7.7.8.1 Overview 232
7.7.8.2 Mapping Specifications 232
7.7.8.2.1 DeriveReqt_Mapping 232
7.7.8.2.2 Refine_Mapping 233
7.7.8.2.3 Requirement_Mapping 234
7.7.8.2.4 Satisfy_Mapping 235
7.7.8.2.5 TestCaseActivity_Mapping 236
7.7.8.2.6 TestCaseVerifyRequirementUsage_Mapping 237
7.7.8.2.7 Trace_Mapping 238
7.7.8.2.8 Verify_Mapping 239
7.7.8.2.9 Model Libraries 240
7.7.8.2.9.1 Verdicts 240
7.7.8.2.9.1.1 VerdictKind 240

List of Tables

1. List of High-Level Action Mappings 66
2. List of High-Level Activities Mappings 104
3. List of High-Level Classification Mappings 120
4. List of High-Level CommonBehavior Mappings 144
5. List of High-Level CommonStructure Mappings 149
6. List of High-Level InformationFlows Mappings 159
7. List of High-Level Interactions Mappings 160
8. List of High-Level Packages Mappings 166
9. List of High-Level SimpleClassifiers Mappings 171
10. List of High-Level StateMachines Mappings 181
11. List of High-Level StructuredClassifiers Mappings 190
12. List of High-Level UseCases Mappings 203
13. List of High-Level Values Mappings 207
14. List of High-Level Activities Mappings 218
15. List of High-Level Allocations Mappings 219
16. List of High-Level Allocations Mappings 220
17. List of High-Level SysML v1 ConstraintBlocks Mappings 222
18. List of High-Level SysML v1 ModelElements Mappings 225
19. List of High-Level Ports & Flows Mappings 227
20. List of High-Level Requirements Mappings 232

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
- Thematix 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 Galey, 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 theirs 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 the 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..*] (not typed)

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.oclIsTypeOf(UML::ActionInputPin)) in
let triggers: Set(UML::Element) =
    src.ownedElement->select(e | e.oclIsKindOf(UML::Trigger)) in
let toElementFMS: Set(UML::Element) =
    src.ownedElement->select(e | e.oclIsKindOf(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)))
```

- createUUID () : String [1]
  Creates a UUID. The specification is implementation-specific and therefore cannot provided here.

- 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.oclIsKindOf(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 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

```
 KerML::MultiplicityRange.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.oclIsUndefined()
  or t.name = ''
  or t.name.oclIsUndefined()
  or t.qualifiedName.oclIsUndefined() 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 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 return "true" for a given model element, this element shall be consider 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'))
```

- 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.oclIsKindOf(UML::Pseudostate) and
    e.oclAsType(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}

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 ToBindingConnector_Init

Description

Initializes the properties of the SysML v2 element BindingConnector.

General Classes

- ToConnector_Init (from KerMLInitializers)

Association Ends

- to : BindingConnector [1]
  {redefines: ToConnector_Init::to}

7.4.2.1.6 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}

7.4.2.1.7 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
```

7.4.2.1.8 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}

7.4.2.1.9 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
```

7.4.2.1.10 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}

7.4.2.1.11 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.oclIsKindOf(factory.srcType) and to.oclIsKindOf(factory.tgtType)
  ```

7.4.2.1.12 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}

7.4.2.1.13 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}

7.4.2.1.14 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] {redefines isDerived}

  ```
  false
  ```

- isEnd () : Boolean [1]

  ```
  false
  ```

- isOrdered () : Boolean [1]

  ```
  false
  ```

- isPortion () : Boolean [1]

  ```
  false
  ```

- isReadOnly () : Boolean [1]

  ```
  false
  ```

- isUnique () : Boolean [1]

  ```
  true
  ```

7.4.2.1.15 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}

7.4.2.1.16 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}

7.4.2.1.17 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

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

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

7.4.2.1.18 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}

7.4.2.1.19 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}

7.4.2.1.20 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]{abstract}
- isDefault () : Boolean [1]

  ```
  false
  ```

- isInitial () : Boolean [1]

  ```
  false
  ```

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

  ```
  Set{self.value()}->reject(v | v.oclIsUndefined())
  ```

- value () : Expression [1]{abstract}

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 ToItemFlow_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}

7.4.2.1.26 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
  ```

7.4.2.1.27 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}

7.4.2.1.28 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}

7.4.2.1.29 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}

7.4.2.1.30 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}

7.4.2.1.31 ToOwningMembership_Init

Description

Initializes the properties of the SysML v2 element OwningMembership.

General Classes

- ToMembership_Init (from KerMLInitializers)

Association Ends

- to : OwningMembership [1]
  {redefines: ToMembership_Init::to}

Operations

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

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

7.4.2.1.32 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}

7.4.2.1.33 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}

Operations

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

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

7.4.2.1.34 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}

7.4.2.1.35 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}

7.4.2.1.36 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}

7.4.2.1.37 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{}
```

7.4.2.1.38 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
  ```

7.4.2.1.39 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}

7.4.2.1.40 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}

7.4.2.1.41 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] {redefines specific}

  ```
  null
  ```

- superclassifier () : Classifier [1] {redefines general}

  ```
  null
  ```

7.4.2.1.42 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{}
  ```

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

7.4.2.1.43 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}

7.4.2.1.44 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}

7.4.2.1.45 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}

7.4.2.1.46 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
  ```

7.4.2.1.47 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}
  {redefines: ToExpression_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 ToEventOccurerenceUsage_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 ToItemDefinition_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 ToItemFeature_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 ToItemUsage_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 ToOccurenceDefinition_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 BindingConnector_Factory

Description

General Classes

- Factory (from Foundations)
- ToBindingConnector_Init (from KerMLInitializers)

Association Ends

- feature1 : Feature [1]
- feature2 : Feature [1]

Operations

- create (in feature1 : Feature, in feature2 : Feature) : BindingConnector [1]
- ownedRelationship () : Relationship [0..*] {redefines ownedRelationship}

```
Set{}
```

7.5.2.2 BindingMembership_Factory

Description

General Classes

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

Association Ends

- feature1 : Feature [1]
- feature2 : Feature [1]

Operations

- create (in feature1 : Feature, in feature2 : Feature) : FeatureMembership [1]
- memberElement () : Element [1] {redefines memberElement}

```
BindingConnector_Factory.create(feature1, feature2)
```

7.5.2.3 Documentation_Factory

Description

General Classes

- Factory (from Foundations)
- ToDocumentation_Init (from KerMLInitializers)

Association Ends

- text : String [1]

Operations

- body () : String [1] {redefines body}

```
text
```

- create (in text : String) : LiteralString [1]

7.5.2.4 DocumentationMembership_Factory

Description

General Classes

- Factory (from Foundations)
- ToOwningMembership_Init (from KerMLInitializers)

Association Ends

- text : String [1]

Operations

- create (in text : String) : LiteralString [1]
- memberElement () : Element [1] {redefines memberElement}

```
Documentation_Factory.create(text)
```

7.5.2.5 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]
- memberElement () : Element [1] {redefines memberElement}

```
EmptyRequirementUsage_Factory.create()
```

7.5.2.6 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(),
UntypedReturnParameterMembership_Factory.create()}
```

7.5.2.7 EndFeature_Factory

Description

General Classes

- ReferenceUsage_Factory (from Factories)

Operations

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

```
 true
```

7.5.2.8 EndFeatureMembership_Factory

Description

General Classes

- ReferenceUsageMembership_Factory (from Factories)

Operations

- memberElement () : Feature [1] {redefines memberElement}

```
 EndFeature_Factory.create(feature, refUsageName, redefine, value)
```

7.5.2.9 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'
```

7.5.2.10 EmptySubjectMembership_Factory

Description

Factory class to create a memberhsip 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]
- memberElement () : Feature [1] {redefines memberElement}

```
EmptySubject_Factory.create()
```

7.5.2.11 FeatureBooleanValue_Factory

7.5.2.12 FeatureStringValue_Factory

Description

General Classes

- FeatureValue_Factory (from Factories)

Association Ends

- value : String [1]
  {redefines: FeatureValue_Factory::value}

Operations

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

```
LiteralString_Factory.create(value)
```

7.5.2.13 FeatureValue_Factory

Description

General Classes

- Factory (from Foundations)

- ToFeatureValue_Init (from KerMLInitializers)

Attributes

- value [1] (not typed)

7.5.2.14 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{

    ReturnParameterMembership_Factory.create(

        Helper.getScalarValueTypeByName('Boolean')

    )

}
```

7.5.2.15 LiteralInteger_Factory

Description

Factory class to create a LiteralRational element.

General Classes

- Factory (from Foundations)
- ToExpression_Init (from KerMLInitializers)

Association Ends

- integer : Integer [1]
- to : LiteralInteger [1]
  {redefines: ToExpression_Init::to}

Operations

- create (in integer : Integer) : LiteralReal [1]

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

```
Set{
    ReturnParameterMembership_Factory.create(
        Helper.getScalarValueTypeByName('Integer')
    )
}
```

7.5.2.16 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{ReturnParameterMembership_Factory.create()}
```

7.5.2.17 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{
    ReturnParameterMembership_Factory.create(
        Helper.getScalarValueTypeByName('Rational')
    )
}
```

7.5.2.18 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{
    ReturnParameterMembership_Factory.create(
        Helper.getScalarValueTypeByName('String')
    )
}
```

7.5.2.19 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{
    ReturnParameterMembership_Factory.create(
        'result',
        Helper.getScalarValueTypeByName('Integer')
    )
}
```

- value () : Integer [1]

```
lowerValue
```

7.5.2.20 MetaclassQNameToMetadataFeature_Factory

Description

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

General Classes

- MetadataFeature_Factory (from Factories)

Attributes

- featureValues [0..*] (not typed) {ordered, nonunique}

Association Ends

- featureNames : String [0..*] {ordered}
- typeQName : String [1]

Operations

- create (in typeQName : String, in featureNames : String, in featureValues) : MetadataFeature [1] {redefines create}
- ownedRelationship () : Relationship [0..*] {redefines ownedRelationship}

```
let type: KerML::Metaclass =
    KerML::Metaclass.allInstances()
        ->any(t | t.qualifiedName = typeQName) in
if type.oclIsUndefined()
    or featureNames->size() <> featureValues->size() then
    Set{}
else
    Set{QNameToFeatureTyping_Factory.create(typeQName)}
    ->union(featureNames->collect(fn |
        let i: Integer = featureNames->indexOf(fn) in
        let feature: KerML::Feature = type.feature
          ->any(f | f.name = fn) in
        if  feature.oclIsUndefined() then
            invalid
        else
            ReferenceUsageMembership_Factory.create(
              feature, '', false, featureValues->at(i)
            )
```

```
            endif)->asSet())
    endif
```

7.5.2.21 MetadataFeature_Factory

Description

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

General Classes

- Factory (from Foundations)
- ToFeature_Init (from KerMLInitializers)

Association Ends

- to : MetadataFeature [1]
  {redefines: ToFeature_Init::to}

Operations

- create () : MetadataFeature [1]

7.5.2.22 MetadataOwningMembership_Factory

Description

General Classes

- Factory (from Foundations)
- ToOwningMembership_Init (from KerMLInitializers)

Attributes

- featureValues [0..*] (not typed) {ordered, nonunique}

Association Ends

- featureNames : String [0..*] {ordered}
- typeQName : String [1]

Operations

- create (in typeQName : String, in featureNames : String, in featureValues) : OwningMembership [1]
- memberElement () : Element [1] {redefines memberElement}

  ```
  MetaclassQNameToMetadataFeature_Factory.create(typeQName, featureNames, featureValues)
  ```

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

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

7.5.2.23 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]
- declaredName () : String [0..1] {redefines declaredName}

```
'multiplicity'
```

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

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

7.5.2.24 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]
- memberElement () : Feature [1] {redefines memberElement}

```
self.lowerBound
```

7.5.2.25 MultiplicityMembership_Factory

Description

General Classes

- Factory (from Foundations)
- ToOwningMembership_Init (from KerMLInitializers)

Association Ends

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

Operations

- create (in lowerValue : Integer, in upperValue : Integer) : FeatureMembership [1]
- declaredName () : String [0..1] {redefines declaredName}

```
'for multiplicity'
```

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

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

7.5.2.26 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] {redefines memberElement}
- memberElement () : Feature [1]

```
self.upperBound
```

7.5.2.27 ParameterMembership_Factory

Description

Factory class to create a ParameterMembership relationship.

General Classes

- Factory (from Foundations)
- ToParameterMembership_Init (from KerMLInitializers)

Attributes

- value [0..1] (not typed)

Association Ends

- name : String [1]
- type : Type [1]

Operations

- create (in name : String, in type : Type, in value) : ParameterMembership [1]
- memberElement () : Element [1] {redefines memberElement}

```
ReferenceUsageAsParameter_Factory.create('result', type)
```

7.5.2.28 QNameToFeatureTyping_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

- typeQName : String [1]

Operations

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

```
SysMLv2::Type.allInstances()->any(t | t.qualifiedName = typeQName)
```

7.5.2.29 ReferenceUsage_Factory

Description

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

General Classes

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

Attributes

- value [0..1] (not typed)

Association Ends

- feature : Feature [1]
- name : String [1]

Operations

- create (in feature : Feature, in name : String, in redefine : Boolean, in value) : ReferenceUsage [1]
- declaredName () : String [0..1] {redefines declaredName}

```
name
```

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

```
(if value.oclIsKindOf(String) then
   Set{FeatureValue_Factory.create(value)}
else
    if value.oclIsUndefined() or not value.oclIsKindOf(KerML::Feature) then
        Set{}
    else
        Set{ReferenceUsageMembership_Factory.create(value, '', false, invalid)}
endif endif)
->including(Subsetting_Factory.create(feature))
->asSet()->reject(r | r.oclIsUndefined())
```

7.5.2.30 ReferenceUsageAsParameter_Factory

Description

General Classes

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

Association Ends

- name : String [1]
- type : Type [1]

Operations

- create (in name : String, in type : Type) : ParameterMembership [1]
- ownedRelationship () : Relationship [0..*] {redefines ownedRelationship}

```
Set{TypeToFeatureTyping_Factory.create(type)}
```

7.5.2.31 ReferenceUsageMembership_Factory

Description

General Classes

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

Attributes

- value [0..1] (not typed)

Association Ends

- feature : Feature [1]
- refUsageName : String [1]

Operations

- create (in feature : Feature, in refUsageName : String, in redefine : Boolean, in value) : FeatureMembership [1]
- memberElement () : Feature [1] {redefines memberElement}

```
ReferenceUsage_Factory.create(feature, refUsageName, redefine, value)
```

7.5.2.32 ReturnParameterMembership_Factory

Description

General Classes

- ParameterMembership_Factory (from Factories)
- ToReturnParameterMembership_Init (from KerMLInitializers)

Attributes

- name [1] : String
- value [0..1] (not typed)

Operations

- create (in name : String, in type : Type) : ParameterMembership [1] {redefines create}
- ownedRelationship () : Relationship [0..*] {redefines ownedRelationship}

```
Set{TypeToFeatureTyping_Factory.create(type)}
```

7.5.2.33 Subsetting_Factory

Description

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

General Classes

- Factory (from Foundations)
- ToSubsetting_Init (from KerMLInitializers)

Association Ends

- subsetted : Feature [1]

Operations

- create (in subsetted : Feature) : Subsetting [1]
- subsettedFeature () : Feature [1] {redefines subsettedFeature}

```
subsetted
```

7.5.2.34 TypeToFeatureTyping_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 : Type [1]

Operations

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

```
type
```

7.5.2.35 UntypedReturnParameterFeature_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'
```

7.5.2.36 UntypedReturnParameterMembership_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]
- memberElement () : Element [1] {redefines memberElement}

```
UntypedReturnParameterFeature_Factory.create()
```

7.5.2.37 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{
   ReturnParameterMembership_Factory.create(
       'result',
       Helper.getScalarValueTypeByName('Integer')
   )
}
```

- value () : Integer [1]

```
upperValue
```

7.6 Mappings from UML4SysML metaclasses

7.6.1 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.

