**Date:** May 2015

Semantic Information Modeling for Federation (*SIMF*)

*Version 0.5*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

**OMG Document Number: ad/2015-05-01**

**Machine readable file(s): ad/2015-05-02**

**The SIMF submission calls for worked examples. The worked example for the SIMF initial submission is the initial submission for an operational threat and risk model, document: sysa/2015-05-01**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Copyright © 2015, Object Management Group, Inc.

Copyright © 2015, Data Access Technologies, Inc (Model Driven Solutions Division)

Copyright © 2015, PNA-Group, Ltd.

Copyright © 2015, No Magic Inc.

Copyright © 2015, 88 Solutions, Inc.

Copyright © 2015, Thematix Partners LLC

USE OF SPECIFICATION - TERMS, CONDITIONS & NOTICES

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

LICENSES

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

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

PATENTS

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

GENERAL USE RESTRICTIONS

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

DISCLAIMER OF WARRANTY

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

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

RESTRICTED RIGHTS LEGEND

Use, duplication or disclosure by the U.S. Government is subject to the restrictions set forth in subparagraph (c) (1) (ii) of The Rights in Technical Data and Computer Software Clause at DFARS 252.227-7013 or in subparagraph (c)(1) and (2) of the Commercial Computer Software - Restricted Rights clauses at 48 C.F.R. 52.227-19 or as specified in 48 C.F.R. 227-7202-2 of the DoD F.A.R. Supplement and its successors, or as specified in 48 C.F.R. 12.212 of the Federal Acquisition Regulations and its successors, as applicable. The specification copyright owners are as indicated above and may be contacted through the Object Management Group, 140 Kendrick Street, Needham, MA 02494, U.S.A.

TRADEMARKS

MDA®, Model Driven Architecture®, UML®, UML Cube logo®, OMG Logo®, CORBA® and XMI® are registered trademarks of the Object Management Group, Inc., and Object Management Group™, OMG™ , Unified Modeling Language™, Model Driven Architecture Logo™, Model Driven Architecture Diagram™, CORBA logos™, XMI Logo™, CWM™, CWM Logo™, IIOP™ , IMM™ , MOF™ , OMG Interface Definition Language (IDL)™ , and OMG SysML™ are trademarks of the Object Management Group. All other products or company names mentioned are used for identification purposes only, and may be trademarks of their respective owners.

COMPLIANCE

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

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

**OMG’s Issue Reporting Procedure**

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

Table of Contents

[1 Scope 1](#_Toc419728380)

[1.1 Business Need 1](#_Toc419728381)

[1.2 Approach 3](#_Toc419728382)

[1.3 Unified Meta Model & Notation 5](#_Toc419728383)

[2 Conformance 5](#_Toc419728384)

[3 Normative References 6](#_Toc419728385)

[4 Terms and Definitions 6](#_Toc419728386)

[5 Symbols 6](#_Toc419728387)

[6 Additional Information 7](#_Toc419728388)

[1.1 Changes to Adopted OMG Specifications [optional] 7](#_Toc419728389)

[6.2 Submitters and Contributors 7](#_Toc419728390)

[7 Introduction to SIMF Concepts 7](#_Toc419728391)

[*7.1* *Pragmatic world view* 7](#_Toc419728392)

[*7.2* *Models* 8](#_Toc419728393)

[*7.3* *Concepts* 9](#_Toc419728394)

[7.3.1 Dictionary Concepts 11](#_Toc419728395)

[*7.4* *Information, Values ands Anything* 11](#_Toc419728396)

[*7.5* *Context* 11](#_Toc419728397)

[*7.6* *Entities* 12](#_Toc419728398)

[*7.7* *Situations* 13](#_Toc419728399)

[7.8 Relations 13](#_Toc419728400)

[*7.9* *Types* 13](#_Toc419728401)

[7.9.1 Individual Types 13](#_Toc419728402)

[7.9.2 Value types 13](#_Toc419728403)

[7.9.3 Situation Types & Situational Roles 13](#_Toc419728404)

[7.9.4 Relation Types 14](#_Toc419728405)

[7.10 Rules 14](#_Toc419728406)

[7.10.1 Production Rules 14](#_Toc419728407)

[7.10.2 As-A Rule 14](#_Toc419728408)

[7.10.3 Logical Grounding Rules 14](#_Toc419728409)

[7.10.4 Constraints 14](#_Toc419728410)

[7.10.5 General Constraints 14](#_Toc419728411)

[7.10.6 Multiplicity Constraints 15](#_Toc419728412)

[7.10.7 Uniqueness Constraints 15](#_Toc419728413)

[7.11 Quantifiers 15](#_Toc419728414)

[7.12 Expressions 15](#_Toc419728415)

[7.13 Properties & Units 15](#_Toc419728416)

[7.14 Naming 16](#_Toc419728417)

[7.15 Representations 16](#_Toc419728418)

[7.16 Important Relations 16](#_Toc419728419)

[7.16.1 Contextualization 16](#_Toc419728420)

[7.16.2 Equivalence 16](#_Toc419728421)

[7.16.3 Generalization 16](#_Toc419728422)

[7.17 SIMF Lexical Scope & Physical Representations 16](#_Toc419728423)

[7.17.1 Transactions 17](#_Toc419728424)

[8 Abstract Syntax [Normative] 18](#_Toc419728425)

[8.1 SIMF Meta Model::Expressions 18](#_Toc419728426)

[8.1.1 Diagram: Expressions 18](#_Toc419728427)

[8.1.2 Class Bound Role 18](#_Toc419728428)

[8.1.3 Association Calculation type 19](#_Toc419728429)

[8.1.4 Class Computed Fact 19](#_Toc419728430)

[8.1.5 Class Constant Reference 19](#_Toc419728431)

[8.1.6 Class Expression 19](#_Toc419728432)

[8.1.7 Association from 19](#_Toc419728433)

[8.1.8 Class Function Call 19](#_Toc419728434)

[8.1.9 Class Function Type 20](#_Toc419728435)

[8.1.10 Class Object Operation Type 20](#_Toc419728436)

[8.1.11 Association Return type 20](#_Toc419728437)

[8.1.12 Association to 20](#_Toc419728438)

[8.1.13 Class Traversal 21](#_Toc419728439)

[8.2 SIMF Meta Model::Lexical scope 21](#_Toc419728440)

[8.2.1 Diagram: Lexical scope 22](#_Toc419728441)

[8.2.2 Class Block 22](#_Toc419728442)

[8.2.3 Class Include 23](#_Toc419728443)

[8.2.4 Class Lexical Reference 23](#_Toc419728444)

[8.2.5 Class Lexical Scope 23](#_Toc419728445)

[8.2.6 Class Model 23](#_Toc419728446)

[8.2.7 Class Model Element 24](#_Toc419728447)

[8.2.8 Class Package 24](#_Toc419728448)

[8.2.9 Class Perspective 24](#_Toc419728449)

[8.3 SIMF Meta Model::Properties 25](#_Toc419728450)

[8.3.1 Diagram: Properties 25](#_Toc419728451)

[8.3.2 Class Measurement Unit 25](#_Toc419728452)

[8.3.3 Class Number 26](#_Toc419728453)

[8.3.4 Class Quantity 26](#_Toc419728454)

[8.3.5 Class Quantity kind 26](#_Toc419728455)

[8.3.6 Class Text 26](#_Toc419728456)

[8.3.7 Class Value 27](#_Toc419728457)

[8.4 SIMF Meta Model::Quantifiers 27](#_Toc419728458)

[8.4.1 Diagram: Quantifiers 27](#_Toc419728459)

[8.4.2 Class All 27](#_Toc419728460)

[8.4.3 Class Most 28](#_Toc419728461)

[8.4.4 Class No 28](#_Toc419728462)

[8.4.5 Class One of 28](#_Toc419728463)

[8.4.6 Class There Exists 28](#_Toc419728464)

[8.5 SIMF Meta Model::Relations 28](#_Toc419728465)

[8.5.1 Diagram: Relations 29](#_Toc419728466)

[8.5.2 Class Constrained Relation Type 29](#_Toc419728467)

[8.5.3 Class Relation 29](#_Toc419728468)

[8.5.4 Class Relation Type 30](#_Toc419728469)

[8.6 SIMF Meta Model::Representation 30](#_Toc419728470)

[8.6.1 Diagram: Representation 31](#_Toc419728471)

[8.6.2 Class Information 31](#_Toc419728472)

[8.6.3 Association Class Representation 31](#_Toc419728473)

[8.6.4 Class Representation Rule 32](#_Toc419728474)

[8.7 SIMF Meta Model::Rules 32](#_Toc419728475)

[8.7.1 Diagram: Rules 33](#_Toc419728476)

[8.7.2 Class As-A Rule 33](#_Toc419728477)

[8.7.3 Class Conditional Rule 33](#_Toc419728478)

[8.7.4 Class Constrant 34](#_Toc419728479)

[8.7.5 Class General Constrant 34](#_Toc419728480)

[8.7.6 Class Implication 34](#_Toc419728481)

[8.7.7 Class Multiplicity Constraint 34](#_Toc419728482)

[8.7.8 Class Production Rule 35](#_Toc419728483)

[8.7.9 Class Rule 35](#_Toc419728484)

[8.7.10 Class Uniqueness Constrant 36](#_Toc419728485)

[8.8 SIMF Meta Model::Situations 36](#_Toc419728486)

[8.8.1 Diagram: Situations 36](#_Toc419728487)

[8.8.2 Class Actuality 36](#_Toc419728488)

[8.8.3 Association Class Binding 37](#_Toc419728489)

[8.8.4 Class Occurance 37](#_Toc419728490)

[8.8.5 Class Phase 37](#_Toc419728491)

[8.8.6 Association Redefinition 37](#_Toc419728492)

[8.8.7 Direct Supertypes 37](#_Toc419728493)

[8.8.8 Class Role 38](#_Toc419728494)

[8.8.9 Class Situation 38](#_Toc419728495)

[8.8.10 Class Situation Type 38](#_Toc419728496)

[8.9 SIMF Meta Model::Templates 39](#_Toc419728497)

[8.9.1 Diagram: Templates 39](#_Toc419728498)

[8.9.2 Class Template 39](#_Toc419728499)

[8.10 SIMF Meta Model::Terms and Identifiers 40](#_Toc419728500)

[8.10.1 Diagram: Terms and identifiers 40](#_Toc419728501)

[8.10.2 Association Identification 40](#_Toc419728502)

[8.10.3 Class Identifier 41](#_Toc419728503)

[8.10.4 Class Name 41](#_Toc419728504)

[8.10.5 Class Name Text 41](#_Toc419728505)

[8.10.6 Class Term 41](#_Toc419728506)

[8.10.7 Class Term Relation 42](#_Toc419728507)

[8.10.8 Class URIIdentifier 42](#_Toc419728508)

[8.11 SIMF Meta Model::Top level 44](#_Toc419728509)

[8.11.1 Diagram: Concept 44](#_Toc419728510)

[8.11.2 Class Anything 44](#_Toc419728511)

[8.11.3 Class Concept 44](#_Toc419728512)

[8.11.4 Class Context 45](#_Toc419728513)

[8.11.5 Class Definition 45](#_Toc419728514)

[8.11.6 Association Documentation 46](#_Toc419728515)

[8.11.7 Association Extent 46](#_Toc419728516)

[8.11.8 Association Generalization 46](#_Toc419728517)

[8.11.9 Class Individual 46](#_Toc419728518)

[8.11.10 Diagram: Context 47](#_Toc419728519)

[8.11.11 Diagram: TopLevel 47](#_Toc419728520)

[8.12 SIMF Meta Model::Transactions 48](#_Toc419728521)

[8.12.1 Diagram: Transactions 48](#_Toc419728522)

[8.12.2 Class External Transaction 48](#_Toc419728523)

[8.12.3 Class Inferred Transaction 48](#_Toc419728524)

[8.12.4 Class Transaction 49](#_Toc419728525)

[8.13 SIMF Meta Model::Types 49](#_Toc419728526)

[8.13.1 Diagram: Types 50](#_Toc419728527)

[8.13.2 Class Classification 50](#_Toc419728528)

[8.13.3 Class Individual Type 50](#_Toc419728529)

[8.13.4 Class Information Type 51](#_Toc419728530)

[8.13.5 Association Instance Relation 51](#_Toc419728531)

[8.13.6 Direct Supertypes 51](#_Toc419728532)

[8.13.7 Class Intersection Type 51](#_Toc419728533)

[8.13.8 Class Key 51](#_Toc419728534)

[8.13.9 Class Phase 52](#_Toc419728535)

[8.13.10 Class Quantification 52](#_Toc419728536)

[8.13.11 Class Type 52](#_Toc419728537)

[8.13.12 Class Union Type 53](#_Toc419728538)

[8.14 Foundational Assumptions 54](#_Toc419728539)

[8.14.1 Multiple representations of overlapping concepts 54](#_Toc419728540)

[8.14.2 Models include data 54](#_Toc419728541)

[8.14.3 Conceptual Models 54](#_Toc419728542)

[8.14.4 Identity and identifiers 54](#_Toc419728543)

[8.14.5 Facts 54](#_Toc419728544)

[8.14.6 REPRESENTATIONs of a CONCEPT 55](#_Toc419728545)

[8.14.7 REPRESENTATION identifiers 55](#_Toc419728546)

[8.14.8 CONCEPT representation identifiers 56](#_Toc419728547)

[8.14.9 Sources 56](#_Toc419728548)

[8.14.10 Ownership 56](#_Toc419728549)

[8.14.11 Lifetime and context of facts 56](#_Toc419728550)

[9 Graphical Notation 58](#_Toc419728551)

[9.1 Context 58](#_Toc419728552)

[9.2 Types 59](#_Toc419728553)

[9.3 Relation Types 60](#_Toc419728554)

[9.4 Directed Relation Types 60](#_Toc419728555)

[9.5 Instances 61](#_Toc419728556)

[9.6 Quantifications 62](#_Toc419728557)

[9.7 LIM Structure 63](#_Toc419728558)

[9.8 Important Relations 63](#_Toc419728559)

[10 Textual Notation 63](#_Toc419728560)

[11 Standard MBR Models 63](#_Toc419728561)

[11.1 OWL Web Ontology Language 64](#_Toc419728562)

[12 Annex A Examples (informative) 65](#_Toc419728563)

[13 Annex B Mapping to OWL 2 (normative) 66](#_Toc419728564)

[13.1 Class 66](#_Toc419728565)

[13.2 Class Generalization 66](#_Toc419728566)

[13.3 Class with Datatype Property 67](#_Toc419728567)

[13.4 Class with Self-Referential Object Property 67](#_Toc419728568)

[13.5 Class with Object Property 68](#_Toc419728569)

[13.6 Property Holder with Datatype Property 68](#_Toc419728570)

[13.7 Property Holder with Self-Referential Object Property 69](#_Toc419728571)

[13.8 Property Holder with Object Property 69](#_Toc419728572)

[13.9 Class with Object Property without Range 70](#_Toc419728573)

[13.10 Class with Subproperty 70](#_Toc419728574)

[13.11 Class with Universal Quantification Constraint on Property I 71](#_Toc419728575)

[13.12 Class with Universal Quantification Constraint on Property II 72](#_Toc419728576)

[13.13 Class with Existential Quantification Constraint on Property 73](#_Toc419728577)

[13.14 Property Holder with Self-Referential Subproperty 74](#_Toc419728578)

[13.15 Property Holder with Subproperty 74](#_Toc419728579)

[13.16 Class with Subproperty without a Range 75](#_Toc419728580)

[13.17 Class with Necessary and Sufficient Property 76](#_Toc419728581)

[13.18 Class With Property Having Unspecified Multiplicity 77](#_Toc419728582)

[14 Annex C UML Profile (normative) 78](#_Toc419728583)

[14.1 Conceptual Modeling Interpretation of UML 78](#_Toc419728584)

[14.1.1 Property Ownership 78](#_Toc419728585)

[14.1.2 Global Properties 78](#_Toc419728586)

[14.2 Subproperty 78](#_Toc419728587)

[14.3 Existential Quantification Constraint 79](#_Toc419728588)

[14.4 Universal Quantification Constraint 79](#_Toc419728589)

[14.5 Necessary and Sufficient Property 80](#_Toc419728590)

[14.6 Conceptual Modeling UML Profile 81](#_Toc419728591)

[14.6.1 UML Subset 81](#_Toc419728592)

[14.6.2 Stereotype: Classification [Classifier] 81](#_Toc419728593)

[14.6.3 Stereotype: Classifies [Generalization, Realization] 82](#_Toc419728594)

[14.6.4 Stereotype: Conceptual Model [Package] 82](#_Toc419728595)

[14.6.5 Stereotype: Disjoint with [Dependency] 82](#_Toc419728596)

[14.6.6 Stereotype: Entity [Class] 83](#_Toc419728597)

[14.6.7 Stereotype: Equivalent with [Dependency] 83](#_Toc419728598)

[14.6.8 Stereotype: Intersection [Classifier] 84](#_Toc419728599)

[14.6.9 Stereotype: Is in context [Dependency] 84](#_Toc419728600)

[14.6.10 Stereotype: Source Reference [Classifier] 85](#_Toc419728601)

[14.6.11 Stereotype: Phase [Class] 85](#_Toc419728602)

[14.6.12 Stereotype: Quantity Kind [Classifier] 85](#_Toc419728603)

[14.6.13 Stereotype: Represents [Realization, Dependency, Generalization] 86](#_Toc419728604)

[14.6.14 Stereotype: Role [Class] 86](#_Toc419728605)

[14.6.15 Stereotype: Union [Classifier] 87](#_Toc419728606)

[A <<Union>> is any context (Package or Classifier) that has an extent which is the complete union of the extents of all context that realize or specialize the Union. 87](#_Toc419728607)

[14.7 Mapping Profile 87](#_Toc419728608)

[14.7.1 Stereotype: Facade [Classifier] 87](#_Toc419728609)

[14.7.2 Stereotype: Filter [Connector, Property] 88](#_Toc419728610)

[14.7.3 Stereotype: Match [Connector, Property] 88](#_Toc419728611)

[14.7.4 Stereotype: Mapping [Package] 88](#_Toc419728612)

[14.7.5 Stereotype: Representation [Package] 89](#_Toc419728613)

[14.7.6 Stereotype: Represents [Realization, Dependency] 89](#_Toc419728614)

[14.7.7 Class: Representation Computation 89](#_Toc419728615)

[14.7.8 Stereotype: Representation Rule [Class] 90](#_Toc419728616)

[14.8 Profile to SIMF Model Mapping 91](#_Toc419728617)

Preface

**OMG**

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

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

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

**OMG Specifications**

As noted, OMG specifications address middleware, modeling and vertical domain frameworks. All OMG Specifications are available from the OMG website at:

*http://www.omg.org/spec*

Specifications are organized by the following categories:

Business Modeling Specifications

Middleware Specifications

1. CORBA/IIOP
2. Data Distribution Services
3. Specialized CORBA

IDL/Language Mapping Specifications

Modeling and Metadata Specifications

1. UML, MOF, CWM, XMI
2. UML Profile

Modernization Specifications

Platform Independent Model (PIM), Platform Specific Model (PSM), Interface Specifications

1. CORBAServices
2. CORBAFacilities

OMG Domain Specifications

CORBA Embedded Intelligence Specifications

CORBA Security Specifications

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

OMG Headquarters  
109 Highland Ave,

Needham, MA 02494 USA  
  
Tel: +1-781-444-0404  
Fax: +1-781-444-0320  
Email: *pubs@omg.org*

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

**Typographical Conventions**

The type styles shown below are used in this document to distinguish programming statements from ordinary English. However, these conventions are not used in tables or section headings where no distinction is necessary.

Times/Times New Roman - 10 pt.: Standard body text

**Helvetica/Arial - 10 pt. Bold:** OMG Interface Definition Language (OMG IDL) and syntax elements.

**Courier - 10 pt. Bold:** Programming language elements.

Helvetica/Arial - 10 pt: Exceptions

NOTE: Terms that appear in italics are defined in the glossary. Italic text also represents the name of a document, specification, or other publication.

**Issues**

The reader is encouraged to report any technical or editing issues/problems with this specification to <http://www.omg.org/report_issue.htm>.

# Scope

## Business Need

Our ability to share, manage, analyze, communicate and act upon information is at the foundation of the modern enterprise and open, collaborative government. Information sharing is essential for an integrated approach to enterprise supply chains, fighting terrorism, business and government intelligence, inter-organizational collaboration and integrating enterprise applications. Yet, this essential capability has remained difficult and expensive to achieve in information systems which are frequently isolated, stove piped, and difficult to integrate. The inability of our systems to share information hampers the ability of our organizations to collaborate and for our processes, services, and information resources to work together. Much of our information technology budgets are consumed by attempts to overcome this “semantic friction” in our systems and organizations are currently spending more on application integration than on building new applications [Gartner2011]. The overall human and financial cost to society from our failure to share and reuse information is many times the cost of the systems’ operation and maintenance.

In general, information sharing can be understood at a number of different levels.

* *Infrastructure* is the technology used to maintain data and move it from one place to another.
* *Format* is the way data are structured.
* *Semantics* deals with how data is interpreted as meaningful information. For an information system, this interpretation is reflected in how the data is processed in order to carry out the business purpose of the system.

We are effective at dealing with data infrastructure today, and we are somewhat effective at handling multiple data formats, albeit via manual and point-to-point integrations. However, we are not very good at understanding how the semantics of data in independent data sources are related. Too often, how each system interprets shared data is implicit in the specific design and operation of the systems. Differences in structure, terminology, viewpoint, and notations make system-specific data structures hard to integrate, negatively impacting the capability to federate these systems.

Full semantic integration requires information systems to all properly and consistently interpret the data exchanged among the systems. This, in turn, requires that there be an explicit understanding of what the desired semantic interpretation *is* at a business level. A semantic *model* can be used to express this understanding in a way that can be validated by the business stakeholders of the systems being integrated. And, given a formal underpinning for such a model, it can then also be used for supporting analyses and deductions necessary to carry out the necessary integration.

Unfortunately, for most existing information systems, the desired semantics have not been properly modeled. The following are some scenarios in which semantic integration is, nevertheless, critical. Diverse and disparate efforts are currently being made to address these scenarios, examples of which are included with the scenario descriptions below. But, as of today, there is no consistent way to address modeling for semantic integration in general across all these areas.

* *Data integration between business systems.* Many large businesses have a critical need to better integrate systems in support of complex products. Not only may their business area have suffered financial distress, but there may be a need for new government reporting or new analytics and integration due to acquisitions. Such organizations typically have multiple layers of existing data bases, middleware specifications and XML schemas for use in web services, event brokers, etc. Most, if not all, of the existing systems and technologies still need to be supported. There may be dozens or even hundreds of enterprise systems involved and hundreds or thousands of small applications and spreadsheets.

