Date: Wednesday 4th November, 2015

The Distributed Ontology, Modeling, and Specification Language (DOL)

Version 0.98a

OMG Document Number: ad/2015-08-01
Machine readable files (normative): ad/2015-08-03, ad-2015-07-02

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.htm`).

Table of Contents

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
  - CORBA/IIOP
  - Data Distribution Services
  - Specialized CORBA
- IDL/Language Mapping Specifications
- Modeling and Metadata Specifications
  - UML, MOF, CWM, XMI
  - UML Profile
- Modernization Specifications
- Platform Independent Model (PIM), Platform Specific Model (PSM), Interface Specifications
  - CORBAServices
  - 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
140 Kendrick Street
Building A, Suite 300
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: Italic text represents names defined in the specification or 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`.

0 Submission-Specific Material

0.1 Submission Preface

Fraunhofer FOKUS, MITRE, and Thematix Partners LLC are pleased to submit this joint proposal in response to the Ontology, Modeling and Specification Integration and Interoperability (OntoIOp) RFP (OMG document ad/2013-12-02). The joint proposal is supported by Athan Services and the Otto-von-Guericke University Magdeburg. The contacts for this submission are:

- Fraunhofer FOKUS, Andreas Hoffmann, andreas.hoffmann@fokus.fraunhofer.de
- MITRE, Leo Obrst, lobrst@mitre.org
- Thematix Partners LLC, Elisa Kendall, ekendall@thematix.com
- Athan Services, Tara Athan, taraathan@gmail.com
- Otto-von-Guericke University Magdeburg, Till Mossakowski, till@iws.cs.uni-magdeburg.de (*lead contact*)

0.2 Mandatory Requirements

ID	RFP requirement	How this proposal addresses requirement
6.5.1(a)	Proposals shall provide a specification of a metalanguage for relationships between the components of logically heterogeneous OMS, particularly, given a language translation from a language L1 to another language L2, the application of the language translation to an OMS that is written in the language L1.	DOL provides the required translation construct using syntax `O with translation t`, see 9.4 and 9.4.2. Moreover, DOL provides heterogeneous interpretations between OMS, see 9.5 and 9.5.2.
6.5.1(b)	Proposals shall provide a specification of a metalanguage for the union of OMS written in different languages, which implicitly involves the application of suitable default translations in order to reach a common target language.	The syntax for unions is `O1 and O2`, see 9.4 and 9.4.2. Default translations are discussed in 9.4, and DOL's notion of heterogeneous logical environment explicitly specifies default translations, see 10.2.
6.5.1(c)	Proposals shall provide a specification of a metalanguage for importation in modular OMS.	DOL allows the import of OMS by their IRI, see 9.4 and 9.4.2.
6.5.1(d)	Proposals shall provide a specification of a metalanguage for relationships between OMS and their extracted modules e.g. the whole theory is a conservative extension of the module.	DOL provides such a construct with syntax `module m : o1 of o2 for sig`, see 9.5 and 9.5.2.
6.5.1(e)	Proposals shall provide a specification of a metalanguage for relationships between OMS and their approximation in less expressive languages such that the approximation is logically implied by the original theory, where the approximation generally has to be maximal in some suitable sense.	DOL provides such a construct with syntax `o keep logic`, see 9.4 and 9.4.2.
6.5.1(f)	Proposals shall provide a specification of a metalanguage for links such as imports, interpretations, refinements, and alignments between OMS/modules.	DOL covers several metalogical relationships, namely entailments, interpretations, equivalences, refinements, alignments and module relations, see 9.5 and 9.5.2.
6.5.1(g)	Proposals shall provide a specification of a metalanguage for combination of OMS along links.	DOL provides such a construct with syntax `combine n`, where `n` is a network of OMS and mappings (links), see 9.4 and 9.4.2.
6.5.2(a)	The constructs of the metalanguage shall be applicable to different logics.	The semantics of DOL is based on a heterogeneous logical environment, which can contain arbitrary logics, see 10.2.

Continued on next page

Table 0.1 – *Continued from previous page*

ID	RFP requirement	How this proposal addresses requirement
6.5.2(b)	The metalanguage shall neither be restricted to OMS in a specific domain, nor to OMS represented in a specific logical language.	The semantics of DOL is based on a heterogeneous logical environment, which can contain arbitrary logics, see 10.2.
6.5.2(c)	The metalanguage shall not replace the object language constructs of the conforming logical languages.	The syntax of a `NativeDocument` is left unspecified in this standard. Rather, here this standard relies on other standards and language definitions. See 9.4 and 9.4.2.
6.5.2(d)	The metalanguage shall provide syntactic constructs for (i) structuring OMS regardless of the logic in which their sentences are formalized and (ii) basic and structured OMS and facilities to identify them in a globally unique way.	The structuring constructs for OMS in 9.4 and 9.4.2 can be used for any logic, see the semantics in 10.2. DOL uses IRIs for referencing both basic and structured OMS, see 9.6.1.
6.5.3(a)	An abstract syntax specified as an SMOF compliant meta model.	The abstract syntax is specified using SMOF, see clause 9. An EBNF variant is given in annex ??.
6.5.3(b)	A human-readable lexical concrete syntax in EBNF and serialization in XML, for the latter XMI shall be used.	The concrete syntax (in EBNF) is specified in clause 9. The XMI representation is automatically derived from the SMOF meta model.
6.5.3(c)	Complete round-trip mappings from the human-readable concrete syntax to the abstract syntax and vice versa.	The metaclasses of the MOF abstract syntax are used as non-terminals of the EBNF concrete syntax (clause 9); this makes a round-trip mapping between both straight-forward. Moreover, the round-trip mapping has been implemented in form of a parser and a printer as part of the heterogeneous tool set (see appendix ?? and `http://hets.eu`).
6.5.3(d)	A formal semantics for the abstract syntax.	The formal semantics is given in clause 10.
6.5.4(a)	Existing OMS in existing serializations shall validate as OMS in the metalanguage with a minimum amount of syntactic adaptation.	Any document providing an OMS in a serialization of a DOL conforming language can be used as-is in DOL, by reference to its IRI. See 9.8.
6.5.4(b)	It shall be possible to refer to existing files/documents from an OMS implemented in the metalanguage without the need for modifying these files/-documents.	Documents can be referenced by IRIs, see 9.6.1.
6.5.4(c)	Translations between logical languages shall preserve (possibly to different degrees) the semantics of the logical languages. Between a given pair of logical languages, several translations are possible.	The semantics of DOL is based on a heterogeneous logical environment, which contains institution comorphisms as translations, see 10.2. Institution comorphisms preserve semantics in a weak form through their satisfaction condition. The DOL Ontology specifies properties of translations (comorphisms) preserving more and more of the semantics, see annex ??.
6.5.5(a)	Informative annexes shall establish the conformance of a number of relevant logical languages. An initial set of language translations may be part of an informative annex.	For conformance of logical languages, see 6.5.5(b) below. Conformance of some translations is established in annex ??.
6.5.5(b)	Conformance of the following subset of logical languages shall be established: OWL2 (with profiles EL, RL, QL), CLIF, RDF, UML class diagrams.	Conformance of the following languages is established: OWL 2 (annex ??), CLIF (annex ??), RDF and RDF Schema (annex ??), UML class diagrams (annex ??).
6.5.5(c)	Conformance of a suitable set of translations among the languages mentioned in the previous bullet point shall be established.	Conformance of some translations is established in annex ??.
6.5.6	Existing standards and best practices for allocating globally unique identifiers shall be reused. The same standards and best practices shall also be applied to associate different representations of the same content to one unique identifier.	DOL uses IRIs to reference documents (both DOL documents, as well as documents written in some conforming language). See 9.6.1.

0.3 Optional Requirements

ID	RFP requirement	How this proposal addresses requirement
6.6.1	Submissions may include additional languages without a standardized model theory.	This has been left for forthcoming versions.
6.6.2	Proposals may provide constructs for non-monotonic logics.	Currently, only monotonic logics are supported. However, DOL provides a circumscription-like non-monotonic structuring construct with syntax `o1 then %minimize o2`, see 9.4 and 9.4.2.
6.6.3	A characterization of the trade-offs among different translations.	This is left for future work.

0.4 Issues to be Discussed

ID	Discussion item	Resolution
6.7.(a)	Do existing language standards need to be extended or adapted in order to make them OntoIOp conforming.	The goal of DOL is to support existing languages without any adaptations, see also 6.5.4(a). However, in order to meet requirement 6.5.6, DOL-conforming languages should support the use of IRIs. If they do not, there is a mechanism for assigning IRIs to (fragments of) language documents even if the language itself does not support this, see 2.2. Moreover, there is a mechanism for injecting IRIs in existing language serializations, see 9.8 and 8.8.
6.7.(b)	Proposals should discuss whether the semantics of the metalanguage shall be included into the standard	The semantics of the DOL metalanguage is included in this specification. The reasons are discussed in the introduction of clause 10.
6.7.(c)	Proposals should discuss the chosen list of logics and translations.	The chosen list of logics and translations is discussed in the introduction of annex ??.
6.7.(d)	Proposals should discuss a meta-ontology of logical languages and theories.	The DOL Ontology is discussed in annex ??.
6.7.(e)	Proposals should discuss the use of QVT for expressing logic translations.	This is discussed in annex ??.
6.7.(f)	Proposals should discuss the role of APIs.	The role of APIs is discussed in section ??.
6.7.(g)	Proposals should discuss availability and use of tools.	Tools for DOL are discussed in annex ??.
6.7.(h)	Proposals should discuss a registry of logical languages.	A registry is discussed in annex ??.

0.5 Evaluation Criteria

ID	Criterion	Comment
6.8(a)	Proposals covering a broader range of features and of use cases will be favored. As a minimum, proposals shall define conformance criteria for logical languages and translations, and their proposed metalanguage shall cover some metalogical relationships and shall be applicable to multiple logics.	Based on the notion of institution, conformance criteria for logical languages are defined in 2.1 and those for translations in 2.1.1. DOL covers several metalogical relationships, namely entailments, interpretations, equivalences, refinements, alignments and module relations, see 9.5 and 9.5.2. DOL is applicable to multiple logics (see also 6.8(c) and 8.5 below).
6.8(b)	Proposals covering existing language standards without (or with fewer) modifications will be favored.	Any document providing an OMS in a serialization of a DOL conforming language can be used as-is in DOL, by reference to its IRI. See 9.8.
6.8(c)	Proposals establishing actually (or making this at least possible in theory) OntoIOp conformance of more logical languages and translations will be favored.	The conformance of OWL 2 (annex ??), Common Logic (annex ??), RDF and RDF Schema (annex ??), UML class diagrams (annex ??) and CASL (annex ??) is established.

0.6 Proof of Concept

Prototypical open source tools for DOL are already available, see annex ??. It is expected that they will reach industrial strength within two or three years.

0.7 Changes to Adopted OMG Specifications

This specification proposes no changes to adopted OMG specifications.

1 Scope

This OMG Specification specifies the Distributed Ontology, Modeling and Specification Language (DOL). DOL is designed to achieve integration and interoperability of ontologies, specifications and MDE models (OMS for short). DOL is a language for distributed knowledge representation, system specification and model-driven development across multiple OMS, particularly OMS that have been formalized in different OMS languages.[1] This OMG Specification responds to the OntoIOp Request for Proposals [?]. Note(1)

1.1 Background Information

Logical languages are used in several fields of computing for the development of formal, machine-processable texts that carry a formal semantics. Among those fields are 1) **O**ntologies formalizing domain knowledge, 2) (formal) **M**odels of systems, and 3) the formal **S**pecification of systems. Ontologies, MDE models and specifications will (for the purpose of this document) henceforth be abbreviated as **OMS**.

An OMS provides formal descriptions, which range in scope from domain knowledge and activities (ontologies, MDE models) to properties and behaviors of hardware and software systems (MDE models, specifications). These formal descriptions can be used for the analysis and verification of domain models, system models and systems themselves, using rigorous and effective reasoning tools. As systems increase in complexity, it becomes concomitantly less practical to provide a monolithic logical cover for all. Instead various MDE models are developed to represent different viewpoints or perspectives on a domain or system. Hence, interoperability becomes a crucial issue, in particular, formal interoperability, i.e. interoperability that is based on the formal semantics of the different viewpoints. Interoperability is both about the ability to interface different domains and systems and the ability to use several OMS in a common application scenario. Further, interoperability is about coherence and consistency, ensuring at an early stage of the development that a coherent system can be reached.