7.6.2 Actions

7.6.2.1 Overview

Table 1. List of High-Level Action Mappings

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
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.
AcceptEventAction	AcceptActionUsage
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.
AddStructuralFeatureValueAction	ActionUsage
AddVariableValueAction	ActionUsage
BroadcastSignalAction	ActionUsage
CallBehaviorAction	ActionUsage
CallOperationAction	ActionUsage
Clause	Mapping is not specified yet.
ClearAssociationAction	ActionUsage
ClearStructuralFeatureAction	ActionUsage
ClearVariableAction	ActionUsage
ConditionalNode	Mapping is not specified yet.
CreateLinkAction	ActionUsage
CreateLinkObjectAction	ActionUsage
CreateObjectAction	ActionUsage
DestroyLinkAction	ActionUsage
DestroyObjectAction	ActionUsage
InputPin	ReferenceUsage
LinkEndCreationData	Mapping is not specified yet.
LinkEndData	Mapping is not specified yet.
LinkEndDestructionData	Mapping is not specified yet.
LoopNode	ActionUsage
OpaqueAction	ActionUsage

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
OutputPin	ReferenceUsage
RaiseExceptionAction	ActionUsage
ReadExtentAction	ActionUsage
ReadIsClassifiedObjectAction	ActionUsage
ReadLinkAction	ActionUsage
ReadLinkObjectEndAction	ActionUsage
ReadSelfAction	ActionUsage
ReadStructuralFeatureAction	ActionUsage
ReadVariableAction	ActionUsage
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.
ReduceAction	ActionUsage
RemoveStructuralFeatureValueAction	ActionUsage
RemoveVariableValueAction	ActionUsage
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.
SendObjectAction	ActionUsage
SendSignalAction	ActionUsage
SequenceNode	ActionUsage
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.
StructuredActivityNode	ActionUsage
TestIdentityAction	CalculationUsage
UnmarshallAction	Mapping is not specified yet.
ValuePin	ReferenceUsage
ValueSpecificationAction	ActionUsage

7.6.2.2 Mapping Specifications

7.6.2.2.1 Accept Event Actions

7.6.2.2.1.1 AcceptActionUsageMembership_Mapping

Description

Creates a membership relationship for *memberElement()*.

General Mappings

ToFeatureMembership_Init
ElementUnique_Mapping

Mapping Source

AcceptEventAction

Mapping Target

FeatureMembership with qualifier: trigger:Trigger

Applicable 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::elementId () : String [1]

  ```
  Helper.getID(from)+'_AAUFMS'
  ```

- FeatureMembership::memberElement (in trigger : Trigger) : Element [1]

  ```
  AcceptEventAction_Mapping.getMapped(from, trigger)
  ```

7.6.2.2.1.2 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 with qualifier: trigger:Trigger

Applicable filters

(none)

7.6.2.2.1.3 AcceptEventActionCommon_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. If the trigger is a change event, then change expression is set as a feature of the action usage and its return parameter is bound to the payload parameter.

SysMLv2 does not support more than one trigger. Therefore only the specified trigger of the action is transformed.

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 with qualifier: trigger:Trigger

Applicable 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 (in trigger : Trigger) : Relationship [0..*]

```
let payloadArg: KerML::Expression
    = if trigger.event.oclIsKindOf(UML::ChangeEvent) then
            Expression_Mapping.getMapped(trigger.event.changeExpression)
else
        invalid
endif in
let payloadParam: KerML::Feature
```

```
    = AcceptEventActionToPLParameter_Mapping.getMapped(from, trigger) in

self.oclAsType(NamedElementUnique_Mapping).ownedRelationship()
->union(if trigger.event.oclIsKindOf(UML::SignalEvent) then
                Set{PayloadParameterMembership_Mapping.getMapped(from, trigger)}
        else if payloadArg.oclIsUndefined() then
                Set{}
        else
                Set{
                PayloadParameterMembership_Mapping.getMapped(from, trigger)
                ,PayloadArgMembership_Mapping.getMapped(trigger)}
        endif endif)
    ->union(if payloadArg.oclIsUndefined() then
        Set{}
else
        Set{BindingMembership_Factory.create(payloadParam
        ,ValueSpecificationToResult_Mapping.getMapped(
        trigger.event.changeExpression)
)}
endif)
```

7.6.2.2.1.4 AcceptEventActionToPLParameter_Mapping

Description

Based on the AcceptEventAction provided as the source, this mapping creates the parameter for the derived payloadArgument feature of the corresponding AcceptActionUsage.

General Mappings

ToReferenceUsage_Init
ElementUnique_Mapping

Mapping Source

AcceptEventAction

Mapping Target

ReferenceUsage with qualifier: trigger:Trigger

Applicable 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::elementId () : String [1]

  ```
  Helper.getID(from)+'_Payload'
  ```

- ReferenceUsage::ownedRelationship (in trigger : Trigger) : Relationship [0..*]

  ```
  if trigger.event.oclIsKindOf(UML::SignalEvent) then
      Set{TypeToFeatureTyping_Mapping.getMapped(trigger.event.signal)}
  else if trigger.event.oclIsKindOf(UML::ChangeEvent) then
      Set{}
  ```

```
else
    Set{}
endif endif
```

7.6.2.2.1.5 PayloadArgMembership_Mapping

Description

Creates a membership relationship for *memberElement()*.

General Mappings

ActionFeatureMembership_Mapping

Mapping Source

Trigger

Mapping Target

FeatureMembership

Applicable 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::memberElement () : Element [1]

```
if from.event.oclIsKindOf(UML::ChangeEvent) then
     Expression_Mapping.getMapped(from.event.changeExpression)
else
    invalid
endif
```

7.6.2.2.1.6 PayloadParameterMembership_Mapping

Description

Creates a membership relationship for *memberElement()*.

General Mappings

ActionParameterMembership_Mapping

Mapping Source

AcceptEventAction

Mapping Target

ParameterMembership with qualifier: trigger:Trigger

Applicable 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::memberElement (in trigger : Trigger) : Element [1]

  ```
  AcceptEventActionToPLParameter_Mapping.getMapped(from, trigger)
  ```

7.6.2.2.1.7 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

Applicable filters

(none)

7.6.2.2.1.8 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

Applicable filters

(none)

7.6.2.2.1.9 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

AcceptEventActionCommon_Mapping

Mapping Source

AcceptEventAction

Mapping Target

AcceptActionUsage with qualifier: trigger:Trigger

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::Ports&Flows::AcceptChangeStructuralFeatureEventAction')
```

Mapping rules

The mapping class only has inherited rules. See the mapping classes in the general mapping section for details.

7.6.2.2.2 Actions

7.6.2.2.2.1 ActionFeatureMembership_Mapping

Description

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

General Mappings

ToFeatureMembership_Init
ElementUnique_Mapping

Mapping Source

Element

Mapping Target

FeatureMembership

Applicable 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::elementId () : String [1]

  ```
  Helper.getID(from)+'_FMS'
  ```

7.6.2.2.2.2 ActionParameterMembership_Mapping

Description

Creates a membership relationship for *memberElement()*.

General Mappings

ActionFeatureMembership_Mapping
ToParameterMembership_Init

Mapping Source

Element

Mapping Target

ParameterMembership

Applicable filters

(none)

7.6.2.2.2.3 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
PropertyLikeNamed_Mapping

Mapping Source

Action

Mapping Target

ActionUsage

Applicable 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::isComposite () : Boolean [1]

  ```
  true
  ```

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

  ```
  let toElementFMS: Set(UML::Element) =
      from.ownedElement->select(e | e.oclIsKindOf(UML::Pin)
          or e.oclIsKindOf(UML::Trigger)) in
  self.oclAsType(NamedElementUnique_Mapping).ownedRelationship()
  ->union(ElementFeatureMembership_Mapping.getMappedColl(toElementFMS)->asSet())
  ->reject(f | f.oclIsUndefined())
  ```

7.6.2.2.2.4 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

Applicable 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..*]

```
self.oclAsType(CommonAction_Mapping).ownedRelationship()
    ->union(from.language
        ->collect(l |
            OpaqueActionSpecificationMembership_Mapping.getMapped(
                from,
                l
            ))
    ->asSet())
```

7.6.2.2.2.5 OpaqueActionSpecification_Mapping

Description

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

General Mappings

OpaqueExpressionLike_Mapping

Mapping Source

OpaqueAction

Mapping Target

TextualRepresentation with qualifier: language:String

Applicable filters

(none)

7.6.2.2.2.6 OpaqueActionSpecificationMembership_Mapping

Description

Creates a membership relationship for *memberElement()*.

General Mappings

ToOwningMembership_Init
ElementUnique_Mapping

Mapping Source

OpaqueAction

Mapping Target

OwningMembership with qualifier: language:String

Applicable 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.memberElement()}
  ```

- OwningMembership::elementId () : String [1]

  ```
  Helper.createUUID()
  ```

- OwningMembership::memberElement () : Element [1]

  ```
  OpaqueActionSpecification_Mapping.getMapped(from, language)
  ```

7.6.2.2.2.7 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
PropertyLikeNamed_Mapping

Mapping Source

Pin

Mapping Target

ReferenceUsage

Applicable 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.oclIsTypeOf(UML::InputPin) then
    KerML::FeatureDirectionKind::_'in'
else if from.oclIsTypeOf(UML::OutputPin) then
    KerML::FeatureDirectionKind::_'out'
else
    invalid
endif endif
```

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

```
self.oclAsType(ElementUnique_Mapping).ownedRelationship()
->including(MultiplicityMembership_Factory.create(from.lower, from.upper))
->including(TypeToFeatureTyping_Mapping.getMapped(from.type))
->reject(r | r.oclIsUndefined())
```

7.6.2.2.2.8 PinToParameter_Mapping

Description

The mapping class maps a UML4SysML::Pin to a ReferenceUsage.

General Mappings

Mapping
ToReferenceUsage_Init

Mapping Source

Pin

Mapping Target

ReferenceUsage

Applicable 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::ownedRelationship () : Relationship [0..*]

```
Set{MultiplicityMembership_Factory.create(from.lower, from.upper)}
->union(if from.type.oclIsUndefined() then
    Set{}
```

```
else
    Set{TypeToFeatureTyping_Mapping.getMapped(from.type)}
endif)
```

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

```
if from.oclIsTypeOf(UML::InputPin) then
    KerML::FeatureDirectionKind::_'in'
else if from.oclIsTypeOf(UML::OutputPin) then
    KerML::FeatureDirectionKind::_'out'
else
    invalid
endif endif
```

7.6.2.2.2.9 PinToParameterMembership_Mapping

Description

Creates a membership relationship for *memberElement()*.

General Mappings

ToParameterMembership_Init
Mapping

Mapping Source

Pin

Mapping Target

ParameterMembership

Applicable 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::memberElement () : Element [1]

```
PinToParameter_Mapping.getMapped(from)
```

7.6.2.2.2.10 PinToReturnParameterMembership_Mapping

Description

Creates a membership relationship for *memberElement()*.

General Mappings

ToReturnParameterMembership_Init
Mapping

Mapping Source

Pin

Mapping Target

ReturnParameterMembership

Applicable 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::memberElement () : Element [1]

  ```
  PinToParameter_Mapping.getMapped(from)
  ```

7.6.2.2.2.11 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

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_Factory.create(from.lower, from.upper)}
```

7.6.2.2.2.12 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

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))
```

7.6.2.2.3 Invocation Actions

7.6.2.2.3.1 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

Applicable filters

(none)

7.6.2.2.3.2 CallAction_Mapping

Description

Abstract mapping class for UML4SysML::CallAction that maps a CallAction to an ActionUsage.

General Mappings

InvocationAction_Mapping

Mapping Source

CallAction

Mapping Target

ActionUsage

Applicable filters

(none)

7.6.2.2.3.3 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

CallAction_Mapping

Mapping Source

CallBehaviorAction

Mapping Target

ActionUsage

Applicable 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..*]

```
self.oclAsType(CommonAction_Mapping).ownedRelationship()
->including(TypeToFeatureTyping_Mapping.getMapped(from.behavior))
```

7.6.2.2.3.4 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

CallAction_Mapping

Mapping Source

CallOperationAction

Mapping Target

ActionUsage

Applicable 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..*]

```
self.oclAsType(CommonAction_Mapping).ownedRelationship()
```

7.6.2.2.3.5 InvocationAction_Mapping

Description

Abstract mapping class for UML4SysML::InvocationAction. It maps an InvocationAction to an ActionUsage.

General Mappings

CommonAction_Mapping

Mapping Source

InvocationAction

Mapping Target

ActionUsage

Applicable filters

(none)

7.6.2.2.3.6 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 SysMLv1Objectl() to target;
}
part def SysMLv1Block;
item def SysMLv1Object;
```

General Mappings

CommonAction_Mapping

Mapping Source

SendObjectAction

Mapping Target

ActionUsage

Applicable filters

(none)

7.6.2.2.3.7 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

Applicable 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..*]

  ```
  self.oclAsType(CommonAction_Mapping).ownedRelationship()
  ```

7.6.2.2.4 Link Actions

7.6.2.2.4.1 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

Applicable filters

(none)

7.6.2.2.4.2 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

Applicable 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.oclIsTypeOf(UML::LinkEndCreationData)) in
let actionInputPin: Set(UML::Element) =
    from.ownedElement->select(e | e.oclIsTypeOf(UML::ActionInputPin)) in
let triggers: Set(UML::Element) =
    from.ownedElement->select(e | e.oclIsKindOf(UML::Trigger)) in
let toElementFMS: Set(UML::Element) =
    from.ownedElement->select(e | e.oclIsKindOf(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)))
```

7.6.2.2.4.3 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

Applicable filters

(none)

7.6.2.2.4.4 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

Applicable 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.oclIsTypeOf(UML::ActionInputPin)) in
let triggers: Set(UML::Element) =
    from.ownedElement->select(e | e.oclIsKindOf(UML::Trigger)) in
let linkData: Set(UML::Element) =
```

```
    from.ownedElement->select( e | e.oclIsKindOf(UML::LinkEndData) or
    e.oclIsKindOf(UML::LinkEndDestructionData)) in
let toElementFMS: Set(UML::Element) =
    from.ownedElement->select(e | e.oclIsKindOf(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)))
```

7.6.2.2.4.5 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

Applicable 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.oclIsTypeOf(UML::ActionInputPin)) in
let triggers: Set(UML::Element) =
    from.ownedElement->select(e | e.oclIsKindOf(UML::Trigger)) in
let linkData: Set(UML::Element) =
    from.ownedElement->select( e | e.oclIsKindOf(UML::LinkEndData)) in
let toElementFMS: Set(UML::Element) =
    from.ownedElement->select(e | e.oclIsKindOf(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)))
```

7.6.2.2.4.6 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

Applicable filters

(none)

7.6.2.2.5 Object Actions

7.6.2.2.5.1 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

Applicable filters

(none)

7.6.2.2.5.2 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

Applicable 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..*]

  ```
  self.oclAsType(CommonAction_Mapping).ownedRelationship()
  ```

7.6.2.2.5.3 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

Applicable filters

(none)

7.6.2.2.5.4 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

Applicable filters

(none)

7.6.2.2.5.5 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

Applicable filters

(none)

7.6.2.2.5.6 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

Applicable filters

(none)

7.6.2.2.5.7 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

Applicable filters

(none)

7.6.2.2.5.8 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

Applicable filters

(none)

7.6.2.2.5.9 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

Applicable 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))
```

7.6.2.2.5.10 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

Applicable 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{PinToParameterMembership_Mapping.getMapped(from.first),
PinToParameterMembership_Mapping.getMapped(from.second),
PinToReturnParameterMembership_Mapping.getMapped(from.result)}
```

7.6.2.2.5.11 TIAResultExpressionMembership_Mapping

Description

Creates a membership relationship for *memberElement()*.

General Mappings

ToFeatureMembership_Init
Mapping

Mapping Source

TestIdentityAction

Mapping Target

ResultExpressionMembership

Applicable 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::memberElement () : Element [1]

  ```
  TIAOperatorExpression_Mapping.getMapped(from)
  ```

7.6.2.2.5.12 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

Applicable filters

(none)

7.6.2.2.6 Other Actions

7.6.2.2.6.1 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

Applicable filters

(none)

7.6.2.2.6.2 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

Applicable filters

(none)

7.6.2.2.7 Structural Feature Actions

7.6.2.2.7.1 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

WriteStructuralFeatureValueAction_Mapping

Mapping Source

AddStructuralFeatureValueAction

Mapping Target

ActionUsage

Applicable filters

(none)

7.6.2.2.7.2 ClearStructuralFeatureAction_Mapping

Description

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

General Mappings

StructuralFeatureAction_Mapping

Mapping Source

ClearStructuralFeatureAction

Mapping Target

ActionUsage

Applicable filters

(none)

7.6.2.2.7.3 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

StructuralFeatureAction_Mapping

Mapping Source

ReadStructuralFeatureAction

Mapping Target

ActionUsage

Applicable filters

(none)

7.6.2.2.7.4 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

Applicable filters

(none)

7.6.2.2.7.5 StructuralFeatureAction_Mapping

Description

Common mapping class for UML4SysML::StructuralFeatureAction. It maps a StructualFeatureAction to an ActionUsage.

General Mappings

CommonAction_Mapping

Mapping Source

StructuralFeatureAction

Mapping Target

ActionUsage

Applicable 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{ElementFeatureMembership_Mapping.getMapped(from.structuralFeature),
ElementFeatureMembership_Mapping.getMapped(from.object)}
->union(self.oclAsType(CommonAction_Mapping).ownedRelationship())
```

7.6.2.2.7.6 WriteStructuralFeatureValueAction_Mapping

Description

TBC: not sure we really need them!