*Example.* A common approach chosen for integrating major business systems is to create a “canonical model” of the domain and then map data into and out of that model using data mapping tools. Unfortunately, while there are various proprietary tools to support such an effort, there is no widely available standard-based tooling for the job. For instance, while UML can be and is used for the modeling part of the job, a general modeling notation such as UML is far from ideal for the conceptual level of modeling required, and there is currently no standard profile to adapt it to the task nor for mapping data into and out of a canonical model in general. (The Model Driven Message Interoperability specification provides some support for the latter, but only limited to message format transformation for the financial services domain.)

* *Data federation across multi-disciplinary teams.* Developing complex systems often involves many parties who are widely distributed in location and time. Such development therefore requires efficient and effective information exchange during the complete development and operations lifecycle of the system. This can only be achieved by realizing semantic integration between all involved parties.

*Example.* The European Cooperation for Space Standardization (ECSS)[[1]](#footnote-1) addresses this issue by introducing the concept of *a global conceptual model.* This model is used in the implementation of “space system data repositories” as federations of physical databases. These databases are geographically dispersed and change over time but are logically integrated in an interoperable architecture, so that data can be exchanged effectively and reliably. Such data repositories need to be stable over a long period of time, so modeling must be at the semantic level *independent* of technology and tools. This modeling allows for upgrading the implementation technology without changing the model and data itself. The primary aim of this is to substantially reduce the system development and operation costs while achieving greater precision and federation.

* *Information federation across an industry.* Entire major industries, such as finance and telecommunications, need to deal with the representation of information relative to multiple contexts, taking into account different business processes, specific modeling goals and needs, visualization and implementation requirements or the existence of overlapping modeling domains. These differing contexts and conditions may require emphasizing different aspects and characteristics of essentially the same information. The representation of a concept in one view may be different from the representation of the same concept in another view as the context-specific details that are relevant differ from view to view. Information can be described using different yet compatible paradigms (e.g., domain-specific languages vs. UML and profiles) yet the meaning and semantics of the information should stay the same regardless of the format. This, again, highlights the need to focus on a common core model of shared semantic concepts.

*Examples.* Some examples of efforts to deal with industry-level information federation are the Shared Information and Data (SID) Model, developed by the TM Forum [TMForum], the Common Information Model (CIM) developed by the Distributed Management Task Force [DMTF] and the Reference Information Model (RIM)developed by Health Level Seven [HL7].

* *Information sharing and federation of threat and risk information,* Threats and risks are increasingly multi-dimensional in nature – spanning physical space and cyber space. Threat actors understand and exploit our stove piped approach to sharing and analyzing information which leads to ineffective collaboration and mediation. Only by federating information across multiple domains such as cyber, physical, critical infrastructure, criminal, intelligence and defense, irrespective of technical and political boundaries, can we effectively counter multi-dimensional intentional threats, natural events and system failures.

*Examples.* Attacks on our critical infrastructure have and will combine cyber attacks with physical attacks. This has been seen in exploits of our electric power grid where physical weaknesses are combined with Cyber to harm our physical infrastructure. By combining Cyber, criminal and terrorist information we will be better able to deal with theses critical threats.

* *Data federation across government organizations.* Information sharing has been recognized by governments as a key enabler for purposes as diverse as fighting terrorism to financial transactions. There has been some progress in standardizing exchange schemas, which is a big step ahead of no standards at all, but the need exists to ensure that there is no ambiguity in the semantics of the exchanged data in order to safely enable the reuse of that data. In addition, any such standard must accept that there are and will be other such standards and that these also need to be federated.

*Example.* The U.S. Information Sharing Environment (ISE) “provides analysts, operators, and investigators with integrated and synthesized terrorism, weapons of mass destruction, and homeland security information needed to enhance national security and help keep our people safe” [ISE]. ISE depends on fixed schemas for information sharing, i.e., the National Information Exchange Model (NIEM) and the Universal Core (UCORE). These schemas provide XML Schema definitions that are claimed to be sufficiently common and universally understood by relevant stakeholders regardless of the IT systems being used within their intended domains. Even within NIEM, though, hundreds of overlapping schemas have been defined.

* *Model federation across different modeling metamodels.* The OMG itself has multiple standards related to modeling. These standards were originally created independently, resulting in difficulties when users try to use them together to share information embodied in models using the different standards. A conceptual model abstracting from the existing OMG modeling standards, would facilitate their comparison, acknowledging the commonality (or lack thereof) between the different concepts and definitions and bridging those concepts.

*Example.* OMG specifications related to just process modeling include BPMN, UML Activities, BPDM, and SPEM. A case in point in the difficulty this has caused relates to the *UML Profile for DODAF and MoDAF* (UPDM), a wide ranging profile supporting US Department of Defense (DOD) and UK Ministry of Defence (MOD) architecture frameworks. The UPDM community wishes, for example, to be able to use BPMN process models in the context of their UML Profile. A stopgap tactic has been to define an additional *UML Profile for BPMN*, which allows BPMN-looking diagrams to be drawn in UML, but it is clear this is not a strategic approach. A better approach would be to create a “process modeling” conceptual domain model that would then permit model bridging relations between BPMN, UML, BPDM and SPEM models, allowing sharing across users’ process models

* *Schema Evolution*. As information systems evolve to support changing enterprise needs, the datasets they use need to evolve as well. While some changes are additive and readily accommodated, others involve factoring and evolving concepts. At their core, such changes require the evolution of the dataset schema underlying the system and the migration of the data from the old to new schemas. Such changes also impact the logic that interacts with the dataset and every external interface and related data structure. While there is some tooling available for schema migration, there is little available to aid in the evolution of the logic and external interfaces. The absence of semantic understanding of the relationship between the schema and external interface data structures makes tooling to aid in the evolution problematic.

*Example.* It is common for an enterprise to represent the concept of *customer* as a composite of information about the person and the role that person plays with respect to the enterprise. Evolving needs, including regulatory requirements, require many enterprises to now factor this concept so that they can represent that the same person may play other roles as well, such as employee. Such semantic understanding is required to enforce constraints such as a prohibition against the same individual playing both the customer and employee role in a transaction. The absence of semantics-based tooling makes such changes labor intensive and error prone.

Current standards for information and data modeling may be effective at defining a particular data model for a particular application using a particular technology to solve a particular problem. But, as highlighted by the above examples, the methodology for using these standards at a higher level of abstraction – namely for cross-domain and cross-organizational semantic modeling – is not as well or as widely understood. As a consequence, the models available within a given organizational context are often not well suited to application across multiple dimensions or technologies, and so poorly support the needs for sharing and federation.

## Approach

As described in Section 6.2.1, the scope of SIMF encompasses conceptual domain modeling, logical information modeling and the modeling of bridging relations between models at all levels. Figure 1 summarizes the general organization of information models into conceptual, logical and physical layers and indicates the scope of SIMF within it. The following more precisely defines the terminology used to describe this scope.

* *Conceptual Domain Model (CDM).* A CDM is not a traditional data model, as such, but, rather, a model of the terms and concepts of an area of concern or *domain[[2]](#footnote-2),* which may be a broad industry area such as telecommunications, finance or even metamodeling or may be more focused on a specific application area. It primarily addresses the semantics, concepts and terminology of a domain, capturing the meaning that usually is not available in a data model, while abstracting out data representation and application specific considerations. The objective of a CDM is to capture the semantics of one or more domains as a well defined set of (potentially federated) concepts, predicates (to express properties about the concepts and to relate them) and integrity rules (constraining instances). For a given domain, many CDMs may co-exist, e.g. CDMs that have been developed by different entities and express differing points of view.

For the purpose of this RFP, conceptual domain modeling is limited to modeling the information concerns of a domain as a conceptyual domain model. Modeling processes and services is considered out of scope. However, this does not imply that all modeling of the dynamics of concepts is necessarily out of scope. (Future RFPs may address further process-oriented conceptual modeling requirements.)

* *Logical Information Model (LIM).* A LIM acts as an intermediary between CDMs and physical data schema (see below). The objective of a LIM is to provide a purpose-specific but implementation technology-independent view of information in terms of logical data structures. There can be multiple different ways to represent the same information from different viewpoints and for specific purposes. Each viewpoint may have its own structure, local vocabulary and subset of all possible information in a domain. These purpose specific commitments are made in the LIM. Elements of a LIM are related to the CDM concepts, predicates and integrity rules they represent (using Model Bridging Relations, see below) and may extend or embed other logical elements. A LIM model addresses a specific viewpoint and purpose and, as such, selects those types, properties and relations of interest and structures them for that purpose.
* *Physical Data Schema (PDS).* A PDS describes how to implement a LIM in a database or exchange format of choice. That is, it defines the application- and technology-specific representations of data. There can be many PDS representations of the same LIM. PDSs grounded in LIMs (using Model Bridging Relations, see below) provide the basis for federation of data defined in those schemas. Such fixed schemas become a particular *projection* of information for a particular purpose, but not the only way to access the same information.
* *Model Bridging Relation (MBR).* An MBR defines a connection between different sets of elements in the same or different models. This connection may be between models across the conceptual, logical and physical modeling layers, or within models of a given layer. MBRs may bridge different models created as part of a single wider effort or they may address connections between independently conceived models. Linking the semantics of information in its different conceptual, logical and physical representations, MBRs are the foundation of federation.



Figure Information Modeling Layers and SIMF Modeling Scope

As indicated in Figure 1, the scope of SIMF includes CDMs, LIMs and MBRs (where the lines in the diagram represent MBRs). While PDSs are out of scope for SIMF, model bridging relations to PDSs are part of SIMF.

## Unified Meta Model & Notation

While SIMF supports modeling at multiple levels, there is a single meta model for SIMF. A model defines the abstraction layer it is intended to model. Likewise, the same syntax is used at all levels. While there is a unified meta model, some model elements are more appropriate for one layer or another – which is defined as an appendix.

# Conformance

The Conformance clause identifies which clauses of the specification are mandatory (or conditionally mandatory) and which are optional in order for an implementation to claim conformance to the specification.

Note: For conditionally mandatory clauses, the conditions must, of course, be specified.

There are five distinct types of conformance. These are listed below. Unless otherwise stated these types of conformance are independent.

1. *Abstract syntax conformance.* A tool demonstrating abstract syntax conformance provides a user interface and/or API that enables instances of concrete SIMF metaclasses to be created, read, updated, and deleted. The tool must also provide a way to validate the well-formedness of models that corresponds to the constraints defined in the SIMF metamodel.

2. *Concrete syntax conformance.* A tool demonstrating concrete syntax conformance provides a user interface and/or API that enables instances of SIMF notation to be created, read, updated, and deleted. Note that a conforming tool may provide the ability to create, read, update and delete additional diagrams and notational elements that are not defined in SIMF.

3. *Model interchange conformance.* A tool demonstrating model interchange conformance can import and export conformant XMI for all valid SIMF models, including models with profiles defined and/or applied. Model interchange conformance implies abstract syntax conformance. A conforming SIMF tool shall be able to load and save XMI in SIMF format.

4. *Diagram interchange conformance.* A tool demonstrating diagram interchange conformance can import and export conformant DI (see Annex B) for all valid SIMF models with diagrams.

5. *Semantic conformance.* A tool demonstrating semantic conformance provides a demonstrable way to interpret SIMF semantics, e.g., data transformers, code generation, model execution, or semantic model analysis.

Where the SIMF specification provides options for a conforming tool, these are explicitly stated in the specification. In a number of other cases, certain aspects of the semantics are listed as "undefined" or “intentionally not specified” or “not specified”, allowing for domain- or application-specific customizations. Only customizations that do not contradict the provisions of this specification will be deemed to conform to it. However, models whose meaning is based on such customizations can only be interchanged without loss with tools that support the same or compatible customizations.

This specification comprises this document together with XMI serialization contained in machine-consumable files as listed on the cover page. If there are any conflicts between this document and the machine-consumable files, the machine-consumable files take precedence.

# Normative References

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

List of normative references.(specific reference to be included)

* UML
* SMOF
* RDF
* OWL
* ODM
* PRR
* JCGM 200:2008

# Terms and Definitions

For the purposes of this specification, the following terms and definitions apply.

See section 1.2

# Symbols

The type styles shown below are used in this document to distinguish programming statements from ordinary English. However, these conventions are not used in tables or section headings where no distinction is necessary.

Times/Times New Roman - 10 pt.: Standard body text

**Helvetica/Arial - 10 pt. Bold:** OMG Interface Definition Language (OMG IDL) and syntax elements.

**Courier - 10 pt. Bold:** Programming language elements.

Helvetica/Arial - 10 pt: Exceptions

NOTE: Terms that appear in italics are defined in the glossary. Italic text also represents the name of a document, specification, or other publication.

# Additional Information

## Changes to Adopted OMG Specifications [optional]

None

## 

## 6.2 Submitters and Contributors

The following companies submitted this specification:

1. Data Access Technologies, Inc. (Model Driven Solutions Division)
2. Nomagic, Inc.
3. PNA-Group, Ltd.
4. 88 Solutions
5. Thematix Partners LLC

Contributors

* Tibco Software Inc.
  + Paul Brown

# Introduction to SIMF Concepts

The following is a high-level, non-normative, description of some of the fundamental SIMF concepts.

The fundamental concepts will be described in a way that most practitioners can easily relate it to their familiar experiences. In this chapter we will gradually build a semantic-conceptual architecture (an architecture that is completely independent of any particular technology and in which there is a clear distinction between the world of the things and the world of the names (representations)).

The prime aim of this chapter is to demonstrate the value that a SIMF semantic-conceptual model and transformations can offer to the ever increasing need of federation of (information) systems in business and government practice. The parentheses are placed intentionally such that SIMF can also be of value to the emerging MBSE (Model Based System Engineering) discipline, federation of risk and threat information as well as the emerging discipline of legal engineering as an aid to produce person based services.

## *Pragmatic world view*

Any conceptual system has strengths and weaknesses and is colored by the influences and domains which influenced the conceptualizations. From a pragmatic point of view there is a good understanding of what a “Tree” is (the word sense of tree that refers to a plant most babies could identify). However when you look very close and at the “edges” such categorizations can be fuzzy. On a similar note when you get to the edges of quantum mechanics and near-light speed travel, our common views of the world can break down, or become unknown or controversial. Those who study language carefully may say that every meaning of every word is contextual and that dictionary and thesaurus don’t completely work.

While all of the above may be true, taken to the extreme it would make communication impossible and synthesizing or transforming information impractical. On the other hand, if you take a pragmatic world view there is a lot that can be done to communicate, even across those with different backgrounds, and most of us just don’t deal with quantum mechanics and “warp speed”. SIMF and the conceptual models built on it use a pragmatic world view and are designed to deal with structured (or at least semi-structured) data. However, such pragmatics are modularized into component models such that those who need to deal with extreme detail in some area may do so without breaking the entire system.

If the “precision dial” were turned all the way up to account for all these edge cases the model would get complex and unusable.

The other extreme is to make definitions and their representation very narrow and specific, perhaps to a system or particular discipline. The other extreme may also have such “fuzzy” definitions that they are no more than hints. This extreme also has its problems and would make wide-scale communications and synthesis impossible. In fact, such overly specific data representations and fuzzy schema are essentially what we are federating with SIMF – the design of specific systems.

So between overly complex and overly specific or fuzzy is the useful middle ground we seek. That middle ground is to be precise and true within this pragmatic world view.

Unless otherwise stated in a particular model, one can expect that:

* Terms mean what they do in a dictionary definition, refined to a word sense
* Where a concept is more specific it will be scoped by appropriate context (e.g. an experts quantification [Keslick, John A. (2004). ["Tree Biology Dictionary"](http://www.treedictionary.com/). Retrieved 2012-07-30.] of what defines a tree may be more specific than common concepts)
* Concepts will have the most general definition and relationships reasonable within the context in which they are defined (note that this sometimes introduces general abstract concepts). Note that this is different from common practices in defining schema where the same essential meaning is repeated everywhere it is used.
* We are treating the world in terms of Newtonian mechanics (e.g. space/time is not relative)
* Where applicable, we are concerned with the surface of the earth and the near vicinity (e.g. in understanding location and gravity). However such assumptions should reference an earth context.
* Generic concepts are better under-defined (having fewer axioms than may be possible) than over-specified. This allows more specific concepts or “theories of the world” to specialize and reuse the generic ones.
* The concepts defined and mapped are being used in some structured way, not unstructured natural language (this does not preclude the structuring of natural language by automated or “human in the loop” means, but this is out of our scope). We do expect that natural language processes may interpret text to define SIMF axioms.

SIMF does provide mechanisms to support the evaluation of context, scope and trust so domains “on the edges” may still use SIMF, but should be careful about what SIMF features or SIMF models may say about common concepts.

## *Models*

SIMF models describe some subject area or domain for the purpose of defining and sharing *information* within and about that domain and with other related domains. SIMF defines a modeling language for this purpose; users of SIMF use that modeling language to describe applicable concepts; in some cases a model captures generic, cross domain concepts while in the remaining cases a model may be more specific, that is specific to a discipline, domain, organization or system (we will consider all of these a “domain”). Ultimately a model focused on a domain, any domain, shares many concepts with other domains – the sharing of such concepts is the basis for communications among people, organizations and systems with different backgrounds and purposes. For this reason domain specific models typically use or specialize more generic cross-domain concepts. To support this requirement SIMF also provides for models that describe general concepts and then provides capabilities to map those general concepts to more specific domain concepts.

**Information**

Models are themselves *information* **about something**. A concept defined in a model is not “the thing”, but a **description of that thing from some perspective**. For this reason the subject of the model and the perspective are important for understanding the content of the model and how it may relate to other models. A model is fundamentally *information*, that information may be represented in different ways: textually, as tables, graphically, etc. SIMF defines some ways to represent information but other representations may be used as is appropriate for a purpose. The same information is typically rendered in multiple different formats for different purposes and stakeholders.

**Bridging technologies**

A model needs some way to store information. SIMF defines an abstract syntax (nevertheless illustrated with sufficient concrete examples) for storing SIMF information and multiple physical representations, in keeping with the SIMF philosophy of *bridging* technologies. Regardless of the technology, a SIMF model can be thought of as a repository of information. Since SIMF is, its self, information about information it is also called a “meta model”.

## *Concepts*

Everything we describe in a SIMF model is considered a *concept*. The term concept is used because a model describes how we conceive or understand something from a perspective. For example, we may all understand there is something we call the “Eiffel tower”. In the figure below this information is represented in a very familiar format, an extended Venn diagram. We see a clear distinction as said above between the world of the things and the world of names and information about the things, an essential distinction in SIMF.

A person preferring a textual representation could use the following formal representation:

(1a) “There is something in the world that we are thinking of when the term “Eiffel tower” is used”.

Or in short but sufficient format:

(1b) There is something in the world that is referred to as Eiffel tower.



Figure

But how we think about it (the thing designated by the term “Eiffel tower”) or describe it may be different. Differences aside, there is something in the world that we are thinking of when the term “Eiffel tower” is used so we want to connect these different descriptions of the same thing – it is the “real thing” that connects us even though there is no way to have that real thing in our minds or models. We connect the dots through our concepts grounded in our mutual understanding of the world.

Concepts in SIMF may be about anything: “Real things” like the Eiffel Tower, Imaginary things like Unicorns, types of things, characteristics of things, relationships between things, rules, even names. In SIMF “Concept” is the root of everything else.

Once you define a concept in a model you can start adding to it, saying more about it – of course each thing you add is a new concept. A SIMF model becomes a connected graph of these concepts and concepts about concepts. Some of those concepts will also come from other models.

In Figure 3 quite a few additional pieces of information are added to say more about the Eiffel tower. We also bring in a second example, in this case the well-known Colosseum. Please note that the English term Colosseum identifies the same thing as the French term Colisée. Figure 3 can be used to illustrate the essential concepts of context and perspective in a way that practitioners easily understand.



Figure

Additional detail on terms and concepts can be found here: <https://github.com/ModelDriven/SIMF/blob/master/SupportingDocuments/PNA-Group/1981-10%20An%20architecture%20for%20knowledge%20base%20software-96-pages.pdf>

### Dictionary Concepts

In many modeling languages you need to know a lot about something to define it in a model. In SIMF, you can invent new concepts any time you like as a dictionary concept. The precise description of this far reaching aspect will be discussed in a separate section. In defining such a new concept you may give it a name and it is best to give it a description. Other concepts are defined more contextually and may not require a name. For example, the concept that the Eiffel Tower was erected in 1889AD or that it is 324 meters tall (including the TV antenna) do not need extra names; they are just facts about the Eiffel tower. Facts are concepts describing one or more other concepts. In the figures above, each line is a fact. However, facts do have “identity” so their provenance can be traced, crucial in trusting any “fact”.

Dictionary concepts may also have qualifiers that define the plurality (e.g. singular or plural), timeframe (past, present or future) and kind of concept (noun, verb, etc.). Dictionary concepts can also be related to one-another in several ways and provide the basis for defining information elements.

## *Information, Values ands Anything*

Values, such as the number 5 or the text “Frog” are not considered concepts, they are information we use to describe concepts. Information and values forms a tree that is distinct with “concepts”. Above concepts and information is the superclass “Anything”. All elements in SIMF derive from “Anything”

## *Context*

In this section we will use Figure 4 as the diagrammatic representation to illustrate contexts and perspectives.



Figure

As descriptions of something from a certain perspective, not all concepts are the same. You may trust some and not others, some may be true of a certain timeframe, and some may only be valid in specific conditions. A central problem in information federation is that the context of information from various information repositories is different. Different authorities may or may not agree on facts, they may or may not use the same or conflicting terms, they may describe different aspects of the same thing. Each description, each concept, needs to be understood within the context in which it is defined and used. SIMF defines “context” as a “first class” concept – that means that dealing with context is part of using SIMF.

In SIMF, a context connects a set of facts with a set of things those facts applies to. The context *contextualizes* a set of concepts that it applies to. Facts and rules are defined in one or more context and apply to everything those contexts contextualize.

There is a fact that the Eiffel Tower is the tallest building in Paris. Who said that? Can you believe it? If I got that fact from Wikipedia, does that make it believable? The source of information is one of the contextual dimensions. What if a taller building were built in 2020? The fact that “the Eiffel Tower is the tallest building in Paris” would be true in the timeframe from 1889-2020. Timeframe can also contextualize a fact. From this example we can see that the applicability and interpretation of a fact is contextual and that multiple contexts may be applicable to any one fact. Your perspective defines what contexts are meaningful and trusted in a situation. SIMF defines context and provides for the provenance of information, the interpretation of that information to form “trust” and “belief” is outside of this specification. Tools that build on SIMF may provide was to evaluate or quantify trust and belief based on the information SIMF provides.

