

Performance Simulation Initiative

Upper-Level Ontology v.2.3 Reference Specification

Authors:

Vadim Ermolayev (ZNU),
Natalya Keberle (ZNU),
Eyck Jentasch (CDNS),
Richard Sohnius (DNS)

VCAD EMEA

Cadence Design Systems GmbH

Mozartstr. 2

D-85622 Feldkirchen

Germany

06.10.2009

Document Information

Project Information:					
Project ID:		Acronym:	PSI		
Full Title:	Performance Simulation Initiative				
Project URL:					
Project Manager:	Mr. Eyck Jentzsch	Partner:	Cadence	E-mail:	jentzsch@cadence.com

Deliverable Information:					
Type:	Documentation			Planned	Actual
Status:	Version 2.3		Date of Delivery:	--	06.10.2009
Dissemination: (to check)	Public: <input type="checkbox"/>	Consortium: <input type="checkbox"/>	Document ID:	PSI-ONTO-TR-2009-2	
URL:					

Author Information:			
Authors (Partner):	Vadim Ermolayev (ZNU), Natalya Keberle (ZNU), Richard Sohnius (CDNS)		
Responsible Author:	Mr. Vadim Ermolayev	Partner:	ZNU
		E-mail:	vadim@ermolayev.com

Document Information:	
Abstract: (for dissemination)	<p>This Specification describes PSI Upper-Level ontology v.2.3 – the upper-level part of PSI Suite of Ontologies v.2.3. Its main function is putting the components of the Suite in line with the commonly accepted metaphysical and cognitive framework of the common sense represented by several reference ontologies and providing semantic bridges to mainstream enterprise, business, and process modeling frameworks. Bridging PSI ontologies to these mainstream process knowledge representations facilitates to easier commitment of Engineering Design Domain professionals to PSI Suite of Ontologies. As many foundational ontologies PSI Upper-Level ontology has a clear cognitive orientation in the sense that it does not pretend being strictly and rigorously referential to the theories describing nature. Instead, it captures ontological categories and contexts based on human common sense reflecting socially dominant views on the Domain – characteristic at least to engineering design professionals. In contrast to foundational ontologies PSI Upper-Level ontology is not foundational in the sense that it is not a profound and a complete theory in philosophical or, more precisely, cognitivist sense. It uses the most abstract part of DOLCE ontology as foundational source for defining PSI Upper-Level ontology components. In addition to being semantic “glue” between the Suite and the outer world, the Upper-Level ontology plays an important role in the methodology of knowledge engineering in PSI. PSI Upper-Level ontology is the resource which is intensively used in the refinement and the evaluation of the PSI Suite of Ontologies.</p>
Keywords:	PSI, upper-level ontology, foundational ontology, DOLCE, engineering design, performance.
Citation:	Ermolayev, V., Jentzsch, E., Keberle, N., and Sohnius, R.: Performance Simulation Initiative. Upper-Level Ontology v.2.3. Reference Specification. Technical Report PSI-ONTO-TR-2009-2, 06.10.2009, VCAD EMEA Cadence Design Systems, GmbH, 75 p.

Reviewers:			
Name:		Partner:	
		E-mail:	
Name:		Partner:	
		E-mail:	

Revision Log				
Version/Revision	Date	Change	Submitted by	Comment
v.2.3.0	04.10.2009	Final draft (RC)	Vadim Ermolayev	
v.2.3.1	06.10.2009	Final draft (RC) after internal cross-check. Corrected several typos and inconsistencies in figures and text	Vadim Ermolayev	

Table of Contents

List of Figures.....	6
List of Tables.....	8
List of Acronyms and Abbreviations.....	9
Executive Summary.....	11
Ontologies in PSI.....	11
The Upper-Level ontology	11
Introduction	13
1 Naming Convention and Used Notations	14
1.1 Naming Convention.....	14
1.2 UML Notation	14
1.3 Transformation from UML Notation to OWL Notation.....	15
1.3.1 UML Concepts Transformed to OWL.....	15
1.3.2 UML Concept Properties Transformed to OWL	15
1.3.3 UML Binary Associations Transformed to OWL.....	16
1.3.4 UML Binary Compositions and Aggregations	17
1.3.5 UML Association Classes.....	17
1.3.6 UML Packages. Relationships between Classes of Different Packages	19
1.4 Colors in UML Diagrams	20
2 Preliminaries.....	22
2.1 Basic Assumptions and Ontological Choices	22
2.2 Overview of the PSI Modeling Approach.....	24
2.3 Vertical and Horizontal Structure of the Suite of Ontologies.....	25
2.4 The Overview of the PSI Upper-Level Ontology	25
3. Ontology Engineering Methodology	31
4 Detailed Specification of the PSI Upper-Level Ontology.....	35
4.1 DOLCE: Particular	35
4.2 DOLCE: Endurant	35
4.3 DOLCE: Perdurant	35
4.4 DOLCE: Quality	35
4.5 A Phenomenon.....	35
4.6 An Event.....	38
4.7 A Process.....	38
4.8 An Environment.....	40
4.9 An AtomicAction and a Decision	41
4.10 A Holon	42
4.11 A State	43
4.12 A Goal.....	44
4.13 A Characteristic and its Specializations.....	45

4.14 A Value and a Measure.....	47
4.15 A TimeInstant.....	48
4.16 A TimeInterval.....	48
4.17 An Object.....	49
4.18 A MaterialObject.....	50
4.19 An Agent.....	51
4.20 A MaterialArtifact.....	52
4.21 A ConsumableResource.....	53
4.22 A Tool.....	53
4.23 An ImmaterialObject.....	53
4.24 An ImmaterialArtifact.....	54
4.25 A Plan and a Strategy.....	55
4.26 A Fact.....	55
4.27 A Belief.....	56
4.28 A Rule.....	56
4.29 A Metric.....	57
4.30 A Requirement.....	57
4.31 A Pattern.....	58
4.32 A BehaviorPattern.....	58
4.33 A ProcessPattern.....	59
4.34 An ActionPattern.....	59
4.35 A StatePattern.....	59
4.36 A Policy.....	60
4.37 A Context.....	60
4.38 A Commitment.....	62
4.39 A NonConsumableResource.....	62
5 Mapping PSI Core and Extension Ontologies to Foundational Theories.....	63
5.1 The Mappings of the PSI Core Ontologies.....	63
5.2 The Mappings of the PSI Extension Ontologies.....	69
6 Future Work.....	71
7 The Index of Concepts, Key Properties, and Terms.....	72
References.....	74