In complex applications, which involve multiple OMS with overlapping concept spaces, it is often necessary to identify correspondences between concepts in the different OMS; this is called OMS alignment. While OMS alignment is most commonly studied for OMS formalized in the same OMS language, the different OMS used by complex applications may also be written in different OMS languages, which may even vary in their expressiveness. This OMG Specification faces this diversity not by proposing yet another OMS language that would subsume all the others. Instead, it accepts the diverse reality and formulates means (on a sound and formal semantic basis) to compare and integrate OMS that are written in different formalisms. It specifies DOL, a formal language for expressing not only OMS but also mappings between OMS formalized in different OMS languages.

Thus, DOL gives interoperability a formal grounding and makes heterogeneous OMS and services based on them amenable to checking of coherence (e.g. consistency, conservativity, intended consequences, and compliance).

1.2 Features Within Scope

The following are within the scope of this OMG Specification:

1. homogeneous OMS as well as heterogeneous OMS (OMS that consist of parts written in different languages);
2. mappings between OMS (which map OMS symbols to OMS symbols);
3. OMS networks (involving several OMS and mappings between them);
4. translations between different OMS languages conforming with DOL (translating a whole OMS to another language);
5. structuring constructs for modeling non-monotonic behavior;
6. annotation and documentation of OMS, mappings between OMS, symbols, and sentences;
7. recommendations of vocabularies for annotating and documenting OMS;
8. a syntax for embedding the constructs mentioned under (1)–(6) as annotations into existing OMS;
9. a syntax for expressing (1)–(5) as standoff markup that points into existing OMS;
10. a formal semantics of (1)–(5);
11. criteria for existing or future OMS languages to conform with DOL.

[1] NOTE: Suggestion from Terry: "DOL is a tool for managing and manipulating distributed knowledge representations, system specifications, and model-driven design/development artifacts among multiple OMS, particularly...". However, I think DOL is not a tool, but a language. Hence, this does not fit. TM

The following are outside the scope of this OMG Specification:

1. the (re)definition of elementary OMS languages, i.e. languages that allow the declaration of OMS symbols (non-logical symbols) and stating sentences about them;
2. algorithms for obtaining mappings between OMS;
3. concrete OMS and their conceptualization and application;
4. mappings between services and devices, and definitions of service and device interoperability;
5. non-monotonic logics[1].

This OMG Specification describes the syntax and the semantics of the Distributed Ontology, Modeling and Specification Language (DOL) by defining an abstract syntax and an associated model-theoretic semantics for DOL.

[1] Only monotonic logics are within scope of this specification. Conformance criteria for non-monotonic logics are still under development. However, closure (i.e. employing a closed-world assumption) provides non-monotonic reasoning in DOL. It is also possible to include non-monotonic logics by construing entailments between formulas as sentences of the logic (formalized as an institution).

2 Conformance

This clause defines conformance criteria for languages and logics that can be used with DOL, as well as conformance criteria for serializations, translations and applications. The conformance of a number of OMS languages (namely OWL 2, Common Logic, RDF and RDF Schema, UML Class Diagrams, TPTP, CASL) as well as translations among these is discussed in informative annexes of this OMG Specification.

2.1 Conformance of an OMS Language/a Logic with DOL

Rationale: for an OMS language to conform with DOL,

- its logical language aspect either needs to satisfy certain criteria related to its own abstract syntax and formal semantics, or there must be a translation (again satisfying certain criteria) to a language that already is DOL-conforming.
- its structuring language aspect (if present) must be compatible with DOL's own structuring mechanisms
- its annotation language aspect must be compatible with DOL's meta-language constructs.

Several conformance levels are defined. They differ with respect to the usage of IRIs as identifiers for all kinds of entities that the OMS language supports.

An OMS language is conforming with DOL if it satisfies the following conditions:

1. its abstract syntax is specified as an SMOF compliant meta model or as an EBNF grammar;
2. it has at least one serialization in the sense of section 2.2;
3. either there exists a translation of it into a conforming language[1], or:
   a) the logical language aspect (for expressing basic OMS) is conforming, and in particular has a semantics (see below),
   b) the structuring language aspect (for expressing structured OMS and relations between those) is conforming (see below), and
   c) the annotation language aspect (for expressing comments and annotations) is conforming (see below).

The *logical language aspect* of an OMS language is conforming with DOL if each logic corresponding to a profile (including the logic corresponding to the whole logical language aspect) is presented as an institution in the sense of Definition 2 in clause 10 , and there is a mapping from the abstract syntax of the OMS language to signatures and sentences of the institution. Note that one OMS language can have several sublanguages or profiles corresponding to several logics (for example, OWL 2 has profiles EL, RL and QL, apart from the whole OWL 2 itself).

The *structuring language aspect* of an OMS language is conforming with DOL if it can be mapped to DOL's structuring language in a semantics-preserving way. The structuring language aspect **may** be empty.

The *annotation language aspect* of an OMS language is conforming with DOL if its constructs have no impact on the semantics. The annotation language aspect **shall** be non-empty; it **shall** provide the facility to express comments.

Concerning item 1. in the definition of DOL conformance of OMS languages above, the following levels of conformance of the abstract syntax of an OMS language with DOL are defined, listed from highest to lowest:

Full IRI conformance The abstract syntax specifies that IRIs be used for identifying all symbols and entities.

No mandatory use of IRIs The abstract syntax does not require IRIs to be used to identify entities. Note that this includes the case of optionally supporting IRIs without enforcing their use (such as in Common Logic).

Any conforming language and logic shall have a machine-processable description as detailed in clause 2.3.

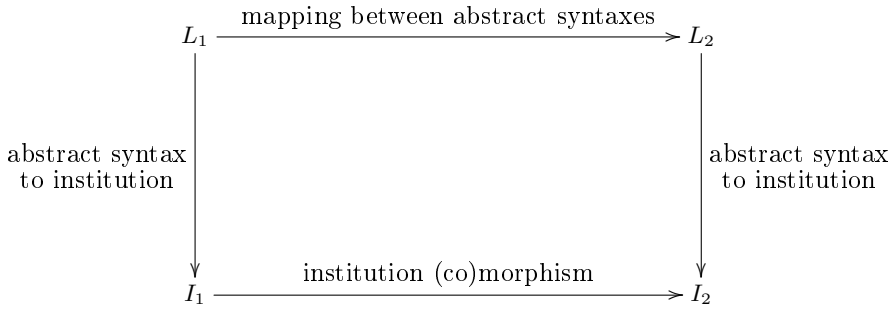
2.1.1 Conformance of language/logic translations with DOL

Rationale: a translation between logics must satisfy certain criteria in order to conform with DOL. Also, a translation between OMS languages based on such logics must be consistent with the translation between these logics. Translations should break neither structuring language aspects nor comments/annotations.

[1] For example, consider the translation of OBO1.4 to OWL, giving a formal semantics to OBO1.4.

A logic translation is conforming with DOL if it is presented either as an institution morphism or as an institution comorphism.

A language translation **shall** provide a mapping between the abstract syntaxes (it **may** also provide mappings between concrete syntaxes). A language translation from language L_1 (based on institution I_1) to language L_2 (based on institution I_2) is conforming with DOL if it is based on a logic translation such that the following diagram commutes (i.e. following both possible paths from L_1 to I_2 leads to the same result):



Language translations **may** also translate the structuring language aspect, in this case, they **shall** preserve the semantics of the structuring language aspect. Furthermore, language translations **should** preserve comments and annotations. All comments attached to a sentence (or symbol) in the source **should** be attached to its translation in the target (if there are more than one sentences (respectively symbols) expressing the translation, to at least one of them).[2] Note(2)

2.2 Conformance of a Serialization of an OMS Language With DOL

Rationale: The main reason for the following specifications is identifier injection. DOL is capable of assigning identifiers to entities (symbols, axioms, modules, etc.) inside fragments of OMS languages that occur in a DOL document, even if that OMS language does not support such identifiers by its own means. Such identifiers will be visible to a DOL tool, but not to a tool that only supports the OMS language. To achieve this without breaking the formal semantics of that OMS language, DOL utilizes the annotation or commenting features that the OMS language supports, in order to place such identifiers inside annotations/ comments. Depending on the nature of a given concrete serialization of the OMS language (be it plain text, some serialization of RDF, XML, or some other structured text format), one can be more specific about what the annotation/commenting facilities of that serialization must look like in order to support this identifier injection. Well-behaved XML and RDF schemas support identifier injection in a 'nice' way (rather than using text-level comments). In the worst case it is not possible to inject something into an OMS language fragment, because the OMS language serialization does not enable the addition of suitable comments. In this case the solution is to point into the OMS language fragment from the enclosing context by using standoff markup.

Further conformance criteria in this section are introduced to facilitate the convenient reuse of verbatim fragments of OMS language inside a DOL document.

Independently from these criteria, several levels of conformance of a serialization are distinguished. They differ with respect to their means of conveniently abbreviating long IRI identifiers.

There are seven levels of conformance of a serialization of an OMS language with DOL.

XMI conformance An XMI serialization for OMS written in the OMS language has been automatically derived from the SMOF specification of the abstract syntax, using the canonical MOF 2 XMI Mapping.

XML conformance The given serialization has to be specified as an XML schema that satisfies all of the following conditions:

1. The elements of the schema belong to one or more non-empty XML namespaces.
2. The serialization shall use XML *elements* to represent all structural elements of an OMS.
3. XML elements that represent structural elements of an OMS shall support identifier injection in at least one of the following two ways:
   a) Such elements shall be able to carry annotations that comprise at least an object (the value of the annotation) and a IRI-valued predicate (the type of annotation), where the structural element is the subject. The value of the predicate shall either be full IRI according to IETF/RFC 3987:2005, or the serialization shall specify a way of interpreting the value of the predicate as a full IRI – for example if it is a relative URI or if an abbreviating notation is used. Analogously, the serialization shall permit the object to be a full IRI or anything that can be interpreted as a full IRI.
   b) The schema shall not forbid both attributes and child elements from foreign namespaces (here: the DOL namespace `http://www.omg.org/spec/DOL/1.0/xml`) on such elements.

[2] NOTE: TM: corrected from wrong English. "there is more than one" needs to be followed by a verb in singular form.

This requirement is necessary because either an annotation or an attribute or a child element is used to inject identifiers into elements of the XML serialization; cf. clause 9.8.

RDF conformance The given serialization has to be specified as an RDF vocabulary that satisfies all of the following conditions:

1. The elements of the vocabulary belong to one or more RDF namespaces identified by absolute URIs.
2. The serialization shall specify ways of giving IRIs or URIs to all structural elements of an OMS. (The rationale is that RDF syntax supports the identification of any kinds of items, so an RDF-based serialization of an OMS language should not forbid making use of such RDF constructs that do allow for identifying arbitrary items.)
3. There shall be no additional rules (stated in writing in the specification of the serialization, or formalized in its implementation in, e.g., OWL) that forbid properties from foreign vocabulary namespaces to be stated about arbitrary subjects for the purpose of annotation.

See annex **??** for an example.

Text conformance The given serialization has to satisfy all of the following conditions:

- The serialization conforms with the requirements for the *text/plain* media type specified in IETF/RFC 2046, section 4.1.3.
- The serialization shall provide a designated comment construct that can be placed sufficiently flexibly as to be uniquely associated with any non-comment construct of the language. That means, for example, one of the following:
  - The serialization provides a construct that indicates the start and end of a comment and may be placed before/after each token that represents a structural element of an OMS.
  - The serialization provides line-based comments (ranging from an indicated position to the end of a line) but at the same time allows the flexible placement of line breaks before/after each token that represents a structural element of an OMS.

Standoff markup conformance The given serialization has to satisfy at least one of the following conditions:

1. The serialization conforms with the requirements for the *text/plain* media type specified in IETF/RFC 2046, section 4.1.3. Note that conformance with *text/plain* is a prerequisite for using, for example, fragment URIs in the style of IETF/RFC 5147 for identifying text ranges.
2. The serialization conforms with XML W3C/TR REC-xml:2008, which is a prerequisite for using XPointer fragment URIs for addressing subresources of an XML resource (cf. W3C/TR REC-xptr-framework:2003).