## *Entities*

Individual things that exist in our world, or some conceived world, are entities. Examples of entities include Barack Obama, The Eiffel Tower, Hurricane Andrew, an agreement between Barack Obama and the country France, the mortgage for my house, etc. Entities include physical things (with a “special/temporal extent) that do exist, may exist in the future or have existed in the past. Entities also include specific things we conceive, such as agreements between specific individuals. Entities have a “lifetime”, and they may change over that lifetime however that lifetime may or may not be known and we may have facts about an entity before, during and after its lifetime

## *Situations*

Various configurations and properties of things exist over some timeframe, as understood from some perspective. Such configurations of things are *situations*. A situation is a description of any configuration of things, including the logical, temporal and physical relationships between them. Situations can be large and comprehensive – like our solar system or as granular as your weight at a particular time. Facts about anything can be contextualized by a situation and that situation may, in turn, be contextualized – perhaps by a timeframe. Situations may be real or imagined, in the past, present or future.

Situations form the basis for relationships and facts in SIMF as any fact or relationship is bound by a time dimension when it was true. As all situations are bound by some timeframe where they are true (regardless of whather that timeframe is known or not), situations are the link between “things” and time.

## Relations

Relations (a kind of situation) are concepts that logically connect other concepts and represent the primary semantic elements of SIMF. For example, the height of the Eiffel Tower is a “Height” relation between the Eiffel Tower and the quantity “324 meters”. That my computer is on my desk may be a “supports” relation between my desk (as the support) and my computer (as the thing supported). While the set of relations describing something may change over time, each relation is considered constant and atomic– the “parts” bound to the relation (the things they relate) don’t change, any change would be a new relation.

Relations can involve any number of “roles” that define how other concepts are connected to the relation, however there are special considerations given for the common case of a relation involving exactly 2 primary entities – binary relations.

It should also be noted that in SIMF relations are “first class”, that is relations may have relations and properties. For example, the timeframe a relation is true is a property of that relation.

## *Types*

Types categorize and describe classes of things. Everything in SIMF has at least one type and may have any number of other types as well. The set of types that categorize a concept may change over time and be specific to a context. Facts describe and may constrain the extent of any type.

### Individual Types

Most common things defined in information models are entity types – kinds of physical or conceptual individuals that exist over a specific timeframe.

### Value types

Most concepts represent things that can change over time – like the Eiffel Tower or the United States. There is another class that represent pure values that do not change and do not have a lifetime. Numbers are the most common kind of such values – you can’t “create” the concept of “5”, it “just is” based on our accepted theory of numbers. Values are typically used as the object of properties (also called attributes) such as something’s size, mass or some other quantity. Numbers, strings and certain fixed enumerated values are considered value types.

### Situation Types & Situational Roles

Situations have their own specialization of the type concept, one that defines additional types –*Roles*. A role represents the state or behavior of something involved in a particular kind of situation. For example, a person may play an “employee” role with respect to a company playing an “employer” role. The concept of the employee and employer is, in this case, specific to a particular employment relationship (a relationship is a kind of situation). Each situation type defines the roles that can be involved with that kind of situation.

Each role also defines the type of entities that can play the role. For example, the employee role can only be played by a person. Roles are bound to individuals in a situation. Roles also serve the purpose of “variables” and “arguments” in programming languages.

Since roles are types, all type relations apply to roles.

Examples:

* Barack Obama is bound to the “President” role in the Organization “United States of America” during the period January 20th, 2009 (Noon EST) to January 20th, 2016 (Noon EST).
* The “Eiffel Tower” is bound to the “subject” role and “324 meters” is bound to the “object” role in the directed relation “Height”.

**Note on naming roles**: In UML it is best practice to name association ends (roles). In SIMF such names are optional. If the role adds no new semantics (other than the involvement with the situation or relation) the name can be inferred from the name of the type that plays the role – it is <type> in the context of the situation. Forcing users to make up more names than is necessary is confusing and increases the terms that have to be understood, thus we only want to name unique concepts.

### Relation Types

Relation types are a specific kind of situation type that represents a semantic connection between the involved roles. Relations are atomic semantic units that cannot be further decomposed or changed. A change in the semantic relationship between concepts requires a new relationship, perhaps of the same type.

## Rules

Rules define something that must be considered true within a context (including situations, types and relations). There are various kinds of rules, some of which are defined in SIMF and some which are defined in SIMF models. Rules may have an <initial situation> which is a template to set up any preconditions for the rule and <apply to> a set of context.

### Production Rules

A production rule computes a set of inferred relations based on a template when the rules <condition> is TRUE. If <implicit> is true the rule is unconditional and will “fire” whenever the condition is TRUE, otherwise the rule must be explicitly invoked.

### As-A Rule

An As-A rule provides for an instance of a type to be viewed as (or coerced to) another type when an instance of the <source type> is bound to a role requiring the <destination type>. As-a rules provides for a temporary façade over the instance, which may be useful as a bridging relation. The As-A rule implies the <destination type> and the <implies> template.

### Logical Grounding Rules

A logical grounding defines one context<applies to> (which is frequently a relation type) as defined by a set of more basic relations <implies> where <condition> is TRUE for <applies to>.

### Constraints

Constraints define some condition that must be TRUE, else the model is not valid. A constraint may be enforced for all time (at once = false) or at a particular instant in time elements (at once = true). SIMF does not infer any elements to attempt to satisfy a constraint, such inferences are done with production rules or by external systems.

Constraints are evaluated at the end of a transaction.

### General Constraints

A general constraint has a constraint expression (condition) that must evaluate to TRUE for the constraint to be satisfied.

### Multiplicity Constraints

A multiplicity defines the maximum or minimum number of relations between two types (normally roles) within the context (normally a relation) the rule applies to.

### Uniqueness Constraints

A uniqueness constraint identifies a set of roles. The values of these rules taken as a set must be unique within the context the rule applies to.

## Quantifiers

When expressing rules it is frequently required to express relations ranging across all or some instances of a type. Specific roles may be defined that range over (quantify) a type. The specific kinds of quantifiers are:

* All – all instances of the type.
* There exists – at least one instance of the type.
* One of – exactly one instance of the type.
* Most – An extension of “there exists” which applies in most situations, but not all. This can be used for defaults.

Quantifiers may be used to make statements about all elements of a type. For example: most teenage boys like supermodels.

## Expressions

SIMF provides for general expression which are then used in other parts of SIMF, for example in rules. The SIMF expression meta model is minimal, the bulk of the functionality to be provided by function libraries. The built-in capabilities are:

* Constant reference
* Role value
* Traversal of a relation
* Function Call

The philosophy of SIMF expressions is to put almost all expression semantics into functions. A function library is anticipated but not yet defined.

## Properties & Units

Properties are values that describe a concept as a quantity or other value. Other terms for Properties are “Characteristics” or “Attributes”. To provide for a semantic definition, properties always have a domain-relevant type. In the case of quantities (a property that can be identified with a number and a Unit), there is always a unit corresponding to some “quantity kind” such as temperature or mass. Units are, of course, critical in understanding what a property really means. Properties are defined with “Property types” that are a kind of directed relation. The “object” of the directed relation for a characteristic is always a value type.

Based on the definitions in JCGM 200:2008 : A “Quantity Property Type” (e.g. Height) has a range of a “Quantity Kind” (e.g. Length) which is represented by a “Measurement Unit” (e.g. Meters) . An instance of a Measurement Unit is a “Quantity Value” (e.g. 324 Meters).

Property values that are not quantities (having a unit) may have any value type as their object type.

A property an unit library is currently defined in the threat and risk submission and will be moved to SIMF.

## Naming

A SIMF concept can have any number of names. A “Term” binds a string label to a concept in some context. A term has a “Priority” to define how preferred that name is in some context. Names may or may not be unique.

## Representations

When bridging from a conceptual model to a logical or physical model there must be ways to represent concepts. Representations are also concepts and a Representation Type is a specialization of “Type”. A representation “Represents” a concept and a Representation Type represents some other type.

While this forms the basic meta-model for representation, rules are required to specify the details of representations.

## Important Relations

### Contextualization

The “Extent” relation exists between a context and the concepts it <contextualizes>. The inverse is <in context of>, Contextualization is the basis for many other relations in SIMF, including the relation between a type and the instances it <categorizes>.

### Equivalence

As has been explained previously, a concept describes something or some set of things. There can be many such descriptions in different context. An *equivalence* relation states that a set of concepts describe exactly the same thing or set of things – they are different concepts of or assertions about the same thing “in the world”.

**Equivalence and open world:** Equivalence is one mechanism that SIMF uses to support an “open world”, that is that any model can assert anything about anything anywhere. This is accomplished by defining concepts of the same thing in different context and stating that they are equivalent.

**Equivalence and consistency**: Equivalence in SIMF does not assert consistency as do some logical languages; different descriptions of the same thing may or may not be consistent. A SIMF tool may be able to determine if two contexts are consistent.

### Generalization

Generalization defines one context (including situations and types) as specializing another and, conversely, that one context generalizes the other. The set of concepts contextualized by the specialized context must be the same as or a subset of the concepts contextualized by the general concept. Further, any rules defined for the generalized concept must apply to the more specialized concept.

Generalization is typically applied between types as subtypes, supertypes. However any context can participate in a generalization.

## SIMF Lexical Scope & Physical Representations

A SIMF model is an information resource and as such has structure and semantics as defined in the SIMF abstract syntax. Only a small subset of that abstract syntax is relevant to the structure of the information, this structure is defined in the “Lexical Scope” portion of the model. Only elements stereotyped as “Structural” are part of the logical information structure, all other elements are represented using that basic foundation. The lexical scope structure is than mapped to various technology formats such as XMI and RDF.

Like most modeling languages, a SIMF model is “block structured”, that is certain elements contain other elements. The model structure is one form of context that may or may not be relevant to its interpretation, SIMF attaches no domain semantics to model structure. Most users will not be concerned with the details of the lexical structure, but an understanding of it is important for implementers, as follows…

A key element is the abstract “*Lexical Scope*” which can reference and/or import any context in any model. Referencing a context makes it visible within the lexical scope, importing a context asserts that external scope within the lexical scope. A lexical scope may also define Terms, which provide labels for concepts.

A *Model* is a lexical scope with no parent, it is an independent data resource.. A model is identified as modeling a particular level of abstraction (Conceptual, Logical, Physical or Binding) and subject area.

A Block (including its subtypes “Type” and “Situation”) are substypes of Lexical Scope and owned by a lexical scope. A block owned by a lexical scope is asserted by that scope. Blocks are similar to UML packages or statements within braces in Java.

Bindings bind an individual to a type (or role) in a situation. Each “end” of an instance of a relation is a binding.

Bindings are possible due to SIMF concept identity. Every concept in SIMF has identity: Every individual, relation, binding and type. For this reason SIMF relations may be made between anything. It may be noted that this is not “first order”, as in “first order logic” (FOL). Supporting a rich conceptual model and mappings requires concepts that are not first order. Portions of SIMF models may be mapped to first order languages to take advantage of their inference capabilities.

**Type coercion**: The referenced or local bound concept may or may not have a type compatible with the role it is playing. If it doesn’t and there is an As-A rule for coercing a type of the referenced concept to the requirements of the role, the As-A rule will fire to create a local “façade” for the concept of the appropriate type. This type coercion is one of SIMF’s model bridging capabilities.

### Transactions

SIMF is intended to support information in multiple repositories using diverse schema being federated. As part of that federation the concept of a Transaction is defined. Any change in a SIMF model visible to a SIMF implementation will generate a transaction. Each transaction will reference the elements created or invalidated by that transaction. A transaction may also spawn sub-transactions, some of which may be the result of rules that cause other changes. Every element of information visible to SIMF implementations can then be tracked to its source.

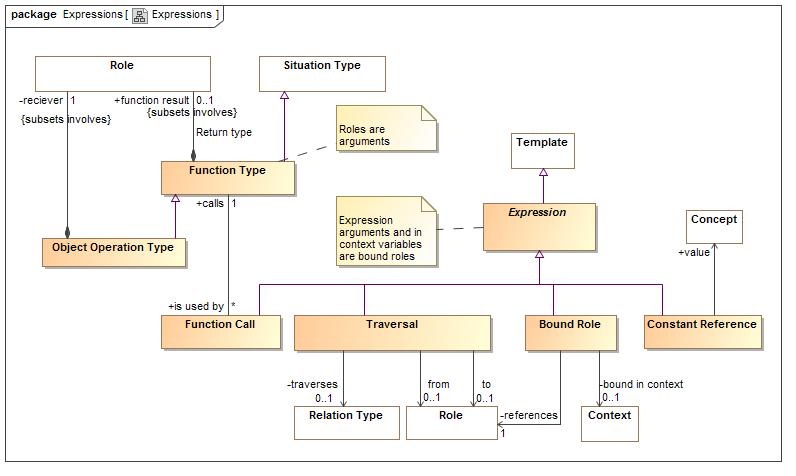
Transactions also assist in integration. It is not the intent that SIMF replicate every semantic of every data technology, that would be impractical. Where changes occur that would impact a data resource the changes (represented by transactions) of that impact are provided to technology adapters such that the change can be made – thus it is not expected or required that data resources would be wholesale converted to one format, changed and converted back – only changes need be propagated. However, transactions are a capability offered – it is up to implementations to define exactly how change is propagated.

# Abstract Syntax [Normative]

This section presents the normative specification for the SIMF metamodel. It begins with an overview of the metamodel structure followed by a description of each sub-package.

## SIMF Meta Model::Expressions

### Diagram: Expressions



1. Expressions

Expressions define computations

### Class Bound Role

A calculation that returns as a result a variable of the expression in context. i.e. a variable reference.

#### Direct Supertypes

[Expression](#_f9bba899ada544a47c36bb071e9024f5)

**package** SIMF Meta Model::Expressions

**extends** [Expression](#_f9bba899ada544a47c36bb071e9024f5)

#### Associations

-304893399.jpg bound in context:[Context](#_66d62b068053cee3464e1e03e6035eed) [0..1]

The context of the bound role, or the current context of not specified.

-304893399.jpg references:[Role](#_a8049a836c9b9b5d6df4b578a5836756) [1]

The role binding to reference

### Association Calculation type

**package** SIMF Meta Model::Expressions

#### Association Ends

-2141820704.jpg calls : [Function Type](#_cff99d2f22ee84a9e95ea582786a897b) [1]

-2141820704.jpg is used by : [Function Call](#_db3e44e523a232e5b77a133d74842e81) [\*]

### Class Computed Fact

The actual evaluation of an expression which may cause the creation of more assertions (i.e. representing the result of a calculation).

**package** SIMF Meta Model::Expressions

### Class Constant Reference

A calculation that returns a concept instance. Not 100% sure this is needed.

#### Direct Supertypes

[Expression](#_f9bba899ada544a47c36bb071e9024f5)

**package** SIMF Meta Model::Expressions

**extends** [Expression](#_f9bba899ada544a47c36bb071e9024f5)

#### Associations

-304893399.jpg value:[Concept](#_eb8398b5a178c638b98597120ec51c4d)

A constant value referenced in an expression.

### Class Expression

The computation of a value which is then bound to the role and context from which it is called.

#### Direct Supertypes

[Template](#_8d9c945b6f864c34fdd7a91d4d62755f)

**package** SIMF Meta Model::Expressions

**extends** [Template](#_8d9c945b6f864c34fdd7a91d4d62755f)

### Association from

**package** SIMF Meta Model::Expressions

#### Association Ends

-304893399.jpg : [Role](#_a8049a836c9b9b5d6df4b578a5836756) [0..1]

-304893399.jpg : [Traversal](#_0492440b12b90a76377a15324efa2182)

### Class Function Call

An element of an expression that performs some operation based on a function type and produces a result. I.e. plus(a,1).

#### Direct Supertypes

[Expression](#_f9bba899ada544a47c36bb071e9024f5)

**package** SIMF Meta Model::Expressions

**extends** [Expression](#_f9bba899ada544a47c36bb071e9024f5)

#### Associations

-2141820704.jpg calls:[Function Type](#_cff99d2f22ee84a9e95ea582786a897b) [1]

### Class Function Type

A kind of function which performs a calculation on arguments (roles) to produce a result (function result). I.e. the definition of plus().  
  
Functions are intended to be side-effect free and context free (they only depend on their arguments and don't change anything) but assertions to specify that certain functions are pure may be required,  
FUNCTION ARGUMENTS ARE QUANTIFICATIONS.

#### Direct Supertypes

[Situation Type](#_50241f5936e61055293ca95f860768d8)

**package** SIMF Meta Model::Expressions

**extends** [Situation Type](#_50241f5936e61055293ca95f860768d8)

#### Associations

-2141820704.jpg is used by:[Function Call](#_db3e44e523a232e5b77a133d74842e81) [\*]

1664306987.jpg function result:[Role](#_a8049a836c9b9b5d6df4b578a5836756) [0..1] *Subsets*: involves:[Role](#_a8049a836c9b9b5d6df4b578a5836756)

### Class Object Operation Type

An operation bound to a spcific "reciever" in the OO sense.

#### Direct Supertypes

[Function Type](#_cff99d2f22ee84a9e95ea582786a897b)

**package** SIMF Meta Model::Expressions

**extends** [Function Type](#_cff99d2f22ee84a9e95ea582786a897b)

#### Associations

1664306987.jpg reciever:[Role](#_a8049a836c9b9b5d6df4b578a5836756) [1] *Subsets*: involves:[Role](#_a8049a836c9b9b5d6df4b578a5836756)

### Association Return type

**package** SIMF Meta Model::Expressions

#### Association Ends

1664306987.jpg function result : [Role](#_a8049a836c9b9b5d6df4b578a5836756) [0..1] *Subsets*: involves:[Role](#_a8049a836c9b9b5d6df4b578a5836756)

1664306987.jpg : [Function Type](#_cff99d2f22ee84a9e95ea582786a897b) *Subsets*: involves:[Role](#_a8049a836c9b9b5d6df4b578a5836756)

### Association to

**package** SIMF Meta Model::Expressions

#### Association Ends

-304893399.jpg : [Role](#_a8049a836c9b9b5d6df4b578a5836756) [0..1] *Subsets*: involves:[Role](#_a8049a836c9b9b5d6df4b578a5836756)

-304893399.jpg : [Traversal](#_0492440b12b90a76377a15324efa2182) *Subsets*: involves:[Role](#_a8049a836c9b9b5d6df4b578a5836756)

### Class Traversal

Traversal from the current context to another across a relation. Where <traverses>, <from> or <to> is not defined, any value will suffice.

#### Direct Supertypes

[Expression](#_f9bba899ada544a47c36bb071e9024f5)

**package** SIMF Meta Model::Expressions

**extends** [Expression](#_f9bba899ada544a47c36bb071e9024f5)

#### Associations

-304893399.jpg traverses:[Relation Type](#_cfa3d81f355fabfe3776e75965ca1870) [0..1]

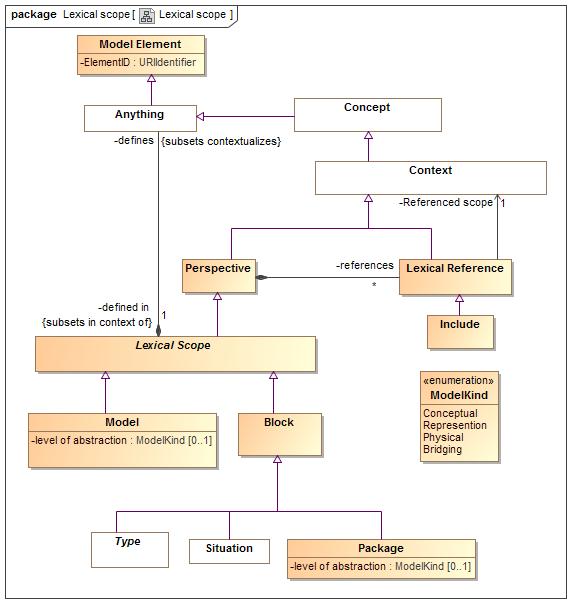
-304893399.jpg :[Role](#_a8049a836c9b9b5d6df4b578a5836756) [0..1]

-304893399.jpg :[Role](#_a8049a836c9b9b5d6df4b578a5836756) [0..1]

## SIMF Meta Model::Lexical scope

Lexical scope defines the structure of models

### Diagram: Lexical scope



1. Lexical scope

An indicator that the referenced context is asserted by (included by reference) the referencing context.

### Class Block

A Block defines a set of types and situations.

#### Direct Supertypes

[Lexical Scope](#_693daf0a0de3f4b82a04aee474c3f151)

**package** SIMF Meta Model::Lexical scope

**extends** [Lexical Scope](#_693daf0a0de3f4b82a04aee474c3f151)

### Class Include

An external scope that is visible and asserted by the owning lexical scope.

#### Direct Supertypes

[Lexical Reference](#_0315319befc74caa0a2a7d36cff333c0)

**package** SIMF Meta Model::Lexical scope

**extends** [Lexical Reference](#_0315319befc74caa0a2a7d36cff333c0)

### Class Lexical Reference

An external scope that is visible to but not necessarily asserted by the owning lexical scope. If "is asserted" is true the lexical scope includes and assets the referenced scope. The reference becomes a proxy for the referenced context.

#### Direct Supertypes

[Context](#_66d62b068053cee3464e1e03e6035eed)

**package** SIMF Meta Model::Lexical scope

**extends** [Context](#_66d62b068053cee3464e1e03e6035eed)

#### Associations

-304893399.jpg Referenced scope:[Context](#_66d62b068053cee3464e1e03e6035eed) [1]

1664306987.jpg :[Perspective](#_260b5def3f83400255a93c832b5c2167)

### Class Lexical Scope

#### Direct Supertypes

[Perspective](#_260b5def3f83400255a93c832b5c2167)

**package** SIMF Meta Model::Lexical scope

**extends** [Perspective](#_260b5def3f83400255a93c832b5c2167)

#### Associations

1664306987.jpg defines:[Anything](#_a52cb0ff6e414b3170b58afe10b6afcb) *Subsets*: contextualizes:[Anything](#_a52cb0ff6e414b3170b58afe10b6afcb)

-2141820704.jpg changed in:[Transaction](#_dd21593da1e0b28c0a75ed488ef07fd3)

### Class Model

A model is an information resource containing SIMF model elements.

#### Direct Supertypes

[Lexical Scope](#_693daf0a0de3f4b82a04aee474c3f151)

**package** SIMF Meta Model::Lexical scope

**extends** [Lexical Scope](#_693daf0a0de3f4b82a04aee474c3f151)

#### Attributes

-1753260495.jpg level of abstraction : [ModelKind](#_f8d5fc9c395254768dfd7ddce8807215) [0..1]

### Class Model Element

Any fact or part of a fact contained in a SIMF model.