General Mappings

StructuralFeatureAction_Mapping

Mapping Source

WriteStructuralFeatureAction

Mapping Target

ActionUsage

Applicable filters

(none)

7.6.2.2.8 Structured Actions

7.6.2.2.8.1 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

Applicable filters

(none)

7.6.2.2.8.2 SequenceNode_Mapping

Description

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

General Mappings

StructuredActivityNode_Mapping

Mapping Source

SequenceNode

Mapping Target

ActionUsage

Applicable filters

(none)

7.6.2.2.8.3 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

Applicable filters

(none)

7.6.2.2.9 Variable Actions

7.6.2.2.9.1 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

Applicable 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 ava : SYSML2::ActionDefinition = SYSML2::ActionDefinition.allInstances()
->any(m | m.qualifiedName = 'SysMLv1Library::AddValueAction') in
let var: KerML::Feature = CommonVariable_Mapping.getMapped(from.variable) in
OrderedSet{TypeToFeatureTyping_Factory.create(ava),
           ReferenceUsageMembership_Factory.create(SYSML2::ReferenceUsage.allInstances()
               ->any(u | u.qualifiedName = 'SysMLv1Library::AddValueAction::insertAt')
               ,'inserAt'
               ,true
               ,invalid),
           ReferenceUsageMembership_Factory.create(SYSML2::ReferenceUsage.allInstances()
               ->any(u | u.qualifiedName = 'SysMLv1Library::AddValueAction::value')
               ,'value'
               ,true
               ,invalid),
           ReferenceUsageMembership_Factory.create(SYSML2::ReferenceUsage.allInstances()
               ->any(u | u.qualifiedName = 'SysMLv1Library::AddValueAction::isReplaceAll')
               ,'isReplaceAll'
               ,true
               ,invalid),
           ReferenceUsageMembership_Factory.create(SYSML2::ReferenceUsage.allInstances()
               ->any(u | u.qualifiedName = 'SysMLv1Library::AddValueAction::target')
               ,'target'
               ,true
               ,var)
}
```

7.6.2.2.9.2 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

Applicable 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..*]

  ```
  self.oclAsType(CommonAction_Mapping).ownedRelationship()
  ```

7.6.2.2.9.3 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

Applicable 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..*]

```
self.oclAsType(CommonAction_Mapping).ownedRelationship()
--Set{RVAFeatureMembership_Mapping.getMapped(from)}
```

7.6.2.2.9.4 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

Applicable 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..*]

```
self.oclAsType(CommonAction_Mapping).ownedRelationship()
```

7.6.3 Activities

7.6.3.1 Overview

Table 2. List of High-Level Activities Mappings

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
Activity	ActionDefinition
ActivityFinalNode	TerminateActionUsage
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.
CentralBufferNode	ActionUsage
ControlFlow	TransitionUsage, SuccessionAsUsage
DataStoreNode	ActionUsage
DecisionNode	DecisionNode
ExceptionHandler	Mapping is not specified yet.
FlowFinalNode	Mapping is not specified yet.
ForkNode	ForkNode
InitialNode	Mapping is not specified yet.
InterruptibleActivityRegion	Mapping is not specified yet.
JoinNode	JoinNode
MergeNode	MergeNode
ObjectFlow	SuccessionFlowUsage, TransitionUsage
Variable	AttributeUsage, ItemUsage

7.6.3.2 Mapping Specifications

7.6.3.2.1 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

ActivityAsDefinitionCommon_Mapping

Mapping Source

Activity

Mapping Target

ActionDefinition

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::Requirements::TestCase')
```

Mapping rules

The mapping class only has inherited rules. See the mapping classes in the general mapping section for details.

7.6.3.2.2 ActivityAsDefinitionCommon_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

Applicable 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..*]

```
self.oclAsType(Behavior_Mapping).ownedRelationship()
    ->union(from.edge->asSet()
        ->union(from.node->select(n | n.oclIsKindOf(UML::Action)
            or n.oclIsKindOf(UML::ControlNode))->asSet()
            ->reject(n | n.oclIsKindOf(UML::AcceptEventAction)))
        ->union(from.variable->asSet())
        ->collect(n | ElementFeatureMembership_Mapping.getMapped(n))->asSet())
    ->union(from.node->select(n | n.oclIsKindOf(UML::AcceptEventAction))
        ->collect(n | n.trigger->collect(t |
            AcceptActionUsageMembership_Mapping.getMapped(n, t)))
        ->flatten()->asSet()
        )
    ->reject(f | f.oclIsUndefined())
```

7.6.3.2.3 ActivityEdge_Mapping

Description

Abstract mapping class which maps a UML4SysML::ActivityEdge to a SysMLv2 Connector.

General Mappings

ToConnector_Init
PropertyLikeNamed_Mapping

Mapping Source

ActivityEdge

Mapping Target

Connector

Applicable filters

(none)

Mapping rules

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

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

```
Set{
    ElementFeatureMembership_Mapping.getMapped(from.source),
    ElementFeatureMembership_Mapping.getMapped(from.target)}
    ->union(
        if from.guard.oclIsUndefined() then
            Set{}
        else
            Set{ElementFeatureMembership_Mapping.getMapped(from.guard)}
    endif)
    ->union(
        if from.weight.oclIsUndefined() then
            Set{}
        else
            Set{MetadataOwningMembership_Factory.create(
                'SysMLv1Library::ActivityEdgeData',
                OrderedSet{'weight'},
```

```
            Sequence{from.weight})
        }
    endif)
->union(
    if Helper.hasStereotypeApplied(
        from,
        'SysML::Activities::Probability') then
        let value: OclAny =
            Helper.getTagValue(
                from,
                'SysML::Activities::Probability',
                'probability') in
        Set{MetadataOwningMembership_Factory.create(
            'SysMLv1Library::Probability',
            OrderedSet{'probability'},
            Sequence{value})
        }
    else
        Set{}
    endif)
->union(if Helper.hasStereotypeApplied(
    from,
    'SysML::Activities::Discrete') then
    Set{MetadataOwningMembership_Factory.create(
        'SysMLv1Library::RateData',
        OrderedSet{'isDiscrete'},
        Sequence{true})
    }
    else if Helper.hasStereotypeApplied(
        from,
        'SysML::Activities::Continuous') then
        Set{MetadataOwningMembership_Factory.create(
            'SysMLv1Library::RateData',
            OrderedSet{'isDiscrete'},
            Sequence{false})
        }
    else if Helper.hasStereotypeApplied(
        from,
        'SysML::Activities::Rate') then
        let values: Sequence(OclAny) =
            Sequence{Helper.getTagValueAsElementColl(
                from,
                'SysML::Activities::Rate',
                'rate')} in
        Set{MetadataOwningMembership_Factory.create(
            'SysMLv1Library::RateData',
            OrderedSet{'rate'},
            values)
        }
    else
        Set{}
    endif endif endif)
->union(if Helper.hasStereotypeApplied(
    from,
    'SysML::Activities::Probability') then
    let values: Sequence(OclAny) =
        Sequence{Helper.getTagValueAsElementColl(
            from,
            'SysML::Activities::Probability',
            'probability')} in
    Set{MetadataOwningMembership_Factory.create(
        'SysMLv1Library::ProbabilityData',
        OrderedSet{'probability'},
```

```
            values)
          }
        else
            Set{}
        endif)
    ->union(self.oclAsType(ElementUnique_Mapping).ownedRelationship())
```

7.6.3.2.4 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

CommonNode_Mapping

Mapping Source

ActivityFinalNode

Mapping Target

TerminateActionUsage

Applicable filters

(none)

7.6.3.2.5 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

CommonNode_Mapping

Mapping Source

CentralBufferNode

Mapping Target

ActionUsage

Applicable filters

(none)

7.6.3.2.6 CommonNode_Mapping

Description

Abstract mapping class for UML4SysML::CommonNode. It maps an ActivityNode to an ActionUsage.

General Mappings

ToActionUsage_Init
PropertyLikeNamed_Mapping

Mapping Source

ActivityNode

Mapping Target

ActionUsage

Applicable 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 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 |
                ElementFeatureMembership_Mapping.getMapped(i))
            ->asSet()
        )
        ->union(from.outgoing
            ->collect(i |
                ElementFeatureMembership_Mapping.getMapped(i))
            ->asSet()
        )->reject(r | r.oclIsUndefined())
else
  Set{}
endif
```

7.6.3.2.7 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

PropertyLikeNamed_Mapping

Mapping Source

Variable

Mapping Target

Feature

Applicable 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]

```
false
```

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

```
false
```

- Feature::isDerived () : Boolean [1]

```
false
```

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

```
if from.type.oclIsUndefined() then
    Set{
        MultiplicityMembership_Factory.create(
            from.lower,
            from.upper
        )
    }
else
    Set{
        MultiplicityMembership_Factory.create(
            from.lower,
            from.upper
        ),
        TypeToFeatureTyping_Mapping.getMapped(from.type)
    }
endif
```

7.6.3.2.8 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

ActivityEdge_Mapping

Mapping Source

ControlFlow

Mapping Target

SuccessionAsUsage

Applicable filters

(none)

7.6.3.2.9 DataStoreNode_Mapping

Description

The mapping of the UML4SysML::DataStoreNode is not defined in detail yet. It will an action usage which contains the behavior of a data store node.

General Mappings

CentralBufferNode_Mapping

Mapping Source

DataStoreNode

Mapping Target

ActionUsage

Applicable filters

(none)

7.6.3.2.10 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

CommonNode_Mapping

Mapping Source

DecisionNode

Mapping Target

DecisionNode

Applicable filters

(none)

7.6.3.2.11 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

CommonNode_Mapping

Mapping Source

ForkNode

Mapping Target

ForkNode

Applicable filters

(none)

7.6.3.2.12 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

CommonNode_Mapping

Mapping Source

JoinNode

Mapping Target

JoinNode

Applicable filters

(none)

7.6.3.2.13 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

CommonNode_Mapping

Mapping Source

MergeNode

Mapping Target

MergeNode

Applicable filters

(none)

7.6.3.2.14 NodeMembership_Mapping

Description

Creates a membership relationship for *memberElement()*.

General Mappings

ElementFeatureMembership_Mapping

Mapping Source

ActivityNode

Mapping Target

FeatureMembership

Applicable filters

(none)

7.6.3.2.15 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
ActivityEdge_Mapping
ToFlowUsage_Init

Mapping Source

ObjectFlow

Mapping Target

SuccessionFlowUsage

Applicable filters

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

```
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..*]

```
self.oclAsType(ActivityEdge_Mapping).ownedRelationship()
->union(
    if from.source.oclIsKindOf(UML::ObjectNode) then
        Set{ObjectFlowItemFeatureMembership_Mapping.getMapped(from)}
    else
        Set{}
    endif)
```

7.6.3.2.16 ObjectFlowGuardSuccessionTargetEndSubsetting_Mapping

DescriptionGeneral Mappings

Mapping Class Specification Description

Mapping Source

No source element.

Mapping Target

No target element.

Applicable filters

(none)

7.6.3.2.17 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

CommonVariable_Mapping

Mapping Source

Variable

Mapping Target

AttributeUsage

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.

7.6.3.2.18 VariableFeatureMembership_Mapping

Description

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

General Mappings

ElementFeatureMembership_Mapping

Mapping Source

Variable

Mapping Target

FeatureMembership

Applicable filters

(none)

7.6.3.2.19 VariableFeatureTyping_Mapping

DescriptionGeneral Mappings

Mapping Class Specification Description

Mapping Source

No source element.
Mapping Target

No target element.

Applicable filters

(none)

7.6.3.2.20 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

CommonVariable_Mapping

Mapping Source

Variable

Mapping Target

ItemUsage

Applicable filters

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

```
not 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.

7.6.4 Classification

7.6.4.1 Overview

Table 3. List of High-Level Classification Mappings

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
Generalization	Subclassification
GeneralizationSet	Mapping is not specified yet.
InstanceSpecification	ConnectionUsage, PartUsage
InstanceValue	FeatureReferenceExpression
Operation	PerformActionUsage
Parameter	ReferenceUsage
ParameterSet	Mapping is not specified yet.
Property	AttributeUsage, OccurrenceUsage, ReferenceUsage, Feature
Slot	Feature
Substitution	Dependency

7.6.4.2 Mapping Specifications

7.6.4.2.1 BehavioralFeature_Mapping

Description

The mapping class is the abstract base class for UML4SysML::BehavioralFeature mappings.

General Mappings

ToUsage_Init
NamedElementUnique_Mapping

Mapping Source

BehavioralFeature

Mapping Target

Usage

Applicable filters

(none)

7.6.4.2.2 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

Applicable 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..*]

```
self.oclAsType(Namespace_Mapping).ownedRelationship()
    ->union(from.substitution
        ->collect(e | ElementOwningMembership_Mapping.getMapped(e))
        ->asSet())
    ->union(from.ownedUseCase
        ->collect(e | ElementOwningMembership_Mapping.getMapped(e))
        ->asSet())
    ->union(from.ownedElement->selectByKind(UML::Feature)
        ->collect(e | ElementFeatureMembership_Mapping.getMapped(e))
        ->asSet())
    ->union(from.generalization
        ->collect(e | Generalization_Mapping.getMapped(e))
        ->asSet())
    ->reject(r | r.oclIsUndefined())
```

- Classifier::isAbstract () : Boolean [1]

```
from.isAbstract
```

7.6.4.2.3 ClassifierToFeature_Mapping

Description

Common mapping class that maps a UML4SysML::Classifier to a Feature.

General Mappings

ToFeature_Init
Mapping

Mapping Source

Classifier

Mapping Target

Feature

Applicable 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{TypeToFeatureTyping_Mapping.getMapped(from)}
  ```

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

  ```
  from.name.toLowerCase()
  ```

7.6.4.2.4 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
NamedElementUnique_Mapping

Mapping Source

ValueSpecification

Mapping Target

FeatureValue

Applicable 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]

```
if from.type.oclIsKindOf(Boolean) then
    LiteralBoolean_Factory.create(from.booleanValue())
else if from.type.oclIsKindOf(String) then
    LiteralString_Factory.create(from.stringValue())
else if from.type.oclIsKindOf(Integer) then
    LiteralInteger_Factory.create(from.integerValue())
else if from.type.oclIsKindOf(UML::UnlimitedNatural) then
    LiteralInteger_Factory.create(from.unlimitedValue())
else if from.type.oclIsKindOf(Real) then
    LiteralRational_Factory.create(from.realValue())
else if from.isNull() then
    LiteralNull_Factory.create()
else
    invalid
endif endif endif endif endif endif
```

7.6.4.2.5 ElementFeatureMembership_Mapping

Description

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

General Mappings

ToFeatureMembership_Init
ElementMembership_Mapping
ElementUnique_Mapping

Mapping Source

Element

Mapping Target

FeatureMembership

Applicable 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::memberElement () : Element [1]

```
if from.oclIsKindOf(UML::Classifier) then
    ClassifierToFeature_Mapping.getMapped(from)
else if from.oclIsKindOf(UML::PackageableElement) or from.oclIsKindOf(UML::Comment) then
```

```
    ElementMain_Mapping.getMapped(from)
else
    PropertyLike_Mapping.getMapped(from)
endif endif
```

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

```
Set{self.memberElement()}->reject(e | e.oclIsUndefined())
```

- FeatureMembership::elementId () : String [1]

```
Helper.getID(from)+'_FMS'
```

- FeatureMembership::visibility () : VisibilityKind [1]

```
if from.oclIsKindOf(UML::NamedElement) then
Helper.getKerMLVisibilityKind(from.oclAsType(UML::NamedElement).visibility)
else KerML::VisibilityKind::public endif
```

7.6.4.2.6 FromPropertyBasic_Mapping

Description

Abstract mapping class that maps a UML4SysML::Property to an Element.