Independently from the conformance levels given above, there is the following hierarchy of conformance w.r.t. CURIEs (compact URIs) as a means of abbreviating IRIs (grammar specified in clause 9.6.2), listed from highest to lowest:

Prefixed CURIE conformance The given serialization allows non-logical symbol identifiers to have the syntactic form of a CURIE, or any subset of the CURIE grammar that allows named prefixes (`prefix:reference`, where a declaration of DOL-conformance of a serialization **may** redefine the separator character to a character different from `:`). A serialization that conforms w.r.t. a prefixed CURIE is **not required** to support CURIEs with no prefix: its declaration of DOL-conformance **may** forbid the use of prefixed CURIEs.
Informative comments:

- In the case that CURIEs are used, a prefix map with multiple prefixes **may** be used to map the non-logical symbol identifiers of a native OMS to IRIs in multiple namespaces (cf. clause 9.6.3)
- The reason for allowing redefinitions of the prefix/reference separator character is that certain serializations of OMS languages may not allow the colon (`:`) in identifiers.

Non-prefixed names only The given serialization only supports CURIEs with no prefix, or any subset of the grammar of the `REFERENCE` nonterminal in the CURIE grammar.
Informative comment: In this case, a binding for the empty prefix **must** be declared, as this is the only possibility of mapping the identifiers of the native OMS to IRIs that are located in one flat namespace.

Any conforming serialization of an OMS language shall have a machine-processable description as detailed in clause 2.3.

2.3 Machine-Processable Description of Conforming Languages, Logics, and Serializations

Rationale: When a parser processes a DOL OMS found somewhere that refers to modules in OMS languages, or includes them verbatim, the parser needs to know what language to expect; further DOL-supporting software needs to know, e.g., what other DOL-conforming languages the module in the given OMS language can be translated to. Therefore, all languages/logics/serializations that conform with DOL are required to describe themselves in a machine-processable way.

For any conforming OMS language, logic, and serialization of an OMS language, it is required that it be assigned an HTTP IRI, by which it can be identified. It is also required that a machine-processable description of this language/-logic/serialization is retrievable by dereferencing this IRI; this requirement follows the linked data principles W3C/TR REC-ldp-20150226:2015. As a minimal requirement, there must be a RDF description conforming to the vocabulary specified in annex **??**. That description must be made available in the RDF/XML serialization when a client requests content of the MIME type *application/rdf+xml*. Descriptions of the language/logic/serialization in further representations, having different content types, may be provided.

2.4 Conformance of a Document With DOL

Rationale: for exchanging DOL documents with other users/tools, nothing that has a formal semantics must be left implicit. One DOL tool may assume that by default any OMS fragments inside a DOL document are in some fixed OMS language unless specified otherwise, but another DOL tool can't be assumed to understand such DOL documents. Defaults are, however, practically convenient, which is the reason for having the following section about the conformance of an *application*.

A document conforms with DOL if it contains a DOL text that is well-formed according to the grammar. That means, in particular, that any information related to logics must be made explicit (as foreseen by the DOL abstract syntax specified in clause 9), such as:

- the logic of each OMS that is part of the DOL document,
- any translation that is employed between two logics (unless it is one of the default translations specified in annex **??**)

However, details about aspects of an OMS that do not have a formal, logic-based semantics, may be left implicit. For example, a conforming document may omit explicit references to matching algorithms that have been employed in obtaining an alignment.

2.5 Conformance of an Application With DOL

In the sequel, "DOL abstract syntax" means an XMI document that conforms to the DOL metamodel. Optionally, further representations (e.g. as JSON) can be supported.

- A *parser* is DOL-conformant if it can parse the DOL textual syntax and produce the corresponding DOL abstract syntax.
- A *printer* is DOL-conformant if it can read DOL abstract syntax and produce DOL textual syntax.
- DOL-conformant software that is used to *edit, format or manage* DOL libraries **must** be capable of reading and writing DOL abstract syntax. Moreover, it **must** meet the requirements for a DOL-conformant parser if it is able to read in DOL textual input. It **must** meet the requirements of a DOL-conformant printer if it is able to generate DOL textual output. However, it is also possible that a software for DOL management will work on the abstract syntax only, delegating the reading and generation of DOL text to external parsers and/or printers.
- a *static analyzer* is DOL-conformant if it can compute the logic and the signature of an OMS according to the semantics defined in section 10. In more detail, a static analyzer can have the following capabilities:
  - *simple analysis*: static analysis of DOL excluding networks and alignments;
  - *full analysis*: static analysis of full DOL.
- a *transformation tool* is DOL-conformant if it implements one (or more) language translations, logic translations, language projections and/or logic projections.
- Software that implements machine *reasoning* about OMS (e.g., theorem proving, approximation) complies with this specification if and only if it interprets DOL documents according to the semantics defined in section 10. In more detail, a reasoning tool can have the following capabilities:
  - *simple logical consequence*, i.e. checking whether all sentences that are marked as `%implied` within basic OMS and extensions are logical consequences of the enclosing OMS;
  - *structured logical consequence*, i.e. checking whether all sentences that are marked as `%implied` are logical consequences of the enclosing OMS and whether all entailments in a DOL document have a defined semantics;
  - *interpretation*, i.e. checking whether all interpretations in a DOL document have a defined semantics;
  - *simple refinement*, i.e. checking whether all refinements of OMS in a DOL document have a defined semantics;
  - *full refinement*, i.e. checking whether all refinements (both of OMS and networks) in a DOL document have a defined semantics;
  - *simple conservativity*, i.e. checking whether all conservativity statements in a DOL document have a defined semantics;

- *full conservativity*, i.e. checking whether all statements about conservative, monomorphic, definitional and weakly definitional extensions in a DOL document have a defined semantics;
- *module extraction*, i.e. the ability to compute modules (typically, a given tool will provide this only for some logics);
- *approximation*, i.e. the ability to compute approximations (typically, a given tool will provide this only for some logics and logic projections);
- *full* DOL *reasoning*, i.e. checking whether an DOL document has a defined semantics.

In practice, DOL-aware *applications* may also deal with documents that are not conforming with DOL according to the criteria established in clause 2.4. However, an application only *conforms* with DOL if it is capable of producing DOL-conforming documents as its output when requested.

DOL-aware applications **shall** support a fixed (possibly extensible) set of OMS languages conforming with DOL.

It is, for example, possible that a DOL-aware application only supports OWL and Common Logic. In that case, the application may process DOL documents that mix OWL and Common Logic ontologies, as well as native OWL and Common Logic documents.

DOL-aware applications also **shall** be able to strip DOL annotations from embedded fragments in other OMS languages. Moreover, they **shall** be able to expand CURIEs into IRIs when requested.

3 Normative References

NR1: W3C/TR REC-ldp-20150226:2015 Linked Data Platform 1.0. W3C Recommendation, 26 February 2015.
http://www.w3.org/TR/2015/REC-ldp-20150226/

NR2: W3C/TR REC-owl2-syntax:2012 OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax (Second Edition). W3C Recommendation, 11 December 2012.
http://www.w3.org/TR/2012/REC-owl2-syntax-20121211/

NR3: ISO/IEC 14977:1996 Information technology – Syntactic metalanguage – Extended BNF

NR4: W3C/TR REC-xml:2008 Extensible Markup Language (XML) 1.0 (Fifth Edition). W3C Recommendation, 26 November 2008.
http://www.w3.org/TR/2008/REC-xml-20081126/

NR5: W3C/TR REC-owl2-primer:2012 OWL 2 Web Ontology Language: Primer (Second Edition). W3C Recommendation, 11 December 2012.
http://www.w3.org/TR/2012/REC-owl2-primer-20121211/

NR6: W3C/TR REC-owl2-profiles:2012 OWL 2 Web Ontology Language: Profiles (Second Edition). W3C Recommendation, 11 December 2012.
http://www.w3.org/TR/2012/REC-owl2-profiles-20121211/

NR7: ISO/IEC 24707:2007 Information technology – Common Logic (CL): a framework for a family of logic-based languages

NR8: OMG Document ptc/2013-09-05: OMG Unified Modeling Language (OMG UML)
http://www.omg.org/spec/UML/Current

NR9: IETF/RFC 3986 Uniform Resource Identifier (URI): Generic Syntax. January 2005.
http://tools.ietf.org/html/rfc3986

NR10: IETF/RFC 3987 Internationalized Resource Identifiers (IRIs). January 2005.
http://tools.ietf.org/html/rfc3987

NR11: IETF/RFC 5147 URI Fragment Identifiers for the text/plain Media Type. April 2008.
http://tools.ietf.org/html/rfc5147

NR12: W3C/TR REC-xptr-framework:2003 XPointer Framework. W3C Recommendation, 25 March 2003.
http://www.w3.org/TR/2003/REC-xptr-framework-20030325/

NR13: W3C/TR REC-rdf11-concepts:2014 RDF 1.1 Concepts and Abstract Syntax. W3C Recommendation, 25 February 2014.
http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/

NR14: W3C/TR REC-xml-names:2009 Namespaces in XML 1.0 (Third Edition). W3C Recommendation, 8 December 2009.
http://www.w3.org/TR/2009/REC-xml-names-20091208/

NR15: W3C/TR REC-rdfa-core:2015 RDFa Core 1.1 – Third Edition. Syntax and processing rules for embedding RDF through attributes. W3C Recommendation, 17 March 2015.
http://www.w3.org/TR/2015/REC-rdfa-core-20150317/

NR16: ISO/IEC 10646 Information technology – Universal Multiple-Octet coded Character Set (UCS)

NR17: W3C/TR REC-rdf-schema:2014 RDF Schema 1.1. W3C Recommendation, 25 February 2014.
http://www.w3.org/TR/2014/REC-rdf-schema-20140225/

NR18: W3C/TR REC-rdf11-mt:2014 RDF 1.1 Semantics. W3C Recommendation, 25 February 2014.
http://www.w3.org/TR/2014/REC-rdf11-mt-20140225/

NR19: W3C/TR REC-owl2-mapping-to-rdf:2012 OWL 2 Web Ontology Language Mapping to RDF Graphs (Second Edition). W3C Recommendation, 11 December 2012
http://www.w3.org/TR/2012/REC-owl2-mapping-to-rdf-20121211/

NR20: DCMI Metadata Terms:2012 DCMI Metadata Terms, DCMI Recommendation, DCMI Usage Board, 14 July 2012.
http://dublincore.org/documents/2012/06/14/dcmi-terms/

NR21: W3C/TR REC-skos-reference:2009 SKOS Simple Knowledge Organization System Reference. W3C Recommendation, 18 August 2009
http://www.w3.org/TR/2009/REC-skos-reference-20090818/

NR22: OMG Specification Metadata:2014 Specification Metadata (SM) Vocabulary. OMG, 18 August 2014
http://www.omg.org/techprocess/ab/20140801/SpecificationMetadata.rdf

NR23: ODM Ontology Definition Metamodel, 2 September 2014.
http://www.omg.org/spec/ODM/1.1/

NR24: MOF Meta Object Facility
http://www.omg.org/spec/MOF/

NR25: SMOF Support for Semantic Structure, April 2013
http://www.omg.org/spec/SMOF/1.0/

NR26: XMI Metadata Interchange (XMI) – using MOF 2 XMI, April 2014
http://www.omg.org/spec/XMI/

NR27: SBVR Semantics Of Business Vocabulary And Rules, November 2013
http://www.omg.org/spec/SBVR/

NR28: DTV Date-Time Vocabulary, August 2013
http://www.omg.org/spec/DTV/1.0/

NR29: RIF Rule Interchange Format, February 2013
http://www.w3.org/TR/rif-overview/

4 Terms and Definitions

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

4.1 Distributed Ontology, Modeling and Specification Language

Distributed Ontology, Modeling and Specification Language; DOL unified metalanguage for the structured and heterogeneous expression of ontologies, specifications, and MDE models, using DOL libraries of OMS, OMS mappings and OMS networks, whose syntax and semantics are specified in this OMG Specification.

DOL library collection of named OMS and OMS networks, possibly written in different OMS languages, linked by named OMS mappings.

4.2 Native OMS, OMS, and OMS Languages

native OMS collection of expressions (like non-logical symbols, sentences and structuring elements) from a given OMS language.
EXAMPLE A UML class diagram, an ontology written in OWL 2 EL, and a specification written in CASL are three different native OMS.
NOTE An OMS can be written in different OMS language serializations.

native document document containing a native OMS.

DOL document document containing a DOL library.

OMS language language equipped with a formal, declarative, logic-based semantics, plus non-logical annotations.
EXAMPLE OMS languages include OWL 2 DL, Common Logic, F-logic, UML class diagrams, RDF Schema, and OBO.
NOTE An OMS language is used for the formal specification of native OMS.
NOTE An OMS language has a logical language aspect, a structuring language aspect, and an annotation language aspect.