**package** SIMF Meta Model::Lexical scope

#### Attributes

-1753260495.jpg ElementID : [URIIdentifier](#_f904ff1da5bfc3387d892b7e0fe9ecb1)

A "universal ID" for a model element, not to be confused with the ID of what the model element represents.

### Class Package

A group of model elements that provides context for those elements.

#### Direct Supertypes

[Block](#_2aa128e17c8314200d6ed8ee7ec053cc)

**package** SIMF Meta Model::Lexical scope

**extends** [Block](#_2aa128e17c8314200d6ed8ee7ec053cc)

#### Attributes

-1753260495.jpg level of abstraction : [ModelKind](#_f8d5fc9c395254768dfd7ddce8807215) [0..1]

### Class Perspective

A perspective defines the set of context meaningful to a stakeholder or lexical scope.

#### Direct Supertypes

[Context](#_66d62b068053cee3464e1e03e6035eed)

**package** SIMF Meta Model::Lexical scope

**extends** [Context](#_66d62b068053cee3464e1e03e6035eed)

#### Associations

1664306987.jpg references:[Lexical Reference](#_0315319befc74caa0a2a7d36cff333c0) [\*]

The context that is referenced by a given perspective.

#### Enumeration ModelKind

package SIMF Meta Model::Lexical scope

public enum ModelKind

{Conceptual, Represention, Physical, Bridging}

##### Literals

114961554.jpg Conceptual

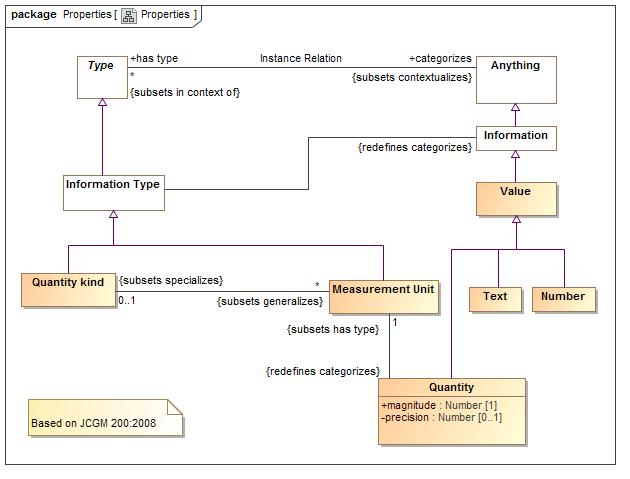
114961554.jpg Represention

114961554.jpg Physical

114961554.jpg Bridging

## SIMF Meta Model::Properties

### Diagram: Properties



1. Properties

Properties define quantities, their kinds and information values.

### Class Measurement Unit

real scalar quantity, defined and adopted by  
convention, with which any other quantity of the  
same quantity kind can be compared to express the ratio of  
the two quantities as a number. e.g. Degrees Centigrade, Miles.

#### Direct Supertypes

[Information Type](#_3f8b371eee11a630805f23f755460b79)

**package** SIMF Meta Model::Properties

**extends** [Information Type](#_3f8b371eee11a630805f23f755460b79)

#### Associations

-2141820704.jpg :[Quantity kind](#_ca1c56b440439615024c837658185d15) [0..1] *Subsets*: specializes:[Context](#_66d62b068053cee3464e1e03e6035eed)

-2141820704.jpg :[Quantity](#_746074a5bfbd6e26906da5d4bd0d2a7f) *Redefines*: categorizes:[Anything](#_a52cb0ff6e414b3170b58afe10b6afcb)

### Class Number

Any reprsentation of a number

#### Direct Supertypes

[Value](#_a739673c8d53da123e392b7e5059ceec)

**package** SIMF Meta Model::Properties

**extends** [Value](#_a739673c8d53da123e392b7e5059ceec)

### Class Quantity

property of a phenomenon, body, or substance,  
where the property has a magnitude that can be  
expressed as a number and a reference. In SIMF, the reference is the type of the quantity value, which is a measurement unit.  
e.g. 5cm.

#### Direct Supertypes

[Value](#_a739673c8d53da123e392b7e5059ceec)

**package** SIMF Meta Model::Properties

**extends** [Value](#_a739673c8d53da123e392b7e5059ceec)

#### Attributes

-1753260495.jpg magnitude : [Number](#_f5a86db7bd9636f0fa472c3859bc9c3c) [1]

-1753260495.jpg precision : [Number](#_f5a86db7bd9636f0fa472c3859bc9c3c) [0..1]

#### Associations

-2141820704.jpg :[Measurement Unit](#_9a97d5f73bf658c81147f5fab194bf88) [1] *Subsets*: has type:[Type](#_dfe1514224ca21cedba7b2b29802db50)

### Class Quantity kind

Aspect common to mutually comparable quantities represented by one or more units. Units with a common quantity kind may be algorithmically converted to any other unit of that quantity kind. e.g. temperature.

#### Direct Supertypes

[Information Type](#_3f8b371eee11a630805f23f755460b79)

**package** SIMF Meta Model::Properties

**extends** [Information Type](#_3f8b371eee11a630805f23f755460b79)

#### Associations

-2141820704.jpg :[Measurement Unit](#_9a97d5f73bf658c81147f5fab194bf88) [\*] *Subsets*: generalizes:[Context](#_66d62b068053cee3464e1e03e6035eed)

### Class Text

Text is a value represented using symbols which have a meaning to stakeholders but otherwise have no formal semantic implication. Properties involving values may have a semantic implication.

#### Direct Supertypes

[Value](#_a739673c8d53da123e392b7e5059ceec)

**package** SIMF Meta Model::Properties

**extends** [Value](#_a739673c8d53da123e392b7e5059ceec)

### Class Value

A value is an atomic piece of information without a specific lifetime or identity independent of the value. Values include numbers, strings and other atomic "primitive" data.

#### Direct Supertypes

[Information](#_1d7f13297e70f68af8189a69a2e52b05)

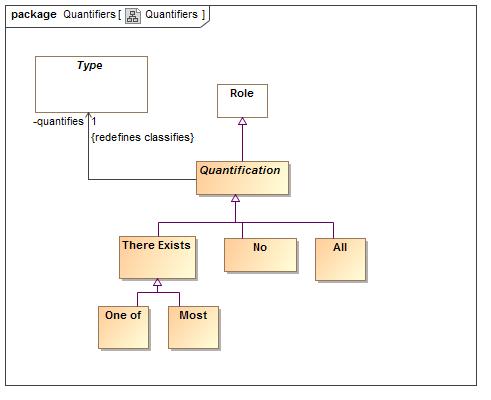
**package** SIMF Meta Model::Properties

**extends** [Information](#_1d7f13297e70f68af8189a69a2e52b05)

## SIMF Meta Model::Quantifiers

Quantifiers define roles that may be used in rules and templates such that these roles apply to a quantified set of instances of a type.  
A quantification can then be used in a relation to define facts about the quantified set.

### Diagram: Quantifiers



1. Quantifiers

### Class All

The universal quantifier - the quantified variable is a stand-in for all elements of the existent of the quantified type

#### Direct Supertypes

[Quantification](#_0983c67ae2bb53c9720e7f68c769280e)

**package** SIMF Meta Model::Quantifiers

**extends** [Quantification](#_0983c67ae2bb53c9720e7f68c769280e)

### Class Most

A stratified existential quantifier with a default for a "typical" value - example: People <typically> have 2 arms.

#### Direct Supertypes

[There Exists](#_573d030933634f018dfa7c402c8b9fd5)

**package** SIMF Meta Model::Quantifiers

**extends** [There Exists](#_573d030933634f018dfa7c402c8b9fd5)

### Class No

A quantifier where no instance of the type fills the role.

#### Direct Supertypes

[Quantification](#_0983c67ae2bb53c9720e7f68c769280e)

**package** SIMF Meta Model::Quantifiers

**extends** [Quantification](#_0983c67ae2bb53c9720e7f68c769280e)

### Class One of

The existential quantifier limited to exactly one of a potentially larger set

#### Direct Supertypes

[There Exists](#_573d030933634f018dfa7c402c8b9fd5)

**package** SIMF Meta Model::Quantifiers

**extends** [There Exists](#_573d030933634f018dfa7c402c8b9fd5)

### Class There Exists

The existential quantifier - at least one.

#### Direct Supertypes

[Quantification](#_0983c67ae2bb53c9720e7f68c769280e)

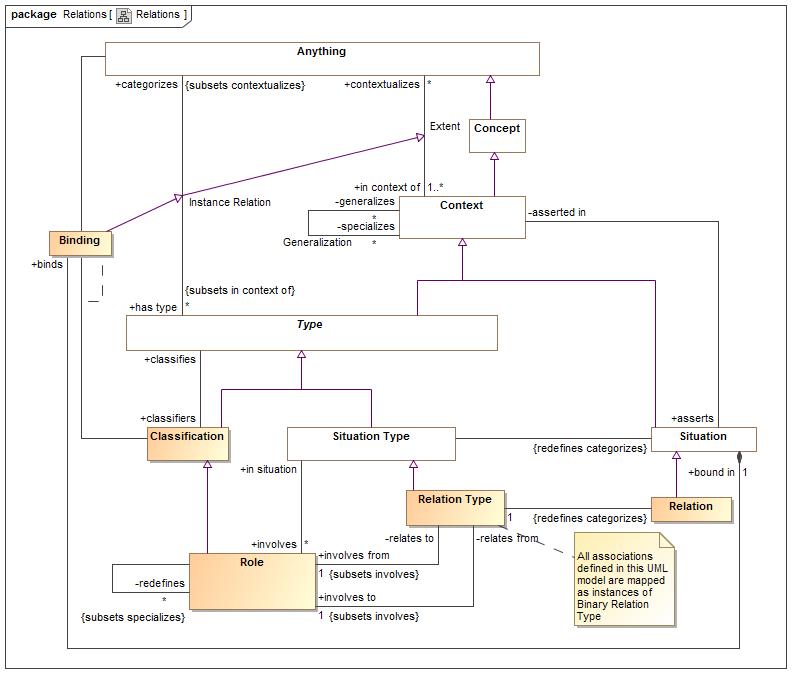
**package** SIMF Meta Model::Quantifiers

**extends** [Quantification](#_0983c67ae2bb53c9720e7f68c769280e)

## SIMF Meta Model::Relations

Relations are primitive facts about anything, relating at least 2 individuals through the roles they play.

### Diagram: Relations



1. Relations

Relations are atomic situations that bind two or more roles as an fact.

### Class Constrained Relation Type

A constrained relation type must specialize another relation. The constrained relation may introduce new roles and rules that correspond to subtypes of the more general relation. If the more general relation is used and any of the role types match the specialized relation, the specialized relation in used instead.

#### Direct Supertypes

[Relation Type](#_cfa3d81f355fabfe3776e75965ca1870)

**package** SIMF Meta Model::Relations

**extends** [Relation Type](#_cfa3d81f355fabfe3776e75965ca1870)

### Class Relation

AN atomic situation that holds between at least 2 entities or values.

#### Direct Supertypes

[Situation](#_8c517cf1950741c0f89edebf828214cc)

**package** SIMF Meta Model::Relations

**extends** [Situation](#_8c517cf1950741c0f89edebf828214cc)

#### Associations

-2141820704.jpg :[Relation Type](#_cfa3d81f355fabfe3776e75965ca1870) [1]

### Class Relation Type

A relation is an "atomic situation", one here its bindings do not change over time. However, the relation as a whole may have conditions and a timespan.  
  
A relation type represents common associations, assertions and attributes.  
  
Relations are considered "primarily binary" and the primary roles are distinguished. However, a relation may involve other roles.

#### Direct Supertypes

[Situation Type](#_50241f5936e61055293ca95f860768d8)

**package** SIMF Meta Model::Relations

**extends** [Situation Type](#_50241f5936e61055293ca95f860768d8)

#### Associations

-2141820704.jpg involves from:[Role](#_a8049a836c9b9b5d6df4b578a5836756) [1] *Subsets*: involves:[Role](#_a8049a836c9b9b5d6df4b578a5836756)

One of the ends of a binary relation - the "from " side. The "from" and "to" distinction may or may not have meaning for a particular relation. Where it does the "From" side represents the "1" in 1:many, the subject or the part in a composite.

-2141820704.jpg involves to:[Role](#_a8049a836c9b9b5d6df4b578a5836756) [1] *Subsets*: involves:[Role](#_a8049a836c9b9b5d6df4b578a5836756)

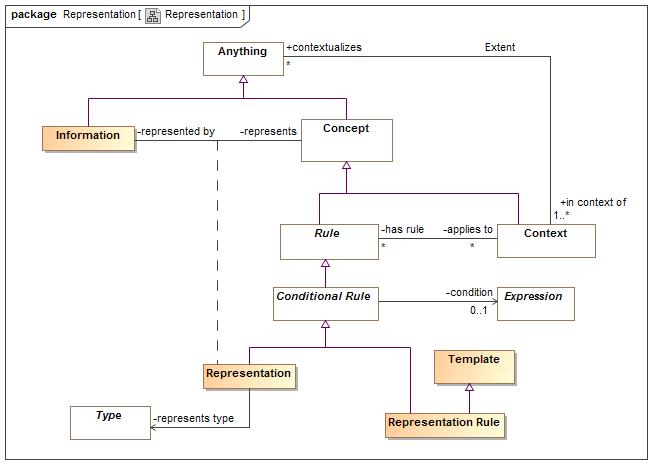
One of the ends of a binary relation - the "to" side. The "from" and "to" distinction may or may not have meaning for a particular relation. Where it does the "From" side represents the "1" in 1:many, the subject or the part in a composite.

-2141820704.jpg :[Relation](#_1a3bc7c4090871c29983008b01e078b1) *Redefines*: categorizes:[Anything](#_a52cb0ff6e414b3170b58afe10b6afcb)

## SIMF Meta Model::Representation

Representations connect a concept to the forms of information that represent facts about that concept.

### Diagram: Representation



1. Representation

Representation rules define how information represents concepts.

### Class Information

Information about something rather than the something it's self.

#### Direct Supertypes

[Anything](#_a52cb0ff6e414b3170b58afe10b6afcb)

**package** SIMF Meta Model::Representation

**extends** [Anything](#_a52cb0ff6e414b3170b58afe10b6afcb)

#### Associations

645683080.jpg represents:[Concept](#_eb8398b5a178c638b98597120ec51c4d)

The represented concept

-2141820704.jpg :[Information Type](#_3f8b371eee11a630805f23f755460b79)

### Association Class Representation

Representation is a relation between a concept and a specific way to represent information about that concept.

#### Direct Supertypes

[Conditional Rule](#_3d425949001fb1cb0502a6157c8cf51e)

**package** SIMF Meta Model::Representation

**extends** [Conditional Rule](#_3d425949001fb1cb0502a6157c8cf51e)

##### Association Ends

645683080.jpg represents : [Concept](#_eb8398b5a178c638b98597120ec51c4d)

The represented concept

645683080.jpg represented by : [Information](#_1d7f13297e70f68af8189a69a2e52b05)

The information that represents the concept

#### Associations

-304893399.jpg represents type:[Type](#_dfe1514224ca21cedba7b2b29802db50)

More specific types that the information represents. For example, if an XML property represents an identifier, a more specific identifier type may be specified.

### Class Representation Rule

A representation rule enforces a pattern of representations between two forms, for example between a physical class and a conceptual class.

#### Direct Supertypes

[Conditional Rule](#_3d425949001fb1cb0502a6157c8cf51e), [Template](#_8d9c945b6f864c34fdd7a91d4d62755f)

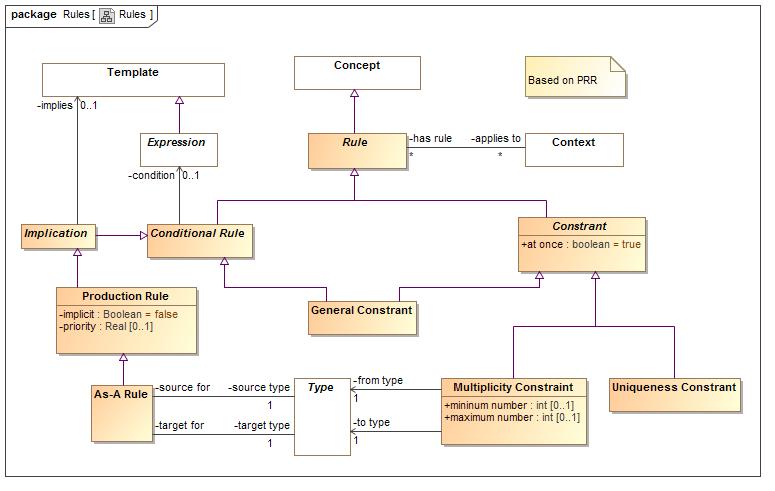
**package** SIMF Meta Model::Representation

**extends** [Conditional Rule](#_3d425949001fb1cb0502a6157c8cf51e), [Template](#_8d9c945b6f864c34fdd7a91d4d62755f)

## SIMF Meta Model::Rules

Rule define constraints or behaviors that must be applied in specified context.

### Diagram: Rules



1. Rules

### Class As-A Rule

Rule for lifting one type to another if the condition is true.   
  
When "implicit" is false: The individual so lifted is only lifted within the context the rule applies to, normally an as-a binding.  
  
When implicit is true: The individual so lifted globally.

#### Direct Supertypes

[Production Rule](#_017937deb28a2a90e53b831d96e041ba)

**package** SIMF Meta Model::Rules

**extends** [Production Rule](#_017937deb28a2a90e53b831d96e041ba)

#### Associations

-2141820704.jpg source type:[Type](#_dfe1514224ca21cedba7b2b29802db50) [1]

-2141820704.jpg target type:[Type](#_dfe1514224ca21cedba7b2b29802db50) [1]

### Class Conditional Rule

A rule with a general expression as a condition

#### Direct Supertypes

[Rule](#_82919e40af9ad2e13647e9d37bbf0956)

**package** SIMF Meta Model::Rules

**extends** [Rule](#_82919e40af9ad2e13647e9d37bbf0956)

#### Associations

-304893399.jpg condition:[Expression](#_f9bba899ada544a47c36bb071e9024f5) [0..1]

Condition that must be TRUE for the rule to "fire".

### Class Constrant

A rule that states that something must be true in the context it applies to..

#### Direct Supertypes

[Rule](#_82919e40af9ad2e13647e9d37bbf0956)

**package** SIMF Meta Model::Rules

**extends** [Rule](#_82919e40af9ad2e13647e9d37bbf0956)

#### Attributes

-1753260495.jpg at once : [boolean](#_704bcc994dd8a2770c2713df173b4bb9) = true

When at once is true, the constraint applies for each snapshot in time but not across snapshots (e.g. a car can have at most one driver at a time). When at once is false the constraint applies across all time (e.g. a person has exactly one birth mother across all time).

### Class General Constrant

A constraint with a general expression as its condition. The expression must be true.

#### Direct Supertypes

[Conditional Rule](#_3d425949001fb1cb0502a6157c8cf51e), [Constrant](#_a965ad950cc9cf33be9b946923c696c7)

**package** SIMF Meta Model::Rules

**extends** [Constrant](#_a965ad950cc9cf33be9b946923c696c7), [Conditional Rule](#_3d425949001fb1cb0502a6157c8cf51e)

### Class Implication

IMplication imples a pattern based on a condition

#### Direct Supertypes

[Conditional Rule](#_3d425949001fb1cb0502a6157c8cf51e)

**package** SIMF Meta Model::Rules

**extends** [Conditional Rule](#_3d425949001fb1cb0502a6157c8cf51e)

#### Associations

-304893399.jpg implies:[Template](#_8d9c945b6f864c34fdd7a91d4d62755f) [0..1]

### Class Multiplicity Constraint

Constrains the number of times members of the constrained context or constrained individual may be involved in situations that bind the constraining role

#### Direct Supertypes

[Constrant](#_a965ad950cc9cf33be9b946923c696c7)

**package** SIMF Meta Model::Rules

**extends** [Constrant](#_a965ad950cc9cf33be9b946923c696c7)

#### Attributes

-1753260495.jpg mininum number : [int](#_0d30278207cac92be6fa561506a22f92) [0..1]

-1753260495.jpg maximum number : [int](#_0d30278207cac92be6fa561506a22f92) [0..1]

#### Associations

-304893399.jpg from type:[Type](#_dfe1514224ca21cedba7b2b29802db50) [1]

-304893399.jpg to type:[Type](#_dfe1514224ca21cedba7b2b29802db50) [1]

Type referenced by the "from type", normally a role or a situation type - but can be any type.

### Class Production Rule

A production rule computes an implied set of concepts based on each element contextualized by what the rule applies to where those elements satisfy the condition. A production rule is expected to contain at least one quantification.

#### Direct Supertypes

[Implication](#_63d69e49de8214503f0947e7f9dbc652)

**package** SIMF Meta Model::Rules

**extends** [Implication](#_63d69e49de8214503f0947e7f9dbc652)

#### Attributes

-1753260495.jpg implicit : [Boolean](#_6119a00b0834641b9fe3f5ae9f58237f) = false

When implicit is true the rule applies whenever the conditions are met. Otherwise the rule must be triggered in some way.

-1753260495.jpg priority : [Real](#_aef4bcae5ebc35dd9653214547b3e3cc) [0..1]

An optional attribute specifying the priority of a rule for use in determining the sequence of execution of Production  
Rules. Rules with higher priority values have higher priority than those with lower priority values.

### Class Rule

A rule is an expression that produces some effect.

#### Direct Supertypes

[Concept](#_eb8398b5a178c638b98597120ec51c4d)

**package** SIMF Meta Model::Rules

**extends** [Concept](#_eb8398b5a178c638b98597120ec51c4d)

#### Associations

-2141820704.jpg applies to:[Context](#_66d62b068053cee3464e1e03e6035eed) [\*]

### Class Uniqueness Constrant

A constraint that, within the context the rule applies to, the set of instances bound to the set of types in the "unique set" relation must be unique. A Uniqueness constraint that is not constrained by a situation indicates that the role may have only one member.  
Uniqueness may be used to define a "key".