General Mappings

Mapping

Mapping Source

Property

Mapping Target

Element

Applicable 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::isReadOnly () : Boolean [1]

```
from.isReadOnly
```

- Element::isDerived () : Boolean [1]

```
from.isDerived
```

- Element::isAbstract () : Boolean [1]

```
false
```

- Element::isEnd () : Boolean [1]

```
if from.association.oclIsUndefined() then
    false
else
    from.association.ownedEnd->includes(from)
endif
```

- Element::isUnique () : Boolean [1]

  ```
  from.isUnique
  ```

- Element::isOrdered () : Boolean [1]

  ```
  from.isOrdered
  ```

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

  ```
  from.name
  ```

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

  ```
  from.isComposite
  ```

7.6.4.2.7 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

ToSubclassification_Init
ElementUnique_Mapping

Mapping Source

Generalization

Mapping Target

Subclassification

Applicable 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)
```

7.6.4.2.8 InstanceSpecificationCommon_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

Applicable 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..*]

```
ElementFeatureMembership_Mapping.getMappedColl(from.slot)->asSet()
->union(TypeToFeatureTyping_Mapping.getMappedColl(from.classifier)->asSet())
->union(self.oclAsType(ElementMain_Mapping).ownedRelationship())
->asSet()
```

7.6.4.2.9 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
InstanceSpecificationCommon_Mapping

Mapping Source

InstanceSpecification

Mapping Target

ConnectionUsage

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.oclIsTypeOf(UML::Association))->size() > 0
```

Mapping rules

The mapping class only has inherited rules. See the mapping classes in the general mapping section for details.

7.6.4.2.10 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
InstanceSpecificationCommon_Mapping

Mapping Source

InstanceSpecification

Mapping Target

PartUsage

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.oclIsTypeOf(UML::Association))->size() = 0
```

Mapping rules

The mapping class only has inherited rules. See the mapping classes in the general mapping section for details.

7.6.4.2.11 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

Applicable 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(if from.type.oclIsUndefined() then
    UntypedReturnParameterMembership_Factory.create()
else
    ReturnParameterMembership_Factory.create(
        'result',
        Classifier_Mapping.getMapped(from.type)
    )
endif)
```

7.6.4.2.12 MultiplicityMembership_Mapping

Description

Creates a membership relationship for *memberElement()*.

General Mappings

ToOwningMembership_Init
ElementUnique_Mapping

Mapping Source

MultiplicityElement

Mapping Target

OwningMembership

Applicable 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::memberElement () : Element [1]

```
MultiplicityElement_Factory.create(from.lower, from.upper)
```

- OwningMembership::elementId () : String [1]

```
Helper.createUUID()
```

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

```
Set{}
```

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

```
'for multiplicity'
```

7.6.4.2.13 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

Applicable 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.oclIsKindOf(UML::Parameter)) in
let parameterSets: Set(UML::Element) =
    from.ownedElement->select(e | e.oclIsKindOf(UML::ParameterSet)) in
self.oclAsType(ElementUnique_Mapping).ownedRelationship()
->union(parameters->collect(e | ParameterMembership_Mapping.getMapped(e))->asSet())
->union(parameterSets->collect(e | ParameterSetMembership_Mapping.getMapped(e))->asSet())
```

7.6.4.2.14 PackageableElementFeatureMembership_Mapping

Description

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

General Mappings

ElementFeatureMembership_Mapping

Mapping Source

PackageableElement

Mapping Target

FeatureMembership

Applicable filters

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

```
not from.owner.oclIsKindOf(UML::State)
```

Mapping rules

The mapping class only has inherited rules. See the mapping classes in the general mapping section for details.

7.6.4.2.15 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
PropertyLikeNamed_Mapping

Mapping Source

Parameter

Mapping Target

ReferenceUsage

Applicable 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]

```
Helper.getKerMLParameterDirectionKind(from.direction)
```

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

```
if from.direction = UML::ParameterDirectionKind::return

    then 'result'

    else from.name endif
```

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

```
self.oclAsType(NamedElementUnique_Mapping).ownedRelationship()
->union(if from.type.oclIsUndefined() then
    Set{}
else if from.type.oclIsKindOf(UML::Enumeration) then
    let type: SysML2::EnumerationDefinition =
         Helper.getEnumerationType(from.type) in
    if type.oclIsUndefined() then
        Set{}
    else
        Set{TypeToFeatureTyping_Factory.create(type)}
    endif
else
    Set{TypeToFeatureTyping_Mapping.getMapped(from.type)}
endif endif)
->union(if from.defaultValue.oclIsUndefined() then
        Set{}
    else
        Set{DefaultValue_Mapping.getMapped(from.defaultValue)}
    endif)
    ->union(if Helper.hasStereotypeApplied(
        from,
        'SysML::Activities::Discrete'
        ) then
        Set{MetadataOwningMembership_Factory.create(
            'SysMLv1Library::RateData',
            OrderedSet{'isDiscrete'}, Sequence{true})
        }
    else if Helper.hasStereotypeApplied(
        from,
        'SysML::Activities::Continuous') then
        Set{MetadataOwningMembership_Factory.create(
            'SysMLv1Library::RateData',
            OrderedSet{'isDiscrete'},
            Sequence{false})
        }
    else if Helper.hasStereotypeApplied(
        from,
        'SysML::Activities::Rate') then
        let values: Sequence(OclAny) =
            Sequence{Helper.getTagValueAsElementColl(
                from,
                'SysML::Activities::Rate', 'rate')
            } in
        Set{MetadataOwningMembership_Factory.create(
            'SysMLv1Library::RateData', OrderedSet{'rate'},
```

```
                values)
            }
        else
            Set{}
        endif endif endif)
    ->including(MultiplicityMembership_Factory.create(from.lower, from.upper))->asSet()
```

7.6.4.2.16 ParameterMembership_Mapping

Description

Creates a membership relationship for *memberElement()*.

General Mappings

ToParameterMembership_Init
ElementFeatureMembership_Mapping

Mapping Source

Parameter

Mapping Target

ParameterMembership

Applicable 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::memberElement () : Element [1]

  ```
  Parameter_Mapping.getMapped(from)
  ```

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

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

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

  ```
  Set{}
  ```

7.6.4.2.17 ParameterReferenceUsage_Mapping

Description

The mapping class creates the reference usage element for the UML4SysML::ParameterSet mapping.

General Mappings

Mapping
ToReferenceUsage_Init

Mapping Source

Parameter

Mapping Target

ReferenceUsage

Applicable 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{TypeToFeatureTyping_Mapping.getMapped(from)}
  ```

7.6.4.2.18 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
NamedElementUnique_Mapping

Mapping Source

ParameterSet

Mapping Target

ReferenceUsage

Applicable 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()
->union(self.oclAsType(ElementUnique_Mapping).ownedRelationship())
->union(if Helper.hasStereotypeApplied(from, 'SysML::Activities::Probability') then
    let values: Sequence(OclAny) =
        Sequence{
            Helper.getTagValueAsElementColl(
                from,
                'SysML::Activities::Probability',
                'probability')
            } in
        Set{MetadataOwningMembership_Factory.create(
            'SysMLv1Library::ProbabilityData',
            OrderedSet{'probability'},
            values)
        }
    else
        Set{}
    endif)
->asSet()
```

7.6.4.2.19 ParameterSetMembership_Mapping

Description

Creates a membership relationship for *memberElement()*.

General Mappings

ToFeatureMembership_Init
NamedElementUnique_Mapping

Mapping Source

ParameterSet

Mapping Target

FeatureMembership

Applicable 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::memberElement () : Feature [1]

  ```
  ParameterSet_Mapping.getMapped(from)
  ```

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

  ```
  Set{}
  ```

- FeatureMembership::elementId () : String [1]

  ```
  Helper.getID(from)+'_PMS'
  ```

7.6.4.2.20 ParameterSetParameterFeatureMembership_Mapping

Description

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

General Mappings

ToFeatureMembership_Init
ElementUnique_Mapping

Mapping Source

ParameterSet

Mapping Target

FeatureMembership with qualifier: parameter:Parameter

Applicable 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::elementId () : String [1]

  ```
  Helper.createUUID()
  ```

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

  ```
  Set{}
  ```

- FeatureMembership::memberElement (in parameter : Parameter) : Feature [1]

  ```
  Parameter_Mapping.getMapped(parameter)
  ```

7.6.4.2.21 PropertyCommon_Mapping

Description

The mapping class is the abstract base class for UML4SysML::Property mappings.

General Mappings

PropertyLikeNamed_Mapping
FromPropertyBasic_Mapping

Mapping Source

Property

Mapping Target

Feature

Applicable 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..*]

```
self.oclAsType(ElementUnique_Mapping).ownedRelationship()
->union(if from.type.oclIsUndefined() then
    Set{}
else
    Set{TypeToFeatureTyping_Mapping.getMapped(from.type)}
endif)
->union(from.subsettedProperty
    ->collect(p | PropertySubsetting_Mapping.getMapped(from, p))->asSet())
->union(PropertyRedefinition_Mapping.getMappedColl(from.redefinedProperty)->asSet())
->union(if from.defaultValue.oclIsUndefined() then
        Set{}
    else
        Set{DefaultValue_Mapping.getMapped(from.defaultValue)}
    endif)
->including(MultiplicityMembership_Factory.create(from.lower, from.upper))->asSet()
```

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

```
from.name
```

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

```
if Helper.hasStereotypeApplied(src, 'SysML::Ports&Flows::FlowProperty') then
    Helper.getFlowDirectionKind(Helper.getTagValue(from,
    'SysML::Ports&Flows::FlowProperty', 'direction'))
else
    invalid
endif
```

7.6.4.2.22 PropertyLike_Mapping

Description

Abstract mapping class that maps a UML4SysML::Element to a Feature.

General Mappings

ElementUnique_Mapping
ToFeature_Init

Mapping Source

Element

Mapping Target

Feature

Applicable filters

(none)

7.6.4.2.23 PropertyLikeNamed_Mapping

Description

Abstract mapping class that maps a UML4SysML::NamedElement to a Feature.

General Mappings

NamedElementUnique_Mapping
PropertyLike_Mapping

Mapping Source

NamedElement

Mapping Target

Feature

Applicable filters

(none)

7.6.4.2.24 PropertyRedefinition_Mapping

Description

Creates a redefinition relationship for the *redefiningFeature()* and the *redefinedFeature()*.

General Mappings

ToRedefinition_Init
PropertySubsetting_Mapping

Mapping Source

Property

Mapping Target

Redefinition

Applicable 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]

  ```
  PropertyCommon_Mapping.getMapped(from)
  ```

7.6.4.2.25 PropertySubsetting_Mapping

Description

Creates a subsetting relationship.

General Mappings

ToSubsetting_Init
Mapping

Mapping Source

Property

Mapping Target

Subsetting

Applicable 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::ownedRelationship () : Relationship [0..*]

  ```
  Set{}
  ```

- Subsetting::subsettedFeature () : Feature [1]

  ```
  PropertyCommon_Mapping.getMapped(from)
  ```

7.6.4.2.26 PropertyToReferenceUsage_Mapping

Description

Creates a reference usage.

General Mappings

Mapping
ToReferenceUsage_Init

Mapping Source

Property

Mapping Target

ReferenceUsage

Applicable 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.type.oclIsUndefined() then
    Set{}
else
    Set{StructuralFeatureToFeatureTyping_Mapping.getMapped(from)}
endif in
```

7.6.4.2.27 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

Mapping Source

Property

Mapping Target

OccurrenceUsage

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.oclIsUndefined() then
        false
    else
        (p.type.oclIsTypeOf(UML::Class) or
        p.type.oclIsTypeOf(UML::Interface) or
        p.type.oclIsKindOf(UML::Actor)) and
        not (p.name.indexOf('base_') > 0) and
        (p.association.oclIsUndefined() 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.

7.6.4.2.28 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

Mapping Source

Property

Mapping Target

ReferenceUsage

Applicable filters

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

```
src.type.oclIsUndefined() 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.

7.6.4.2.29 RegularPropertyFeatureMembership_Mapping

Description

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

General Mappings

StructuralFeatureMembership_Mapping

Mapping Source

StructuralFeature

Mapping Target

FeatureMembership

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 p.association.oclIsUndefined()
```

Mapping rules

The mapping class only has inherited rules. See the mapping classes in the general mapping section for details.

7.6.4.2.30 Slot_Mapping

Description

A UML4SysML::Slot is mapped to a SysML v2 Feature.

General Mappings

PropertyLike_Mapping

Mapping Source

Slot

Mapping Target

Feature

Applicable filters

(none)

7.6.4.2.31 SlotMembership_Mapping

Description

Creates a membership relationship for *memberElement()*.

General Mappings

ElementFeatureMembership_Mapping

Mapping Source

Slot

Mapping Target

FeatureMembership

Applicable filters

(none)

7.6.4.2.32 StructuralFeatureMembership_Mapping

Description

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

General Mappings

ElementFeatureMembership_Mapping

Mapping Source

StructuralFeature

Mapping Target

FeatureMembership

Applicable filters

(none)

7.6.4.2.33 TypeToFeatureTyping_Mapping

Description

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

General Mappings

ToFeatureTyping_Init
Mapping

Mapping Source

Classifier

Mapping Target

FeatureTyping

Applicable 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.oclIsKindOf(UML::PrimitiveType) then
    Helper.getScalarValueType(from)
else if from.oclIsKindOf(UML::Enumeration) then
    Helper.getEnumerationType(from)
else
    Classifier_Mapping.getMapped(from)
endif endif
```

7.6.5 CommonBehavior

7.6.5.1 Overview

Table 4. List of High-Level CommonBehavior Mappings

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
AnyReceiveEvent	Mapping is not specified yet.
CallEvent	The concept of a CallEvent is not supported by SysML v2.
ChangeEvent	CalculationUsage
FunctionBehavior	ActionDefinition
OpaqueBehavior	ActionDefinition
SignalEvent	Mapping is not specified yet.
TimeEvent	CalculationUsage
Trigger	AcceptActionUsage

7.6.5.2 Mapping Specifications

7.6.5.2.1 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

Applicable 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..*]

```
self.oclAsType(ClassCommon_Mapping).ownedRelationship()
->union(from.ownedElement->select(e | e.oclIsKindOf(UML::Parameter))
    ->collect(p |ParameterMembership_Mapping.getMapped(p))->asSet())
->union(from.ownedElement->select(e | e.oclIsKindOf(UML::ParameterSet))
    ->collect(p | ParameterSetMembership_Mapping.getMapped(p))->asSet())
```

7.6.5.2.2 ChangeEvent_Mapping

Description

Main mapping class for the mapping of UML4SysML::ChangeEvent.