List of Figures

- Fig. 1.1:** The notation for specifying relationships in UML
- Fig. 1.2:** The notation for specifying concepts (classes)
- Fig. 1.3:** The notation for specifying concept properties
- Fig. 1.4:** The notation for specifying complex datatype properties
- Fig. 1.5:** The notation for specifying binary associations
- Fig. 1.6:** The notation for specifying binary compositions and aggregations
- Fig. 1.7:** The notation for specifying UML association classes
- Fig. 1.8:** The notation for specifying relationships among classes of different UML packages
- Fig. 2.1:** The model of a Process
- Fig. 2.2:** The model of a State
- Fig. 2.3:** The models of an Object and an Agent
- Fig. 2.4:** The model of a Rule
- Fig. 2.5:** The models of a Characteristic, a Value, and a Measure
- Fig. 3.1:** The phases of ontology refinement iteration in a Shaker Modeling process
- Fig. 4.1:** The taxonomy of the PSI Upper-Level ontology
- Fig. 4.2:** The holonymy/meronymy relationships and associations in the PSI Upper-Level ontology
- Fig. 4.3:** The semantic context of the concept of a **Process**
- Fig. 4.4:** The semantic context of the concept of an **Environment**
- Fig. 4.5:** The semantic context of the concept of an **AtomicAction**
- Fig. 4.6:** The semantic context of the concept of a **Holon**
- Fig. 4.7:** The semantic context of the concept of a **State**
- Fig. 4.8:** The semantic context of the concept of a **Characteristic**
- Fig. 4.9:** The semantic context of the concept of an **Object**
- Fig. 4.10:** The semantic context of the concept of a **MaterialObject**
- Fig. 4.11:** The semantic context of the concept of an **Agent**
- Fig. 4.12:** The semantic context of the concept of an **ImmaterialObject**
- Fig. 4.13:** The semantic context of the concept of a **Rule**
- Fig. 4.14:** The semantic context of the concept of a **Context**
- Fig. 5.1:** The taxonomy of the PSI Core Time ontology and the mappings of its concepts to DOLCE and SUMO through the PSI Upper-Level ontology
- Fig. 5.2:** The taxonomy of the PSI Core Environment, Event, and Happening ontology and the mappings of its concepts to DOLCE and SUMO+WordNet through the PSI Upper-Level ontology
- Fig. 5.3:** The taxonomy of the PSI Core Actor ontology and the mappings of its concepts to DOLCE and SUMO+WordNet through the PSI Upper-Level ontology
- Fig. 5.4:** The taxonomy of the PSI Core Organization ontology and the mappings of its concepts to DOLCE and SUMO+WordNet through the PSI Upper-Level ontology
- Fig. 5.5:** The taxonomy of the PSI Core Process Pattern ontology and the mappings of its concepts to DOLCE and SUMO+WordNet through the PSI Upper-Level ontology
- Fig. 5.6:** The taxonomy of the PSI Core Process ontology and the mappings of its concepts to DOLCE and SUMO+WordNet through the PSI Upper-Level ontology

- Fig. 5.7:** The taxonomy of the PSI Core Design Artifact ontology and the mappings of its concepts to DOLCE and SUMO+WordNet through the PSI Upper-Level ontology
- Fig. 5.8:** The taxonomy of the PSI Core DASpecific ontology and the mappings of its concepts to DOLCE and SUMO+WordNet through the PSI Upper-Level ontology
- Fig. 5.9:** The joint taxonomy of the PSI Core ontologies and the PSI Upper-Level ontology
- Fig. 5.10:** The joint taxonomy of the Mediators of the IMS Resource, Tool ontologies and the used concepts of the IMS Library ontology. The mappings of these concepts to DOLCE and SUMO+WordNet through the PSI Upper-Level ontology.
- Fig. 5.11:** The taxonomy of the Software Tool Evaluation ontology. The mappings of its concepts to DOLCE and SUMO+WordNet through the PSI Upper-Level ontology.
- Fig. 5.12:** The taxonomy of the Ability ontology. The mappings of its concepts to DOLCE and SUMO+WordNet through the PSI Upper-Level ontology.
- Fig. 5.13:** The taxonomy of the Generic Negotiation ontology. The mappings of its concepts to DOLCE and SUMO+WordNet through the PSI Upper-Level ontology.

List of Tables

Table 1.1: Structural coloring in PSI Suite of Ontologies

Table 1.2: Change coloring in PSI Suite of Ontologies

Table 2.1: Foundational ontologies and their ontological choices [22]

List of Acronyms and Abbreviations

AM	ABox Migration
B	Bridging
BFO	Basic Formal Ontology
CA	Commonsense Alignment
CDNS, Cadence	Cadence Design Systems, GmbH
DA	Design Artifact
DASpecific	Design Artifact Complexity and Quality ontology
DEDP	Dynamic Engineering Design Process
DOLCE	Descriptive Ontology for Linguistic and Cognitive Engineering
DOR	Domain Ontology Refinement
EMEA	Europe, Middle East & Africa
E2H	Environment, Event, and Happening ontology
EO	Enterprise Ontology
Fig.	Figure
FSU-metheval	FSU-metheval – Friedrich-Schiller-University Jena: Department of methodology and evaluation research
IMS	Fraunhofer Institute of Microelectronic Circuits and Systems
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
MFR	Modeling Framework Refinement
OCHRE	Object-Centered High-level Reference Ontology
OntoClean	The methodology for formal evaluation of the taxonomical structure of an ontology
OpenCYC	A freely available version of the CYC knowledgebase owned by CYCorp
OWL	Web Ontology Language
PPRJ	Internal Protégé format for representing ontologies in Protégé formal syntax. File name extension of such files
PRODUKTIV+	Reference System for Measuring Design Productivity of Nanoelectronic Systems
PSI	Performance Simulation Initiative
PSI-ULO	PSI Upper-Level ontology
PSL	Process Specification Language
RDF(S)	Resource Description Framework (Schema)
SUMO	Suggested Upper Merged Ontology
TOVE	Toronto Virtual Enterprise Ontology
TR	Taxonomy Refinement
UE	User Evaluation
UML	Unified Modeling Language
UOR	Upper Ontology Refinement
v	Version