#### Direct Supertypes

[Constrant](#_a965ad950cc9cf33be9b946923c696c7)

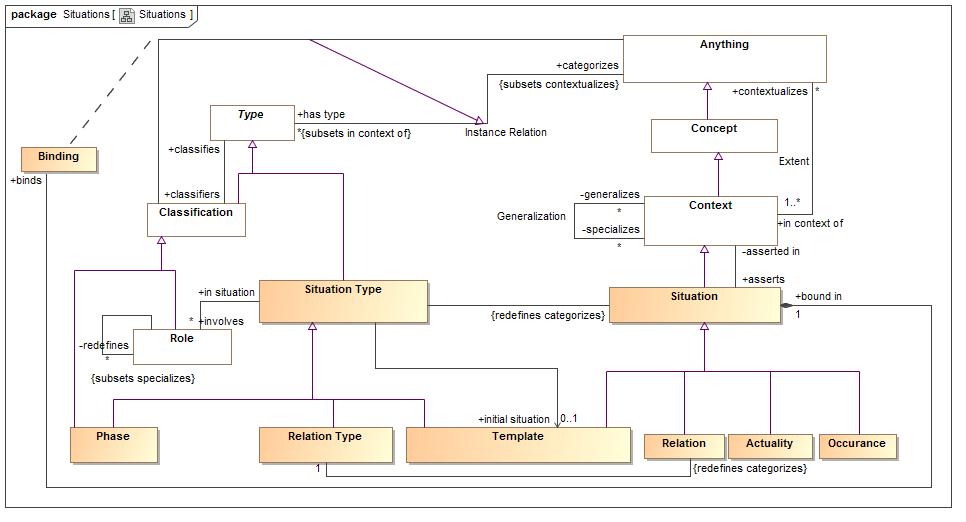
**package** SIMF Meta Model::Rules

**extends** [Constrant](#_a965ad950cc9cf33be9b946923c696c7)

## SIMF Meta Model::Situations

Situations are arrangements of individuals, assertions and the relations and assertions between them over a timespan. Any "condition" that exists is a situation - including relations.

### Diagram: Situations



1. Situations

A set of role bindings about a specific subject

### Class Actuality

An individual situation that actually exists or will exist

#### Direct Supertypes

[Situation](#_8c517cf1950741c0f89edebf828214cc)

**package** SIMF Meta Model::Situations

**extends** [Situation](#_8c517cf1950741c0f89edebf828214cc)

### Association Class Binding

A binding binds a particular individual to a role in a situation.   
For relations (which are situation), this binds an association "end".

#### Direct Supertypes

[Instance Relation](#_7930d7b301f56f0155603422a27ad833)

**package** SIMF Meta Model::Situations

**extends** [Instance Relation](#_7930d7b301f56f0155603422a27ad833)

##### Association Ends

645683080.jpg : [Anything](#_a52cb0ff6e414b3170b58afe10b6afcb)

645683080.jpg : [Classification](#_3b2e69eb6121d1e3a1180bbe8ee64013)

#### Associations

1664306987.jpg bound in:[Situation](#_8c517cf1950741c0f89edebf828214cc) [1]

### Class Occurance

A situation that is happening (changing) over time.

#### Direct Supertypes

[Situation](#_8c517cf1950741c0f89edebf828214cc)

**package** SIMF Meta Model::Situations

**extends** [Situation](#_8c517cf1950741c0f89edebf828214cc)

### Class Phase

A situation involving a specific set of individuals with values and relations constant over a specific timespan. e.g. a typical "record" or DBMS. The set of individuals are those contextualized by the snapshot. Probably an SBVR "Actuality"

#### Direct Supertypes

[Classification](#_3b2e69eb6121d1e3a1180bbe8ee64013), [Situation Type](#_50241f5936e61055293ca95f860768d8)

**package** SIMF Meta Model::Situations

**extends** [Situation Type](#_50241f5936e61055293ca95f860768d8), [Classification](#_3b2e69eb6121d1e3a1180bbe8ee64013)

### Association Redefinition

THe substitution for one role for another in a situation. Used for redefinition and role/pattern synthesis.

### Direct Supertypes

[Generalization](#_9668c4b6815cfd111fa02c95e2a3802a)

**package** SIMF Meta Model::Situations

**extends** [Generalization](#_9668c4b6815cfd111fa02c95e2a3802a)

#### Association Ends

-2141820704.jpg redefines : [Role](#_a8049a836c9b9b5d6df4b578a5836756) [\*]

THe Role(s) a situational role redefines, or substitutes for. A kind of generalization. Note that UML "Subsets" equates to generalization of a role tole.

-2141820704.jpg : [Role](#_a8049a836c9b9b5d6df4b578a5836756)

### Class Role

A role with respect to some situation, e.g. Joe is teaching class A123. Roles owned by relations are the "ends" of that relation.

#### Direct Supertypes

[Classification](#_3b2e69eb6121d1e3a1180bbe8ee64013)

**package** SIMF Meta Model::Situations

**extends** [Classification](#_3b2e69eb6121d1e3a1180bbe8ee64013)

#### Associations

-2141820704.jpg in situation:[Situation Type](#_50241f5936e61055293ca95f860768d8)

1664306987.jpg :[Function Type](#_cff99d2f22ee84a9e95ea582786a897b)

1664306987.jpg :[Object Operation Type](#_e6c2e5d52e1652a6c3d27d411345c754)

-2141820704.jpg :[Object Operation Type](#_e6c2e5d52e1652a6c3d27d411345c754)

-2141820704.jpg redefines:[Object Operation Type](#_e6c2e5d52e1652a6c3d27d411345c754)

THe Role(s) a situational role redefines, or substitutes for. A kind of generalization. Note that UML "Subsets" equates to generalization of a role tole.

-2141820704.jpg relates to:[Relation Type](#_cfa3d81f355fabfe3776e75965ca1870)

The relation owning the from side role

-2141820704.jpg relates from:[Relation Type](#_cfa3d81f355fabfe3776e75965ca1870)

The relation owning the to side role

### Class Situation

An identifiable arrangement of individuals, assertions and the relations and assertions between them over a timespan. Any "condition" that exists is a situation

#### Direct Supertypes

[Block](#_2aa128e17c8314200d6ed8ee7ec053cc), [Context](#_66d62b068053cee3464e1e03e6035eed)

**package** SIMF Meta Model::Situations

**extends** [Context](#_66d62b068053cee3464e1e03e6035eed), [Block](#_2aa128e17c8314200d6ed8ee7ec053cc)

#### Associations

-2141820704.jpg :[Situation Type](#_50241f5936e61055293ca95f860768d8)

1664306987.jpg binds:[Binding](#_4050167f9abcfb2187d46a6d3c360798)

Bindings asserted in the situation.

-2141820704.jpg asserted in:[Context](#_66d62b068053cee3464e1e03e6035eed)

Context in which the situation is considered true.

### Class Situation Type

A type of identifiable arrangement of individuals, assertions and the relations and assertions between them over a timespan

#### Direct Supertypes

[Type](#_dfe1514224ca21cedba7b2b29802db50)

**package** SIMF Meta Model::Situations

**extends** [Type](#_dfe1514224ca21cedba7b2b29802db50)

#### Associations

-304893399.jpg initial situation:[Template](#_8d9c945b6f864c34fdd7a91d4d62755f) [0..1]

The template for a new situation of a situation type, establishing initial values and relations.

-2141820704.jpg involves:[Role](#_a8049a836c9b9b5d6df4b578a5836756) [\*]

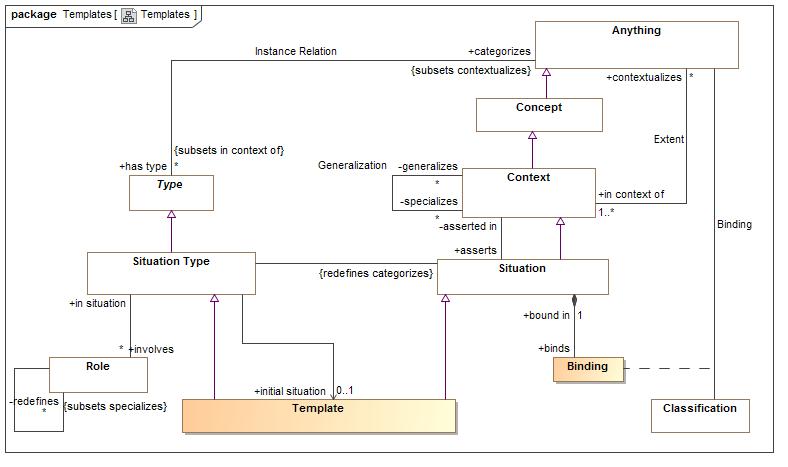
The roles involved in (bound to) a kind of situation.

-2141820704.jpg :[Situation](#_8c517cf1950741c0f89edebf828214cc) *Redefines*: categorizes:[Anything](#_a52cb0ff6e414b3170b58afe10b6afcb)

## SIMF Meta Model::Templates

Templates are patterns that may then be expressed as instances of the pattern.

### Diagram: Templates



1. Templates

### Class Template

A template represents a pattern. A template provides the initial relations, which may be expressions, to be asserted for any situation of a given situation type. e.g. initial or default values.   
The structure of the template is defined by the bindings and asserted (sub) situations that derive from the template being a situation.  
  
In many cases the roles of a situation will utilize quantifiers.

#### Direct Supertypes

[Situation](#_8c517cf1950741c0f89edebf828214cc), [Situation Type](#_50241f5936e61055293ca95f860768d8)

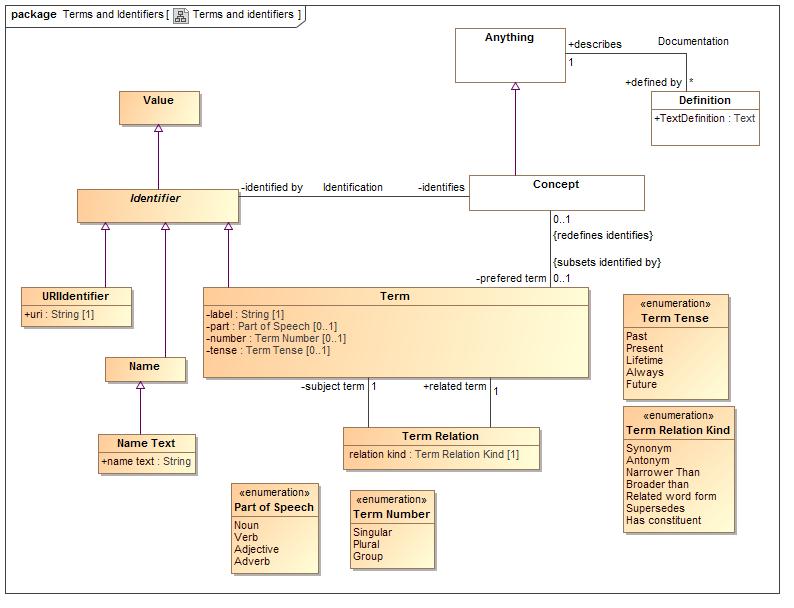
**package** SIMF Meta Model::Templates

**extends** [Situation Type](#_50241f5936e61055293ca95f860768d8), [Situation](#_8c517cf1950741c0f89edebf828214cc)

## SIMF Meta Model::Terms and Identifiers

Terms and identifiers provide for signs for (ways to identify) anything.

### Diagram: Terms and identifiers



1. Terms and identifiers

### Association Identification

**package** SIMF Meta Model::Terms and Identifiers

#### Association Ends

-2141820704.jpg identifies : [Concept](#_eb8398b5a178c638b98597120ec51c4d) *Redefines*: categorizes:[Anything](#_a52cb0ff6e414b3170b58afe10b6afcb)

A named concept

-2141820704.jpg identified by : [Identifier](#_18f8ef1b23e6cdf9278bd94f24f73c26) *Redefines*: categorizes:[Anything](#_a52cb0ff6e414b3170b58afe10b6afcb)

Names for a concept

### Class Identifier

Anything that is used to identify a concept

#### Direct Supertypes

[Value](#_a739673c8d53da123e392b7e5059ceec)

**package** SIMF Meta Model::Terms and Identifiers

**extends** [Value](#_a739673c8d53da123e392b7e5059ceec)

#### Associations

-2141820704.jpg identifies:[Concept](#_eb8398b5a178c638b98597120ec51c4d)

A named concept

### Class Name

a word or set of words by which a person, animal, place, or thing is known, addressed, or referred to

#### Direct Supertypes

[Identifier](#_18f8ef1b23e6cdf9278bd94f24f73c26)

**package** SIMF Meta Model::Terms and Identifiers

**extends** [Identifier](#_18f8ef1b23e6cdf9278bd94f24f73c26)

### Class Name Text

The text of a name, e.g. "Sam".

#### Direct Supertypes

[Name](#_c280f15fe0cba2ff4ebb6fcedeb9b6f4)

**package** SIMF Meta Model::Terms and Identifiers

**extends** [Name](#_c280f15fe0cba2ff4ebb6fcedeb9b6f4)

#### Attributes

-1753260495.jpg name text : [String](#_e8a6ce315d976318da3ab784a645ea44)

### Class Term

A term / element name: the binding between a label and a named concept.

#### Direct Supertypes

[Identifier](#_18f8ef1b23e6cdf9278bd94f24f73c26)

**package** SIMF Meta Model::Terms and Identifiers

**extends** [Identifier](#_18f8ef1b23e6cdf9278bd94f24f73c26)

#### Attributes

-1753260495.jpg label : [String](#_e8a6ce315d976318da3ab784a645ea44) [1]

An actual name of a concept as a string

-1753260495.jpg part : [Part of Speech](#_f85696be9951cc62b0bddea9998d56d4) [0..1]

-1753260495.jpg number : [Term Number](#_c6b743d51f58c4a241a52b46c1acd3eb) [0..1]

-1753260495.jpg tense : [Term Tense](#_c03d984bacd331889a1842e7fa9670ba) [0..1]

#### Associations

-2141820704.jpg :[Term Relation](#_69f840439731998eac0ac7698bb30c3a)

-2141820704.jpg :[Concept](#_eb8398b5a178c638b98597120ec51c4d) [0..1] *Redefines*: identifies:[Concept](#_eb8398b5a178c638b98597120ec51c4d)

-2141820704.jpg term relations:[Term Relation](#_69f840439731998eac0ac7698bb30c3a)

### Class Term Relation

A generic constraint stating that there is some relationship between concepts

**package** SIMF Meta Model::Terms and Identifiers

#### Attributes

-1753260495.jpg relation kind : [Term Relation Kind](#_b61706b9062af50db1ad9c38493dda87) [1]

#### Associations

-2141820704.jpg related term:[Term](#_1945edd0888993a52c5dc6467a7b3ef8) [1]

-2141820704.jpg subject term:[Term](#_1945edd0888993a52c5dc6467a7b3ef8) [1]

### Class URIIdentifier

A URI Identifier for a concept, Todo: Ref spec.

#### Direct Supertypes

[Identifier](#_18f8ef1b23e6cdf9278bd94f24f73c26)

**package** SIMF Meta Model::Terms and Identifiers

**extends** [Identifier](#_18f8ef1b23e6cdf9278bd94f24f73c26)

#### Attributes

-1753260495.jpg uri : [String](#_e8a6ce315d976318da3ab784a645ea44) [1]

#### Enumeration Part of Speech

package SIMF Meta Model::Terms and Identifiers

public enum Part of Speech

{Noun, Verb, Adjective, Adverb}

##### Literals

114961554.jpg Noun

114961554.jpg Verb

114961554.jpg Adjective

114961554.jpg Adverb

#### Enumeration Term Number

package SIMF Meta Model::Terms and Identifiers

public enum Term Number

{Singular, Plural, Group}

##### Literals

114961554.jpg Singular

One individual. (e.g. Bird)

114961554.jpg Plural

A set or type of individuals. (e.g. Birds)

114961554.jpg Group

A group of individuals. (e.g. flock)

#### Enumeration Term Relation Kind

package SIMF Meta Model::Terms and Identifiers

public enum Term Relation Kind

{Synonym, Antonym, Narrower Than, Broader than, Related word form, Supersedes, Has constituent}

##### Literals

114961554.jpg Synonym

All related concepts represent the same extent of individuals and/or types - nouns or verbs.

114961554.jpg Antonym

Relates concepts that have opposite meanings

114961554.jpg Narrower Than

A constraint specifying that the subject term has a narrower definition than related term

114961554.jpg Broader than

A constraint specifying that the subject term has a broader definition than related term

114961554.jpg Related word form

A relation identifying terms that are equivalent except for one or more qualifiers (kind, plurality, timeframe). eg: (dog & dogs) or( run, ran). Thus the different terms relate to the same family of concepts.

114961554.jpg Supersedes

The subject term replaces the related term

114961554.jpg Has constituent

The subject term includes the related term as a constituent of a phrase

#### Enumeration Term Tense

package SIMF Meta Model::Terms and Identifiers

public enum Term Tense

{Past, Present, Lifetime, Always, Future}

##### Literals

114961554.jpg Past

Past tense

114961554.jpg Present

Present tense

114961554.jpg Lifetime

True for the lifetime of the identified concept

114961554.jpg Always

Always true

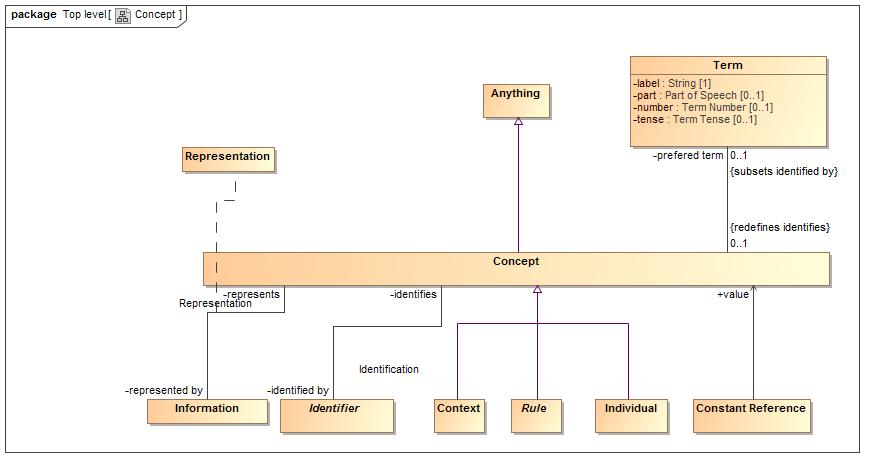
114961554.jpg Future

Future tense

## SIMF Meta Model::Top level

The top level objects provide the foundation for all objects in the model

### Diagram: Concept



1. Concept

### Class Anything

A model element representing anything in a model

#### Direct Supertypes

[Model Element](#_11c6f71d6b180ede328c4b04dc60b6a8)

**package** SIMF Meta Model::Top level

**extends** [Model Element](#_11c6f71d6b180ede328c4b04dc60b6a8)

#### Associations

-2141820704.jpg created by:[Transaction](#_dd21593da1e0b28c0a75ed488ef07fd3) [1]

-2141820704.jpg invalidated by:[Transaction](#_dd21593da1e0b28c0a75ed488ef07fd3) [0..1]

-2141820704.jpg defined by:[Definition](#_1a6d88e097d757268d09f68af82fbd34) [\*]

-2141820704.jpg has type:[Type](#_dfe1514224ca21cedba7b2b29802db50) [\*] *Subsets*: in context of:[Context](#_66d62b068053cee3464e1e03e6035eed)

One of the types defining a concept

-2141820704.jpg in context of:[Context](#_66d62b068053cee3464e1e03e6035eed) [1..\*]

The set of context that contextualizes a concept

645683080.jpg :[Classification](#_3b2e69eb6121d1e3a1180bbe8ee64013)

1664306987.jpg defined in:[Lexical Scope](#_693daf0a0de3f4b82a04aee474c3f151) [1] *Subsets*: in context of:[Context](#_66d62b068053cee3464e1e03e6035eed)

### Class Concept

A representation of the conception of anything, this includes individuals, types, axioms, situations, speech acts, information structures, numbers, etc.

#### Direct Supertypes

[Anything](#_a52cb0ff6e414b3170b58afe10b6afcb)

**package** SIMF Meta Model::Top level

**extends** [Anything](#_a52cb0ff6e414b3170b58afe10b6afcb)

#### Associations

645683080.jpg represented by:[Information](#_1d7f13297e70f68af8189a69a2e52b05)

The information that represents the concept

-2141820704.jpg identified by:[Identifier](#_18f8ef1b23e6cdf9278bd94f24f73c26)

Names for a concept

-2141820704.jpg prefered term:[Term](#_1945edd0888993a52c5dc6467a7b3ef8) [0..1] *Subsets*: identified by:[Identifier](#_18f8ef1b23e6cdf9278bd94f24f73c26)

### Class Context

A grouping of concepts contextualized in some way and with some common attributes and/or relations.

#### Direct Supertypes

[Concept](#_eb8398b5a178c638b98597120ec51c4d)

**package** SIMF Meta Model::Top level

**extends** [Concept](#_eb8398b5a178c638b98597120ec51c4d)

#### Associations

-2141820704.jpg contextualizes:[Anything](#_a52cb0ff6e414b3170b58afe10b6afcb) [\*]

The set of concepts "in" a context

-2141820704.jpg has rule:[Rule](#_82919e40af9ad2e13647e9d37bbf0956) [\*]

-2141820704.jpg generalizes:[Rule](#_82919e40af9ad2e13647e9d37bbf0956) [\*]

The more general role of a generalization.

-2141820704.jpg specializes:[Rule](#_82919e40af9ad2e13647e9d37bbf0956) [\*]

The more specific role of a generalization.

-2141820704.jpg asserts:[Situation](#_8c517cf1950741c0f89edebf828214cc)

All asserted situations (including bindings and relations) are true for anything contextualized by the context.

### Class Definition

Natural language definition of a concept

#### Direct Supertypes

[Anything](#_a52cb0ff6e414b3170b58afe10b6afcb)

**package** SIMF Meta Model::Top level

**extends** [Anything](#_a52cb0ff6e414b3170b58afe10b6afcb)

#### Attributes

-1753260495.jpg TextDefinition : [Text](#_0e6e6fe0a29fb43221940aa4118b04a2)

Text describing a concept in natural language

#### Associations

-2141820704.jpg describes:[Anything](#_a52cb0ff6e414b3170b58afe10b6afcb) [1]

### Association Documentation

**package** SIMF Meta Model::Top level

#### Association Ends

-2141820704.jpg describes : [Anything](#_a52cb0ff6e414b3170b58afe10b6afcb) [1]

-2141820704.jpg defined by : [Definition](#_1a6d88e097d757268d09f68af82fbd34) [\*]

### Association Extent

The association between a context and the set of concepts contextualized by that context.