DOL structured OMS syntactically valid DOL expression denoting an OMS that is built from smaller OMS as building blocks.

NOTE DOL structured OMS, typically, use basic OMS as building blocks for defining other structured OMS, OMS mappings or OMS networks.
NOTE All DOL structured OMS are structured OMS.

ontology logical theory that is used as a shard conceptualization

MDE model logical theory that is used as an abstract representation of a domain or of a system, in the sense of model-driven engineering (MDE)
NOTE Not to be confused with the term model in the sense of logic (model theory).

specification logical theory that is used to express formal constraint in mathematical structures, software systems and/or hardware systems

OMS (ontology, specification or MDE model) basic OMS or structured OMS.
NOTE
An OMS is either a basic OMS (which is always a native OMS, and can occur as a text fragment in a DOL document) or a structured OMS (which can be either a native structured OMS contained in some native document, or a DOL structured OMS contained in a DOL document).

NOTE An OMS has a single signature and model class over that signature as its model-theoretic semantics.

basic OMS; flat OMS native OMS that does not utilize any elements from the structuring language aspect of its language.
NOTE Basic OMS are self-contained in the sense that their semantics does not depend on some other OMS. In particular, a basic OMS does not involve any imports.
NOTE Since a basic OMS has no structuring elements, it consists of (or at least denotes) a signature equipped with a set of sentences and annotations.
NOTE In signature-free logics like Common Logic or TPTP, a basic OMS only consists of sentences. A signature can be obtained a posteriori by collecting all non-logical symbols occuring in the sentences.

non-logical symbol; OMS symbol atomic expression or syntactic constituent of an OMS that requires an interpretation through a model.
NOTE This differs from the notion of "atomic sentence": such sentences may involve several non-logical symbols.

EXAMPLE Non-logical symbols in OWL W3C/TR REC-owl2-syntax:2012 (there called "entities") comprise

- individuals (denoting objects from the domain of discourse),
- classes (denoting sets of objects; also called concepts), and
- properties (denoting binary relations over objects; also called roles).

These non-logical symbols are distinguished from logical symbols in OWL, e.g., those for intersection and union of classes.

EXAMPLE Non-logical symbols in Common Logic ISO/IEC 24707:2007 comprise

- names (denoting objects from the domain of discourse),
- sequence markers (denoting sequences of objects).

These non-logical symbols are distinguished from logical symbols in Common Logic, e.g. logical connectives and quantifiers.

signature; vocabulary set (or otherwise structured collection) of non-logical symbols of an OMS.
NOTE The signature of a term is the set of all non-logical symbols occurring in the term. The notion of signature depends on the OMS language or logic.
NOTE The signature of an OMS is usually unequivocally determinable.

model semantic interpretation of all non-logical symbols of a signature.
NOTE A model of an OMS is a model of the signature of the OMS that also satisfies all the additional constraints expressed by the OMS. In case of flattenable OMS, these constraints are expressed by the axioms of the OMS.
NOTE This term refers to *model* in the sense of model theory (a branch of logic). It is not to be confused with MDE model in the sense of modeling (i.e., the "M" in OMS).
NOTE The notion of model depends on the OMS language or logic.

expression a finite combination of symbols that are well-formed according to applicable rules (depending on the language)

term syntactic expression either consisting of a single non-logical symbol or recursively composed of other terms (a.k.a. its subterms).
NOTE A term belongs to the logical language aspect of an OMS language.

sentence term that is either true or false in a given model, i.e. which is assigned a truth value in this model.
NOTE In a model, on the one hand, a sentence is always true or false. In an OMS, on the other hand, a sentence can have several logical statuses. For example, a sentence can be: an axiom, if postulated to be true; a theorem, if proven from other axioms and theorems; or a conjecture, if expecting to be proven from other axioms and theorems.
NOTE A sentence can conform to one or more signatures (namely those signatures containing all non-logical symbols used in the sentence).
NOTE It is quite common that sentences are required to be closed (i.e. have no free variables). However, this depends on the OMS language at hand.
NOTE A sentence belongs to the logical language aspect of an OMS language.
NOTE The notion of sentence depends on the OMS language or logic.

satisfaction relation relation between models and sentences indicating which sentences hold true in the model.
NOTE The satisfaction relation depends on the OMS language or logic.

logical theory signature equipped with a set of sentences over the signature.

NOTE Each logical theory can also be written a basic OMS, and conversely each basic OMS has as its semantics a logical theory.

entailment; logical consequence; specialization relation between two OMS (or an OMS and a sentence, or two OMS networks, or an OMS network and an OMS) expressing that the second item (the conclusion) is logically implied by the first one (the premise).
NOTE Entailment expresses that each model satisfying the premise also satisfies the conclusion.

NOTE The converse is generalization.

axiom sentence that is postulated to be valid (i.e. true in every model).

theorem sentence that has been proven from other axioms and theorems and therefore has been demonstrated to be a logical consequence of the axioms.

tool software for processing DOL libraries and OMS.

theorem proving process of demonstraing that a sentence (or OMS) is the logical consequence of some OMS.

theorem prover tool implementing theorem proving.

4.3 Structured OMS

structured OMS OMS that results from other (basic and structured) OMS by import, union, combination, OMS translation, OMS reduction or other structuring operations.
NOTE Structured OMS are either DOL structured OMS or native OMS that utilize elements of the structuring language aspect of their OMS language.

flattenable OMS OMS that can be seen, by purely syntactical means, to be logically equivalent to a flat OMS.
NOTE More precisely, an OMS is flattenable if and only if it is either a basic OMS or it is an extension, union, translation, module, approximation, filtering, or reference of named OMS involving only flattenable OMS.

elusive OMS OMS that is not flattenable.

subOMS OMS whose associated sets of non-logical symbols and sentences are subsets of those present in a given larger OMS.

import reference to an OMS behaving as if it were verbatim included; also import of DOL libraries.
NOTE Semantically, an import of O_2 into O_1 is equivalent to the verbatim inclusion of O_2 in place of the import declaration.
NOTE The purpose of O_2 importing O_1 is to make non-logical symbols and sentences of O_1 available in O_2.
NOTE Importing O_1 into O_2 turns O_2 into an extension of O_1.
NOTE An owl:import in OWL is an import.
NOTE The import of a whole DOL library into another DOL library is also called import.

union DOL structured OMS expressing the aggregation of several OMS to a new OMS, without any renaming.

OMS translation DOL structured OMS expressing the assignment of new names to some non-logical symbols of an OMS, or translation of an OMS along a language translation.
NOTE An OMS translation results in an OMS mapping between the original and the renamed OMS.
NOTE Typically, the resulting OMS mapping of a translation is surjective: the symbols of the original OMS can be identified by the renaming, but no new symbols are added.

OMS reduction DOL structured OMS expressing the restriction of an OMS to a smaller signature.

local environment context for an OMS, being the signature built from all previously-declared symbols and axioms.

extension structured OMS extending a given OMS with new symbols and sentences.
NOTE The new symbols and sentences are interpreted relative to the local envorinment, which is the signature of the "given OMS".

extension mapping inclusion OMS mapping between two OMS where the sets of non-logical symbols and sentences of the second OMS are supersets of those present in the first OMS.
NOTE The second OMS is said to extend the first, and is an extension of the first OMS.

conservative extension extension that does not add new logical properties with respect to the signature of the extended OMS.
NOTE An extension is a consequence-theoretic or model-theoretic conservative extension. If used without qualification, the consequence-theoretic version is meant.

consequence-theoretic conservative extension extension that does not add new theorems (in terms of the unextended signature).
NOTE An extension O_2 of an OMS O_1 is a consequence-theoretic conservative extension, if all properties formulated in the signature of O_1 hold for O_1 whenever they hold for O_2.

model-theoretic conservative extension extension that does not lead to a restriction of class of models of an OMS.
NOTE An extension O_2 of an OMS O_1 is a model-theoretic conservative extension, if each model of O_1 can be expanded to a model of O_2.
NOTE Each model-theoretic conservative extension is also a consequence-theoretic one, but not vice versa.

monomorphic extension extension whose newly introduced non-logical symbols are interpreted in a way unique up to isomorphism.
NOTE An extension O_2 of an OMS O_1 is a monomorphic extension, if each model of O_1 can be expanded to a model of O_2 that is unique up to isomorphism.
NOTE Each monomorphic extension is also a model-theoretic conservative extension but not vice versa.

definitional extension extension whose newly introduced non-logical symbols are interpreted in a unique way.
NOTE An extension O_2 of an OMS O_1 is a definitional extension, if each model of O_1 can be uniquely expanded to a model of O_2.
NOTE O_2 being a definitional extension of O_1 implies a bijective correspondence between the classes of models of O_2 and O_1.
NOTE Each definitional extension is also a monomorphic extension but not vice versa.

weak definitional extension extension whose newly introduced non-logical symbols can be interpreted in at most one way.
NOTE An extension O_2 of an OMS O_1 is a weak definitional extension, if each model of O_1 can be expanded to at most one model of O_2.
NOTE An extension is definitional if and only if it is both weakly definitional and model-theoretically conservative.

implied extension model-theoretic conservative extension that does not introduce new non-logical symbols.
NOTE A conservative extension O_2 of an OMS O_1 is an implied extension, if and only if the signature of O_2 is the signature of O_1. O_2 is an implied extension of O_1 if and only if the model class of O_2 is the model class of O_1.
NOTE Each implied extension is also a definitional extension but not vice versa.

consistency property of an OMS expressing that it has a non-trivial set of logical consequences in the sense that not every sentence follows from the OMS.
NOTE The opposite is inconsistency.
NOTE In many (but not all) logics, consistency of an OMS equivalently can be defined as *false* not being a logical consequence of the OMS. However, this does not work for logics that e.g. do not feature a *false*. See [**?**] for a more detailed discussion.

satisfiability property of an OMS expressing that it is satisfied by least one model.
NOTE The opposite is unsatisfiability.
NOTE Any satisfiable OMS is consistent, but there are some logics where the converse does not hold.

model finding process that finds models of an OMS and thus proves it to be satisfiable.

model finder tool that implements model finding.

module structured OMS expressing a subOMS that conservatively extends to the whole OMS.
NOTE The conservative extension can be either model-theoretic or consequence-theoretic; without qualification, the consequence-theoretic version is used.

module extraction activity of obtaining from an OMS concrete modules to be used for a particular purpose (e.g. to contain a particular sub-signature of the original OMS).
NOTE Cited and slightly adapted from [?].
NOTE The goal of module extraction is "decomposing an OMS into smaller, more manageable modules with appropriate dependencies" [?].
EXAMPLE Assume one extracts a module about white wines from an OWL DL ontology about wines of any kind. That module would contain the declaration of the non-logical symbol "white wine", all declarations of non-logical symbols related to "white wine", and all sentences about all of these non-logical symbols.

approximant logically implied theory (possibly after suitable translation) of an OMS in a smaller signature or a sublanguage.

maximum approximant best possible approximant of an OMS in a smaller signature or a sublanguage.
NOTE Technically, a maximum approximant is a uniform interpolant, see [?].

approximation structured OMS that expresses a maximum approximant.

filtering structured OMS expressing the verbatim removal of symbols or axioms from an OMS.
NOTE If a symbol is removed, all axioms containing that symbol are removed, too.

closed world assumption assumption that facts whose status is unknown are true.

closure; circumscription structured OMS expressing a variant of the closed world assumption by restricting the models to those that are minimal, maximal, free or cofree (with respect to the local environment).
NOTE Symbols from the local environment are assumed to have a fixed interpretation. Only the symbols newly declared in the closure are forced to have minimal or maximal interpretation.
NOTE DOL supports four different forms of closure: minimization and maximization as well as freeness and cofreeness (explained below).