```
calc sysMLv1ChangeEvent1 {
  language "language"
  /* change expression */
}
```

General Mappings

Event_Mapping
ToCalculationUsage_Init

Mapping Source

ChangeEvent

Mapping Target

CalculationUsage

Applicable 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..*]

```
self.oclAsType(NamedElementMain_Mapping).ownedRelationship()
->including(ElementOwningMembership_Mapping.getMapped(from.changeExpression))
->reject(r | r.oclIsUndefined())
```

7.6.5.2.3 Event_Mapping

Description

Abstract mapping class for UML4SysML::Event that maps an Event to an Element.

General Mappings

NamedElementMain_Mapping

Mapping Source

Event

Mapping Target

Element

Applicable filters

(none)

7.6.5.2.4 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

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::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..*]

```
self.oclAsType(Behavior_Mapping).ownedRelationship()
->union(from.language
    ->collect(l | OpaqueBehaviorMembership_Mapping.getMapped(from, l))->asSet())
```

7.6.5.2.5 OpaqueBehaviorMembership_Mapping

Description

Creates a membership relationship for *memberElement()*.

General Mappings

ToOwningMembership_Init
ElementUnique_Mapping

Mapping Source

OpaqueBehavior

Mapping Target

OwningMembership with qualifier: language:String

Applicable 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::elementId () : String [1]

```
Helper.createUUID()
```

- OwningMembership::memberElement (in language : String) : Element [1]

```
OpaqueBehaviorSpecification_Mapping.getMapped(from, language)
```

7.6.5.2.6 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
OpaqueExpressionLike_Mapping

Mapping Source

OpaqueBehavior

Mapping Target

TextualRepresentation with qualifier: language:String

Applicable filters

(none)

7.6.5.2.7 TimeEvent_Mapping

Description

Main mapping class for the mapping of UML4SysML::TimeEvent.

```
calc sysMLv1TimeEvent1 {
  language "language"
  /* duration */
}
```

General Mappings

Event_Mapping
ToCalculationUsage_Init

Mapping Source

TimeEvent

Mapping Target

CalculationUsage

Applicable 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..*]

```
self.oclAsType(NamedElementMain_Mapping).ownedRelationship()
->including(ElementOwningMembership_Mapping.getMapped(from.when))
->reject(r | r.oclIsUndefined())
```

7.6.5.2.8 Trigger_Mapping

Description

A UML4SysML::Trigger is mapped to a SysML v2 AcceptActionUsage. Note: the following UML metaclasses are associated to Trigger, so we have to check that we have consistent translation rules for them: ReplyAction State Transition AcceptEventAction

General Mappings

NamedElementUnique_Mapping
ToActionUsage_Init

Mapping Source

Trigger

Mapping Target

AcceptActionUsage

Applicable 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..*]

```
self.oclAsType(NamedElementUnique_Mapping).ownedRelationship()
->including(TriggerParameterMembership_Mapping.getMapped(from))
->including(ParameterMembership_Factory.create())
```

7.6.6 CommonStructure

7.6.6.1 Overview

Table 5. List of High-Level CommonStructure Mappings

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
Abstraction	Dependency
Comment	Comment
Constraint	ConstraintDefinition
Dependency	Dependency

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
ElementImport	MembershipImport
PackageImport	NamespaceImport
Realization	Dependency
Usage	Dependency

7.6.6.2 Mapping Specifications

7.6.6.2.1 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

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::Requirements::Refine')
or Helper.hasStereotypeApplied(src, 'SysML::Requirements::Trace')
or Helper.hasStereotypeApplied(src, 'SysML::Requirements::DeriveReqt')
or Helper.hasStereotypeApplied(src, 'SysML::Requirements::Satisfy')
or Helper.hasStereotypeApplied(src, 'SysML::Allocations::Allocate'))
```

Mapping rules

The mapping class only has inherited rules. See the mapping classes in the general mapping section for details.

7.6.6.2.2 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 SysMLv1Activitiy {
        comment about SysMLv1Activity, SysMLv1Block1
                /* comment body */
```

```
}
comment about SysMLv1Block1, SysMLv1Block /* comment body */
```

General Mappings

ElementMain_Mapping
ToComment_Init

Mapping Source

Comment

Mapping Target

Comment

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::body () : String [1]

  ```
  if from.body->isEmpty() then '' else from.body endif
  ```

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

  ```
  self.oclAsType(ElementUnique_Mapping).ownedRelationship()
      ->union(from.annotatedElement
          ->collect(e | CommentAnnotation_Mapping.getMapped(from, e))->asSet())
      ->union(if Helper.hasStereotypeApplied(src, 'SysML::ModelElements::Problem') then
              Set{MetadataOwningMembership_Factory.create('ModelingMetadata::Issue'
                  ,OrderedSet{'text'}
                  ,Sequence{from._'body'}->reject(b | b.oclIsUndefined())
                  )}
          else
              Set{}
          endif)
      ->union(if Helper.hasStereotypeApplied(src, 'SysML::ModelElements::Rationale') then
              Set{MetadataOwningMembership_Factory.create('ModelingMetadata::Rationale'
                  ,OrderedSet{'text'}
                  ,Sequence{from._'body'}->reject(b | b.oclIsUndefined())
                  )}
          else
              Set{}
          endif)
  ```

7.6.6.2.3 CommentAnnotation_Mapping

Description

The mapping class creates the annotation relationship for the UML4SysML::Comment mapping. Refactoring: as a qualified mapping this one shall be unique

General Mappings

ToAnnotation_Init
UniqueMapping

Mapping Source

Comment

Mapping Target

Annotation with qualifier: annotatedElement:Element

Applicable 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]

  ```
  invalid
  ```

- Annotation::annotatedElement (in annotatedElement : Element) : Element [1]

  ```
  if annotatedElement.oclIsKindOf(UML::PackageableElement)
      or annotatedElement.oclIsKindOf(UML::Comment) then
          ElementMain_Mapping.getMapped(annotatedElement)
      else
          PropertyLike_Mapping.getMapped(annotatedElement)
      endif
  ```

7.6.6.2.4 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

Applicable 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),
        ReturnParameterMembership_Factory.create(
            'result',
            Helper.getScalarValueTypeByName('Boolean')
        )
      })
```

7.6.6.2.5 Dependency_Mapping

Description

A UML4SysML::Dependency relationship is mapped to a SysML v2 Dependency relationship.

General Mappings

DirectedRelationship_Mapping
NamedElementMain_Mapping

Mapping Source

Dependency

Mapping Target

Dependency

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::Requirements::DeriveReqt')
or Helper.hasStereotypeApplied(src, 'SysML::Requirements::Satisfy'))
```

Mapping rules

The mapping class only has inherited rules. See the mapping classes in the general mapping section for details.

7.6.6.2.6 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

Applicable 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::client () : Element [0..*]

```
from.source->collect(e | if e.oclIsKindOf(UML::PackageableElement) then
    ElementMain_Mapping.getMapped(e)
else
    PropertyLike_Mapping.getMapped(e)
endif)
```

- Relationship::supplier () : Element [0..*]

```
from.target->collect(e | if e.oclIsKindOf(UML::PackageableElement) then
    ElementMain_Mapping.getMapped(e)
else
    PropertyLike_Mapping.getMapped(e)
endif)
```

7.6.6.2.7 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
ElementUnique_Mapping

Mapping Source

Element

Mapping Target

Element

Applicable filters

(none)

7.6.6.2.8 ElementMembership_Mapping

Description

TBC (or merged in ElementOwningMembership_Mapping

General Mappings

ToMembership_Init
Mapping

Mapping Source

Element

Mapping Target

Membership

Applicable 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.oclIsKindOf(UML::PackageableElement)
    or from.oclIsKindOf(UML::Comment)
    or from.oclIsKindOf(UML::InteractionFragment) then
    ElementMain_Mapping.getMapped(from)
else
    PropertyLike_Mapping.getMapped(from)
endif
```

- Membership::visibility () : VisibilityKind [1]

```
if (from.oclIsKindOf(UML::NamedElement)) then
    from.oclAsType(UML::NamedElement).visibility
```

```
else
    KerML::VisibilityKind::public
endif
```

7.6.6.2.9 ElementOwningMembership_Mapping

Description

Creates a owning membership relationship for *ownedMemberElement()*.

General Mappings

ElementMembership_Mapping
ToOwningMembership_Init
ElementUnique_Mapping

Mapping Source

Element

Mapping Target

OwningMembership

Applicable 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.memberElement()}
  ```

- OwningMembership::elementId () : String [1]

  ```
  Helper.createUUID()
  ```

7.6.6.2.10 ElementUnique_Mapping

Description

Abstract mapping class for unique mappings of UML4SysML::Element to Element.

General Mappings

UniqueMapping

Mapping Source

Element

Mapping Target

Element

Applicable 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->collect(c| ElementOwningMembership_Mapping.getMapped(c))
    ->reject(r | r.oclIsUndefined())->asSet()
```

- Element::elementId () : String [1]

```
Helper.getID(from)
```

7.6.6.2.11 NamedElementMain_Mapping

Description

The mapping class is the abstract base class for mappings of UML4SysML::NamedElements.

General Mappings

ElementMain_Mapping
NamedElementUnique_Mapping

Mapping Source

NamedElement

Mapping Target

Element

Applicable filters

(none)

7.6.6.2.12 NamedElementUnique_Mapping

Description

Abstract mapping class for unique mappings of UML4SysML::NamedElement to Element.

General Mappings

ElementUnique_Mapping

Mapping Source

NamedElement

Mapping Target

Element

Applicable 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
  ```

7.6.6.2.13 Namespace_Mapping

Description

The mapping class is the abstract base class for UML*4*SysML::Namespace mappings.

General Mappings

ToNamespace_Init
NamedElementMain_Mapping

Mapping Source

Namespace

Mapping Target

Namespace

Applicable 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..*]

  ```
  self.oclAsType(ElementMain_Mapping).ownedRelationship()
  ->union(PackageImport_Mapping.getMappedColl(from.packageImport)->asSet())
  ->union(ElementImport_Mapping.getMappedColl(from.elementImport)->asSet())
  ```

7.6.6.2.14 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

Applicable 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))
```

7.6.7 InformationFlows

7.6.7.1 Overview

Table 6. List of High-Level InformationFlows Mappings

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
InformationFlow	Mapping is not specified yet.
InformationItem	ItemDefinition

7.6.7.2 Mapping Specifications

7.6.7.2.1 InformationItem_Mapping

Description

A UML4SysML::InformationItem is mapped to a SysML v2 ItemDefinition.

General Mappings

Classifier_Mapping

Mapping Source

InformationItem

Mapping Target

ItemDefinition

Applicable filters

(none)

7.6.8 Interactions

7.6.8.1 Overview

Table 7. List of High-Level Interactions Mappings

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
ActionExecutionSpecification	ActionUsage
BehaviorExecutionSpecification	ActionUsage
CombinedFragment	Interaction
ConsiderIgnoreFragment	Interaction
Continuation	Mapping is not specified yet.
DestructionOccurrenceSpecification	Mapping is not specified yet.
ExecutionOccurrenceSpecification	Mapping is not specified yet.
Gate	Mapping is not specified yet.
GeneralOrdering	Mapping is not specified yet.
Interaction	Interaction, Behavior
InteractionConstraint	ConstraintDefinition
InteractionOperand	Interaction, Namespace
InteractionUse	Step
Lifeline	PartUsage
Message	Flow
MessageOccurrenceSpecification	Mapping is not specified yet.
OccurrenceSpecification	Mapping is not specified yet.
PartDecomposition	Step
StateInvariant	Invariant

7.6.8.2 Mapping Specifications

7.6.8.2.1 ActionExecutionSpecification_Mapping

Description

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

General Mappings

ToActionUsage_Init
PropertyLike_Mapping

Mapping Source

ActionExecutionSpecification

Mapping Target

ActionUsage

Applicable filters

(none)

7.6.8.2.2 BehaviorExecutionSpecification_Mapping

Description

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

General Mappings

ToActionUsage_Init
PropertyLike_Mapping

Mapping Source

BehaviorExecutionSpecification

Mapping Target

ActionUsage

Applicable filters

(none)

7.6.8.2.3 CombinedFragment_Mapping

Description

A UML4SysML::CombinedFragment is mapped to a SysMLv2 Interaction.

General Mappings

NamedElementMain_Mapping
ToInteraction_Init

Mapping Source

CombinedFragment

Mapping Target

Interaction

Applicable 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..*]

```
self.oclAsType(ElementMain_Mapping).ownedRelationship()
    ->union(ElementFeatureMembership_Mapping.getMappedColl(
        from.ownedElement->selectByKind(UML::InteractionOperand))->asSet())
    ->union(ElementFeatureMembership_Mapping.getMappedColl(
        from.ownedElement->selectByKind(UML::OccurrenceSpecification))->asSet())
    ->reject(r | r.oclIsUndefined())
```

7.6.8.2.4 ExecutionSpecificationMembership_Mapping

Description

Creates a membership relationship for *memberElement()*.

General Mappings

ToEndFeatureMembership_Init
ElementFeatureMembership_Mapping

Mapping Source

ExecutionSpecification

Mapping Target

EndFeatureMembership

Applicable filters

(none)

7.6.8.2.5 Interaction_Mapping

Description

A UML4SysML::Interaction is mapped to a SysMLv2 Interaction.

General Mappings

Behavior_Mapping
ToInteraction_Init

Mapping Source

Interaction

Mapping Target

Interaction

Applicable 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..*]

```
self.oclAsType(Behavior_Mapping).ownedRelationship()
    ->union(ExecutionSpecificationMembership_Mapping.getMappedColl(
        from.fragment->selectByKind(UML::ExecutionSpecification))->asSet())
    ->union(ElementFeatureMembership_Mapping.getMappedColl(
        from.lifeline)->asSet())
    ->union(ElementFeatureMembership_Mapping.getMappedColl(
        from.ownedElement->selectByKind(UML::MessageOccurrenceSpecification))->asSet())
    ->union(ElementFeatureMembership_Mapping.getMappedColl(
        from.ownedElement->selectByKind(UML::CombinedFragment))->asSet())
    ->union(ElementFeatureMembership_Mapping.getMappedColl(
        from.fragment->selectByKind(UML::StateInvariant))->asSet())
    ->union(ElementFeatureMembership_Mapping.getMappedColl(
        from.fragment->selectByKind(UML::InteractionUse))->asSet())
    ->reject(r | r.oclIsUndefined())
```

7.6.8.2.6 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

Applicable 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.oclIsKindOf(UML::ExecutionSpecification)) in
let occurrencesSpecs: Set(UML::Element) =
    from.ownedElement->select(e | e.oclIsKindOf(UML::OccurrenceSpecification)) in
let continuations: Set(UML::Element) =
    from.ownedElement->select(e | e.oclIsKindOf(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.oclAsType(ElementMain_Mapping).ownedRelationship())
->union(executionOccurrences
    ->collect(e |ExecutionSpecificationMembership_Mapping.getMapped(e))->asSet())
```

7.6.8.2.7 InteractionUse_Mapping

Description

A UML4SysML::InteractionUse is mapped to a SysML v2 Step.

General Mappings

ToStep_Init
PropertyLike_Mapping

Mapping Source

InteractionUse

Mapping Target

Step

Applicable 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(ElementUnique_Mapping).ownedRelationship()

      ->including(TypeToFeatureTyping_Mapping.getMapped(from.refersTo))
  ```

7.6.8.2.8 LifelinePartUsage_Mapping

Description

A UML4SysML::Lifeline is mapped to a SysML v2 PartUsage.