**package** SIMF Meta Model::Top level

#### Association Ends

-2141820704.jpg contextualizes : [Anything](#_a52cb0ff6e414b3170b58afe10b6afcb) [\*]

The set of concepts "in" a context

-2141820704.jpg in context of : [Context](#_66d62b068053cee3464e1e03e6035eed) [1..\*]

The set of context that contextualizes a concept

### Association Generalization

The set of instances contextualized by the specialized context is a subset of the set of instances contextualized by the generalized context.

**package** SIMF Meta Model::Top level

#### Association Ends

-2141820704.jpg generalizes : [Context](#_66d62b068053cee3464e1e03e6035eed) [\*]

The more general role of a generalization.

-2141820704.jpg specializes : [Context](#_66d62b068053cee3464e1e03e6035eed) [\*]

The more specific role of a generalization.

### Class Individual

An individual is a person or a specific object, process, agreement, etc. Individuals do not have to be physical but do not include types, categories or values. Individuality (or selfhood) is the state or quality of being an individual; particularly of being a separate from other individuals and possessing identity. Individuals typically have a lifetime and may change over that lifetime. Individuals may have parts and processes that together help define the individual but may change over time. Also known as "Endurant".

#### Direct Supertypes

[Concept](#_eb8398b5a178c638b98597120ec51c4d)

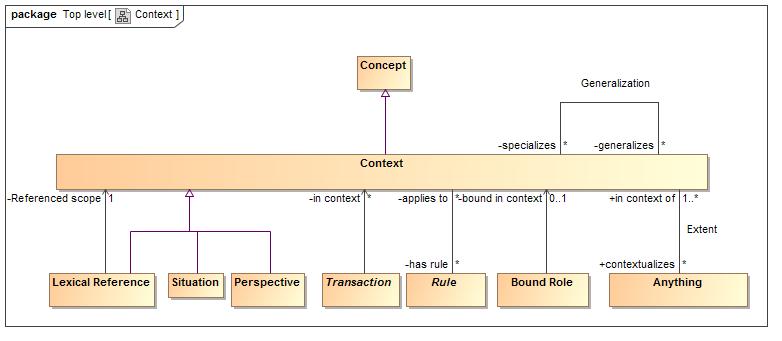
**package** SIMF Meta Model::Top level

**extends** [Concept](#_eb8398b5a178c638b98597120ec51c4d)

#### Associations

-2141820704.jpg transacts:[External Transaction](#_e50aee5acabb112d25a288860b20c3f8) [\*]

### Diagram: Context

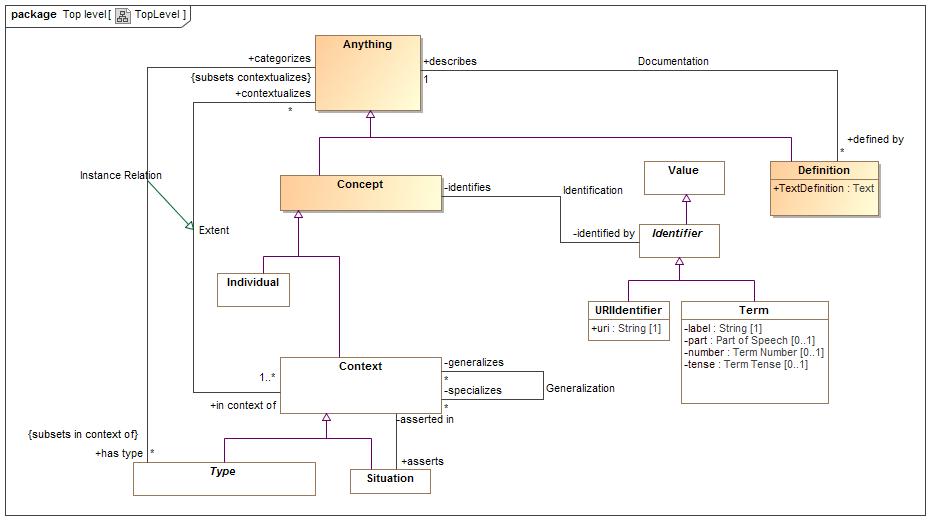


1. Context

#### Known other classes

[Class Anything](#_a52cb0ff6e414b3170b58afe10b6afcb), [Class Concept](#_eb8398b5a178c638b98597120ec51c4d), [Class Context](#_66d62b068053cee3464e1e03e6035eed), [Class Definition](#_1a6d88e097d757268d09f68af82fbd34), [Association Documentation](#_7c14183741a6bbb21d2fd7dc5685175f), [Association Extent](#_52c887644007b8e51a1f6e976113707a), [Association Generalization](#_9668c4b6815cfd111fa02c95e2a3802a), [Class Individual](#_67e305a274efbb1a54872a90447bb34e)

### Diagram: TopLevel



1. TopLevel

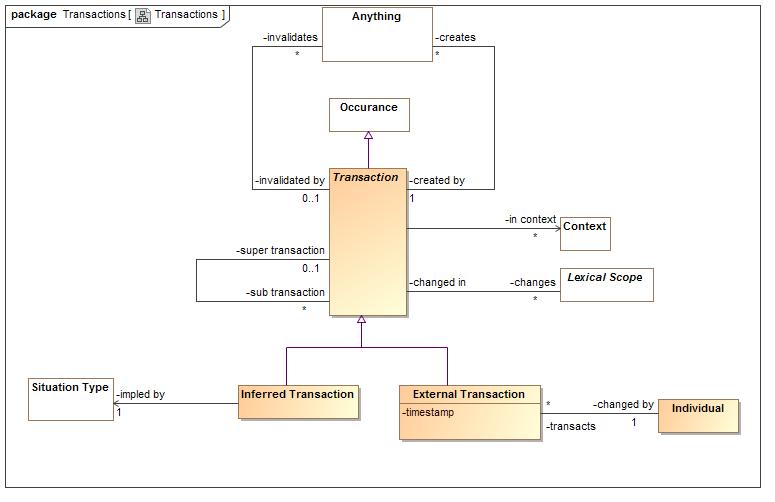
#### Known other classes

[Class Anything](#_a52cb0ff6e414b3170b58afe10b6afcb), [Class Concept](#_eb8398b5a178c638b98597120ec51c4d), [Class Context](#_66d62b068053cee3464e1e03e6035eed), [Class Definition](#_1a6d88e097d757268d09f68af82fbd34), [Association Documentation](#_7c14183741a6bbb21d2fd7dc5685175f), [Association Extent](#_52c887644007b8e51a1f6e976113707a), [Association Generalization](#_9668c4b6815cfd111fa02c95e2a3802a), [Class Individual](#_67e305a274efbb1a54872a90447bb34e)

## SIMF Meta Model::Transactions

Transactions provide for information changing over time, this provides the basis for synchronization of data sources and destinations.

### Diagram: Transactions



1. Transactions

### Class External Transaction

A transaction initated by an actor outside of the SIMF system.

#### Direct Supertypes

[Transaction](#_dd21593da1e0b28c0a75ed488ef07fd3)

**package** SIMF Meta Model::Transactions

**extends** [Transaction](#_dd21593da1e0b28c0a75ed488ef07fd3)

#### Attributes

-1753260495.jpg timestamp

#### Associations

-2141820704.jpg changed by:[Individual](#_67e305a274efbb1a54872a90447bb34e) [1]

### Class Inferred Transaction

A transaction that is the result of the execution of a rule.

#### Direct Supertypes

[Transaction](#_dd21593da1e0b28c0a75ed488ef07fd3)

**package** SIMF Meta Model::Transactions

**extends** [Transaction](#_dd21593da1e0b28c0a75ed488ef07fd3)

#### Associations

-304893399.jpg impled by:[Situation Type](#_50241f5936e61055293ca95f860768d8) [1]

### Class Transaction

A transaction is an atomic state change in an information system and is used as the basis for synchronization between different representations of the same concepts.

#### Direct Supertypes

[Occurance](#_63eecd782b89142ca4b95f2aad7bbd26)

**package** SIMF Meta Model::Transactions

**extends** [Occurance](#_63eecd782b89142ca4b95f2aad7bbd26)

#### Associations

-2141820704.jpg changes:[Lexical Scope](#_693daf0a0de3f4b82a04aee474c3f151) [\*]

-2141820704.jpg super transaction:[Lexical Scope](#_693daf0a0de3f4b82a04aee474c3f151) [\*]

-2141820704.jpg sub transaction:[Lexical Scope](#_693daf0a0de3f4b82a04aee474c3f151) [\*]

-304893399.jpg in context:[Context](#_66d62b068053cee3464e1e03e6035eed) [\*]

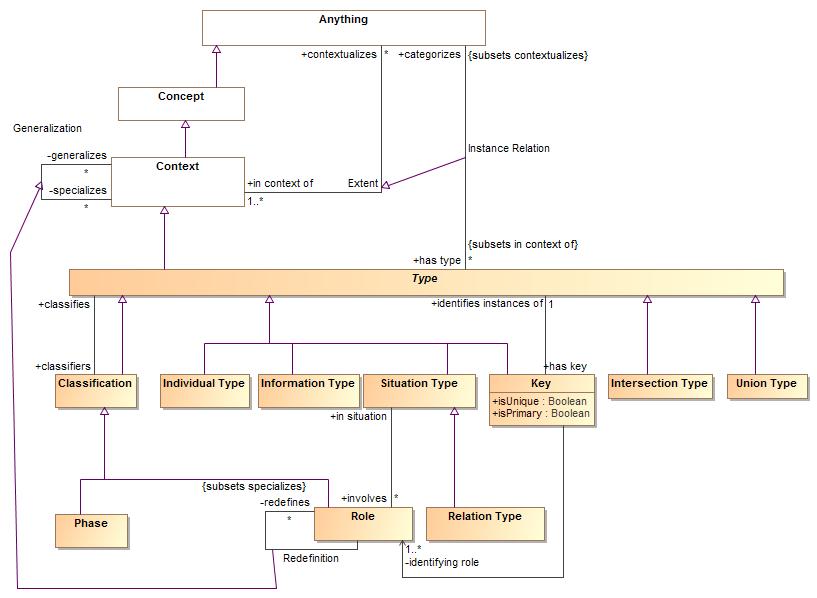
-2141820704.jpg creates:[Anything](#_a52cb0ff6e414b3170b58afe10b6afcb) [\*]

-2141820704.jpg invalidates:[Anything](#_a52cb0ff6e414b3170b58afe10b6afcb) [\*]

## SIMF Meta Model::Types

Type provide for ways to categorize anything based on what it is, the roles it plays or the phases it may be in.  
Something may be categorized by any number of types (multiple classification assumption).

### Diagram: Types



1. Types

### Class Classification

A classification is a "mix in" type that defines some aspect of something but does not represent the "fundamental type" of that thing, but some potentially transient role, phase or other way to classify. Something must have at least one type that is not a classification.

#### Direct Supertypes

[Type](#_dfe1514224ca21cedba7b2b29802db50)

**package** SIMF Meta Model::Types

**extends** [Type](#_dfe1514224ca21cedba7b2b29802db50)

#### Associations

-2141820704.jpg classifies:[Type](#_dfe1514224ca21cedba7b2b29802db50)

The type of individual that may be classified. Where more than one type is nominated, instances of any one of the types may be classified. If classifies is the null set, any type may be classified.

645683080.jpg :[Anything](#_a52cb0ff6e414b3170b58afe10b6afcb)

### Class Individual Type

Type of an identifiable individual with a lifetime

#### Direct Supertypes

[Type](#_dfe1514224ca21cedba7b2b29802db50)

**package** SIMF Meta Model::Types

**extends** [Type](#_dfe1514224ca21cedba7b2b29802db50)

### Class Information Type

A type representing values. E.g. numbers and strings. Values have no state or lifetime - they conceptually exist forever. The identity of a value is equivalent to the value.

#### Direct Supertypes

[Type](#_dfe1514224ca21cedba7b2b29802db50)

**package** SIMF Meta Model::Types

**extends** [Type](#_dfe1514224ca21cedba7b2b29802db50)

#### Associations

-2141820704.jpg :[Information](#_1d7f13297e70f68af8189a69a2e52b05) *Redefines*: categorizes:[Anything](#_a52cb0ff6e414b3170b58afe10b6afcb)

### Association Instance Relation

The relation between a type and the concepts that type categorizes, the instances

### Direct Supertypes

[Extent](#_52c887644007b8e51a1f6e976113707a)

**package** SIMF Meta Model::Types

**extends** [Extent](#_52c887644007b8e51a1f6e976113707a)

#### Association Ends

-2141820704.jpg categorizes : [Anything](#_a52cb0ff6e414b3170b58afe10b6afcb) *Redefines*: categorizes:[Anything](#_a52cb0ff6e414b3170b58afe10b6afcb)

The set of concepts of individuals described by a type.

-2141820704.jpg has type : [Type](#_dfe1514224ca21cedba7b2b29802db50) [\*] *Redefines*: categorizes:[Anything](#_a52cb0ff6e414b3170b58afe10b6afcb)

One of the types defining a concept

### Class Intersection Type

An intersection is a type that has an extent which is the complete intersection of the extents of all supertypes (or super context). Intersection is a stronger statement than a subtype as a subtype may not be a complete intersection. If an instance has all supertypes the intersection type will be inferred.  
For intersection,

#### Direct Supertypes

[Type](#_dfe1514224ca21cedba7b2b29802db50)

**package** SIMF Meta Model::Types

**extends** [Type](#_dfe1514224ca21cedba7b2b29802db50)

### Class Key

A key is an identifier composed from the state of an object, it identifies one or more instances of a type.

#### Direct Supertypes

[Type](#_dfe1514224ca21cedba7b2b29802db50)

**package** SIMF Meta Model::Types

**extends** [Type](#_dfe1514224ca21cedba7b2b29802db50)

#### Attributes

-1753260495.jpg isUnique : [Boolean](#_6119a00b0834641b9fe3f5ae9f58237f)

-1753260495.jpg isPrimary : [Boolean](#_6119a00b0834641b9fe3f5ae9f58237f)

#### Associations

-2141820704.jpg identifies instances of:[Type](#_dfe1514224ca21cedba7b2b29802db50) [1]

Instances of type that a key identifies

-304893399.jpg identifying role:[Role](#_a8049a836c9b9b5d6df4b578a5836756) [1..\*]

### Class Phase

A phase (or state) is a characteristic of something that exists for limited time(s). Something takes on or looses a phase as a result of some event. States can include "potential roles" such as "John is a teacher".

#### Direct Supertypes

[Classification](#_3b2e69eb6121d1e3a1180bbe8ee64013)

**package** SIMF Meta Model::Types

**extends** [Classification](#_3b2e69eb6121d1e3a1180bbe8ee64013)

#### Associations

-304893399.jpg state of:[Type](#_dfe1514224ca21cedba7b2b29802db50) [1..\*]

### Class Quantification

An assertion that defines a quantification (based on a subtype) over a context. A concept is considered the quantified variable.  
e.g. for all people p: People is the context and P is the quantified variable. In SIMF the quantified variable wold typically be named <quantifier> <type>. So the above quantified variable would be named "all people". The quantified variable will be asserted to have the quantified type.  
  
A Quantifier would be "mixed in" with a concrete subtype of Role.

#### Direct Supertypes

[Role](#_a8049a836c9b9b5d6df4b578a5836756)

**package** SIMF Meta Model::Types

**extends** [Role](#_a8049a836c9b9b5d6df4b578a5836756)

#### Associations

-304893399.jpg quantifies:[Type](#_dfe1514224ca21cedba7b2b29802db50) [1] *Redefines*: classifies:[Type](#_dfe1514224ca21cedba7b2b29802db50)

### Class Type

A categorization of concepts about individuals or types based on specific criteria

#### Direct Supertypes

[Block](#_2aa128e17c8314200d6ed8ee7ec053cc), [Context](#_66d62b068053cee3464e1e03e6035eed)

**package** SIMF Meta Model::Types

**extends** [Block](#_2aa128e17c8314200d6ed8ee7ec053cc), [Context](#_66d62b068053cee3464e1e03e6035eed)

#### Associations

-2141820704.jpg categorizes:[Anything](#_a52cb0ff6e414b3170b58afe10b6afcb) *Subsets*: contextualizes:[Anything](#_a52cb0ff6e414b3170b58afe10b6afcb)

The set of concepts of individuals described by a type.

-2141820704.jpg source for:[As-A Rule](#_fbe140d2e61a092c9ceffa2d5308ff5e)

-2141820704.jpg target for:[As-A Rule](#_fbe140d2e61a092c9ceffa2d5308ff5e)

-2141820704.jpg classifiers:[Classification](#_3b2e69eb6121d1e3a1180bbe8ee64013)

-2141820704.jpg has key:[Key](#_1b4bfdc2f6552991ff5cecb390e7e2af)

Key for a type

### Class Union Type

A Union is any type that has an extent which is the complete union of the extents of all context that specialize the Union.   
  
Note: The semantic different between a union and a supertype is not clear but it is included for coverage of FOL.

#### Direct Supertypes

[Type](#_dfe1514224ca21cedba7b2b29802db50)

**package** SIMF Meta Model::Types

**extends** [Type](#_dfe1514224ca21cedba7b2b29802db50)

Representation of SIMF concepts

## Foundational Assumptions

### Multiple representations of overlapping concepts

A base assumption of SIMF is that there are and will be multiple information resources that represent information about the same and overlapping things. These different representations may use different definitional languages (e.g. XML Schema, SQL, RDF, UML) and well as different domain terminology and human languages (e.g. Risk, Danger, Achtung, Risque). Our task is to federate information from multiple sources as well as express information from one source in another form.

### Models include data

Any structured set of information about anything is considered a “model”. This usage of the term model is consistent with the ontological usage but is an expansion of the term as typically used in modeling languages such as UML.

### Conceptual Models

For a SIMF implementation there will exist a conceptual model that contains a all the concepts to be interpreted and various mappings to physical or logical data representations. We will use the label CONCEPT and REPRESENTATION thorough to represent elements at both levels. It should also be noted that REPRESENTATION may or may not be fully physical, some REPRESENTATIONs may, in turn, have various syntaxes. For example, there are multiple syntaxes for RDF. We are treating all these layers of REPRESENTATION the same with the assumption there is an isomorphic transform between elements in the same REPRESENTATION family (e.g. the RDF logical model and RDF in Turtle Syntax). The conceptual model will be the “hub” or “pivot point” to integrate multiple REPRESENTATION models.

Conceptual models are intended to be models of the world – real world things. Facts about real world things are represented by data models in a data REPRESENTATION form but these facts also have a CONCEPT counterpart. While any implementation of a conceptual model must have some representation and/or behavior, the representation is abstract in terms of the specification (there is no normative syntax for data about instances of a conceptual model).

There are many possible implementation choices for CONCEPT models. Implementations MAY use the conceptual model only as a “virtual” rule set and have no physical representation of information about CONCEPT instances. Implementations MAY instantiate a physical representation corresponding directly to the conceptual model.

There is no assumption of any single or distinguished conceptual model. CONCEPT is a role for a model, some conceptual models may be derived from existing REPRESENTATIONs – particularly model based representations such as UML or OWL. Of course, some models will make better conceptual hubs than others. The Threat/RISK model is designed to be a conceptual model and will be refactored as required to represent the set of concepts represented by the REPRESENTATIONs it is intended to federate

### Identity and identifiers

There are different forms of identity and identifiers. There is the identity of the “real world” thing that some representation is stating facts about and there is the identity of the representation (the data or assertion - facts or statements). An identifier for a thing is called its “sign”. The sign of the real-world thing and the sign of data about that thing are not the same. Most REPRESENTATION models have representation identifiers but sometimes confuse these as signs for (identifiers for) the real world thing. Further, anything may have multiple identifiers – however many technologies only have one. “Keys” tend to be signs for real things, but this is not consistent.

### Facts

The term “fact” as used here is based on “fact modeling” but does not necessarily mean that something is “true”, a fact is some statement that asserts something - it may or may not be true and that truth may be contextual. Other terms could be “assertion” or “statement”.

### REPRESENTATIONs of a CONCEPT

Where there exists some CONCEPT of something there is a REPRESENTATION data element representing facts about this concept. The concept is also a representation is some respects, but this representation is playing the role of representing the real-world thing, not a data structure.

#### Relation

* Verb form: representation pattern🡪represents🡪CONCEPT pattern
* Noun form: Representation(representation pattern, CONCEPT pattern)

“Represents” provides the basis for the relation between a REPRESENTATION and a CONCEPT. Both ends or the represents relation require an identity for the “fact” or “facts” being represented. Thus “represents” depends on the ability to identify both the REPRESENTATION and CONCEPT fact. See 1.5.

The represents relation asserts that the REPRESENTATION element represents information about the CONCEPT. The information is not assumed to be complete or consistent. For various types of concepts this implies:

* **Individuals:** That the REPRESENTATION individual is a sign for facts (data) about the CONCEPT individual. Individuals are identifiable instances where the extent of the sign represents a single “thing” in the world. Types, context, relations and values are not individuals.
* **Types**: That the REPRESENTATION type describes the CONCEPT type. The extent of the REPRESENTATION type MAY be a subset of the extent of the CONCEPT type.
* **Context**: That the REPRESENTATION context contextualizes all or a subset of the CONCEPT context.
* **Relations**: That the relation semantics holds true for the bound elements of both the REPRESENTATION and CONCEPT

### REPRESENTATION identifiers

A REPRESENTATION resource contains a set of facts, such a resource may change over time so the set of facts may also change. At a particular time, as seen from a “transaction[[3]](#footnote-3)”, a resource has a fixed set of facts. Within a transaction a particular fact has an IDENTITY. The IDENTITY may be the same across multiple transactions from the same resource. The combination of the identity of the transaction and the identity of the fact within that transaction must be universally unique.

A fact in a REPRESENTATION resource has an IDENTIFIER. The form and structure of that identifier is technology dependent and not determined in this specification but it must be representable as a text string. Given a transaction/fact identifier combination any fact must be traceable to its origin. Where this pattern is replicated to the source of the fact, full provenance of the information is provided. A REPRESENTATION model may provide additional provenance information as part of a transaction and/or resource source reference but the semantics of that provenance is not assured unless the source is also a SIMF model.

Identical REPRESENTATION identifiers, within the same resource, are assumed to identify the same concept. Identical identifiers within different resources MAY be interpreted as identifying the same concept. Such identifiers are frequently called UUIDs or GUIDS.

#### Example physical identifiers

The STIX fragment:

<stix:TTP id="opensource:ttp-62ff7599-e1db-414f-b742-908def2ec219"

timestamp="2014-11-06T17:46:28.913306+00:00"

xsi:type="ttp:TTPType" version="1.1.1">

<ttp:Title>ZeuS</ttp:Title>

</stix:TTP>

Could use an XPATH expression for the ID, such as: ‘//\*[@ID=”opensource:ttp-62ff7599-e1db-414f-b742-908def2ec219]’ as the physical ID for the TTP fact (asserting the existence of the TTP) and "‘//\*[@ID=”opensource:ttp-62ff7599-e1db-414f-b742-908def2ec219”:]/Title[1]’ as the ID for the title fact.