VCAD	Virtual CAD
WordNet	WordNet Linguistic Ontology (Princeton University)
Wrt	With respect to
XML	eXtensible Markup Language
ZNU	Zaporozhye National University

Executive Summary

PSI project deals with the development of the methodology and the software toolset for assessing and optimizing the performance of engineering design processes in microelectronic and integrated circuit design. The design technology in this domain is well defined. However, many factors make engineering design processes very stochastic, non-deterministic, structurally ramified, time-bound – in a phrase, loosely defined and highly dynamic. The examples of such factors are: the human factor, the innovative character, the pace of technology change, the peculiarities of the market and the customer requirements, etc. In difference to many alternative and competitive approaches to assessing the performance of engineering design PSI goes deeper in the details of a design system and a process and uses simulation for modeling performance in its real dynamics and with sufficiently detailed picture to make the assessment grounded. To emphasise the focus on dynamics the processes of engineering design in PSI are qualified as Dynamic Engineering Design Processes (DEDP). Simulation approach also allows playing “what-if” games to model the unpredictable character of the real business world in the domain.

Ontologies in PSI

Due to the omnipresence of the complicating and de-linearizing factors mentioned above the finely grained and complete knowledge of a process is the central intellectual asset which allows the PSI methodology be convincing and produce grounded assessments. This knowledge is formalized using the Suite of PSI Ontologies that forms a logically sound descriptive theory of the domain. Indeed, if someone intends to imagine an arbitrary process of designing something, most certainly he or she will think in terms of: a goal – the state of affairs to be reached; an action that may bring the process closer to its goal; an object to apply actions to; a designer who acts and applies actions to objects; an instrument to be used by an actor for executing actions; and an environment where the process occurs. All these are modeled in dynamics – depending on time and on events which manifest the changes in a design system that is the environment of a DEDP. The structure of the PSI Suite of Ontologies reflects this approach. It comprises eight Core ontologies: the Time ontology; the Environment, Event, and Happening ontology; the Actor ontology; the Organization ontology; the Process Pattern ontology; the Process ontology; the Design Artifact ontology; and the Design Artifact Complexity and Quality ontology. The “corolla” of this Core is formed by the Extension ontologies. Some of the Extensions have been developed in PSI: The Software Tool Evaluation ontology, the Ability ontology, the Generic Negotiation ontology. The others are the results of the accomplished PRODUKTIV+¹ project: the Resource ontology, the Library ontology, the Tool ontology. For the ontologies adopted from PRODUKTIV+ we provide ontology mediators that allow us deriving the ongoing development from the knowledge artifacts that have already been accomplished and can not be changed.

PSI ontologies are used: (i) to formally describe the Domain of Discourse – to model Design Systems and DEDPs; (ii) to provide soundly defined unified (standardized) lexicon for Cadence and its customers; (iii) to structure the PSI knowledgebase used in performance assessment experiments; (iv) to derive the specifications of the software tools under development. The major use of the PSI ontologies is the knowledge representation and schema for the assessment of industrial engineering design processes using PSI approach. Consequently the goal of the assessment process is to gradually accumulate the fine-grained knowledge about a design system and performed DEDPs in the phases of Measurement, Simulation, and Evaluation. Each phase adds new pieces of knowledge to the collection acquired at the previous step. This knowledge is formalized using PSI ontologies.

The Upper-Level ontology

PSI Upper-Level ontology is the upper-level part of PSI Suite of Ontologies. Its main function is putting the components of the Suite in line with the commonly accepted metaphysical and cognitive framework of the common sense represented by several reference ontologies like SUMO [11], DOLCE [14] and highly reputable linguistic resources like WordNet [12]. One more objective of introducing the upper-level of the Suite is providing semantic bridges to mainstream enterprise, business, and process modeling frameworks like the Enterprise Ontology [8], Toronto Virtual Enterprise Ontology [9], Process Specification Language [10]. Bridging PSI ontologies to these mainstream

¹ PRODUKTIV+ (Referenzsystem zur Messung der Produktivität beim Entwurf nanoelektronischer Systeme) is the accomplished R&D project funded by the German Bundesministerium für Bildung und Forschung (BMBF). Core project partners are: Advanced Micro Devices, Inc. (<http://www.amd.com/>), Robert Bosch GmbH (<http://www.bosch.de/>), Cadence Design Systems GmbH (<http://www.cadence.com/>), Infineon Technologies AG (<http://www.infineon.com/>).

theories of process knowledge representation facilitates to easier commitment of Engineering Design Domain professionals to PSI Suite of Ontologies.

In difference to the mentioned enterprise, business, and process modeling frameworks, which are to a certain extent Domain independent or model manufacturing Domain, PSI Upper-Level ontology defines an upper-level theory for the Domain of Engineering Design processes and environments.

As many foundational ontologies PSI Upper-Level ontology has a clear cognitive orientation in the sense that it does not pretend being strictly and rigorously referential to the theories describing nature. Instead, it captures ontological categories and contexts based on human common sense reflecting socially dominant views on the Domain – characteristic at least to engineering design professionals. As such, the categories introduced in PSI Upper-Level ontology are not related to the intrinsic nature of the world but are rather thought of as “cognitive artifacts ultimately depending on human perception, cultural imprints and social conventions” (c.f. [14]). Therefore, these categories assist in making already formed conceptualizations of the PSI Suite of Ontologies explicit and referenced by the common sense. PSI Upper-Level ontology also plays an integration and harmonization role of a foundational ontology [23] because it represents a rather domain-independent upper-level descriptive theory based on formal principles for harmonizing and integrating the underlying domain dependent modules with other relevant ontologies.