General Mappings

ToPartUsage_Init
PropertyLike_Mapping

Mapping Source

Lifeline

Mapping Target

PartUsage

Applicable 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(ElementUnique_Mapping).ownedRelationship()-
>union(if from.represent.oclIsUndefined() or from.represent.type.oclIsUndefined() then
    Set{}
else
    Set{TypeFeatureTyping_Mapping.getMapped(from.represent.type)}
endif)
```

7.6.8.2.9 Message_Mapping

Description

A UML4SysML::Message is mapped to a SysML v2 ItemFlow.

General Mappings

ToItemFlow_Init
PropertyLike_Mapping

Mapping Source

Message

Mapping Target

Flow

Applicable filters

(none)

7.6.8.2.10 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

Applicable 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(QNameToFeatureTyping_Factory.create('ScalarValues::Boolean'))
```

7.6.9 Packages

7.6.9.1 Overview

Table 8. List of High-Level Packages Mappings

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
Extension	ConnectionDefinition
ExtensionEnd	The mapping of the extension relationship is performed in the context of Stereotype_Mapping.
Image	Mapping is not specified yet.
Model	Package
Package	Package
PackageMerge	The concept of the PackageMerge relationship is not supported by SysML v2.
Profile	Package
ProfileApplication	Mapping is not specified yet.
Stereotype	MetadataDefinition

7.6.9.2 Mapping Specifications

7.6.9.2.1 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
ElementUnique_Mapping

Mapping Source

ElementImport

Mapping Target

MembershipImport

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::visibility () : VisibilityKind [1]

  ```
  Helper.getKerMLVisibilityKind(from.visibility)
  ```

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

  ```
  from.alias
  ```

7.6.9.2.2 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

Applicable 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()
    ->union(if from.viewpoint.oclIsUndefined()
                or from.viewpoint = '' then
                Set{}
            else
                Set{MetadataOwningMembership_Factory.create(
                    'ModelData',
                    OrderedSet{'viewpoint'},
                    Sequence{from.viewpoint}
                )}
            endif)
```

7.6.9.2.3 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

Applicable 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(Namespace_Mapping).ownedRelationship()
    ->union(let types: Set(UML::Type) =
        from.ownedType->reject(t | t.oclIsKindOf(UML::Association)
                and t.memberEnd->exists(e | e.type.oclIsKindOf(UML::UseCase)))
                ->asSet() in
    ElementOwningMembership_Mapping.getMappedColl(types)->asSet())
    ->union(ElementOwningMembership_Mapping.getMappedColl(
        from.packagedElement->reject(e | e.oclIsKindOf(UML::Type))
            ->select(e | Helper.hasMainMapping(e))->asSet()))
    ->union(
        if from.URI.oclIsUndefined() or from.URI = '' then
            Set{}
        else
            Set{MetadataOwningMembership_Factory.create(
                'PackageData'
                ,OrderedSet{'URI'}
                ,Sequence{from.URI})}
        endif
    )->asSet()->reject(r | r.oclIsUndefined())
```

7.6.9.2.4 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
ElementUnique_Mapping

Mapping Source

PackageImport

Mapping Target

NamespaceImport

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::PackageImport) then
    Helper.isInScope(src.oclAsType(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)
  ```

7.6.9.2.5 Profile_Mapping

Description

A UML4SysML::Profile is mapped to a SysML v2 Package.

General Mappings

Package_Mapping

Mapping Source

Profile

Mapping Target

Package

Applicable 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))
  ```

7.6.9.2.6 Stereotype_Mapping

Description

A UML4SysML::Stereotype is mapped to a SysML v2 MetadataDefinition.

General Mappings

Class_Mapping

Mapping Source

Stereotype

Mapping Target

MetadataDefinition

Applicable filters

(none)

7.6.10 SimpleClassifiers

7.6.10.1 Overview

Table 9. List of High-Level SimpleClassifiers 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

7.6.10.2 Mapping Specifications

7.6.10.2.1 Attribute_Mapping

Description

An UML4SysML::Property is mapped to a SysMLv2 AttributeUsage.

General Mappings

PropertyCommon_Mapping

Mapping Source

Property

Mapping Target

AttributeUsage

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::Property) and not
    Helper.hasStereotypeApplied(src.owner,
        'SysML::ConstraintBlocks::ConstraintBlock') then
        let p: UML::Property = src.oclAsType(UML::Property) in
        if p.type.oclIsUndefined() then
            false
        else
         p.type.oclIsKindOf(UML::DataType) and
          (p.association.oclIsUndefined() 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.

7.6.10.2.2 BehavioredClassifier_Mapping

Description

The abstract mapping class maps the abstract metaclass UML4SysML::BehavioredClassifiers to a SysMLv2 Classifier. The mapping class is used by concrete mapping classes, for example, Block_Mapping.

General Mappings

Classifier_Mapping

Mapping Source

BehavioredClassifier

Mapping Target

Classifier

Applicable 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..*]

  ```
  self.oclAsType(Classifier_Mapping).ownedRelationship()
  ->union(if from.classifierBehavior.oclIsUndefined() then
      Set{}
  else
      Set{ElementFeatureMembership_Mapping.getMapped(from.classifierBehavior)}
  endif)
  ```

7.6.10.2.3 BehaviorToActionUsage_Mapping

Description

The BehavioredClassifierToPerformActionUsage_Mapping class creates a PerformActionUsage element to call the transformed SysML v1 classifier behavior.

General Mappings

ToActionUsage_Init
PropertyLikeNamed_Mapping

Mapping Source

Behavior

Mapping Target

ActionUsage

Applicable 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{TypeToFeatureTyping_Mapping.getMapped(from)}
```

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

```
if from.owner.oclIsKindOf(UML::BehavioredClassifier)
    and from.owner.asTypeOf(UML::BehavioredClassifier).classifierBehavior = from then
    'classifierBehavior'
else
    invalid
endif
```

7.6.10.2.4 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

Applicable filters

(none)

7.6.10.2.5 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

Applicable 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::isVariation () : Boolean [1]

  ```
  true
  ```

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

  ```
  self.oclAsType(Classifier_Mapping).ownedRelationship()
      ->union(from.ownedLiteral
  ```

```
->collect(e | EnumerationVariantMembership_Mapping.getMapped(e))
->asSet())
```

7.6.10.2.6 EnumerationLiteral_Mapping

Description

A UML4SysML::EnumerationLiteral is mapped to a SysML v2 EnumerationUsage.

General Mappings

PropertyLikeNamed_Mapping

Mapping Source

EnumerationLiteral

Mapping Target

EnumerationUsage

Applicable filters

(none)

7.6.10.2.7 EnumerationVariantMembership_Mapping

Description

The EnumerationVariantMembership_Mapping class creates the variant membership relationship between the enumeration definition and a enumeration usage.

General Mappings

ToOwningMembership_Init
ElementOwningMembership_Mapping

Mapping Source

EnumerationLiteral

Mapping Target

VariantMembership

Applicable 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
  ```

7.6.10.2.8 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)
  ```

7.6.10.2.9 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
NamedElementUnique_Mapping

Mapping Source

Interface

Mapping Target

ConjugatedPortDefinition

Applicable 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)}
  ```

7.6.10.2.10 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

ElementOwningMembership_Mapping

Mapping Source

Interface

Mapping Target

OwningMembership

Applicable 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::memberElement () : Element [1]

```
InterfaceConjugatedPortDefinition_Mapping.getMapped(from)
```

7.6.10.2.11 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
NamedElementUnique_Mapping

Mapping Source

Interface

Mapping Target

PortConjugation

Applicable 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 () : PortDefinition [1]

  ```
  from
  ```

- PortConjugation::conjugatedType () : Type [1]

  ```
  SysMLv2::ConjugatedPortDefinition.allInstances()
  ->collect(cpd | cpd.owningRelationship)
  ->select(r | r.oclIsKindOf(SysMLv2::Membership))
  ->any(m | m.memberName = from.name)
  ```

7.6.10.2.12 InterfaceRealization_Mapping

Description

A UML4SysML::InterfaceRealization is mapped to a SysMLv2 Subclassification relationship.

General Mappings

ToSpecialization_Init
NamedElementUnique_Mapping

Mapping Source

InterfaceRealization

Mapping Target

Subclassification

Applicable 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)
  ```

7.6.10.2.13 PrimitiveType_Mapping

Description

The PrimitiveType_Mapping class maps a UML4SysML::PrimitiveType from the standard SysML v1 library to its counterpart AttributeDefinition in the standard SysML v2 library.

General Mappings

NamedElementUnique_Mapping

Mapping Source

PrimitiveType

Mapping Target

AttributeDefinition

Applicable filters

(none)

Mapping rules

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

- AttributeDefinition::_to () : AttributeDefinition [1]

  ```
  let pt: SysML::AttributeDefinition = Helper.getScalarValueType(from) in
  if pt.oclIsUndefined() then
      PrimitiveType2_Mapping.getMapped(from)
  else
      pt
  endif
  ```

7.6.10.2.14 PrimitiveType2_Mapping

Description

The PrimitiveType2_Mapping class maps a user defined UML4SysML::PrimitiveType that is not part of the ones defined in the SysML v1 standard library to a SysML v2 AttributeDefinition.

General Mappings

DataType_Mapping

Mapping Source

PrimitiveType

Mapping Target

AttributeDefinition

Applicable filters

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

```
let predefined: Set(String)
    = Set{'Boolean', 'Integer', 'Real', 'String', 'UnlimitedNatural'} in
    predefined->excludes(from.name)
```

Mapping rules

The mapping class only has inherited rules. See the mapping classes in the general mapping section for details.

7.6.10.2.15 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

Applicable 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(ElementUnique_Mapping).ownedRelationship()

      ->including(TypeToFeatureTyping_Mapping.getMapped(from.signal))
  ```

7.6.10.2.16 Signal_Mapping

Description

A UML4SysML::Signal is mapped to a SysML v2 ItemDefinition.

General Mappings

Classifier_Mapping

Mapping Source

Signal

Mapping Target

ItemDefinition

Applicable filters

(none)

7.6.11 StateMachines

7.6.11.1 Overview

Table 10. List of High-Level StateMachines Mappings

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
ConnectionPointReference	StateUsage
FinalState	StateUsage
Pseudostate	ActionUsage, StateUsage
Region	StateUsage
State	StateUsage
StateMachine	StateDefinition
Transition	TransitionUsage

7.6.11.2 Mapping Specifications

7.6.11.2.1 CommonPseudostate_Mapping

Description

Abstract mapping class for common rules for pseudostates mappings.

General Mappings

PropertyLikeNamed_Mapping
ToActionUsage_Init

Mapping Source

Pseudostate

Mapping Target

ActionUsage

Applicable 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..*]

```
self.oclAsType(PropertyLike_Mapping).ownedRelationship()
    ->union((from.ownedElement - from.ownedComment)
        ->collect(e | ElementFeatureMembership_Mapping.getMapped(e))
        ->asSet())
```

7.6.11.2.2 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

Applicable 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 toFeatureMS : Set(UML::Element) =
    from.ownedElement->select(e | e.oclIsKindOf(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())
```

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

```
false
```

7.6.11.2.3 DoBehaviorStateSubactionMembership_Mapping

Description

Creates a state subaction membership relationship for *memberFeature()*.

General Mappings

StateBehaviorStateSubactionMembership_Mapping

Mapping Source

Behavior

Mapping Target

StateSubactionMembership

Applicable 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
```

7.6.11.2.4 EntryBehaviorStateSubactionMembership_Mapping

Description

Creates a state subaction membership relationship for *memberFeature()*.

General Mappings

StateBehaviorStateSubactionMembership_Mapping

Mapping Source

Behavior

Mapping Target

StateSubactionMembership

Applicable 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
  ```

7.6.11.2.5 ExitBehaviorStateSubactionMembership_Mapping

Description

Creates a state subaction membership relationship for *memberFeature()*.

General Mappings

StateBehaviorStateSubactionMembership_Mapping

Mapping Source

Behavior

Mapping Target

StateSubactionMembership

Applicable 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
  ```

7.6.11.2.6 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

Applicable filters

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

```
src.oclIsTypeOf(UML::FinalState)
```

Mapping rules

The mapping class only has inherited rules. See the mapping classes in the general mapping section for details.

7.6.11.2.7 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

Applicable filters

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

```
(src.kind = UML::PseudostateKind::initial)
```

Mapping rules

The mapping class only has inherited rules. See the mapping classes in the general mapping section for details.

7.6.11.2.8 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

Applicable filters

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

```
(src.kind <> UML::PseudostateKind::initial)
```

Mapping rules

The mapping class only has inherited rules. See the mapping classes in the general mapping section for details.

7.6.11.2.9 Region_Mapping

Description

A UML4SysML::Region is mapped to SysML v2 StateUsage.

General Mappings

PropertyLikeNamed_Mapping
ToStateUsage_Init

Mapping Source

Region

Mapping Target

StateUsage

Applicable 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..*]

```
self.oclAsType(ElementUnique_Mapping).ownedRelationship()
    ->union((from.ownedElement - from.ownedComment)
        ->collect(e | ElementFeatureMembership_Mapping.getMapped(e))
        ->asSet())
    ->reject(e | e.oclIsUndefined())
```

7.6.11.2.10 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

PropertyLikeNamed_Mapping
ToStateUsage_Init

Mapping Source

State

Mapping Target

StateUsage

Applicable 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..*]

```
self.oclAsType(ElementUnique_Mapping).ownedRelationship()
->union(if (from.entry.oclIsUndefined()) then
        Set{}
    else
    Set{EntryBehaviorStateSubactionMembership_Mapping.getMapped(from.entry)}
    endif)
->union(if (from.doActivity.oclIsUndefined()) then
```

```
        Set{}
    else
        Set{DoBehaviorStateSubactionMembership_Mapping.getMapped(from.doActivity)}
    endif)
->union(if (from.exit.oclIsUndefined()) then
        Set{}
    else
        Set{ExitBehaviorStateSubactionMembership_Mapping.getMapped(from.exit)}
    endif)
->reject(e | e.oclIsUndefined())
```

- StateUsage::isParallel () : Boolean [1]

  ```
  from.isComposite
  ```

7.6.11.2.11 StateBehaviorStateSubactionMembership_Mapping

Description

Abstract mapping class for mapping classes for state behavior mappings (enty, do and exit).

General Mappings

ToStateSubactionMembership_Init
ElementFeatureMembership_Mapping

Mapping Source

Behavior

Mapping Target

StateSubactionMembership

Applicable filters

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

```
from.owner.oclIsKindOf(UML::State)
```

Mapping rules

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

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

  ```
  from.name
  ```

7.6.11.2.12 StateDefinition_Mapping

Description

A UML4SysML::StateMachine is mapped to a SysML v2 StateDefinition.

General Mappings

Behavior_Mapping

Mapping Source

StateMachine

Mapping Target

StateDefinition

Applicable 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::isParallel () : Boolean [1]

```
from.region->size() > 1
```

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

```
self.oclAsType(ElementUnique_Mapping).ownedRelationship()
    ->union((from.ownedElement - from.ownedComment)
        ->collect(e | ElementOwningMembership_Mapping.getMapped(e))
        ->asSet())
    ->reject(e | e.oclIsUndefined())
```

7.6.11.2.13 Transition_Mapping

Description

A UML4SysML::Transition is mapped to a SysML v2 TransitionUsage.

General Mappings

PropertyLikeNamed_Mapping
ToTransitionUsage_Init

Mapping Source

Transition

Mapping Target

TransitionUsage

Applicable 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(ElementUnique_Mapping).ownedRelationship()
    ->union(((from.ownedElement - from.ownedComment) - from.trigger)
        ->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 | ElementFeatureMembership_Mapping.getMapped(e))
    ->asSet())
    ->including(TransitionSuccession_Mapping.getMapped(from))
```

7.6.11.2.14 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
PropertyLikeNamed_Mapping

Mapping Source

Transition

Mapping Target

Succession

Applicable 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{EndFeatureMembership_Mapping.getMapped(from.source),
EndFeatureMembership_Mapping.getMapped(from.target)}
```

7.6.12 StructuredClassifiers

7.6.12.1 Overview

Table 11. List of High-Level StructuredClassifiers Mappings

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
Association	ConnectionDefinition
AssociationClass	OccurrenceDefinition, ConnectionDefinition
Class	OccurrenceDefinition

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
Connector	ConnectionUsage
ConnectorEnd	Feature
Port	OccurrenceUsage, PortUsage, Feature, AttributeUsage

7.6.12.2 Mapping Specifications

7.6.12.2.1 Association_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

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 not Helper.isConnectionDef(src)
```

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..*]

  ```
  self.oclAsType(Classifier_Mapping).ownedRelationship()->asOrderedSet()
  ->union(EndFeatureMembership_Mapping.getMappedColl(from.memberEnd)->asOrderedSet())
  ->asOrderedSet()
  ```

7.6.12.2.2 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

ClassCommon_Mapping

Mapping Source

Class

Mapping Target

OccurrenceDefinition

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')
and not Helper.hasStereotypeApplied(src, 'SysML::ConstraintBlocks::ConstraintBlock')
```

Mapping rules

The mapping class only has inherited rules. See the mapping classes in the general mapping section for details.

7.6.12.2.3 ClassCommon_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

BehavioredClassifier_Mapping

Mapping Source

Class

Mapping Target

OccurrenceDefinition

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 Helper.hasStereotypeApplied(src, 'SysML::ModelElements::Stakeholder')
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.

7.6.12.2.4 ConnectionDefEnd_Mapping

Description

A UML4SysML::Property element is mapped to a SysML v2 Feature.