Physical identifiers are not typically exposed to the user.

### CONCEPT representation identifiers

All conceptual facts, or “CONCEPTS” reside in some knowledge base, this knowledge base has a unique physical and logical identity. Within a knowledge base there exists a set of concepts with an identifier unique within that knowledge base. The combination of the knowledge base identifier and the concept identifier must be universally unique.

### Sources

All concepts come from some source outside of the conceptual system (even if this is a rule or inference operating in the conceptual system). Concepts have a SOURCE. All sources produce concept representations via some rule (inferences are considered rules). A REPRESENTATION of a concept (represents relation) is one such rule. Thus all concepts may be traced to their source as it existed at a particular time and was interpreted or computed based on a rule at that time.

#### Relation:

* Verb form: Fact🡪has source🡪Source Record
* Noun form: Source(Fact, Source Record)

Every conceptual fact has a source, that source identifies:

* The transaction. Transactions identify any physical resource as well as the time of the transaction.
* The rule that produced the fact
* The REPRESENTATION source identifier(s)

Since facts come from an identifiable source, that source and any internal structure MAY be used as a hint in producing the fact identifier. For resources with internal structure, this would typically be the Resource followed by a set of Packages followed by some name, ID or path of the REPRESENTATION fact.

The same source MAY result in multiple concepts. A representation concept is unique to a source but may be declared as equivalent to other representations concept.

### Ownership

Representation information, shall be “owned” directly or indirectly by a model, this ownership shall correspond with the structure of the source where such structure is important in interpreting the concept. The statement is owned, the ownership of the statement has no semantic interpretation with relation to what the concept means. Ownership of the concept may have a semantic interpretation as to the authority, scope or timeframe of the concept.

Ownership is one dimension of “context”, a concept may be contextualized by any number of context. Context may impact the interpretation of or trust in a concept. The context set of an owner in included in the context sets of concepts it owns.

### Lifetime and context of facts

Logically concepts are never “deleted”, they may become irrelevant or dated as would be determined by statements about the concept. Implementations MAY optimize a knowledge base by deleting facts that have been determined to be irrelevant for all purposes of the knowledge base (i.e. they would never impact the truth of any computation or query). Deletion of facts is not, in general, recommended (deletion is so 20th century).

Example: the fact “Thomas Woodrow Wilson is president of the United States” was true “March 4, 1913 through March 4, 1921”. We accept it will always be true that this fact was true in this timeframe, so there is no sense in “deleting” the fact. The timeframe fact “March 4, 1913 through March 4, 1921” is a fact about the fact: “Thomas Woodrow Wilson is president of the United States”. Another fact about this fact is that its source resource was <http://www.whitehouse.gov> on January 14th, 2015 at 9:05PM-EST.

Note that the above implies that there may be more than one FACT that says exactly the same thing, but they may come from different sources or be stated in different context or timeframes. Or, different sources or context may assert conflicting facts about the same thing. Trust is a function of the interpretation of a set of facts, including their source, timeframe and context. Interpretation is time dependent (i.e. transactional), with respect to a given set of context operating on a given knowledge base at a given time. (Note that a knowledge base may include all or part of the contents of other repositories and that repositories may be physical or virtual). A knowledge base could be as small as one or more files on your tablet or as large as Google or Wikipedia.

# Graphical Notation

The following provides some initial experiments for graphical notation, this notation may change. As of this revision most work as been done in either UML or in a “fact modeling” notation. Additional considerations for fact modeling may be found at: <https://github.com/ModelDriven/SIMF/tree/master/SupportingDocuments/PNA-Group>

## Context



## Types



## Relation Types



## Directed Relation Types



## Instances



## Quantifications



## LIM Structure



## Important Relations



# Textual Notation

# Standard MBR Models

Standards MBR models for existing languages are under development.

## OWL Web Ontology Language

OWL semantics are currently defined in terms of the UML profile, this will be expanded to include the SIMF meta model.

# Annex A Examples (informative)

**The SIMF submission calls for worked examples. The worked examples for the SIMF initial submission is the initial submission for an operational threat and risk model, document: sysa/2015-05-01**

# Annex B Mapping to OWL 2 (normative)

Examples are given below that show the transformation of UML modeled in SIMF to an exported OWL 2 ontology. The OWL ontologies are presented in OWL Functional Syntax.

The first diagram below, for a simple UML class, shows the ontology is transformed as the package containing the UML class. Subsequent diagrams do not show the package in the diagram for the sake of brevity.

## Class



Ontology(<http://nomagic.com/ontology/example-case/case-01>

Declaration(

Class(:Person)

)

AnnotationAssertion(rdfs:label :Person "Person"@en)

)

## Class Generalization



Ontology(<http://nomagic.com/ontology/example-case/case-04>

Declaration(

Class(:FutsalTeam)

)

Declaration(

Class(:SoccerTeam)

)

AnnotationAssertion(rdfs:label :FutsalTeam "Futsal Team"@en)

SubClassOf(:FutsalTeam :SoccerTeam)

AnnotationAssertion(rdfs:label :SoccerTeam "Soccer Team"@en)

)

## Class with Datatype Property



Ontology(<http://nomagic.com/ontology/example-case/case-02>

Import(<http://www.omg.org/spec/PrimitiveTypes/20100901>)

Declaration(

Class(:Person)

)

Declaration(

DataProperty(:hasName)

)

Declaration(

AnnotationProperty(<http://purl.org/dc/terms/description>)

)

Declaration(

Datatype(xsd:string)

)

AnnotationAssertion(rdfs:label :Person "Person"@en)

SubClassOf(

:Person

ObjectIntersectionOf(

DataMaxCardinality(1 :hasName xsd:string)

DataMinCardinality(1 :hasName xsd:string)

)

)

AnnotationAssertion(rdfs:label :hasName "has name"@en)

DataPropertyDomain(:hasName :Person)

DataPropertyRange(:hasName xsd:string)

AnnotationAssertion(<http://purl.org/dc/terms/description> <http://www.omg.org/spec/PrimitiveTypes/20100901#String> "An instance of String defines a piece of text. The semantics of the string itself depends on its purpose, it can be a comment, computational language expression, OCL expression, etc. It is used for String attributes and String expressions in the metamodel."@en)

)

## Class with Self-Referential Object Property



Ontology(<http://nomagic.com/ontology/example-case/case-02a>

Declaration(

Class(:Person)

)

Declaration(

ObjectProperty(:isRelatedTo)

)

AnnotationAssertion(rdfs:label :Person "Person"@en)

SubClassOf(

:Person

ObjectIntersectionOf(

ObjectMinCardinality(1 :isRelatedTo :Person)

)

)

AnnotationAssertion(rdfs:label :isRelatedTo "is related to"@en)

ObjectPropertyDomain(:isRelatedTo :Person)

ObjectPropertyRange(:isRelatedTo :Person)

)

## Class with Object Property



Ontology(<http://nomagic.com/ontology/example-case/case-03>

Declaration(

Class(:SoccerPlayer)

)

Declaration(

Class(:SoccerTeam)

)

Declaration(

ObjectProperty(:consistsOf)

)

AnnotationAssertion(rdfs:label :SoccerPlayer "Soccer Player"@en)

AnnotationAssertion(rdfs:label :SoccerTeam "Soccer Team"@en)

SubClassOf(

:SoccerTeam

ObjectIntersectionOf(

ObjectMaxCardinality(11 :consistsOf :SoccerPlayer) ObjectMinCardinality(5 :consistsOf :SoccerPlayer)

)

)

AnnotationAssertion(rdfs:label :consistsOf "consists of"@en)

ObjectPropertyDomain(:consistsOf :SoccerTeam)

ObjectPropertyRange(:consistsOf :SoccerPlayer)

)

## Property Holder with Datatype Property



Ontology(<http://nomagic.com/ontology/example-case/case-03a>

Import(<http://www.omg.org/spec/PrimitiveTypes/20100901>)

Declaration(

DataProperty(:hasName)

)

Declaration(

AnnotationProperty(<http://purl.org/dc/terms/description>)

)

Declaration(

Datatype(xsd:string)

)

SubClassOf(

owl:Thing

ObjectIntersectionOf(

DataMaxCardinality(3 :hasName xsd:string)

DataMinCardinality(2 :hasName xsd:string)

)

)

AnnotationAssertion(rdfs:label :hasName "has name"@en)

DataPropertyRange(:hasName xsd:string)

AnnotationAssertion(<http://purl.org/dc/terms/description>

<http://www.omg.org/spec/PrimitiveTypes/20100901#String> "An instance of String defines a piece of text. The semantics of the string itself depends on its purpose, it can be a comment, computational language expression, OCL expression, etc. It is used for String attributes and String expressions in the metamodel."@en)

)

## Property Holder with Self-Referential Object Property



Ontology(<http://nomagic.com/ontology/example-case/case-03b>

Declaration(

ObjectProperty(:isRelatedTo)

)

SubClassOf(

owl:Thing

ObjectIntersectionOf(

ObjectMinCardinality(1 :isRelatedTo)

)

)

AnnotationAssertion(rdfs:label :isRelatedTo "is related to"@en)

)

## Property Holder with Object Property



Ontology(<http://nomagic.com/ontology/example-case/case-03c>

Declaration(

Class(:Liquid)

)

Declaration(

ObjectProperty(:isDissolvedBy)

)

AnnotationAssertion(rdfs:label :Liquid "Liquid"@en)

SubClassOf(

owl:Thing

ObjectIntersectionOf(

ObjectMinCardinality(1 :isDissolvedBy :Liquid)

)

)

AnnotationAssertion(rdfs:label :isDissolvedBy "is dissolved by"@en)

ObjectPropertyRange(:isDissolvedBy :Liquid)

)

## Class with Object Property without Range



Ontology(<http://nomagic.com/ontology/example-case/case-03d>

Declaration(

Class(:Receptacle)

)

Declaration(

ObjectProperty(:holds)

)

AnnotationAssertion(rdfs:label :Receptacle "Receptacle"@en)

AnnotationAssertion(rdfs:label :holds "holds"@en)

ObjectPropertyDomain(:holds :Receptacle)

)

## Class with Subproperty



Ontology(<http://nomagic.com/ontology/example-case/case-05>

Declaration(

Class(:FutsalPlayer)

)

Declaration(

Class(:FutsalTeam)

)

Declaration(

Class(:SoccerPlayer)

)

Declaration(

Class(:SoccerTeam)

)

Declaration(

ObjectProperty(:composedOf)

)

Declaration(

ObjectProperty(:consistsOf)

)

AnnotationAssertion(rdfs:label :FutsalPlayer "Futsal Player"@en)

SubClassOf(:FutsalPlayer :SoccerPlayer)

AnnotationAssertion(rdfs:label :FutsalTeam "Futsal Team"@en)

SubClassOf(:FutsalTeam :SoccerTeam)

SubClassOf(

:FutsalTeam

ObjectIntersectionOf(

ObjectMaxCardinality(5 :composedOf :FutsalPlayer) ObjectMinCardinality(5 :composedOf :FutsalPlayer)

)

)

AnnotationAssertion(rdfs:label :SoccerPlayer "Soccer Player"@en)

AnnotationAssertion(rdfs:label :SoccerTeam "Soccer Team"@en)

SubClassOf(

:SoccerTeam

ObjectIntersectionOf(

ObjectMaxCardinality(11 :consistsOf :SoccerPlayer) ObjectMinCardinality(5 :consistsOf :SoccerPlayer)

)

)

AnnotationAssertion(rdfs:label :composedOf "composed of"@en)

SubObjectPropertyOf(:composedOf :consistsOf)

ObjectPropertyDomain(:composedOf :FutsalTeam)

ObjectPropertyRange(:composedOf :FutsalPlayer)

AnnotationAssertion(rdfs:label :consistsOf "consists of"@en)

ObjectPropertyDomain(:consistsOf :SoccerTeam)

ObjectPropertyRange(:consistsOf :SoccerPlayer)

)

## Class with Universal Quantification Constraint on Property I



Ontology(<http://nomagic.com/ontology/example-case/case-06>

Declaration(

Class(:Dog)

)

Declaration(

Class(:DogOwner)

)

Declaration(

Class(:Person)

)

Declaration(

Class(:Pet)

)

Declaration(

ObjectProperty(:has)

)

AnnotationAssertion(rdfs:label :Dog "Dog"@en)

SubClassOf(:Dog :Pet)

AnnotationAssertion(rdfs:label :DogOwner "Dog Owner"@en)

SubClassOf(:DogOwner :Person)

SubClassOf(

:DogOwner

ObjectIntersectionOf(

ObjectMinCardinality(1 :has :Dog)

ObjectAllValuesFrom(:has :Dog)

)

)

AnnotationAssertion(rdfs:label :Person "Person"@en)

AnnotationAssertion(rdfs:label :Pet "Pet"@en)

AnnotationAssertion(rdfs:label :has "has"@en)

ObjectPropertyDomain(:has :Person)

ObjectPropertyRange(:has :Pet)

)

## Class with Universal Quantification Constraint on Property II

This example differs from the previous example primarily in that the superclasses “Person” and “Pet” are from a different package than their subclasses “Dog Lover” and “Dog,” respectively. This is reflected in the OWL ontology by the import of this namespace.

The superclasses “Person” and “Pet”, defined in the package “Case 06”, are a different color and a lighter shade than the classes defined in the package “Case 07”. This is to distinguish them from the classes defined in this package. MagicDraw’s AutoStyler plugin can automatically set the properties for classes and other UML elements “defined elsewhere,” that is in a package not containing the defining diagram for the UML element (See section 2.2, Automatic Styling of Concept Models.).



Ontology(<http://nomagic.com/ontology/example-case/case-07>

Import(<http://nomagic.com/ontology/example-case/case-06>)

Declaration(

Class(<http://nomagic.com/ontology/example-case/case-06#Person>)

)

Declaration(

Class(<http://nomagic.com/ontology/example-case/case-06#Pet>)

)

Declaration(

Class(:Dog)

)

Declaration(

Class(:DogLover)

)

Declaration(

ObjectProperty(<http://nomagic.com/ontology/example-case/case-06#has>)

)

AnnotationAssertion(rdfs:label :Dog "Dog"@en)

SubClassOf(:Dog <http://nomagic.com/ontology/example-case/case-06#Pet>)

AnnotationAssertion(rdfs:label :DogLover "Dog Lover"@en)

SubClassOf(:DogLover <http://nomagic.com/ontology/example-case/case-06#Person>)

SubClassOf(

:DogLover ObjectIntersectionOf(

ObjectAllValuesFrom(<http://nomagic.com/ontology/example-case/case-06#has> :Dog)

)

)

)

## Class with Existential Quantification Constraint on Property



Ontology(<http://nomagic.com/ontology/example-case/case-08>

Import(<http://nomagic.com/ontology/example-case/case-06>)

Declaration(

Class(<http://nomagic.com/ontology/example-case/case-06#Person>)

)

Declaration(

Class(<http://nomagic.com/ontology/example-case/case-06#Pet>)

)

Declaration(

Class(:Dog)

)

Declaration(

Class(:DogLover)

)

Declaration(

ObjectProperty(<http://nomagic.com/ontology/example-case/case-06#has>)

)

AnnotationAssertion(rdfs:label :Dog "Dog"@en)

SubClassOf(:Dog <http://nomagic.com/ontology/example-case/case-06#Pet>)

AnnotationAssertion(rdfs:label :DogLover "Dog Lover"@en)

SubClassOf(:DogLover <http://nomagic.com/ontology/example-case/case-06#Person>)

SubClassOf(

:DogLover

ObjectIntersectionOf(

ObjectMinCardinality(1 <http://nomagic.com/ontology/example-case/case-06#has> :Dog) ObjectSomeValuesFrom(<http://nomagic.com/ontology/example-case/case-06#has> :Dog)

)

)

)

## Property Holder with Self-Referential Subproperty



Ontology(<http://nomagic.com/ontology/example-case/case-11>

Declaration(

ObjectProperty(:contains)

)

Declaration(

ObjectProperty(:holds)

)

AnnotationAssertion(rdfs:label :contains "contains"@en)

SubObjectPropertyOf(:contains :holds)

AnnotationAssertion(rdfs:label :holds "holds"@en)

)

## Property Holder with Subproperty



Ontology(<http://nomagic.com/ontology/example-case/case-18>

Declaration(

Class(:Acid)

)

Declaration(

Class(:Liquid)

)

Declaration(

ObjectProperty(:isCorrodedBy)

)

Declaration(

ObjectProperty(:isDissolvedBy)

)

AnnotationAssertion(rdfs:label :Acid "Acid"@en)

SubClassOf(:Acid :Liquid)

AnnotationAssertion(rdfs:label :Liquid "Liquid"@en)

SubClassOf(

owl:Thing

ObjectIntersectionOf(

ObjectMinCardinality(1 :isCorrodedBy :Acid)

)

)

SubClassOf(

owl:Thing

ObjectIntersectionOf(

ObjectMinCardinality(1 :isDissolvedBy :Liquid)

)

)

AnnotationAssertion(rdfs:label :isCorrodedBy "is corroded by"@en)

SubObjectPropertyOf(:isCorrodedBy :isDissolvedBy)

ObjectPropertyRange(:isCorrodedBy :Acid)

AnnotationAssertion(rdfs:label :isDissolvedBy "is dissolved by"@en)

ObjectPropertyRange(:isDissolvedBy :Liquid)

)

## Class with Subproperty without a Range



Ontology(<http://nomagic.com/ontology/example-case/case-16>

Declaration(

Class(:Game)

)

Declaration(

Class(:SoccerMatch)

)

Declaration(

ObjectProperty(:isACompetitionBetween)

)

Declaration(

ObjectProperty(:isPlayedBetween)

)

AnnotationAssertion(rdfs:label :Game "Game"@en)

SubClassOf(

:Game

ObjectIntersectionOf(

ObjectMinCardinality(2 :isPlayedBetween)

)

)

AnnotationAssertion(rdfs:label :SoccerMatch "Soccer Match"@en)

SubClassOf(:SoccerMatch :Game)

SubClassOf(

:SoccerMatch

ObjectIntersectionOf(

ObjectMaxCardinality(2 :isACompetitionBetween) ObjectMinCardinality(2 :isACompetitionBetween)

)

)

AnnotationAssertion(rdfs:label :isACompetitionBetween "is a competition between"@en)

SubObjectPropertyOf(:isACompetitionBetween :isPlayedBetween)

ObjectPropertyDomain(:isACompetitionBetween :SoccerMatch)

AnnotationAssertion(rdfs:label :isPlayedBetween "is played between"@en)

ObjectPropertyDomain(:isPlayedBetween :Game)

)

## Class with Necessary and Sufficient Property



Ontology(<http://nomagic.com/ontology/example-case/case-20>

Declaration(

Class(:CarManufacturer)

)

Declaration(

Class(:Manufacturer)

)

Declaration(

Class(:SteeringWheelManufacturer)

)

Declaration(

Class(:WindshieldManufacturer)

)

Declaration(

ObjectProperty(:hasContractWith)

)

AnnotationAssertion(rdfs:label :CarManufacturer "Car Manufacturer"@en)

EquivalentClasses(

:CarManufacturer

ObjectIntersectionOf(

ObjectMinCardinality(1 :hasContractWith :SteeringWheelManufacturer) ObjectSomeValuesFrom(:hasContractWith :SteeringWheelManufacturer)

)

)

EquivalentClasses(

:CarManufacturer

ObjectIntersectionOf(

ObjectMinCardinality(1 :hasContractWith :WindshieldManufacturer)

ObjectSomeValuesFrom(:hasContractWith :WindshieldManufacturer)

)

)

SubClassOf(:CarManufacturer :Manufacturer)

AnnotationAssertion(rdfs:label :Manufacturer "Manufacturer"@en)

AnnotationAssertion(rdfs:label :SteeringWheelManufacturer "Steering Wheel Manufacturer"@en)

SubClassOf(:SteeringWheelManufacturer :Manufacturer)

AnnotationAssertion(rdfs:label :WindshieldManufacturer "Windshield Manufacturer"@en)

SubClassOf(:WindshieldManufacturer :Manufacturer)

AnnotationAssertion(rdfs:label :hasContractWith "has contract with"@en)

ObjectPropertyDomain(:hasContractWith :Manufacturer)

ObjectPropertyRange(:hasContractWith :Manufacturer)

)

## Class With Property Having Unspecified Multiplicity

UML allows the cardinality of a property to be left unspecified. The concept modeling profile interprets unspecified cardinalities as being zero to many (“0..\*”).



Ontology(<http://nomagic.com/ontology/example-case/case-21>

Declaration(

Class(:SoccerPlayer)

)

Declaration(

Class(:SoccerTeam)

)

Declaration(ObjectProperty(:consistsOf))

AnnotationAssertion(rdfs:label :SoccerPlayer "Soccer Player"@en)

AnnotationAssertion(rdfs:label :SoccerTeam "Soccer Team"@en)

AnnotationAssertion(rdfs:label :consistsOf "consists of"@en)

ObjectPropertyDomain(:consistsOf :SoccerTeam)

ObjectPropertyRange(:consistsOf :SoccerPlayer)

)

# Annex C UML Profile (normative)

## Conceptual Modeling Interpretation of UML

In order to improve UML’s suitability for modeling real-world concepts, this profile interprets standard UML to allow subproperties, existential quantification constraints, and universal quantification constraints. In addition to those interpretations, this profile uses a small UML profile to achieve things that cannot be interpreted from standard UML, such as global properties and necessary and sufficient properties. The following subsections describe how this profile achieves these capabilities.

### Property Ownership

The concept modeling profile of UML interprets the owner of a property as a context in which that property must conform to certain constraints. These constraints can include multiplicity, which includes a minimum cardinality and a maximum cardinality, a type for the property, existential quantification, and universal quantification, which is the default. When an instance is a member of an owning class, all of that class’ constraints must be met.

UML allows the cardinality of a property to be left unspecified. The concept modeling profile interprets unspecified cardinalities as being zero to many (“0..\*”).

### Global Properties

Global properties are property declarations that can be used by any instance. Normally, a UML property cannot be defined outside of a classifier, so a global property declaration is represented as a UML property owned by a class that is stereotyped as a **«**PropertyHolder**».** The concept of a property holder was introduced in the NIEM-UML standard for a similar purpose. In the concept modeling profile, every property holder is equivalent to one topmost class (⊤) of which all other classes are subclasses. Thus, a property of a property holder is inherited by all subclasses. In addition, while the name of a property holder is irrelevant, consistently naming property holders “Thing”, “Concept”, or “Entity” in all concept models avoids any confusion with normal classes.