In addition to being semantic “glue” between the Suite and the outer world of knowledge representation, the Meta-Ontology plays an important role in the methodology of knowledge engineering in PSI. PSI Upper-Level ontology is the resource which is intensively used in the refinement and the evaluation of PSI Core Ontologies as described in Section 3 of this specification.

Introduction

This Specification provides the upper-level description framework for modeling design processes and environments in microelectronic engineering design. The framework is presented in the form of PSI Upper-Level ontology v.2.2. This ontology is provided in the form of UML diagrams² and is coded in the Semantic Web Ontology Language (OWL)³.

PSI Upper-Level ontology is designed and implemented in a way to obey the principle of the minimal ontological commitment [1]. This principle states that an ontology should require the minimal ontological commitment from its users sufficient to support the intended usage in knowledge sharing activities. “An ontology should make as few claims as possible about the world being modeled, allowing the parties committed to the ontology freedom to specialize and instantiate the ontology as needed. Since ontological commitment is based on *consistent use of vocabulary*, ontological commitment can be minimized by specifying the weakest theory (allowing the most models) and defining only those terms that are essential to the communication of knowledge consistent with that theory.” One of the consequences of obeying this principle is that PSI Upper-Level ontology is the ontology of particulars [14] – the entities that do not have instances. We do not need instances on the upper level of knowledge representation. In difference to the Upper-Level ontology, the concepts of the PSI Core and Extension ontologies have instances that constitute the PSI knowledge base of Design Systems and DEDPs.

The structure of the Specification is as follows:

- Section 1 describes the naming convention, the notations and the transformation rules between the notations used in the design of the PSI Upper-Level ontology.
- Section 2 overviews the modeling approach used as the basics for the design of the ontology and briefly describes our approach to modeling engineering design processes and engineering design environments.
- Section 3 describes the methodology used in the development and evaluation of the consecutive versions of PSI Theoretical Framework, PSI Upper-Level ontology, and the PSI Suite of Ontologies.
- Section 4 provides the detailed specifications of the concepts of PSI Upper-Level ontology and their semantic contexts.
- Section 5 reports on the mappings of PSI Core ontologies to DOLCE as well as to SUMO+WordNet. SUMO and WordNet are used as the reference sources of the common sense knowledge representation. The reasons for choosing these particular reference sources are given in Section 2.
- Section 6 lists the problems which are identified by PSI team but not yet resolved in v.2.3. This list is the agenda for the future work in the development of PSI Upper-Level ontology.
- Section 7 is the index of major terms, concepts and properties provided for cross-reference purposes

This fixed version of the specification together with the sub-ordinate files (UML diagrams, .owl, and .pprj files) is stored to the **v2.3/PSI-ULO** subfolder of the PSI ontologies repository. The root directory for the repository is **PSI-Ontologies**. Each of the versions of the Suite of Ontologies is stored to its own branch. The components stored for the Meta-Ontology version are:

- Specification version fix – stored to the version root folder **v2.3/PSI-ULO**
- UML diagrams of the version fix – stored to the **UML** subfolder of the root
- OWL and Protégé PPRJ files of the version fix – stored to the **RDFS-OWL** subfolder of the root

Working notes, discussion documents and the drafts of UML, OWL, and Pprj files are stored in the subfolders of the **Discussion** subfolder:

- All working notes and discussion records – stored to the **Working-Docs** sub-folder
- The drafts of this specification – stored to the **Spec-Drafts** folder
- The drafts of UML diagrams – stored to the **UML-Drafts** folder
- The drafts of OWL and Protégé PPRJ files – stored to the **RDFS-OWL-Drafts** folder

² The UML diagrams, OWL and PPRJ codes of PSI Upper-Level ontology v.2.3 can be downloaded from PSI documents repository at <https://psi.vcad-vlab.net/edit/documents/PSI-Ontologies/v2.3/PSI-ULO/UML/> and <https://psi.vcad-vlab.net/edit/documents/PSI-Ontologies/v2.3/PSI-ULO/RDFS-OWL/> respectively.

³ Web Ontology Language (OWL). OWL Reference: <http://www.w3.org/TR/owl-ref/>

1 Naming Convention and Used Notations

This section describes the naming convention, the notations and the transformation between the notations used in the design of the PSI ontologies and, in particular, the PSI Upper-Level ontology v.2.3.

1.1 Naming Convention

The names of the concepts, relationships, properties are derived from the words of English language which best outline their semantics. For example, suppose we invent the name of the concept representing an artifact produced in a design process. Then, the following combination of English words seems to be appropriate for deriving the name: design, artifact. The name of the concept is derived of this set of the key words by the concatenation of either the words or their meaningful abbreviations: DesignArtifact. In some cases the delimiters are used – please see the notations.

It has been agreed to use the following name formats for the components of the ontologies.

Concept (class) names: the so-called “CaMeI” notation with the first capital letter

If the name of a concept is derived from one English word – the name has the only (the first) capital letter. For example, Interface. If the name of a concept is derived from several (>1) English words – the name has capital letters at the beginning of each part taken from a separate word. For example, DesignArtifact (derived from design, and artifact), PhysicalVR (derived from physical, verification, and runset).

Relationship and property names: the so-called “caMeI” notation with the first lowercase letter. Examples: name, valueRange, componentOf, ableToPerform.

1.2 UML Notation

UML class diagrams of the ontologies are presented using UML 1.4. To disambiguate the way for specifying the multiplicities in relationships the following convention has been adopted.