NOTE See [?], [?].

minimization form of closure that restricts the models to those that are minimal (with respect to the local environment).

maximization form of closure that restricts the models to those that are maximal (with respect to the local environment).

freeness special type of closure, restriction of models to those that are free (with respect to the local environment).
NOTE In first-order logic (and similar logics), freeness means minimal interpretation of predicates and minimal equality among data values. Freeness can be used for the specification of inductive datatypes like numbers, lists, trees, bags etc. In order to specify e.g. lists over some elements, the specification of the elements should be in the local environment.

cofreeness special type of closure, restriction of models to those that are cofree (with respect to the local environment).
NOTE In first-order logic (and similar logics), cofreeness means maximal interpretation of predicates and equality being observable equivalence. Cofreeness can be used for the specification of coinductive datatypes like infinite lists and streams.

combination structured OMS expressing the aggregation of all the OMS in an OMS network, where non-logical symbols are shared according to the OMS mappings in the OMS network.
EXAMPLE Consider an ontology involving a concept `Person`, and another one involving `Human being`, and an alignment that relates these two concepts. In the combination of the ontologies along the alignment, there is only one concept, representing both `Person` and `Human being`.

sharing property of OMS symbols being mapped to the same symbol when computing a combination of an OMS network.
NOTE Sharing is always relative to a given OMS network that relates different OMS. That is, two given OMS symbols can share with respect to one OMS network, and not share with respect to some other OMS network.

4.4 Mappings Between OMS

OMS mapping; link relationship between two OMS.

symbol map item pair of symbols of two OMS, indicating how a symbol from the first OMS is mapped by a signature morphism to a symbol of the second OMS
NOTE A symbol map item is given as $s_1 \mapsto s_2$, where s_1 is a symbol from the *source* OMS and s_2 is a symbol from the *target* of the OMS mapping.
NOTE Similar to correspondence.

signature morphism mapping between two signatures, preserving the structure of the source signature within the target signature
NOTE Each signature morphism has an underlying list of symbol map items. Conversely, a list of symbol map items may induce a signature morphism (but generally, it does not so in all cases).

interpretation; view; refinement OMS mapping that postulates a specialization relation between two OMS along a morphism between their signatures.
NOTE An interpretation typically leads to proof obligations, i.e. one has to prove that translations of axioms of the source OMS along the morphism accompanying the interpretation are theorems in the target OMS.

equivalence OMS mapping ensuring that two OMS share the same definable concepts.
NOTE Two OMS are equivalent if they have a common definitional extension. The OMS may be written in different OMS languages.

interface signature signature mediating between an OMS and a module of that OMS in the sense that it contains those non-logical symbols that the sentences of the module and the sentences of the OMS have in common.
NOTE Adapted from [?].

module relation OMS mapping stating that one OMS is a module of the other one.

alignment an OMS mapping expressing a collection of semantic relations between entities of the two OMS.
NOTE Alignments consist of correspondences, each of which may have a confidence value. If all confidence values are 1, the alignment can be given a formal, logic-based semantics.

correspondence relationship between an non-logical symbol e_1 from an OMS O_1 and an non-logical symbol e_2 from an OMS O_2, or between an non-logical symbol e_1 from O_1 and a term t_2 formed from non-logical symbols from O_2, with a confidence level.
NOTE A correspondence is given as a quadruple $(e_1, R, \left\{ \begin{array}{c} e_2 \\ t_2 \end{array} \right\}, c)$, where R denotes the type of relationship that is asserted to hold between the two non-logical symbols/terms, and $0 \leq c \leq 1$ is a confidence value. R and c may be omitted: When R is omitted, it defaults to the equivalence relation, unless another default relation has been explicitly specified; when c is omitted, it defaults to 1.
NOTE A confidence value of 1 does not imply logical equivalence (cf. [?] for a worked-out example).
NOTE Not all OMS languages implement logical equivalence. For example, OWL does not implement logical equivalence in general, but separately implements equivalence relations restricted to individuals (*owl:sameAs*), classes (*owl:equivalentClass*) and properties (*owl:equivalentProperty*).
NOTE A default correspondence can be used for stating that all symbols with the same name in the two ontologies are equivalent. A correspondence block can be used for specifying the relation and/or the confidence value of several single correspondences in the same time: if the relation or the confidence value of a single correspondence in a block are missing, they will be replaced with those specified as parameters of the block.

matching algorithmic procedure that generates an alignment for two given OMS.
NOTE For both matching and alignment, see [?, ?].

matcher tool that implements matching.

OMS network; distributed OMS; hyperontology graph with OMS as nodes and OMS mappings as edges, showing how the OMS are interlinked.
NOTE In [?], a distinction between focused and distributed heterogeneous specifications is made. In the terminology of this standard, this is the distinction between OMS and OMS networks.
NOTE An OMS network is a diagram of OMS in the sense of category theory, but different from a diagram in the sense of model-driven engineering.
NOTE The links between the nodes of a network can be given using interpretations or alignments. Imports between the nodes of a network are automatically included in the network. By including an interpretation or an alignment in a distributed OMS, the involved nodes are automatically included.
EXAMPLE Consider two ontologies and an interpretation between them. In the network of the interpretation there are two nodes, one for each ontology, and one edge from the source ontology to the target ontology of the interpretation.

category a collection of objects with suitable morphisms between them.
NOTE In this standard, objects of a category are usually signatures or OMS, and morphisms are signature morphisms, or OMS mappings. In principle, no assumption about the exact nature of objects and morphisms is made.
NOTE The morphisms determine which part of the structure of the objects is relevant, i.e. preserved by morphisms. Hence, objects can be seen as "sets with structure", and morphisms as "structure-preserving maps". However note that not all categories can be obtained in this way.

4.5 Features of OMS Languages

mapping; function relation between a set of inputs and a set of permissible outputs with the property that each input is related to exactly one output.
NOTE In some cases is a morphism, as in category theory.

language mapping mapping between languages
NOTE This is a general term, subsuming OMS language translation, logic translation and logic reduction below.

OMS language translation mapping from constructs in the source OMS language to their equivalents in the target OMS language.
NOTE An OMS language translation shall satisfy the property that the result of a translation is a well-formed text in the target language.

graph set of objects (nodes) that are connected by links (edges).

OMS language graph graph of OMS languages and OMS language translations, typically used in a heterogeneous environment.
NOTE In an OMS language graph, some of the OMS language translations can be marked to be default translations.

default translation specially marked OMS language translation or logic translation that will be used whenever a translation is needed and no explicit translation is given.

heterogeneous environment environment for the expression of homogeneous and heterogeneous OMS, comprising a logic graph, an OMS language graph and supports relations.
NOTE The support relations specify which language supports which logics and which serializations, and which language translation supports which logic translation or reduction. Moreover, each language has a default logic and a default serialization.
NOTE Although in principle, there can be many heterogeneous environments, for ensuring interoperability, there will be a global heterogeneous environment (maintained in some registry), with subenvironments for specific purposes.

sublanguage syntactically specified subset of a given language, consisting of a subset of its meta classes (abstract syntax) and terminal and nonterminal symbols and grammar rules (concrete syntax).

language aspect a set of language constructs of a given language, not necessarily forming a sublanguage.

logical language aspect the (unique) language aspect of an OMS language that enables the expression of non-logical symbols and sentences in a logic.

structuring language aspect the (unique) language aspect of an OMS language that covers structured OMS as well as the relations of basic OMS and structured OMS to each other, including, but not limited to imports, OMS mappings, conservative extensions, and the handling of prefixes for CURIEs.

annotation language aspect the (unique) language aspect of an OMS language that enables the expression of comments and annotations.

profile (syntactic) sublanguage of an OMS language interpreted according to a particular logic that targets specific applications or reasoning methods.
EXAMPLE Profiles of OWL 2 include OWL 2 EL, OWL 2 QL, OWL 2 RL, OWL 2 DL, and OWL 2 Full.
NOTE Profiles typically correspond to sublogics.
NOTE Profiles can have different logics, even with completely different semantics, e.g. OWL 2 DL versus OWL 2 Full.
NOTE The logic needs to support the language.

4.6 Logic

logic specification of valid reasoning that comprises signatures (user defined vocabularies), models (interpretations of these), sentences (constraints on models), and a satisfaction relation between models and sentences.
NOTE Most OMS languages have an underlying logic.
EXAMPLE $\mathcal{SROIQ}(D)$ is the logic underlying OWL 2 DL.
NOTE See annex **??** for the organization of the relation between OMS languages and their logics and serializations.

supports relation relation between OMS languages and logics expressing the logical language aspect of the former, namely that the constructs of the former lead to a logical theory in the latter.
NOTE There is also a supports relation between OMS languages and serializations, and one betwwen language translations and logic translations/reductions.

exact logical expressivity strengthening of the supports relation between languages and logics, stating that the language has exactly the expressivity of the logic.

institution metaframework mathematically formalizing the notion of a logic in terms of notions of signature, model, sentence and satisfaction.
NOTE In order to support a broad range of OMS languages and enable interoperability between them, the DOL semantics has to abstract from the differences of the logic language aspects of OMS languages. Institutions provide a formal framework that enables this abstraction.
NOTE The notion of institution uses category theory for providing formal interfaces for the notions of signature, model, sentence and satisfaction.

NOTE See Definition 2 in clause 10 for a formal definition.

plain mapping logic mapping that maps signatures to signatures and therefore does not use infrastructure axioms.

translation mapping between languages or logics representing all structure, in contrast to reduction.

reduction mapping between languages or logics forgetting parts of the structure, projection to a smaller language or logic.

logic translation translation of a source logic into a target logic (mapping signatures, sentences and models) that keeps or encodes the logical content of OMS.

logic reduction reduction of a source logic onto a (usually less expressive) target logic (mapping signatures, sentences and models) that forgets those parts of the logical structure not fitting the target logic.

simple theoroidal logic translation translation that maps signatures of the source logic to theories (i.e. signatures and sets of sentences, playing the role of infrastructure axioms) of the target logic.
EXAMPLE The translation from OWL to multi-sorted first-order logic translates each OWL built-in type to its first-order axiomatization as a datatype.

infrastructure axiom axiom that is used in the target of a logic translation in order to encode a signature of the source logic
EXAMPLE The translation from OWL to multi-sorted first-order logic translates each OWL built-in type to its first-order axiomatization as a datatype. These first order axioms are infrastructure axioms.

sublogic a logic that is a syntactic restriction of another logic, inheriting its semantics.

logic graph graph of logics, logic translations and logic reductions, typically used in a heterogeneous environment.
NOTE In a logic graph, some of the logic translations and reductions can be marked to be default translations.

homogeneous OMS OMS whose parts are all formulated in one and the same logic.
NOTE The opposite of heterogeneous OMS.

heterogeneous OMS OMS whose parts are formulated in different logics.
NOTE The opposite of homogeneous OMS.
EXAMPLE See section ??.

faithful mapping logic mapping that preserves and reflects logical consequence.

model-expansive mapping logic mapping that has a surjective model translation (ensuring faithfulness of the mapping).

model-bijective mapping logic mapping that has a bijective mapping of models.

exact mapping logic mapping that is compatible with certain DOL structuring constructs, e.g. union, OMS translation and OMS reduction.

weakly exact mapping logic mapping that is weakly compatible with certain DOL structuring constructs, e.g. union, OMS translation and OMS reduction.

embedding logic mapping that embeds the source into the target logic, using components that are embeddings and (in the case of model translations) isomorphism.

sublogic logic embedding that is "syntactic" in the sense that signature and sentence translations are inclusions.

adjointness relation between a logic translation and a logic reduction, expressing that they share their sentence and model translations, while the signature translations are adjoint to each other (in the sense of category theory).

4.7 Interoperability

logically interoperable property of structured OMS, which may be written in different OMS languages supporting different logics, of being usable jointly in a coherent way (via suitable OMS language translations), such that the notions of their overall consistency and logical entailment have a precise logical semantics.
NOTE Within ISO 19763 and ISO 20943, metamodel interoperability is equivalent to the existence of mapping, which are statements that the domains represented by two MDE models intersect and there is a need to register details of the correspondence between the structures in the MDE models that semantically represent this overlap. Within these standards, an MDE model is a representation of some aspect of a domain of interest using a normative modeling facility and modeling constructs.

The notion of logical interoperability is distinct from the notion of interoperability used in ISO/IEC 2381-1 Information Technology Vocabulary – Part 1: Fundamental Terms, which is restricted to the capability to communicate, execute programs, or transfer data among various hardware or software entities in a manner that requires the user to have little or no knowledge of the unique characteristics of those entities.

OMS interoperability relation among OMS (via OMS alignments) which are logically interoperable.