General Mappings

UniqueMapping
End_Mapping

Mapping Source

Property

Mapping Target

Feature

Applicable filters

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

```
not src.oclAsType(UML::Property).association.oclIsUndefined()
and Helper.isConnectionDef(src.oclAsType(UML::Property).association)
```

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{MultiplicityMembership_Factory.create(from.lower,from.upper)}
    else
        Set{}
    endif in
let typings: Set(KerML::FeatureTyping) = if from.type.oclIsUndefined() then
    Set{}
else
    Set{TypeToFeatureTyping_Mapping.getMapped(from.type)}
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.oclIsUndefined() then
        Set{}
```

```
    else
        Set{DefaultValue_Mapping.getMapped(from)}
    endif in
crossFMultiplicity->union(typings)
    ->union(subsettings)->union(defaultValue)
    ->including(MultiplicityMembership_Factory.create(1,1))->asSet()
```

7.6.12.2.5 ConnectionEndToSubsetting_Mapping

Description

Creates a subsetting relationship.

General Mappings

ToSubsetting_Init
ElementUnique_Mapping

Mapping Source

ConnectorEnd

Mapping Target

Subsetting

Applicable 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::subsettedFeature () : Feature [1]

  ```
  ConnectorEndToSubsettedFeature_Mapping.getMapped(from)
  ```

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

  ```
  let propertyPath: OrderedSet(UML::Property) =
      Helper.getTagValueAsElementColl
          (from, 'SysML::Blocks::NestedConnectorEnd','propertyPath')
      ->asOrderedSet() in
  if propertyPath->notEmpty() then
          OrderedSet{self.oclAsType(ElementUnique_Mapping).ownedRelationship()
          ,ConnectorEndToSubsettedFeatureMembership_Mapping.getMapped(from)}
  else
          OrderedSet{self.oclAsType(ElementUnique_Mapping).ownedRelationship()}
  endif
  ```

7.6.12.2.6 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

PropertyLikeNamed_Mapping
ToConnector_Init

Mapping Source

Connector

Mapping Target

ConnectionUsage

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

  ```
  from.name
  ```

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

  ```
  let typings: Set(KerML::FeatureTyping) =
      if from.type.oclIsUndefined() then
          Set{}
      else
          Set{TypeToFeatureTyping_Mapping.getMapped(from.type)}
      endif in
  let redefinitions: Set(KerML::Redefinition) = from.redefinedProperty
      ->collect(p | PropertyRedefinition_Mapping.getMapped(p))->asSet() in
  let endMBS: Set(KerML::FeatureMembership) =
      from.end->collect(e | ElementFeatureMembership_Mapping.getMapped(e))
      ->asSet() in
          self.oclAsType(ElementUnique_Mapping).ownedRelationship()
      ->union(typings)->union(redefinitions)->union(endMBS)
      ->asSet()
  ```

7.6.12.2.7 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

Applicable 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
  ```

7.6.12.2.8 ConnectorEndToOwnedFeature_Mapping

Description

The mapping class creates the SysML v2 Feature element for the UML4SysML::ConnectorEnd mapping.

General Mappings

ConnectorEndToFeatureCommon_Mapping
PropertyLikeNamed_Mapping

Mapping Source

ConnectorEnd

Mapping Target

Feature

Applicable 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
let rel: OrderedSet(KerML::Relationship) = OrderedSet{
    MultiplicityMembership_Factory.create(from.lower, from.upper)
    ,self.oclAsType(ElementUnique_Mapping).ownedRelationship()} in
if subsetting.oclIsUndefined() then
    rel
else
    rel->including(subsetting)
endif
```

7.6.12.2.9 ConnectorEndToSubsettedFeature_Mapping

Description

The mapping class maps UML4SysML::ConnectorEnd that are part of a SysML::Ports&Flows::NestedConnectorEnd.

General Mappings

ConnectorEndToFeatureCommon_Mapping
ElementUnique_Mapping

Mapping Source

ConnectorEnd

Mapping Target

Feature

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::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_Factory.create(from.lower, from.upper)})
```

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

```
'featureChain'
```

7.6.12.2.10 ConnectorType_Mapping

Description

A UML4SysML::Association is mapped to a SysML v2 ConnectionDefinition.

General Mappings

Classifier_Mapping
Relationship_Mapping

Mapping Source

Association

Mapping Target

ConnectionDefinition

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 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..*]

```
self.oclAsType(Classifier_Mapping).ownedRelationship()
    ->asOrderedSet()
    ->union(from.memberEnd
        ->collect(e | EndFeatureMembership_Mapping.getMapped(e))
        ->asOrderedSet())
    ->asOrderedSet()
```

7.6.12.2.11 CrossSubsetting_Mapping

Description

Creates a subsetting relationship.

General Mappings

ToSubsetting_Init
UniqueMapping

Mapping Source

Property

Mapping Target

CrossSubsetting

Applicable 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::subsettedFeature () : Feature [1]

  ```
  NonOwnedEnd_Mapping.getMapped(from)
  ```

7.6.12.2.12 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

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
not src.oclAsType(UML::Property).association.oclIsUndefined()
```

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
  ```

7.6.12.2.13 EndFeatureMembership_Mapping

Description

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

General Mappings

ElementMembership_Mapping
ToEndFeatureMembership_Init
ElementUnique_Mapping

Mapping Source

Element

Mapping Target

EndFeatureMembership

Applicable filters

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

```
src.oclIsKindOf(UML::TypedElement)
or src.oclIsKindOf(UML::Feature)
or src.oclIsKindOf(UML::Classifier)
or src.oclIsKindOf(UML::Action)
```

Mapping rules

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

- EndFeatureMembership::memberElement () : Element [1]

```
if from.oclIsKindOf(UML::Classifier) then
    ClassifierToFeature_Mapping.getMapped(from)
else if from.oclIsKindOf(UML::PackageableElement)
    or from.oclIsKindOf(UML::Comment) then
    ElementMain_Mapping.getMapped(from)
else if from.oclIsKindOf(UML::Property)
    and not from.association.oclIsUndefined()
    and from.association.ownedEnd->includes(from) then
    PropertyLike_Mapping.getMapped(from)
else
    NonOwnedEnd_Mapping.getMapped(from)
endif endif endif
```

- EndFeatureMembership::elementId () : String [1]

```
Helper.getID(from)+'_FMS'
```

- EndFeatureMembership::visibility () : VisibilityKind [1]

```
if from.oclIsKindOf(UML::NamedElement) then
Helper.getKerMLVisibilityKind(from.oclAsType(UML::NamedElement).visibility)
else KerML::VisibilityKind::public endif
```

7.6.12.2.14 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

ToFeature_Init
Mapping

Mapping Source

NamedElement

Mapping Target

Feature

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

```
'nonOwnedEnd'
```

- Feature::isEnd () : Boolean [1]

```
true
```

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

```
let mult: KerML::Membership = if from.oclIsKindOf(UML::MultiplicityElement) then
    MultiplicityMembership_Factory.create(from.lower, from.upper)
else
    MultiplicityMembership_Factory.create(1, 1)
endif in

let feature: KerML::Feature = if from.oclIsKindOf(UML::Action) then
    CommonAction_Mapping.getMapped(from)
else if from.type.oclIsKindOf(UML::DataType) then
    Attribute_Mapping.getMapped(from)
else
   PropertyTypedByClassInterface_Mapping.getMapped(from)
endif endif in
Set{mult, Subsetting_Factory.create(feature)}
```

7.6.12.2.15 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

Mapping Source

Property

Mapping Target

Feature

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
let association: UML::Association = p.association in

not association.oclIsUndefined()
and association.ownedEnd->includes(p)
and not Helper.isConnectionDef(association)
and p.association.memberEnd->collect(e | e.type)->selectByKind(UML::UseCase)->isEmpty()
```

Mapping rules

The mapping class only has inherited rules. See the mapping classes in the general mapping section for details.

7.6.12.2.16 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

Mapping Source

Port

Mapping Target

PortUsage

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, 'SysML::Ports&Flows::FullPort')
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
```

Mapping rules

The mapping class only has inherited rules. See the mapping classes in the general mapping section for details.

7.6.12.2.17 PropertyToFeatureChaining_Mapping

Description

The mapping class creates the SysML v2 FeatureChaining for the UML4SysML::Property mapping.

General Mappings

ToFeatureChaining_Init
Mapping

Mapping Source

Property

Mapping Target

FeatureChaining

Applicable 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]

  ```
  PropertyCommon_Mapping.getMapped(from)
  ```

7.6.13 UseCases

7.6.13.1 Overview

Table 12. List of High-Level UseCases Mappings

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
Actor	PartDefinition
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.
Include	Mapping is not specified yet.
UseCase	UseCaseDefinition

7.6.13.2 Mapping Specifications

7.6.13.2.1 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

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::Stakeholder')
```

Mapping rules

The mapping class only has inherited rules. See the mapping classes in the general mapping section for details.

7.6.13.2.2 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
PropertyLike_Mapping

Mapping Source

Include

Mapping Target

IncludeUseCaseUsage

Applicable 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..*]

  ```
  self.oclAsType(ElementUnique_Mapping).ownedRelationship()
  ->including(TypeToFeatureTyping_Mapping.getMapped(from.addition))
  ```

7.6.13.2.3 IncludeMembership_Mapping

Description

Creates a membership relationship for *memberElement()*.

General Mappings

ElementFeatureMembership_Mapping

Mapping Source

Include

Mapping Target

FeatureMembership

Applicable filters

(none)

7.6.13.2.4 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

Mapping Source

UseCase

Mapping Target

UseCaseDefinition

Applicable 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 actors : Set(UML::Property) =
    UML::Association.allInstances()
        ->select( a | a.memberEnd->exists(e | e.type = from))
        ->collect( a | a.memberEnd)->flatten()
        ->select( m | m.type <> from)->asSet() in
let extensionPoints : Sequence(UML::Element) =
    from.ownedElement->select(e | e.oclIsKindOf(UML::ExtensionPoint)) in
let extend : Sequence(UML::Element) =
    from.ownedElement->select(e | e.oclIsKindOf(UML::Extend)) in
let include : Sequence(UML::Element) =
    from.ownedElement->select(e | e.oclIsKindOf(UML::Include)) in

self.oclAsType(BehavioredClassifier_Mapping).ownedRelationship()
->union(from.subject->collect(s |
    let type: KerML::Classifier = Classifier_Mapping.getMapped(s) in
        ParameterMembership_Factory.create(s.name.toLowerCase(), type, Set{}))
        ->asSet())
->union(ElementFeatureMembership_Mapping.getMappedColl(from.include)->asSet())
->union(UseCaseActorMembership_Mapping.getMappedColl(actors)->asSet())
```

7.6.13.2.5 UseCaseActorMembership_Mapping

Description

Creates a membership relationship for *memberElement()*.

General Mappings

ToActorMembership_Init
ElementFeatureMembership_Mapping

Mapping Source

Property

Mapping Target

ActorMembership

Applicable filters

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

```
from.type.oclIsKindOf(UML::Actor)
```

Mapping rules

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

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

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

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

  ```
  Set{}
  ```

7.6.14 Values

7.6.14.1 Overview

Table 13. List of High-Level Values Mappings

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
Duration	Expression
DurationConstraint	ConstraintDefinition
DurationInterval	Expression
DurationObservation	Mapping is not specified yet.
Expression	Expression, OperatorExpression
Interval	Expression
IntervalConstraint	ConstraintDefinition
LiteralBoolean	LiteralBoolean
LiteralInteger	LiteralInteger
LiteralNull	NullExpression
LiteralReal	LiteralRational

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
LiteralString	LiteralString
LiteralUnlimitedNatural	LiteralInfinity
OpaqueExpression	CalculationUsage
StringExpression	Expression, OperatorExpression
TimeConstraint	ConstraintDefinition
TimeExpression	TriggerInvocationExpression
TimeInterval	Expression
TimeObservation	Mapping is not specified yet.

7.6.14.2 Mapping Specifications

7.6.14.2.1 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

Applicable 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
  ```

7.6.14.2.2 LiteralBoolean_Mapping

Description

The mapping class maps UML4SysML::LiteralBoolean to SysML v2 LiteralBoolean.

General Mappings

LiteralSpecificationCommon_Mapping

Mapping Source

LiteralBoolean

Mapping Target

LiteralBoolean

Applicable 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
  ```

7.6.14.2.3 LiteralInteger_Mapping

Description

The mapping class maps UML4SysML::LiteralInteger to SysML v2 LiteralInteger.

General Mappings

LiteralSpecificationCommon_Mapping

Mapping Source

LiteralInteger

Mapping Target

LiteralInteger

Applicable 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
  ```

7.6.14.2.4 LiteralNull_Mapping

Description

The mapping class maps UML4SysML::LiteralNull to SysML v2 NullExpression.

General Mappings

LiteralSpecificationCommon_Mapping

Mapping Source

LiteralNull

Mapping Target

NullExpression

Applicable filters

(none)

7.6.14.2.5 LiteralReal_Mapping

Description

The mapping class maps UML4SysML::LiteralReal to SysML v2 LiteralRational.

General Mappings

LiteralSpecificationCommon_Mapping

Mapping Source

LiteralReal

Mapping Target

LiteralRational

Applicable 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
  ```

7.6.14.2.6 LiteralSpecificationCommon_Mapping

Description

The mapping class the is abstract base class for all concrete UML4SysML::LiteralSpecification mappings.

General Mappings

ValueSpecification_Mapping

Mapping Source

LiteralSpecification

Mapping Target

LiteralExpression

Applicable filters

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

```
not src.owner.oclIsKindOf(UML::MultiplicityElement)
```

Mapping rules

The mapping class only has inherited rules. See the mapping classes in the general mapping section for details.

7.6.14.2.7 LiteralString_Mapping

Description

The mapping class maps UML4SysML::LiteralString to the SysML v2 LiteralString.

General Mappings

LiteralSpecificationCommon_Mapping

Mapping Source

LiteralString

Mapping Target

LiteralString

Applicable 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
  ```

7.6.14.2.8 LiteralUnlimitedUnbounded_Mapping

Description

The mapping class maps UML4SysML::LiteralUnlimited to SysML v2 LiteralInfinity if it is the unlimited value.

General Mappings

LiteralSpecificationCommon_Mapping

Mapping Source

LiteralUnlimitedNatural

Mapping Target

LiteralInfinity

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.

7.6.14.2.9 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

Applicable filters

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

```
from.value >= 0
```

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
  ```

7.6.14.2.10 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

ToCalculationUsage_Init
ValueSpecification_Mapping

Mapping Source

OpaqueExpression

Mapping Target

CalculationUsage

Applicable filters

(none)

7.6.14.2.11 OpaqueExpressionLike_Mapping

Description

Abstract mapping class for UML4SysML::OpaqueExpression that maps Element to TextualRepresentation.