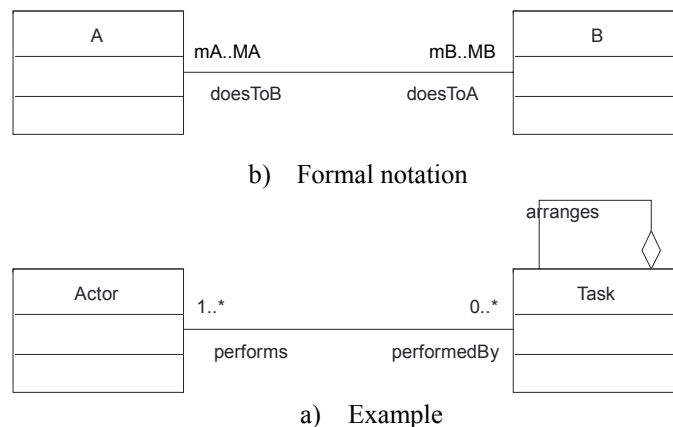


Fig. 1.1: The notation for specifying relationships in UML

For any type of the relationship **R** connecting concepts **A** and **B** the notation for **R** is given in Fig. 1.1a, where:

- **R** is the name of the relationship (rarely used in PSI ontologies – for specifying the names of the binary relationships of a concept to itself, e.g., **A** to **A**)

The example of such a “symmetric” relationship is: **Task** – **arranges** – **Task**.

- **doesToA**, **doesToB** are the names of the relationship ends specified from **A** to **B** (**doesToB**) and from **B** to **A** (**doesToA**) respectively.

Such “asymmetric” relationships are used quite often. An example is: **Actor performs (0..*) Task** and **Task performedBy (1) Actor** (Fig. 1.1b).

- (**mA**, **MA**), (**mB**, **MB**) are the multiplicities denoting the (instance) cardinality of the relationship [2]:

mA specifies how many (minimally) instances of **A** may be related to the ONE SPECIFIC instance of **B**, and

MA specifies how many (maximally) instances of **A** may be related to the ONE SPECIFIC instance of **B**.

mB specifies how many (minimally) instances of **B** may be related to the ONE SPECIFIC instance of **A**, and

MB specifies how many (maximally) instances of **B** may be related to the ONE SPECIFIC instance of **A**.

For example, in the performance relationship (Fig. 1.1b) **one and only one** instance of an **Actor** may perform one specific instance of an **AssociatedTask**, and **zero or more** instances of an **AssociatedTask** may be performed by one specific instance of an **Actor**.

1.3 Transformation from UML Notation to OWL Notation

There are common rules for transformation of UML diagrams into OWL descriptions:

1.3.1 UML Concepts Transformed to OWL

UML concepts (classes) are presented in OWL as OWL Classes (Fig. 1.2).

E.g. UML class **Role** will have the following representation in OWL:

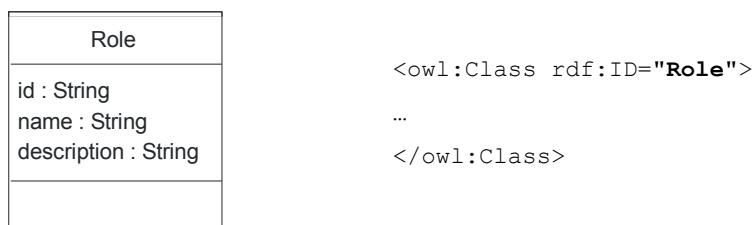


Fig. 1.2: The notation for specifying concepts (classes).

1.3.2 UML Concept Properties Transformed to OWL

To transform UML datatype properties into OWL, we use longer names of properties (Fig. 1.3), e.g. **Role-description** instead of simple **description**, in order to distinguish the property “**description**” of class **Role** from all other “**description**” properties, defined for other UML/OWL classes [3].

Datatype properties, belonging to a UML class, are presented as OWL DatatypeProperties.

E.g., datatype property **description** (String), belonging to the class **Role**, will have the following OWL representation.

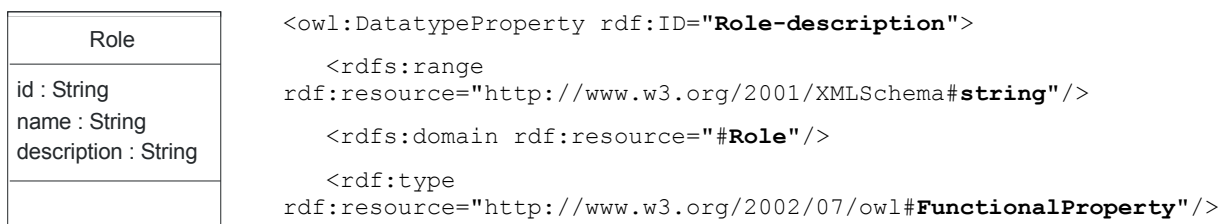


Fig. 1.3: The notation for specifying concept properties.

Note, that each individual of OWL Class **Role** will have one and only one description. This is achieved with explicit use of OWL construct “**FunctionalProperty**”.

There is only one exception when datatype property of a UML class is presented as OWL ObjectProperty. This is the situation, when a datatype property has values not from basic UML types (string, integer, double, boolean etc), but from another UML Class, e.g. TimeInstant (package Time in v.2.3, Fig. 1.4). The idea behind this situation is that sometimes it is necessary to restrict possible values of a datatype property, e.g. to allow these values to have complex structure.

E.g. datatype property **beginning** (TimeInstant), belonging to the class **Project**, will have the following OWL representation. Note, that this property has long name **Project-beginning-TimeInstant**, as other OWL ObjectProperties.

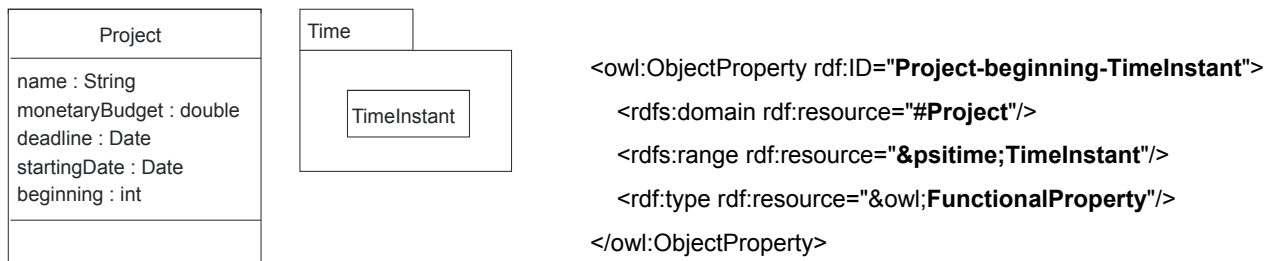


Fig. 1.4: The notation for specifying complex datatype properties.