4.8 Abstract and Concrete Syntax

concrete syntax ; serialization specific syntactic encoding of a given OMS language or of DOL.
NOTE Serializations serve as standard formats for exchanging DOL documents and OMS between human beings and tools.
EXAMPLE OWL uses the term "serialization"; the following are standard OWL serializations: OWL functional-style syntax, OWL/XML, OWL Manchester syntax, plus any standard serialization of RDF (e.g. RDF/XML, Turtle, ...). However, W3C specifications only require an RDF/XML implementation for OWL2 tools.
EXAMPLE Common Logic uses the term "dialect"; the following are standard Common Logic dialects: Common Logic Interchange Format (CLIF), Conceptual Graph Interchange Format (CGIF), eXtended Common Logic Markup Language (XCL).

document result of serializing an OMS or DOL library using a given serialization.

standoff markup way of providing annotations to subjects in external resources, without embedding them into the original resource (here: OMS).

abstract syntax; parse tree term language for representing documents in a machine-processable way
NOTE An abstract syntax can be specified as a MOF metamodel. Then abstract abstract syntax documents can be represented as XMI documents.

4.9 Semantics

formalization precise mathematical entity capturing an informal or semi-formal entity.

formal semantics assignment of a mathematical meaning to a language by mapping the abstract syntax to suitable semantic domains.
NOTE A formal semantics is a formalization of the meaning of a language.

semantic domain mathematically-defined set of values that can represent the intended meanings of language constructs.

semantic rule specification of a mapping from expressions for some meta class in the abstract syntax to a semantic domain.

global environment mapping from identifiers (IRIs) to values in semantics domains representing the global knowledge about OMS.

4.10 Semantic Web

resource something that can be globally identified.
NOTE IETF/RFC 3986:2005, Section 1.1 deliberately defines a resource as "in a general sense [...] whatever might be identified by [an IRI]". The original source refers to URIs, but DOL uses the compatible IRI standard IETF/RFC 3987:2005 for identification.
EXAMPLE Familiar examples include an electronic document, an image, a source of information with a consistent purpose (e.g., "today's weather report for Los Angeles"), a service (e.g., an HTTP-to-SMS gateway), and a collection of other resources. A resource is not necessarily accessible via the Internet; e.g., human beings, corporations, and bound books in a library can also be resources. Likewise, abstract concepts can be resources, such as the operators and operands of a mathematical equation, the types of a relationship (e.g., "parent" or "employee"), or numeric values (e.g., zero, one, and infinity). See IETF/RFC 3986:2005, Section 1.1

element (of an OMS) any resource in an OMS (e.g. a non-logical symbol, a sentence, a correspondence, the OMS itself, ...) or a named set of such resources.

linked data structured data that is published on the Web in a machine-processable way, according to principles specified in W3C/TR REC-ldp-20150226:2015[1].
NOTE The linked data principles (adapted from W3C/TR REC-ldp-20150226:2015 and its paraphrase at [?]) are the following:

1. Use IRIs as names for things.
2. Use HTTP IRIs so that these things can be referred to and looked up ("dereferenced") by people and user agents. (I.e., the IRI is treated as a URL (uniform resource locator).)
3. Provide useful machine-processable (plus optionally human-readable) information about the thing when its IRI is dereferenced, using standard formats.
4. Include links to other, related IRIs in the exposed data to improve discovery of other related information on the Web.

NOTE RDF, serialized as RDF/XML [?], is the most common format for publishing linked data. However, its usage is not mandatory.
NOTE Using HTTP content negotiation [?] it is possible to serve representations in different formats from the same URL.

4.11 OMS Annotation and Documentation

annotation additional information without a logical semantics that is attached to an element of an OMS.
NOTE Formally, an annotation is given as a (subject, predicate, object) triple as defined by SOURCE: W3C/TR REC-rdf11-concepts:2014, Section 3.1. The subject of an annotation is an element of an OMS. The predicate is an RDF property defined in an external OMS and describes in what way the annotation object is related to the annotation subject.
NOTE According to the preceding note, it is possible to interpret annotations under an RDF semantics. "Without a logical semantics" in this definition means that annotations to an OMS are not considered sentences of that OMS.

OMS documentation set of all annotations to an OMS, plus any other documents and explanatory comments generated during or after development or deployment of the OMS.
NOTE Adapted from [?].

[1] The original source is widely accepted but not formally a standard [?].

5 Symbols

As listed below, these symbols and abbreviations are generally for the main clauses of the OMG Specification. Some annexes may introduce their own symbols and abbreviations which will be grouped together within that annex.

CASL	Common Algebraic Specification Language, specified by the Common Framework Initiative
CGIF	Conceptual Graph Interchange Format
CL	Common Logic
CLIF	Common Logic Interchange Format
CURIE	Compact URI expression
DDL	Distributed description logic
DOL	Distributed Ontology, Modeling and Specification Language
DTV	Date-Time Vocabulary
EBNF	Extended Backus-Naur Form
E-connections	a modular ontology language (closely related to DDL)
F-logic	frame logic, an object-oriented ontology language
IRI	Internationalized Resource Identifier
MOF	Meta-Object Facility
OCL	Object Constraint Language
OWL 2	Web Ontology Language (W3C), version 2: family of knowledge representation languages for authoring ontologies
OWL 2 DL	description logic profile of OWL 2
OWL 2 EL	a sub-Boolean profile of OWL 2 (used often e.g. in medical ontologies)
OWL 2 Full	the language that is determined by RDF graphs being interpreted using the OWL 2 RDF-Based Semantics [?]
OWL 2 QL	profile of OWL 2 designed to support fast query answering over large amounts of data
OWL 2 RL	fragment of OWL 2 designed to support rule-based reasoning
OWL/XML	XML-based serialization of the OWL 2 language
P-DL	Package-based description logic
RDF	Resource Description Framework, a graph data model
RDFS	RDF Schema
RDFa	a set of XML attributes for embedding RDF graphs into XML documents
RDF/XML	an XML serialization of the RDF data model
RIF	Rule Interchange Format
SBVR	Semantics of Business Vocabulary and Business Rules
SMOF	MOF Support for Semantic Structures
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
XMI	XML Metadata Interchange
XML	eXtensible Markup Language

6 Additional Information

(Informative)

6.1 Changes to Adopted OMG Specifications

This specification does not require or request any change to any other OMG specification.

6.2 How to Read This Specification

The initial five clauses of this specification describe the scope of the specification, determine conformance criteria, provide normative references, define terms and definitions, and introduce symbols that are used in the specification. The next three clauses are *informative*. This clause provides some background information, the next two provide a high-level summary of usage scenarios and goals (clause 7) and an overview over the design of DOL (clause 8).

Clause 9 defines the abstract syntax of DOL (*normative*) as an SMOF compliant meta model. Further, the same clause also provides a human friendly text serialization of the abstract syntax of DOL (*normative*). Annex ?? contains the abstract syntax specified using Extended Backus–Naur Form (EBNF) (*informative*).

Clause 10 defines the model-theoretic semantics of DOL on the abstract syntax, and also makes the notion of heterogeneous logical environment (providing languages, logics and translations) precise (*normative*).

Annex ?? specifies an RDF vocabulary for the terms in clause 4, and for OMS languages and translation that conform with DOL (*normative*).

Various languages are shown to conform to DOL in informative annexes: OWL2 (annex ??), Common Logic (annex ??), RDF and RDF Schema (annex ??), UML class diagrams (annex ??), TPTP (annex ??), and CASL (annex ??).

Annex ?? provides a core graph of logics and translations, covering those OMS languages whose conformance with DOL is established in the preceding annexes (*informative*). Annex ?? extends the graph presented in Annex ?? by a list of OMS language whose conformance with DOL will be established by a registry (*informative*).

Annex ?? discusses an extension of DOL by queries. This extension is needed to support query languages (e.g., SQL or SPARQL) in DOL and to enable query related constructs for OMS in other DOL conformant languages (*informative*).

Annex ?? provides of DOL texts, which provide examples for all DOL constructs, which are specified in the abstract syntax (*informative*).

Annex ?? gives an overview of available software tools for DOL. Annex ?? discusses the implementation of a linked-data compliant IRI scheme used in one of these tools (*informative*).

The bibliography contains ?? references to the literature that is cited in this document (*informative*).

6.3 Acknowledgments

6.3.1 Submitting and supporting organizations

The following OMG members are submitting this specification:

- Fraunhofer FOKUS
- MITRE
- Thematix Partners LLC

The following organizations are supporting this specification:

- Otto-von-Guericke University Magdeburg
- Athan Services

6.3.2 Participants

The following people contributed directly to the development of this specification.

- Tara Athan, Athan Services, USA
- Conrad Bock, National Institute of Standards and Technology, USA
- Mihai Codescu, Otto-von-Guericke University Magdeburg, Germany
- Daniel Couto Vale, University of Bremen, Germany
- Martin Glauer, Otto-von-Guericke University Magdeburg, Germany
- Michael Gruninger, University of Toronto, Canada
- Stephan Günther, Otto-von-Guericke University Magdeburg, Germany
- Maria Hedblom, Otto-von-Guericke University Magdeburg, Germany
- Andreas Hoffmann, Fraunhofer FOKUS, Germany
- Yazmin Angelica Ibañez, University of Bremen, Germany
- Maria Keet, University of Cape Town, South Africa
- Elisa Kendall, Thematix Partners LLC, USA
- Alexander Knapp, University of Augsburg, Germany
- Oliver Kutz, Free University of Bolzano, Italy
- Christoph Lange, University of Bonn and Fraunhofer IAIS, Germany
- Terry Longstreth, Independent Consultant, USA
- Christian Maeder, Jacobs University Bremen, Germany
- Till Mossakowski, Otto-von-Guericke University Magdeburg, Germany
- Fabian Neuhaus, Otto-von-Guericke University Magdeburg, Germany
- Leo Obrst, MITRE, USA
- Tim Reddehase, University of Bremen, Germany
- Bernd Reichel, Otto-von-Guericke University Magdeburg, Germany
- Madhura Thosar, Otto-von-Guericke University Magdeburg, Germany

7 Goals and Usage Scenarios

(Informative)

Often, engineering tasks require the use of several different OMS, which represent knowledge about a given domain or specify a given system from different perspectives or for different purposes. (E.g., a software engineer will typically use different OMS to model different aspects of a software system, including its behavior, its components, and its interactions with other systems.) Further, the OMS are often represented in different OMS languages (e.g., UML class diagrams, OWL, or Common Logic), which may differ in style, expressivity, and different computational properties.

The use of different OMS within the same context leads to several challenges in the design and deployment of OMS, which have been addressed by current research in ontological engineering, formal software specification and formal modeling:

- How is it possible to support shareability and reusability of OMS within the same domain?
- How is it possible to merge OMS in different domains, particularly in the cases in which the OMS are axiomatized in different logical languages?
- What notions of modularity play a role when only part of an OMS is being shared or reused?
- What are the relationships between versions of an OMS axiomatized in different logical languages?

To illustrate these challenges, this clause presents a set of usage scenarios that involve the use of more than one OMS. These scenarios address the areas of ontology design, formal specification, and model-driven development. In spite of their many differences, they all highlight one common theme: the use of multiple OMS leads to interoperability challenges.

The purpose of DOL is to provide a standardized representation language, which can be used to represent structured OMS and the relations between OMS as part of OMS networks in a semantically well-defined way. Thus, tools that implement DOL are able to integrate different OMS into a coherent whole, thereby enabling users of DOL to overcome the different kind of interoperability issues that are illustrated by the usage scenarios in this clause.

Most of the following subsections are illustrated with sample DOL libraries. These are always written in DOL, see the DOL Text Serialization in clause 9. Naturally, they also contain parts written in different OMS languages (e.g. OWL), the syntax of which is not described in this standard, but in other standard documents.

7.1 Use Case Onto-1: Interoperability Between OWL and FOL Ontologies

In order to achieve interoperability during ontology development it is often necessary to describe concepts in a language more expressive than OWL. Therefore, it is common practice to informally annotate OWL ontologies with FOL axioms (e.g., Keet's mereotopological ontology [?], Dolce Lite [?], BFO-OWL). OWL is used because of better tool support, FOL because of greater expressiveness. However, relegating FOL axioms to informal annotations means that these are not available for machine processing. Another example of this problem is the following: For formally representing concept schemes (including taxonomies, thesauri and classification schemes) and provenance information there are the two W3C standards SKOS (Simple Knowledge Organization System; W3C/TR REC-skos-reference:2009) and PROV, as well as [3]ISO and other domain-specific standards for metadata representation. The semantics for the SKOS and PROV languages are largely specified as OWL ontologies; however, as OWL cannot capture the full semantics, the rest is specified using some informal first-order rules. In other words, valid instance models that use SKOS or PROV may be required to satisfy both OWL and FOL axioms. When solving reasoning tasks over either SKOS or PROV ontologies, OWL reasoners are not able to consider the FOL axioms. Hence, the information contained in these axioms is lost. Note(3)

DOL allows the user to replace such informal annotations by formal axioms in a suitable ontology language. The relation between the OWL ontology and the FOL axioms is that of a heterogeneous import. In the result, both the OWL and the FOL axioms are amenable to, e.g., automated consistency checking and theorem proving. Hence, all available information can be used in the reasoning process. For example, the ontology below extends the OWL definition of `isProperPartOf` as an asymmetric relation with a first-order axiom (in Common Logic) asserting that the relation is also transitive.