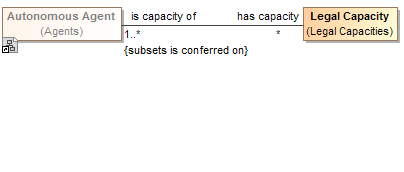


## Subproperty

A subproperty is a more specific kind of property than some other property, and a super property is a more general kind of property than some other property. For example, “has father” is a more specific property than “has parent”, and “has parent” is a more general property than either “has mother” or “has father”. In the concept modeling interpretation of UML, subsetting a property creates a subproperty when the subsetting property has a different name than the subsetted property. (See section XXX, Existential Quantification Constraint, for when the name is the same or is omitted.) UML provides a {subsets} constraint that asserts that the values within a subsetting property are also in the set of values within a subsetted property. The concept modeling profile interprets a subsetting property having a different name as a subproperty.

The diagram below shows that the property “is capacity of” (owned by the class “Legal Capacity) is a subset of the global property “is conferred on” (from the property holder “Thing”).

**Note that to create a subproperty, the subsetting property must have a *different* name than the property it subsets.**



## Existential Quantification Constraint

A property is not limited to a minimum and a maximum cardinality (known as multiplicity) for just one type. A property can have a multiplicity for a superclass, while at the same time having a more specific multiplicity for one or more subclasses of that superclass. This is known as an *existential quantification* (∃) or qualified constraint. This type of constraint is an assertion that, among other possible values, the number of values of one of these subclasses is between some minimum and maximum cardinality. Adding an existential quantification constraint does not define a new property, rather it constrains an existing property. Note that an existential quantification constraint must have a minimum cardinality of at least one, in order to meet the definition of “existential” for the constraint. In the concept modeling interpretation of UML, subsetting a property without giving the new property a different name (or leaving off the new property name altogether) creates an existential quantification constraint. As {subsets} with an omitted name is not well defined in UML, in the concept modeling profile it is used to state that a subset of values must meet the stated cardinality and type constraints of the subsetting property. It does not create a new property, although it does create a context in which this constraint holds: the owning class and its subclasses.

The diagram below shows an existential quantification constraint on the global property “is conferred by” (from the property holder “Thing). The multiplicity is such that at least one of the instances of the property constraint must be one of the types in the union.

**Note that the property adding the constraint is unnamed. This is equivalent, in this case, to naming this property the same as the property being constrained (“is conferred by” from the property holder “Thing”).**

## Universal Quantification Constraint

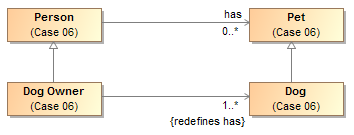
Sometimes it is necessary, in the context of some class, to constrain *all* the values of a property to a particular type. This is known as a *universal* quantification or *for-all* constraint (∀). This kind of constraint is an assertion that only values of the specified type are valid, and the number of values must be between some minimum and maximum cardinality. In the concept modeling interpretation of UML, introducing a new property or redefining an existing property creates a universal quantification constraint in the context of the owning class. This interpretation is based on {redefines} in UML, which allows adding more specific constraints to an existing property, without defining a new property.

The diagram below shows the introduction of a new property “consists of”, defining a universal quantification constraint on the property. The constraint states that, in the context of Soccer Team and any of its subclasses, all values of this property must be of the type “Soccer Player” and that there must be between 5 and 11 values of this property.



The diagram below shows a universal quantification constraint on the property “has” (owned by the class “Person”). It states that, in the context of “Dog Owner”, all values of the property “has” must be of type “Dog” and at least one value of this property must exist.

**Note that a property that is redefined must have the *same* name as the redefined property. In this case, leaving the redefined property unnamed is equivalent to naming the property the same as the property being redefined (“has” from the class “Person”).**



## Necessary and Sufficient Property

A property's multiplicity or type is declared in the context of an owning class or a property holder. These declarations are always *necessary* conditions for an instance to be a member of the owning class, or, in the case of a property holder, for an instance to be valid at all.

Another kind of condition is known as *necessary and sufficient*. A class with at least one necessary and sufficient condition is known as a *defined* class, which means the differentiating characteristics of the class that make it distinguishable from its parent and sibling classes are defined.

In the concept modeling interpretation of UML, a property that has the {sufficient} constraint applied to it indicates that when an instance satisfies the multiplicity and type constraints for the property’s values, not only is a *necessary* condition for being an instance of the class met, a *sufficient* condition is also met. This necessary and sufficient condition allows an inferencing engine to classify that instance as a member of the class that owns the property. Once an instance is classified automatically, the conditions on any other properties that have the {sufficient} constraint, including those inherited from superclasses, merely become *necessary* conditions the instance must also meet to be a *valid* member of the owning class. In other words, satisfying any one {sufficient} constraint is enough for an inferencing engine to classify an instance.

The diagram below shows that when an instance with the property “has contract with” satisfies specific multiplicity (“1..\*”) and type constraints (of type ‘Steering Wheel Manufacturer” or “Windshield Manufacturer”) for the property’s values, the instance meets a necessary and sufficient condition to be a member of the class “Car Manufacturer”. Therefore, an inferencing engine would classify this instance as an instance of the class “Car Manufacturer”. Note that, as discussed above, an instance meeting any one of these necessary and sufficient conditions is enough to classify the instance, regardless of conditions on the values of any other properties with the {sufficient} constraint owned by the class “Car Manufacturer”. The conditions on the values of these properties become necessary conditions on an instance to be a valid member of class “Car Manufacturer.” Note also that an instance meeting any one of these necessary and sufficient conditions is enough to distinguish instances of the class “Car Manufacturer’ from its parent class “Manufacturer.”



## Conceptual Modeling UML Profile

The following documents the UML stereotypes defined for SIMF models, as such not all of these stereotypes are necessarily used within the threat/risk mode, but all are available for use. Each stereotype has a mapping to the SIMF meta model and logical grounding.

### UML Subset

The SIMF conceptual modeling profile uses a subset of UML, only the following concepts have an interpretation in SIMF:

* Classes {Map to Entities}
* Data types and primitive types {Map to Values}
* Associations, associations ends {Map to Relation types}
* Properties for values {Map to Simple property relations}
* Property defaults {Map to a “Most” quatification}
* Association classes (Map to relations - however, all associations and properties are considered “first class”)
* Subsets and redefines of association ends {Map to role generalization}
* Cardinalities {Map to Constraint}
* Packages & Package URI {Map to Lexical and logical context}
* Realization {Map to Representation realizes concept}
* Structured classifiers {Map to templates}

SIMF does not preclude other UML constructs, it just does not interpret them. Extended SIMF profiles MAY interpret additional UML elements.

### Stereotype: Classification [Classifier]

A classification describes a particular type definition to which an individual conforms at a particular time and in a particular set of context. Classifications may be added to or removed from an individual over time and in different context. All subtypes of a classification are classifications.

* Known subtypes: Role, Phase

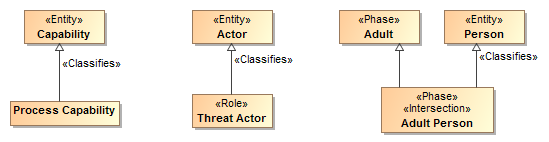
For examples see “Classifies”.

See also:

### Stereotype: Classifies [Generalization, Realization]

A classification defined by a <<Classifies>> generalization or realization is an arbitrary classification of an entity beyond any fundamental entity type. <<Classifies>> implies that the subtype is a <<Classification>>. An instance must be typed by the classifies supertype for it to also be classified as the classifies subtype. A classification may be contextual, such as within a relation, situation and/or timeframe.

Examples



* Process capability is a classification that classifies Capabilities.
* Threat Actor is a role that classifies Actor
* An adult person is a phase of a person that is also an adult.

### Stereotype: Conceptual Model [Package]

A conceptual model represents signs, assertions and classifications for “real world” objects. A conceptual model models the world (or a conceptualization of it), not data or computation. Conceptual models are the foundation of this specification.

##### URI:String (From UML)

A unique logical identifier for the model.

Examples



The package “Threat Risk Concepts” is marked as a conceptual model. The URI for this model is: ID://models.modeldriven.org/threat-risk.

Note: As a convention we are using “ID” for “Identity” to distinguish such URIs from internet addresses but otherwise use the HTTP naming convention. The relationship between an ID and any physical resource is outside of this specification.

### Stereotype: Disjoint with [Dependency]

Disjoint specifies that the extent of real-world individuals for which the dependent elements are a sign, classification for or context of have no elements in common.

Note: We do NOT assume that anything has only one type or instance representation unless stated otherwise.

Examples



* No people are also planets
* Fido (A dog) is not Mars (APlanet). {the set of things that are Fido is disjoint with the set of things that are Mars}.
* Fido (A dog) is not a person (Even if Fido’s owner things he is). {the set of things that are Fido is disjoint with the set of things that are a Person}.

Disjoint may also be specified using generalization sets.

### Stereotype: Entity [Class]

An entity is a foundational type of a class, what it essentially “is” as opposed to the roles it may play, phases it may be in or other arbitrary classifications. It is not expected that the entity type of an individual would change (TBD: Should it be illegal for a fundamental type of an instance to change?).

Examples

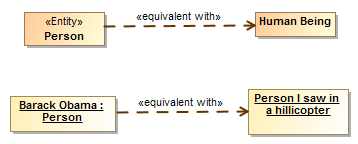


Person and Place are fundimental types.

### Stereotype: Equivalent with [Dependency]

Equivalent is an assertion that two elements in a model or data resource represent the same thing or set of things. Unlike ontological languages it is not assumed that the two elements are consistent as statements from different context may or may not agree.

Examples



* Person and Human are equivelent types, they have the same extent.
* Barack Obama is the person I saw in a helicopter last night.

See also

* ODM/OWL: SameAs, EquivalentClass, EquivalentProperty

### Stereotype: Intersection [Classifier]

An intersection is a class that has an extent which is the complete intersection of the extents of all supertypes. Intersection is a stronger statement than a subtype as a subtype may not be a complete intersection. If an instance has all supertypes the intersection type will be inferred.

For intersection, SIMF considers generalization and realization equivalent. This is due to ownership and legacy considerations in UML. Generalization is the preferred representation.

Examples



All dogs that are pets are pet dogs.

#### See also

* ODM/OWL: intersectionOf

### Stereotype: Is in context [Dependency]

<<Is in context>> is an assertion that the source of the relationship is in the context of the target of the relationship. All assertions and rules defined in the target context apply to the source and everything in the context of the source (in context is transitive). Packages, classes, situations and instances are typical context. Note that <<Is in context>> is the default interpretation of a dependency, if no stereotype is specified it will be interpreted as <<Is in context>>.

#### Examples



* GPS coordinates apply within the context of earth.
* The Radar detector law applies in the context of the state of Virginia and is in the context of rules of the road.

### Stereotype: Source Reference [Classifier]

<<Reference>> provides traceability to the source of a piece of information in some information resource to facilitate provenance. Reference is a statement about the model data and has no semantic implication. Source reference may impact the trust in a statement but the evaluation of trust is outside of this specification. Source is typically populated by the SIMF tooling.

#### Tag definitions

* Kind:String Specifies the kind of resource being referenced. (E.G. “XML”). The values of “Kind” are implementation specific. (Note: look for standard to site).
* Resource:URL Specifies the location of the resource
* Path:String Specifies the location of the information in the source. The form of expression of the path is dependent on the Kind.. Kinds and path languages are not defined in this specification.
* When: DateTime Specifies when the information was first asserted in the resource.
* TransactionID:String Transaction ID for the information.

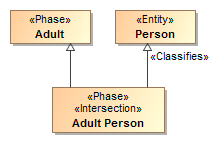
### Stereotype: Phase [Class]

Phase classifies some portion of an entities lifetime. Also known as a “State”.

Supertypes

* Classification

Examples



Adult is a phase (what it classifies not stated here). An Adult Person represents a phase in the life of a person.

### Stereotype: Quantity Kind [Classifier]

<<Quantity Kind>> is an aspect common to mutually comparable quantities represented by one or more units. Units with a common quantity kind may be algorithmically converted to any other unit of that quantity kind. e.g. temperature.[ JCGM 200:2008].

Typically Units <<represents>> quantity kinds.

Examples



See also

* JCGM 200:2008

### Stereotype: Represents [Realization, Dependency, Generalization]

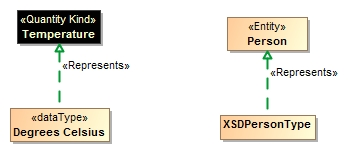
<<Represents>> is an assertion that the source type or feature provides a more concrete way to represent the target type or feature. Represents may be used within conceptual models or from a physical model to a conceptual model.

* A representation that is a dependency or realization makes no assumption that the types are substitutable
* A representation that is a generalization is substitutable for what it represents

Tag definitions

* Condition:Value specification – expression that must evaluate to true for the rule to be valid
* In context:Context – context that scopes the applicability of the representation (may also be specified as a dependency FROM the represents rule).
* type:Classifier – Potentially more specific type that a property is mapped to

Examples



* Degrees Celsius is a way to represent Temperature. Note that there could still be different ways to represent Degrees Celsius (e.g. As a real or integer)
* XSDPersonType (From a physical model) is a way to represent data about a Person (From a conceptual model)
* To-do: Rule and context examples

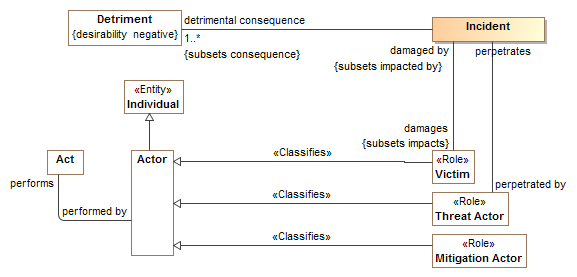
### Stereotype: Role [Class]

A <<Role>> is a classification of an entity based on that entities behavior, participation in a situation or capabilities. A <<Role>> <<Classifies>> the types that may play that role.

Supertypes

* Classification

Examples



Victims, Threat Actors and Mitigation Actors are roles of actors which may be related to specific incidents.

### Stereotype: Union [Classifier]

A <<Union>> is any context (Package or Classifier) that has an extent which is the complete union of the extents of all context that realize or specialize the Union.

For union, SIMF considers generalization and realization equivalent. This is due to ownership and legacy consideration in UML.

Example



A Marine Mammal is either a Dolphin, Orka or Whale.

See Also

* OWL/ODM: UnionOf

## Mapping Profile

### Stereotype: Facade [Classifier]

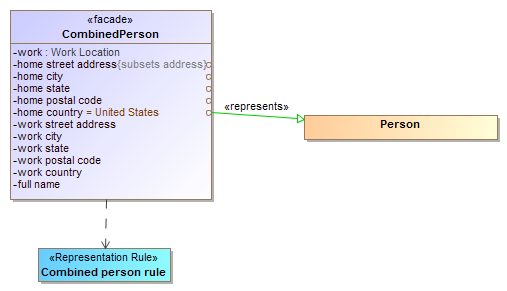
Classifies an element as being a view of (facade of) one or more other elements. Facades usually define additional properties that match some external view of a conceptual model element. Façade’s will use one of two methods to relate the façade properties to the conceptual Model:

* A dependency to a <<Representation Rule>>
* Applying the <<computation>> stereotype and Subclassing “Representation Computation”

A Façade should use a generalization with <<represents>> to specify the class for which it is a façade.

Note that a façade may be used as a pure specification or may actually be instantiated, depending on implementation styles.

Examples



In the above example CombinedPerson is a façade for Person and adds “flattened” properties for home and work address as well as a textual full name.

### Stereotype: Filter [Connector, Property]

A filter categorizes a connector or property within a representation rule structure of a facade as having a condition (A boolean expression restricting the population of elements) and/or a default (an expression creating a new object where one does not exist).

TBD: Expression to use ALF syntax.

### Stereotype: Match [Connector, Property]

Match specifies a an element in a representation rule structure that must match the conceptual model

### Stereotype: Mapping [Package]

A <<Mapping>> package represents signs and classification for mappings between different models.

##### URI:String (From UML)

A unique logical identifier for the model.

Examples



### Stereotype: Representation [Package]

A <<Representation>> package represents signs and classification for a viewpoint, data and/or technology specific representation of concepts. If a model is not otherwise tagged, it is assumed to be Physical. Note that Physical models may also be logical or platform independent models.

### Stereotype: Represents [Realization, Dependency]

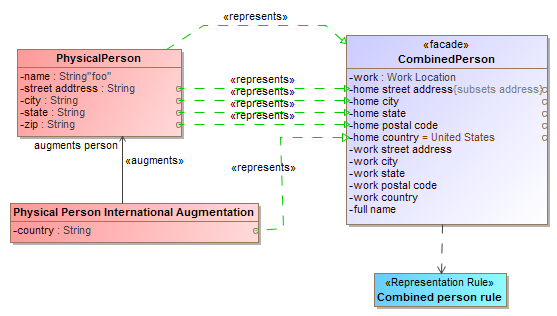
<<Represents>> is an assertion that the source type or feature provides a more concrete way to represent the target type or feature. Represents may be used within conceptual models or from a physical model to a conceptual model.

* A representation that is a dependency or realization makes no assumption that the types are substitutable (Used in mappings)
* A representation that is a generalization is substitutable for what it represents (used in the conceptual model)

Tag definitions

* Condition:Value specification – expression that must evaluate to true for the rule to be valid
* In context:Context – context that scopes the applicability of the representation. Context may also be specified by an un-stereotyped dependency from the rule to the context.
* type:Classifier – Potentially more specific type that a property is mapped to

Examples



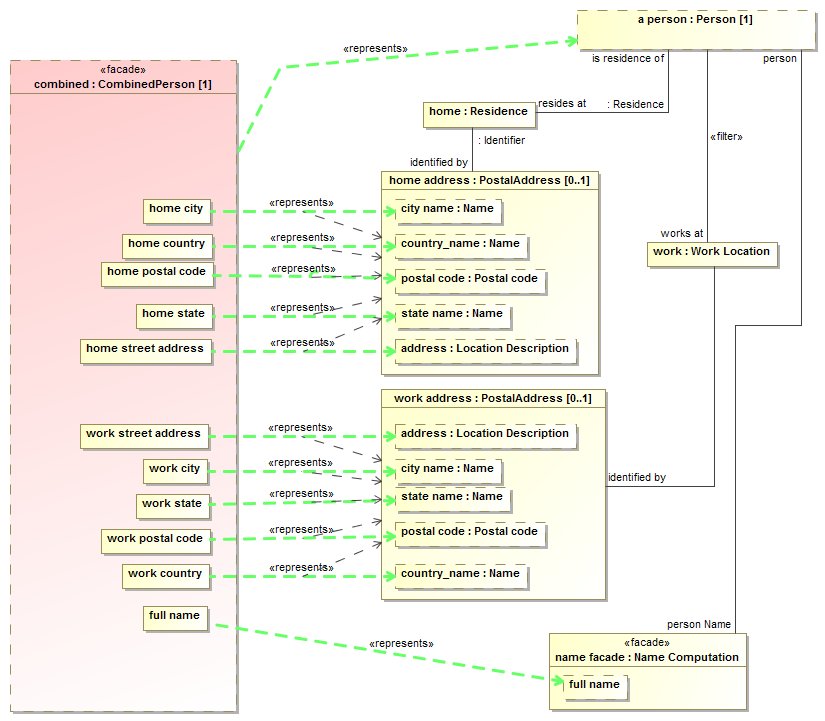
### Class: Representation Computation

A facade that includes calculations as defined by push() and pull() methods.

* Operation: push() - An operation called to evoke the behavior associated with a new facade element being created or modified.
* Operation: pull() - An operation called to evoke the behavior associated with a facade representing existing elements.

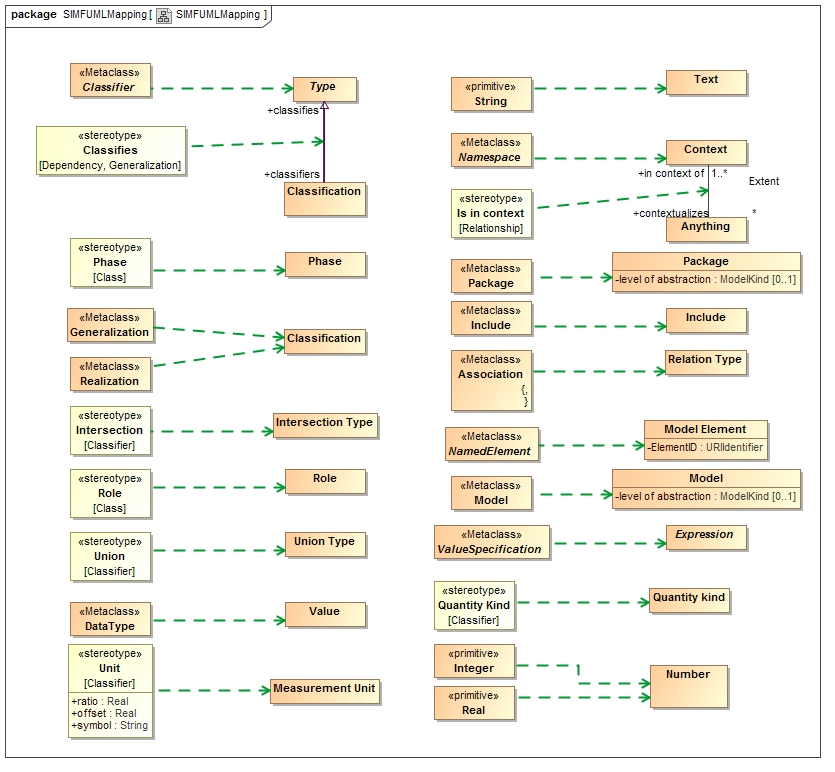
### Stereotype: Representation Rule [Class]

A representation rule is a pattern structure described by a structured classifier that shows how both "sides" of a representation must match and any traversals through those structures. Such traversals are links which may also have filters to more precisely define the pattern. The mapping engine ensures that the patterns match, bidirectionally.



## Profile to SIMF Model Mapping

The following diagram provides a high-level mapping of the UML meta model and profile to the SIMF meta model



1. ECSS is an initiative established to develop a coherent, single set of user-friendly standards for use in all European space activities [ECSS]. [↑](#footnote-ref-1)
2. Not to be confused with how “domain” is used in the intelligence community, which refers to levels of security [↑](#footnote-ref-2)
3. “transaction” in this context simply means a set of statemetns asserted at once, it may or may not have ACID properties as do DBMS transactions. [↑](#footnote-ref-3)