1.3.3 UML Binary Associations Transformed to OWL

To transform UML binary associations (Fig.1.5) to OWL, we also use longer names of association ends, e.g. **Actor-performs-Task** instead of simple **performs**, in order to distinguish the association “**performs**” of the binary association between **Actor** and **Task** from all other “**performs**” associations, defined for other UML/OWL classes [3].



Fig. 1.5: The notation for specifying binary associations.

Binary associations between UML classes are presented as OWL ObjectProperties.

For example, binary association between UML classes **Actor** and **Task** will be presented as the following **two** OWL object properties:

```

<owl:ObjectProperty rdf:ID="Actor-performs-Task">
  <rdfs:domain rdf:resource="#Actor"/>
  <rdfs:range rdf:resource="#Task"/>
  <owl:inverseOf>
    <owl:ObjectProperty rdf:about="#Task-performedBy-Actor"/>
  </owl:inverseOf>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="Task-performedBy-Actor">
  <rdfs:domain rdf:resource="#Task "/>
  <rdfs:range rdf:resource="#Actor"/>
  <owl:inverseOf>
    <owl:ObjectProperty rdf:about="# Actor-performs-Task"/>
  </owl:inverseOf>
</owl:ObjectProperty>

```

Multiplicities of binary association ends are defined with additional restrictions, posed over OWL classes, participating in the binary association.

For example, the fact that each instance of an **Task** should be related to **one or more** instances of an **Actor** via the object property **Task-performedBy-Actor**, and **nothing except for Actor** instances can be at the opposite end of this object property, is presented as the following OWL code:

```

<owl:Class rdf:about="#Task">
  <rdfs:subClassOf>
    <owl:Restriction>

```



```

    <owl:minCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:minCardinality>
    <owl:onProperty>
        <owl:ObjectProperty rdf:ID="Task-performedBy-Actor"/>
    </owl:onProperty>
</owl:Restriction>
</rdfs:subClassOf>
</rdfs:subClassOf>
<owl:Restriction>
    <owl:onProperty>
        <owl:ObjectProperty rdf:about="#Task-performedBy-Actor"/>
    </owl:onProperty>
    <owl:allValuesFrom rdf:resource=" #Actor"/>
</owl:Restriction>
</rdfs:subClassOf>
</owl:Class>

```

1.3.4 UML Binary Compositions and Aggregations

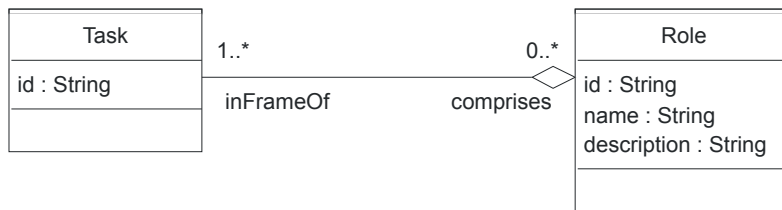


Fig. 1.6: The notation for specifying binary compositions and aggregations.

OWL does not distinguish between associations, compositions and aggregations (Fig.1.6). All necessary information can be defined using proper multiplicities [4]

1.3.5 UML Association Classes

To transform UML association classes (Fig. 1.7) to OWL, we introduce these association classes as OWL classes, and define additional binary associations.

Corresponding OWL code:

Illustrate the relationships between newly introduced OWL class **Ability**, and existing class **Actor**:

```

<owl:Class rdf:about="#Ability">
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty>
                <owl:ObjectProperty rdf:ID="Ability-ableToPerform-Actor"/>
            </owl:onProperty>
            <owl:allValuesFrom rdf:resource="#Actor"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>