[3] NOTE: CL: Did we mean something like "ISO 12345" here, i.e. some specific ISO standard that we reference by number? Terry: (re: footnote 29) Probably the family headed by ISO 11179 Information Technology – Metadata registries (MDR) (http://metadata-stds.org/) The 11179 standard is a multipart standard that includes the following parts:· ' Part 1: Framework · Part 2: Classification · Part 3: Registry metamodel and basic attributes · Part 4: Formulation of data definitions · Part 5: Naming and identification principles · Part 6: Registration Other standards in the series are ISO 19583 Concepts and usage of metadata ISO 19763-* Metamodel interoperability ISO 19773 Metadata modules ISO 24706 Metadata for technical stds ISO 24707 Common logic

```
%prefix( lang:  <http://purl.net/DOL/languages/>
                %% descriptions of languages ...
         trans: <http://purl.net/DOL/translations/> )%
                %% ... and translations

language lang:CommonLogic
ontology Parthood =
ObjectProperty: isProperPartOf
  Characteristics: Asymmetric
  SubPropertyOf: isPartOf
with translation trans:SROIQtoCL
then
  (if (and (isProperPartOf x y) (isProperPartOf y z))
       (isProperPartOf x z))
```

OWL can express transitivity, but not together with asymmetry.

7.2 Use Case Onto-2: Ontology Integration by Means of a Foundational Ontology

One major use case for ontologies in industry is to achieve interoperability and data integration. However if ontologies are developed independently and used within the same domain, the differences between the ontologies may actually impede interoperability. One strategy to avoid this problem is the use of a shared foundational ontology (e.g., DOLCE or BFO), which can be used to harmonize different domain ontologies. One challenge for this approach is that foundational ontologies typically rely on expressive ontology languages (e.g., Common Logic), while domain ontologies may be represented in languages that are optimized for performance (e.g., OWL EL). For this reason, currently the role of the foundational ontology is mainly to provide a conceptual framework that may be reused by the domain ontologies; further, watered-down versions of the foundational ontologies in OWL (like DOLCE-lite or the OWL version of BFO) are used as basis for the development of domain ontologies, be this as is, in an even less expressive version (e.g., a DOLCE-lite in OWL 2 EL), or only a relevant subset thereof (e.g., only the branch of endurants). A sample interplay between foundational and domain ontologies in various languages is depicted in Figure 8.1 below.

DOL provides the framework for integrating different domain ontologies, aligning these to foundational ontologies [**?**, **?**] and combining the aligned ontologies into a coherent integrated ontology – even across different ontology languages. Thus, DOL enables ontology developers to utilize the complete, and most expressive, foundational ontologies for ontology integration and validation purposes.

The foundational ontology (FO) repository Repository of Ontologies for MULtiple USes (ROMULUS)[1] contains alignments between a number of foundational ontologies, expressing semantic relations between the aligned entities. For this use-case three such ontologies are considered, containing spatial and temporal concepts: DOLCE[2], GFO[3] and BFO[4], and present alignments between them using DOL syntax:

```
%prefix(
          gfo:   <http://www.onto-med.de/ontologies/>
          dolce: <http://www.loa-cnr.it/ontologies/>
          bfo:   <http://www.ifomis.org/bfo/>
          lang:  <http://purl.net/DOL/languages/>

       )%

language lang:OWL

alignment DolceLite2BFO :
  dolce:DOLCE-Lite.owl
  to
  bfo:1.1 =
 endurant = IndependentContinuant,
 physical-endurant = MaterialEntity,
 physical-object = Object,   perdurant = Occurrent,
 process = Process,          quality = Quality,
 spatio-temporal-region = SpatiotemporalRegion,
 temporal-region = TemporalRegion,  space-region = SpatialRegion

alignment DolceLite2GFO :
```

[1] See http://www.thezfiles.co.za/ROMULUS/home.html
[2] See http://www.loa.istc.cnr.it/DOLCE.html
[3] See http://www.onto-med.de/ontologies/gfo/
[4] See http://www.ifomis.org/bfo/

```
  dolce:DOLCE-Lite.owl to gfo:gfo.owl =
        particular = Individual, endurant = Presential,
        physical-object = Material_object, amount-of-matter = Amount_of_substrate,
        perdurant = Occurrent,  quality = Property,
        time-interval = Chronoid, generic-dependent < necessary_for,
        part < abstract_has_part, part-of < abstract_part_of,
        proper-part  <   has_proper_part,       proper-part-of  < proper_part_of,
        generic-location < occupies,    generic-location-of < occupied_by

alignment BFO2GFO :
  bfo:1.1 to gfo:gfo.owl =
        Entity = Entity, Object = Material_object,
        ObjectBoundary  = Material_boundary, Role < Role ,
        Occurrent = Occurrent,  Process = Process, Quality = Property
        SpatialRegion   = Spatial_region, TemporalRegion = Temporal_region
```

DOL can be used to combine ontologies, while taking into account the semantic dependencies given by the alignments. In the following example the ontology Space is defined as a combination of three different ontologies (BFO, GFO, DolceLite) along three alignments.

```
ontology Space =
 combine BFO2GFO, DolceLite2GFO, DolceLite2BFO
```

7.3 Use Case Onto-3: Module Extraction From Large Ontologies

Especially in the biomedical domain, ontologies tend to become very large (e.g., SNOMED CT, FMA) with over 100000 concepts and relationships. Yet, none of these ontologies covers all aspects of a domain, and frequently provide coverage at various levels of specificity, with excessive detail in some areas that may not be required for all usage scenarios. Often, for a given knowledge representation problem in industry, only relevant knowledge from two such large reference ontologies needs to be integrated, so a comprehensive integration would be both unfeasible and unwieldy. Hence, parts (modules) of these ontologies are obtained by selecting the concepts and relationships (roles) relevant for the intended application. An integrated version will then be based on these excerpts from the original ontologies (i.e., modules). For example, the Juvenile Rheumatoid Arthritis ontology JRAO has been created using modules from the NCI thesaurus and GALEN medical ontology. (See [4]Figure 7.1) DOL supports the description of such subsets (modules) of ontologies, as well as their alignment and integration. Note(4)

```
%prefix( lang:  <http://purl.net/DOL/languages/> )%
library GalenModule
language lang:OWL
ontology myGalen =
  http://purl.bioontology.org/ontology/GALEN extract Drugs, Joints, Bodyparts
end

module myGalenIsAModule : myGalen of http://purl.bioontology.org/ontology/GALEN
  for Drugs, Joints, Bodyparts
end
```

7.4 Use Case Onto-4: Interoperability Between Closed-World Data and Open-World Metadata

Data collection has become easier and much more widespread over the years. This data has to be assigned a meaning somehow, which occurs traditionally in the form of metadata annotations. For instance, consider geographical datasets derived from satellite data and raw sensor readings. Current implementations in, e.g., ecological economics [?] require manual annotation of datasets with the information relevant for their processes. While there have been attempts to standardize such information [?], metadata for datasets of simulation results are more difficult to standardize. Moreover, it is resource-consuming to link the data to the metadata, to ensure the metadata itself is of good quality and consistent, and to actually exploit the metadata when querying the data for data analysis.

The data is usually represented in a database or RDF triple store, which work with a closed world assumption on the dataset, and are not expressive enough to incorporate the metadata 'background knowledge', such as the conditions for validity of the physical laws in the MDE model of the object of observation. These metadata require a more expressive language, such as OWL or Common Logic, which operate under an open-world semantics. However, it is unfeasible to

[4]NOTE: CL: can we please have a vector graphic here?

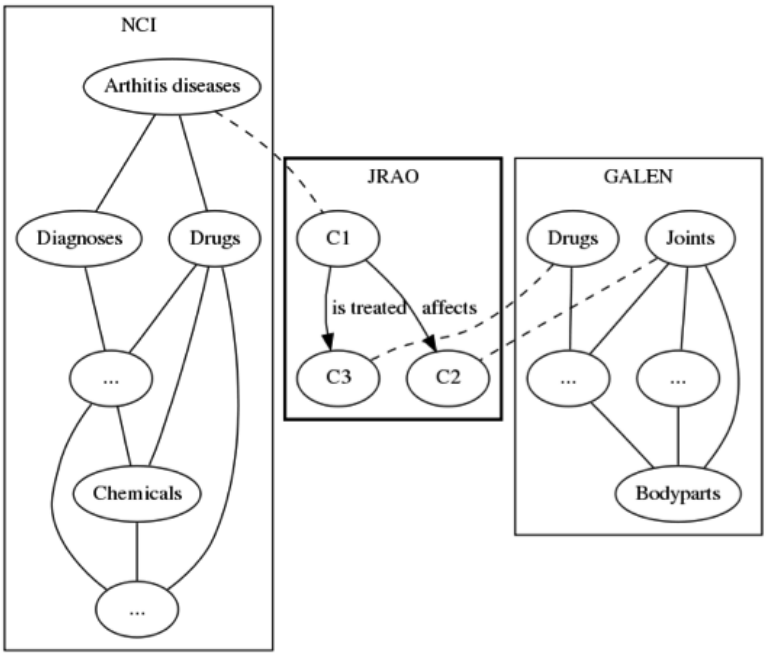


Figure 7.1: JRAO – Example for Module Extraction

translate the whole large dataset into OWL or first-order logic. To 'meet in the middle', it is possible to declare bridge rules (i.e., a mapping layer) that can link the metadata to the data. This approach can be used for intelligent data analysis that combines the data and metadata through querying the system. It enables the analysis of the data on the conceptual layer, instead of users having to learn the SQL/SPARQL query languages and how the data is stored. There are various tools and theories to realize this, which is collectively called Ontology-Based Data Access/Management, see also [?].

The languages for representing the metadata or ontology, for representing the bridge rules or mapping assertions, and for representing the data are different yet they need to be orchestrated and handled smoothly in the system, be this for data analytics for large enterprises, for formulating policies, or in silico biology in the sciences.

DOL provides the framework for expressing such bridge rules in a systematic way, maintaining these, and building tools for them.

7.5 Use Case Onto-5: Verification of Rules Translating Dublin Core Into PROV

The Dublin Core Metadata terms, which have been formalized as an RDF Schema vocabulary, developed initially by the digital library community, are less comprehensive but more widely used than PROV (cf. subclause 7.1). The rules for translating Dublin Core to the OWL subset of PROV (and, with restrictions, vice versa) are not known to yield valid instances of the PROV data model, i.e. they are not known to yield OWL ontologies consistent with respect to the OWL axioms that capture part of the PROV data model. This may disrupt systems that would like to reason about the provenance of an entity, and thus the assessment of the entity's quality, reliability or trustworthiness. The Dublin Core to PROV ontology translation[5] is expressed partly by a symbol mapping and partly by FOL rules. These FOL rules are implemented by CONSTRUCT patterns in the SPARQL RDF query language.[6] SPARQL has a formal specification of the evaluation semantics of its algebraic expressions, which is different from the model-theoretic semantics of the OWL and RDF Schema languages; nevertheless SPARQL CONSTRUCT is a popular and immediately executable syntax for expressing translation rules between ontologies in RDF-based languages in a subset of FOL. DOL not only supports the reuse of the existing Dublin Core RDF Schema and PROV OWL ontologies as modules of a distributed ontology (= OMS network), but it is also able to support the description of the FOL translation rules in a sufficiently expressive ontology language, e.g. Common Logic, and thus enable formal verification of the translation from Dublin Core to PROV.

[5] http://www.w3.org/TR/2013/NOTE-prov-dc-20130430/
[6] E.g., http://www.w3.org/TR/2013/NOTE-prov-dc-20130430/#dct-creator

7.6 Use Case Onto-6: Maintaining Different Versions of an Ontology in Languages with Different Expressivity

Often is useful to maintain different versions of an ontology within languages, which differ in their expressivity.