General Mappings

ToTextualRepresentation_Init
ElementUnique_Mapping

Mapping Source

Element

Mapping Target

TextualRepresentation with qualifier: language:String

Applicable 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
  ```

7.6.14.2.12 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

OpaqueExpressionLike_Mapping

Mapping Source

OpaqueExpression

Mapping Target

TextualRepresentation with qualifier: language:String

Applicable filters

(none)

7.6.14.2.13 ReturnParameterMembership_Mapping

Description

Creates a membership relationship for *memberElement()*.

General Mappings

ToReturnParameterMembership_Init
Mapping

Mapping Source

Classifier

Mapping Target

ReturnParameterMembership

Applicable 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::memberElement () : Feature [0..1]

```
Parameter_Mapping.getMapped(from)
```

7.6.14.2.14 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

TriggerInvocationExpression

Applicable 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::kind () : TriggerKind [1]

```
SYSML2::TriggerKing::when
```

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

```
self.oclAsType(ValueSpecification_Mapping).ownedRelationship()
->union(if from.expr.oclIsUndefined() then
  Set{}
else
  Set{ElementOwningMembership_Mapping.getMapped(from.expr)}
endif)
```

7.6.14.2.15 TypeToReturnParameterFeature_Mapping

Description

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

General Mappings

ToFeature_Init
Mapping

Mapping Source

Classifier

Mapping Target

Feature

Applicable 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{TypeToFeatureTyping_Mapping.getMapped(from)}
```

7.6.14.2.16 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

Applicable 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..*]

```
self.oclAsType(ElementMain_Mapping).ownedRelationship()
    ->including(ValueSpecificationToResultMembership_Mapping.getMapped(from))
```

7.6.14.2.17 ValueSpecificationToResult_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
PropertyLikeNamed_Mapping

Mapping Source

ValueSpecification

Mapping Target

ReferenceUsage

Applicable 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.type.oclIsUndefined() then
    Set{}
else if from.type.oclIsKindOf(UML::Enumeration) then
    let type: SysML2::EnumerationDefinition =
         Helper.getEnumerationType(from.type) in
    if type.oclIsUndefined() then
        Set{}
    else
        Set{TypeToFeatureTyping_Factory.create(type)}
    endif
else
    Set{TypeToFeatureTyping_Mapping.getMapped(from.type)}
endif endif)
->including(MultiplicityMembership_Factory.create(1, 1))->asSet()
```

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

```
'result'
```

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

```
Helper.getKerMLParameterDirectionKind(UML::ParamterDirectionKind::return)
```

7.6.14.2.18 ValueSpecificationToResultMembership_Mapping

Description

Creates a membership relationship for *memberElement()*.

General Mappings

ElementUnique_Mapping
ToReturnParameterMembership_Init

Mapping Source

ValueSpecification

Mapping Target

ReturnParameterMembership

Applicable 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::memberElement () : Element [1]

  ```
  ValueSpecificationToResult_Mapping.getMapped(from)
  ```

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

  ```
  Set{}
  ```

7.7 Mappings from SysML v1.7 stereotypes

7.7.1 Overview

7.7.2 Activities

7.7.2.1 Overview

Table 14. List of High-Level Activities Mappings

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
Continuous	Mapping is not specified yet.
ControlOperator	The concept that an action can control other actions is not supported by SysML v2.
Discrete	Mapping is not specified yet.
NoBuffer	Mapping is not specified yet.

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
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.
Probability	Mapping is not specified yet.
Rate	Mapping is not specified yet.

7.7.2.2 Mapping Specifications

7.7.2.2.1 Model Libraries

7.7.2.2.1.1 ControlValues

7.7.2.2.1.1.1 ControlValueKind

The enumeration ControlValueKind is mapped to the SysML v2 enumeration definition SysMLv1Library::Enumerations::ControlValueKind (see 7.3.2).

7.7.3 Allocations

7.7.3.1 Overview

Table 15. List of High-Level Allocations Mappings

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
Allocate	Mapping not specified yet.
AllocationUsage	Mapping not specified yet.
AllocateActivityPartition	Mapping not specified yet.

7.7.3.2 Mapping Specifications

7.7.3.2.1 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
Dependency_Mapping

Mapping Source

Abstraction

Mapping Target

AllocationUsage

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.

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

```
self.oclAsType(Dependency_Mapping).ownedRelationship()->asOrderedSet()
->union(EndFeatureMembership_Mapping.getMappedColl(from.source)->asOrderedSet())
->union(EndFeatureMembership_Mapping.getMappedColl(from.target)->asOrderedSet())
```

7.7.4 Blocks

7.7.4.1 Overview

Table 16. List of High-Level Allocations Mappings

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
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.
BindingConnector	
BindingConnectorAsUsage	
Block	PartDefinition
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.

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
ParticipantProperty	Mapping is not specified yet.
PropertySpecificType	Mapping is not specified yet.
ValueType	AttributeDefinition

7.7.4.2 Mapping Specifications

7.7.4.2.1 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

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.

7.7.4.2.2 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

ClassCommon_Mapping

Mapping Source

Class

Mapping Target

PartDefinition

Applicable filters

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

```
not src.oclIsTypeOf(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

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

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

```
self.oclAsType(ClassCommon_Mapping).ownedRelationship()
    ->including( let values: Sequence(OclAny) =
        Sequence{Helper.getTagValueAsElementColl(
            from,
            'SysML::Blocks::Block',
            'isEncapsulated')} in
    MetadataOwningMembership_Factory.create(
        'SysMLv1Library::BlockData',
        OrderedSet{'isEncapsulated'}, values))
```

7.7.5 ConstraintBlocks

7.7.5.1 Overview

Table 17. List of High-Level SysML v1 ConstraintBlocks Mappings

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
ConstraintBlock	ConstraintDefinition

7.7.5.2 Mapping Specifications

7.7.5.2.1 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

ClassCommon_Mapping

Mapping Source

Class

Mapping Target

ConstraintDefinition

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..*]

```
self.oclAsType(Namespace_Mapping).ownedRelationship()
    ->union(ConstraintParameterMembership_Mapping.getMappedColl(
        from.ownedAttribute)->asSet())
    ->union(ElementFeatureMembership_Mapping.getMappedColl(from.ownedRule)
    ->asSet())
    ->union(from.generalization->collect(e | Generalization_Mapping.getMapped(e))
    ->asSet())
```

7.7.5.2.2 ConstraintParameter_Mapping

Description

The mapping class maps SysML v1 constraint parameter to SysML v2 attribute usages.

General Mappings

PropertyCommon_Mapping

Mapping Source

Property

Mapping Target

ReferenceUsage

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::Property) and
Helper.hasStereotypeApplied(src.owner, 'SysML::ConstraintBlocks::ConstraintBlock') then
    let p: UML::Property = src.oclAsType(UML::Property) in
    if p.type.oclIsUndefined() 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.

7.7.5.2.3 ConstraintParameterMembership_Mapping

Description

Creates a membership relationship for *memberElement()*.

General Mappings

ElementMembership_Mapping
ToFeatureMembership_Init
ElementUnique_Mapping

Mapping Source

Property

Mapping Target

FeatureMembership

Applicable 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::memberElement () : Element [1]

  ```
  ConstraintParameter_Mapping.getMapped(from)
  ```

- FeatureMembership::visibility () : VisibilityKind [1]

  ```
  Helper.getKerMLVisibilityKind(from.visibility)
  ```

7.7.6 Model Elements

7.7.6.1 Overview

Table 18. List of High-Level SysML v1 ModelElements Mappings

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
Conform	Mapping is not specified yet.
ElementGroup	Package
Expose	Mapping is not specified yet.
Problem	Comment
Rationale	Comment
Stakeholder	ItemDefinition
View	Mapping is not specified yet.
Viewpoint	Mapping is not specified yet.

7.7.6.2 Mapping Specifications

7.7.6.2.1 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

ToPackage_Init
ElementMain_Mapping

Mapping Source

Comment

Mapping Target

Package

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')
and not Helper.hasStereotypeApplied(src, 'SysML::ModelElements::Problem')
and not 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.

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

  ```
  Helper.getTagValueAsString(from, 'SysML::ModelElements::ElementGroup', 'name')
  ```

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

  ```
  let criterion: String = Helper.getTagValueAsString(from, 'ElementGroup', 'criterion') in
  self.oclAsType(ElementMain_Mapping).ownedRelationship()
  ->union(Helper.getTagValueAsElementColl(from,
          'SysML::ModelElements::ElementGroup', 'member')
      ->collect(e | ElementMembership_Mapping.getMapped(e))->asSet())
  ->including(MetadataOwningMembership_Factory.create('ElementGroupData'
      ,OrderedSet{'criterion'}
      ,Sequence{criterion}->reject(c | c.oclIsUndefined())
      ))->asSet()
  ```

7.7.6.2.2 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

Classifier_Mapping

Mapping Source

Classifier

Mapping Target

ItemDefinition

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..*]

```
self.oclAsType(Classifier_Mapping).ownedRelationship()
    ->including(MetadataOwningMembership_Factory.create(
        'SysMLv1Library::StakeholderData',
        OrderedSet{'isStakeholder'}, Sequence{true}))
```

7.7.7 PortsAndFlows

7.7.7.1 Overview

Table 19. List of High-Level Ports & Flows Mappings

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
AcceptChangeStructuralFeatureEventAction	AcceptActionUsage
AddFlowPropertyValueOnNestedPortAction	Mapping is not specified yet.
ChangeStructuralFeatureEvent	Mapping is not specified yet.
DirectedFeature	PerformActionUsage
FlowProperty	AttributeUsage, OccurrenceUsage, ReferenceUsage
FullPort	PortUsage
InterfaceBlock	PortDefinition
InvocationOnNestedPortAction	ItemFlow
ProxyPort	
TriggerOnNestedPort	Mapping is not specified yet.

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
~InterfaceBlock	PortDefinition

7.7.7.2 Mapping Specifications

7.7.7.2.1 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

AcceptEventActionCommon_Mapping

Mapping Source

AcceptEventAction

Mapping Target

AcceptActionUsage with qualifier: trigger:Trigger

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.

7.7.7.2.2 CommonInterfaceBlock_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

Applicable filters

(none)

7.7.7.2.3 ConjugatedInterfaceBlock_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

CommonInterfaceBlock_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}
  ```

7.7.7.2.4 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

Mapping Source

Port

Mapping Target

PortUsage

Applicable filters

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

```
(not src.type.oclIsUndefined()) and
Helper.hasStereotypeApplied(src, 'SysML::Ports&Flows::FullPort')
```

Mapping rules

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

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

```
self.oclAsType(Port_Mapping).ownedRelationship()
    ->including(MetadataOwningMembership_Factory.create(
        'SysMLv1Library::PortData',
        OrderedSet{'isFullPort'},
        Sequence{true}))
```

7.7.7.2.5 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

CommonInterfaceBlock_Mapping

Mapping Source

Class

Mapping Target

PortDefinition

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

The mapping class only has inherited rules. See the mapping classes in the general mapping section for details.

7.7.7.2.6 PortConjugation_Mapping

Description

Creates a PortConjugation between a PortDefinition and a ConjugatedPortDefinition element.

General Mappings

ToConjugation_Init
ElementUnique_Mapping

Mapping Source

Class

Mapping Target

PortConjugation

Owned Mappings

- conjugatedInterfaceBlock : ConjugatedInterfaceBlock_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::conjugatedType () : Type [1]

  ```
  conjugatedInterfaceBlock.to
  ```

- PortConjugation::originalPortDefinition () : Type [1]

  ```
  InterfaceBlock_Mapping.getMapped(from)
  ```

7.7.7.2.7 TriggerToReceiver_Mapping

DescriptionGeneral Mappings

No general mappings.

Mapping Source

No source element.

Mapping Target

No target element.

Applicable filters

(none)

7.7.8 Requirements

7.7.8.1 Overview

Table 20. List of High-Level Requirements Mappings

SysML v1 Abstract Syntax/Stereotype	SysML v2 Abstract Syntax
Copy	The copy relationship is not covered by SysML v2.
DeriveReqt	ConnectionUsage
Refine	Dependency
Requirement	RequirementUsage
Satisfy	SatisfyRequirementUsage
TestCase	VerificationCaseDefinition
Trace	Dependency
Verify	RequirementVerificationMembership

7.7.8.2 Mapping Specifications

7.7.8.2.1 DeriveReqt_Mapping

Description

A SysML::Requirements::DeriveReqt 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

Relationship_Mapping
ToConnectionUsage_Init

Mapping Source

Abstraction

Mapping Target

ConnectionUsage

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::DeriveReqt')
```

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..*]

```
EndFeatureMembership_Mapping.getMappedColl(from.source)->asOrderedSet()
->union(EndFeatureMembership_Mapping.getMappedColl(from.target)->asOrderedSet())
->including(QNameToFeatureTyping_Factory.create('DerivationConnections::Derivation'))
->union(self.oclAsType(Relationship_Mapping).ownedRelationship()->asOrderedSet())
```

7.7.8.2.2 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 <'id1'> SysMLv1Requirement {
        doc /*
               * requirement text
               */
}
use case def SysMLv1UseCase;

dependency from SysMLv1UseCase to SysMLv1Requirement {
        @SysMLv1Library::RefineData {isRefine = true;}
}
```

General Mappings

Dependency_Mapping

Mapping Source

Abstraction

Mapping Target

Dependency

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))
```

7.7.8.2.3 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

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(ElementUnique_Mapping).ownedRelationship()
    ->including(ReturnParameterMembership_Factory.create(
        'result',
        Helper.getScalarValueTypeByName('Boolean')))
    ->union(let text: String =
        Helper.getTagValueAsString(
            from,
            'SysML::Requirements::Requirement', 'text'
        ) in
if text.oclIsUndefined() then
    Set{}
else
    Set{DocumentationMembership_Factory.create(text)}
endif)
->including(EmptySubjectMembership_Factory.create())
```

- RequirementUsage::reqId () : String [1]

```
let stereotype: UML::Stereotype = Helper.getRequirementStereotype(from) in
Helper.getTagValueAsString(from, stereotype.qualifiedName, 'id')
```

7.7.8.2.4 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 sysML1BlockUsage.sysMLv1PartProperty {
        sysMLv1BlockUsage : SysMLv1Block;
}
```

General Mappings

ToOccurrenceUsage_Init
Relationship_Mapping

Mapping Source

Abstraction

Mapping Target

SatisfyRequirementUsage

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..*]

```
self.oclAsType(Relationship_Mapping).ownedRelationship()->asOrderedSet()
    ->including(TypeToFeatureTyping_Mapping.getMapped(
        from.supplier->any(s | true)))
    ->including(ReturnParameterMembership_Factory.create(
        'result',
        Helper.getScalarValueTypeByName('Boolean')))
    ->union(let clt: OclAny = from.client->any(c | true) in
if clt.oclIsKindOf(UML::Property) then
    OrderedSet{ReferenceUsageMembership_Factory.create(
        PropertyToReferenceUsage_Mapping.getMapped(clt),
        '',
        invalid)
    }
else
    OrderedSet{}
endif)
->including(EmptySubjectMembership_Factory.create())
```

7.7.8.2.5 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

ActivityAsDefinitionCommon_Mapping

Mapping Source

Activity

Mapping Target

VerificationCaseDefinition

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..*]

```
self.oclAsType(ActivityAsDefinitionCommon_Mapping).ownedRelationship()
->union(from.ownedElement->select(e | e.oclIsKindOf(UML::Parameter) and
    (e.oclAsType(UML::Parameter).type.name = 'VerdictKind'))
    ->collect(p | ReturnParameterMembership_Factory.create(
        'result',
        Helper.getEnumerationType(p.type)))->asSet())
->union(from.ownedElement->select(e | e.oclIsKindOf(UML::Parameter) and
        e.oclAsType(UML::Parameter).type.name <> 'VerdictKind')
    ->collect(p | ParameterMembership_Mapping.getMapped(p))->asSet())
->union(from.clientDependency
    ->select( v | Helper.hasStereotypeApplied(v, 'SysML::Requirements::Verify'))
    ->collect(v | Verify_Mapping.getMapped(v))->asSet())
```

7.7.8.2.6 TestCaseVerifyRequirementUsage_Mapping

Description

The mapping class creates the requirements usage of the SysML v2 test case for the verify relationship.

General Mappings

ToRequirementUsage_Init
Mapping

Mapping Source

Abstraction

Mapping Target

RequirementUsage

Applicable 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..*]

  ```
  from.supplier->collect(r | Subsetting_Factory.create(Requirement_Mapping.getMapped(r)))
  ```

7.7.8.2.7 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 <'id1'> SysMLv1Requirement1 {
        doc /*
               * requirement text
               */
}
requirement <'id2'> SysMLv1Requirement2 {
        doc /*
               * requirement text
               */
}
dependency from SysMLv1Requirement1 to SysMLv1Requirement2 {
        @SysMLv1Library::TraceData {isTrace = true;}
}
```

General Mappings

Dependency_Mapping

Mapping Source

Abstraction

Mapping Target

Dependency

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))
```

7.7.8.2.8 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 <'id1'> SysMLv1Requirement {
        doc /*
               * requirement text
               */
}
verification def SysMLv1TestCase {
        objective objective_SysMLv1TestCase {
                verify SysMLv1Requirement;
        }
        return verdict : VerificationCases::VerdictKind;
}
```

General Mappings

ToRelationship_Init
ElementFeatureMembership_Mapping

Mapping Source

Abstraction

Mapping Target

RequirementVerificationMembership

Applicable 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::memberElement () : Element [0..*]

```
TestCaseVerifyRequirementUsage_Mapping.getMapped(from)
```

7.7.8.2.9 Model Libraries

7.7.8.2.9.1 Verdicts

7.7.8.2.9.1.1 VerdictKind

The enumeration VerdictKind is mapped to the SysML v2 VerificationCases::VerdictKind model library element.
Systems Modeling