```

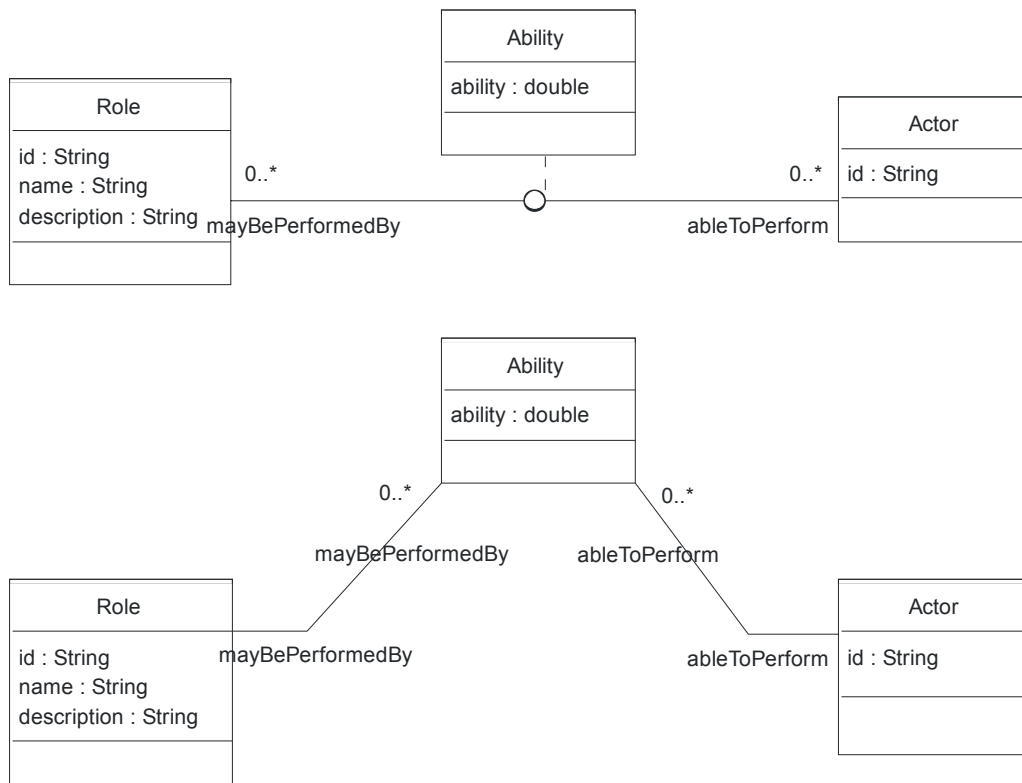


Fig. 1.7: The notation for specifying UML association classes.

```

</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:cardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
    <owl:onProperty>
      <owl:ObjectProperty rdf:about="#Ability-ableToPerform-Actor"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
For OWL class Actor:
<owl:Class rdf:ID="Actor">
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:allValuesFrom>
      <owl:Class rdf:ID="Ability"/>
    </owl:allValuesFrom>
    <owl:onProperty>
      <owl:ObjectProperty rdf:ID="Actor-ableToPerform-Ability"/>
    </owl:onProperty>

```

```

    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

```

As we may see the core name of the relationship – **ableToPerform** – remains the same both for class **Ability** and class **Actor**.

1.3.6 UML Packages. Relationships between Classes of Different Packages

UML package groups all classes describing the part of the Universe of Discourse in one package.

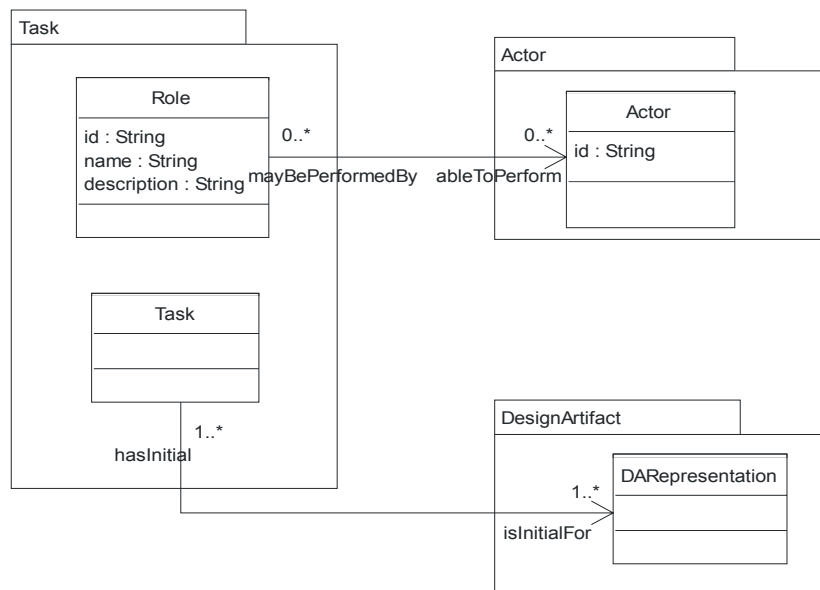


Fig. 1.8: The notation for specifying relationships among classes of different UML packages.

OWL supports the similar technology – a **namespace** – for grouping all related OWL classes (or, more generally, XML/RDF/RDFS constructs) in one module.

All classes and properties, defined in some namespace, are reachable from other namespaces through the `<owl:import>` construct. For example, if UML package **Task-Activity** defines the relationships to the classes in UML packages **Design Artifact** and **Actor** (Fig. 9), and there are OWL ontologies **DA.owl** and **Actor.owl**, then the OWL description of the **Task-Activity** ontology will contain the following statements:

```

<owl:Ontology rdf:about="">
  <owl:imports rdf:resource="URI-for-DA-ontology/da"/>
  <owl:imports rdf:resource="URI-for-Actor-ontology/actor"/>
</owl:Ontology>

```

The convention adopted in the PSI Suite of Ontologies v.2.3 specifies that the arrow notation is used for explicitly specifying which of the interrelated ontologies have to import the other one. As pictured in Fig. 1.8 the ontology that defines the relationship to the concept of the target ontology (the end of the relationship marked by the arrow) imports the target ontology. For the example in Fig. 9 the **Task** ontology imports the **DesignArtifact** and the **Actor** ontologies.

Let's show the definition of OWL classes and their object properties, in case when one object property relates OWL classes from different user-defined namespaces. For example, UML class **Role** (UML package **Task**) has relationship to the other UML class, **Actor**, from the other UML package **Actor**.

The following OWL code describes the OWL class **Role** (belongs to **Task** ontology file), and its relationship to **Actor** (**Actor** belongs to **Actor** ontology file):

```

<owl:Class rdf:ID="Role">
  <rdfs:subClassOf>
    <owl:Restriction>

```

```

<owl:allValuesFrom>
  <rdf:Description rdf:about="URI-for-Actor-ontology#Actor">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:allValuesFrom rdf:resource="#Role"/>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="Actor-ableToPerform-Role"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
  </rdf:Description>
</owl:allValuesFrom>
<owl:onProperty>
  <owl:ObjectProperty rdf:ID="Role-mayBePerformedBy-Actor"/>
</owl:onProperty>
</owl:Restriction>
</rdfs:subClassOf>

```

The following OWL code (belonging also to the **Task-Activity** ontology code) describes the *opposite* direction of the relationship, from **Actor** to **Role**:

```

<owl:ObjectProperty rdf:about="#Actor-ableToPerform-Role">
  <rdfs:range rdf:resource="#Role"/>
  <owl:inverseOf>
    <owl:ObjectProperty rdf:about="#Role-mayBePerformedBy-Actor"/>
  </owl:inverseOf>
  <rdfs:domain rdf:resource="URI-for-Actor-ontology#Actor"/>
</owl:ObjectProperty>







```

1.4 Colors in UML Diagrams

There are two reasons to use colors in the UML diagrams of this Specification. The first reason is to highlight that the element of a diagram belongs to the certain structural component of the Suite: The Core, the Extension, the Upper-Level ontology – please refer to Section 3 for the explanation of the structure. For the extensions different colors are used as well to identify the lead partner responsible for its development. The second reason is to identify the changes associated with the element on a diagram (a package or a class).