For example, DOLCE is a foundational ontology that has primarily been formalized in the first-order logic ontology language KIF (a predecessor of Common Logic), but also in OWL ("DOLCE Lite") [?]. This "OWLized" version was targeting use in semantic web services and domain ontology interoperability, and to provide the generic categories and relationships to aid domain ontology development. DOLCE has been used also for semantic middleware, and in OWL-formalized ontologies of different domains, including neuroimaging, computing, and ecology. Given the differences in expressivity between KIF and OWL, DOLCE Lite had to simplify certain notions. For example, the DOLCE Lite formalization of "temporary parthood" (something is part of something else at a certain point or interval in time) omits any information about the time, as OWL only supports binary predicates (a.k.a. "properties"). That leaves ambiguities for modeling a view from DOLCE Lite to the first-order DOLCE, as such a view would have to reintroduce the third (temporal) component of such predicates:

- Should a relation asserted in terms of DOLCE Lite be assumed to hold for *all* possible points/intervals in time, i.e. should it be universally quantified?
- Or should such a relation be assumed to hold for *some* points/intervals in time, i.e. should it be existentially quantified?
- Or should a concrete value for the temporal component be assumed, e.g. "0" or "now"?

DOL supports the formalization of all of these views. Given suitable consistency checking tools, DOL enables the analysis of whether any such view satisfies all further axioms that the first-order DOLCE states about temporal parthood.

7.7 Use Case Onto-7: Metadata within OMS Repositories

DOL provides a language for the metadata within OMS Repositories. For example, the Common Logic Repository (COLORE) [7] is an open repository of more than 150 ontologies as of December 2011, all formalized in Common Logic. COLORE stores metadata about its ontologies, which are represented using a custom XML schema that covers the following aspects[8], without specifying a formal semantics for them:

module provenance author, date, version, description, keyword, parent ontology[9]

axiom source provenance name, author, year[10]

direct relations maps (signature morphisms), definitional extension, conservative extension, inconsistency between ontologies, imports, relative interpretation, faithful interpretation, definable equivalence

DOL provides built-in support for a subset of the "direct relations" and specifies a formal semantics for them. In addition, it supports the implementation of the remainder of the COLORE metadata vocabulary as an ontology, reusing suitable existing metadata vocabularies such as OMV, and it supports the implementation of one or multiple Common Logic ontologies plus their annotations as one coherent DOL library.

7.8 Use Case Spec-1: Modularity of Specifications

Often specifications become so large that it is necessary to structure them in a modular way, for human readability and maintainability, and for more efficient tool support. The lack of a standard for such modular structuring hinders interoperability among different development efforts and the reuse of specifications. DOL provides a notion of structured modular specification that is equally applicable to all DOL-conforming logical languages.

Structuring pays off even for small specifications. For example, it makes structuring a simple specification of sorting lists in the following way enhances both readability and potential for re-use of specifications:

```
%prefix( lang:  <http://purl.net/DOL/languages/> )%

library Sorting

language lang:CASL
%right_assoc __::__
spec TotalOrder =
  sort Elem
  pred __<=__ : Elem * Elem
  forall x,y,z : Elem
  . x <= x                          %(reflexive)%
```

[7] http://stl.mie.utoronto.ca/colore/
[8] http://stl.mie.utoronto.ca/colore/metadata.html
[9] Note that this use of the term "module" in COLORE corresponds to the term structured OMS in this OMG Specification.
[10] Note that this may cover any sentences in the sense of this OMG Specification.

```
  . x <= z if x <= y /\ y <= z      %(transitive)%
  . x = y if x <= y /\ y <= x       %(antisymmetric)%
  . x <= y \/ y <= x                %(dichotomous)%
end

spec Nat =
  free type Nat ::= 0 | suc(Nat)
end

spec List =
  Nat
then
  sort Elem
  free type List ::= [] | __::__(Elem; List)
  op count : Elem * List -> Nat
  forall x,y : Elem; L : List
  . count(x,[]) = 0
  . count(x,x :: L) = suc(count(x,L))
  . count(x,y :: L) = count(x,L) if not x=y
end

spec Sorting =
  TotalOrder and List
then
  preds is_ordered : List;
        permutation : List * List
  vars x,y:Elem; L,L1,L2:List
  . is_ordered([])
  . is_ordered(x::[])
  . is_ordered(x::y::L) <=> x<=y /\ is_ordered(y::L)
  . permutation(L1,L2) <=> (forall x:Elem . count(x,L1) = count(x,L2))
then
  op sorter : List->List
  var L:List
  . is_ordered(sorter(L))
  . permutation(L,sorter(L))
hide is_ordered, permutation
end
```

In the last step, the structuring operation of hiding is used to restrict the specification to an export interface: predicates `is_ordered` and `permutation` are hidden, because they are only auxiliary and need not be implemented.

7.9 Use Case Spec-2: Specification Refinements

Formal software and hardware development methods are often used to ensure the correct function of systems which have safety-critical requirements or which may not be easily accessible for repair or replacement. Examples of such requirements can be found in safety-critical areas such as medical systems, or in the automotive, avionics and aerospace industries, as well as in components used by those industries such as in microprocessor design.

Typically, a requirement specification is refined into a design specification and then an implementation, often involving several intermediate steps (see, e.g. the V-model [**?**], although this does not require formal specification). There are numerous specification formalisms in use, including the OMG's SysML language; moreover, often during development, the formalism needs to be changed (e.g. from a specification to a programming language, or from a temporal logic to a state machine). For each of these formalisms, notions of refinement have been defined and implemented. However, the lack of a standardized, logically sound language and methodology for such refinement hinders interoperability among different development efforts and the reuse of refinements. DOL provides the capability to represent refinement that is equally applicable to all DOL-conforming logical languages, and that covers at least the most relevant of the industrial use cases of specification refinement.

A simple example is the refinement of the (purely declarative) sorting specification from use case in section 7.8 into a specification of a particular sorting algorithm (for simplicity, insert sort is used for demonstration):

```
spec InsertSort =
  TotalOrder and List
then
  ops insert : Elem*List -> List;
      insert_sort : List->List
  vars x,y:Elem; L:List
  . insert(x,[]) = x::[]
  . insert(x,y::L) = x::insert(y,L) when x<=y else y::insert(x,L)
  . insert_sort([]) = []
  . insert_sort(x::L) = insert(x,insert_sort(L))
```

```
 hide insert
end

%% refinement from abstract sorting to insert sort
refinement InsertSortCorrectness =
   Sorting refined via sorter |-> insert_sort to InsertSort
end
```

Note that hiding is essential here to make the signatures of both specifications compatible. If the predicates `is_ordered` and `permutation` had not been hidden in the `Sorting` specification, a refinement would not have been possible, since `InsertSort` does not implement these predicates (and it would be rather artificial to add an implementation for them).

Refinements can be composed. A simple example below illustrates this by expressing that natural numbers with addition form a monoid, and that natural numbers can be efficiently represented for implementation as lists of binary digits, together with several equivalent ways of composing these refinements.

```
spec Monoid =
 sort Elem
 ops 0 : Elem;
        __+__ : Elem * Elem -> Elem, assoc, unit 0
end

spec NatWithSuc = %mono
 free type Nat ::= 0 | suc(Nat)
 op __+__ : Nat * Nat -> Nat, unit 0
 forall x , y : Nat . x + suc(y) = suc(x + y)
 op 1:Nat = suc(0)
end

spec Nat =
  NatWithSuc hide suc
end

spec NatBin =
generated type Bin ::= 0 | 1 | __0(Bin) | __1(Bin)

ops __+__ , __++__ : Bin * Bin -> Bin
forall x, y : Bin
 .  0 0 = 0  .  0 1 = 1
 .  not  (0 = 1)  .  x 0 = y 0 => x = y .  not  (x 0 = y 1)  .  x 1 = y 1 => x = y
 .  0 + 0 = 0  .  0 ++ 0 = 1
 .  x 0 + y 0 = (x + y) 0  .  x 0 ++ y 0 = (x + y) 1
 .  x 0 + y 1 = (x + y) 1  .  x 0 ++ y 1 = (x ++ y) 0
 .  x 1 + y 0 = (x + y) 1  .  x 1 ++ y 0 = (x ++ y) 0
 .  x 1 + y 1 = (x ++ y) 0  .  x 1 ++ y 1 = (x ++ y) 1
end

refinement R1 =
 Monoid refined via Elem |-> Nat to Nat
end

refinement R2 =
 Nat refined via Nat |-> Bin to NatBin
end

refinement R3 =
 Monoid refined via Elem |-> Nat to
 Nat refined via Nat |-> Bin to NatBin
end

refinement R3' =
 Monoid refined via Elem |-> Nat to R2
end

refinement R3'' =
 Monoid refined via Elem |-> Nat to Nat then R2
end

refinement R3''' = R1 then R2
```

7.10 Use Case Model-1: Consistency Among UML Diagrams of Different Types

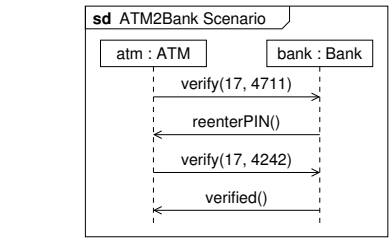
A typical UML model involves diagrams of different types. Such UML models may have intrinsic errors because diagrams of different types may specify conflicting requirements. Typical questions that arise in this context are ask for semantic consistency, e.g.,

- whether the multiplicities in a class diagram are semantically consistent with each other;
- whether the sequential composition of actions in an interaction diagram is justified by an accompanying OCL specification;
- whether cooperating state machines comply with pre-/post-conditions and invariants;
- whether the behavior prescribed in an interaction diagram is realizable by several state machines cooperating according to a composite structure diagram.

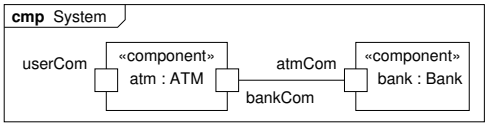
Such questions are currently hard to answer in a systematic manner. One method to answer these questions and find such errors is a check for semantic consistency. Under some restrictions, the proof of semantic consistency can be (at least partially) performed using model-checking tools like Hugo/RT [?]. Once a formal semantics for the different diagram types has been chosen (see, e.g. [?]), it is possible to use DOL to specify in which sense the diagrams need to be consistent, and check this by suitable tools.

7.10.1 The ATM Example

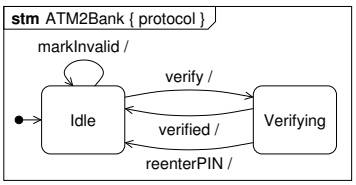
The ATM example, which illustrates model-driven development using UML, is taken from [?]. The example involves the design of a traditional automatic teller machine (ATM) connected to a bank. For simplicity, the example focuses on the ATM's processing of card and PIN entry actions. After entering the card, one has three trials for entering the correct PIN (which is checked by the bank). After three unsuccessful trials the card is kept.



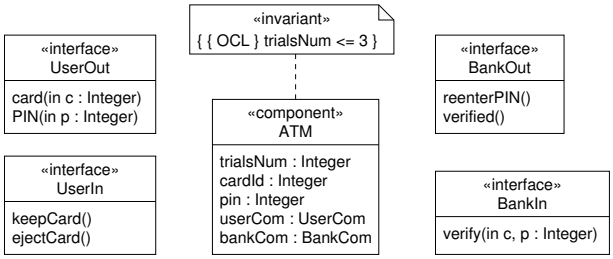
(a) Interaction



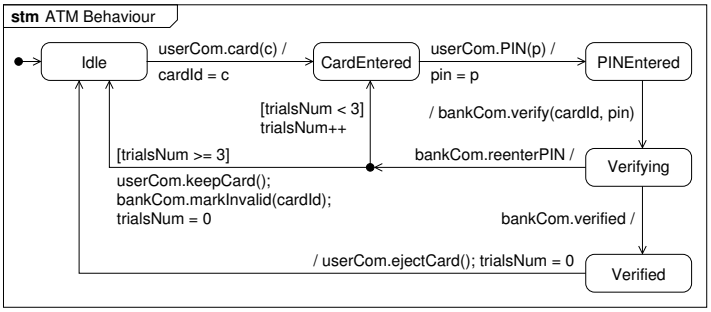
(b) Composite structure



(c) Protocol state machine



(d) Interfaces and components



(e) State machine

Figure 7.2: ATM example