Structural coloring refers to the UML packages which represent the ontologies of the Suite as shown in Table 1.1.




Table 1.1: Structural coloring in PSI Suite of Ontologies

	The package (ontology module) belongs to the PSI Core ontology
	The package (ontology module) belongs to the PSI Extension ontology developed in PSI (Cadence)
	The package (ontology module) belongs to the Extension ontology developed by IMS
	The package (ontology module) belongs to the PSI Extension ontology developed by FSU-metheval
	The package represents the language or the ontology which is a publicly available.
	The package (ontology module) is the PSI Upper-Level ontology

Change coloring refers to the UML classes which represent the concepts in ontologies as shown in Table 1.2.

Table 1.2:

Change coloring in PSI Suite of Ontologies

	The class represents a concept which has not been changed in the current revision compared to previous one
	The class represents a concept which has been changed in the current revision compared to previous one
	The class represents a new concept or a new package that has been introduced in the current revision of the described ontology

2 Preliminaries

This section presents the role of PSI Upper-Level ontology and describes how it shapes out the modular structure and the semantics of the Core ontologies of the PSI Suite. First, the assumptions and general ontological choices used in the development of PSI Upper-Level ontology are presented. Secondly, the place of PSI Upper-Level ontology in PSI knowledge pyramid is shown and explained. Finally, a high-level overview of PSI Upper-Level ontology is given.

The role of foundational ontologies is to serve as a starting point for building new ontologies, to provide a reference point for easy and rigorous comparisons among different ontological approaches, and to create a foundational framework for analyzing, harmonizing and integrating existing ontologies and metadata standards. They are conceptualizations that contain specifications of domain independent concepts and relations based on formal principles derived from linguistics, philosophy, and mathematics.

2.1 Basic Assumptions and Ontological Choices

PSI Upper-Level ontology is the upper-level part of the PSI Suite of Ontologies [25, 30]. Its main function is putting the components of the Suite in line with the commonly accepted metaphysical and cognitive framework of the common sense represented by several reference ontologies like Suggested Upper Merged Ontology (SUMO) [11], Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) [14] and highly reputable linguistic resources like WordNet Linguistic Ontology (WordNet) [12]. One more objective of introducing the meta-level of the Suite is providing semantic bridges to mainstream enterprise, business, and process modeling frameworks like the Enterprise Ontology (EO) [8], Toronto Virtual Enterprise Ontology (TOVE) [9], Process Specification Language (PSL) [10].

In difference to the mentioned enterprise, business, and process modeling frameworks, which are to a certain extent Domain independent (TOVE, PSL) or model manufacturing Domain (EO), PSI Upper-Level ontology defines an upper-level theory for the Domain of Engineering Design processes and environments.

As many foundational ontologies PSI Upper-Level ontology has a clear cognitive orientation in the sense that it does not pretend being strictly and rigorously referential to the theories describing nature. Instead, it captures ontological categories and contexts based on human common sense reflecting socially dominant views on the Domain – characteristic at least to engineering design professionals. As such, the categories introduced in PSI Upper-Level ontology are not related to the intrinsic nature of the world but are rather thought of as “cognitive artifacts ultimately depending on human perception, cultural imprints and social conventions” (c.f. [14]). Therefore, these categories assist in making already formed conceptualizations of the PSI Suite of Ontologies explicit and referenced by the common sense. PSI Upper-Level ontology also plays an integration and harmonization role of a foundational ontology [23] because it represents a rather domain-independent descriptive theory based on formal principles for harmonizing and integrating the underlying domain dependent modules with other relevant ontologies.

In contrast to foundational ontologies PSI Upper-Level ontology is not foundational in the sense that it is not a profound and a complete theory in philosophical or, more precisely, cognitivist sense. For example, PSI Upper-Level ontology does not deal with many problems characteristic for foundational theories like: differences between abstract and concrete objects, particulars and universals; spatio-temporal co-localization of things; mereological axiomatization, etc. It also does not provide rich axiomatic sets for rigorously describing the semantics of the contained entities. Instead, other highly reputable foundational ontologies are used as reference sources for defining PSI Upper-Level ontology components. The mappings of these components to those reference sources are explicitly specified.

Choosing the most appropriate reference foundational ontologies among possible candidates is not an easy task because it requires ontological commitment to the chosen ontologies and their ontological choices. Typical ontological choices (also called meta-criteria) are:

- Descriptivism vs. Revisionarism
- Multiplicativism vs. Reductionism
- Possibilism vs. Actualism
- Endurantism vs. Perdurantism

A good comparative analysis of several well known foundational ontologies and their ontological choices has been undertaken in SmartWeb project [22]. Five most promising candidates among approximately a dozen available worldwide has been analysed: Basic Formal Ontology (BFO) , DOLCE, Object-Centered High-level Reference Ontology (OCHRE) , OpenCYC, and SUMO. The results are given in Table 2.1.

Table 2.1: Foundational ontologies and their ontological choices [22].

Requirement Alternative	BFO	DOLCE	OCHRE	OpenCYC	SUMO
Descriptivism	-	+	-	+	+
Multiplicativism	-	+	unclear	unclear	+
Actualism	+	-	-	unclear	unclear
Perdurantism	+	+	-	unclear	+

Legend: + – the ontology supports the ontological choice; - – the ontology does not support the ontological choice; unclear – it is not clear if the ontology supports the ontological choice.

Typical ontological choices in line with modeling requirements of PSI project are discussed below. Base on this discussion we make our choice of reference foundational ontologies for the design of PSI Upper-Level ontology.

Descriptivism vs. Revisionarism. A descriptive ontology aims at describing the ontological assumptions based on the surface structure of natural language and human common sense. For example, a descriptive ontology distinguishes between physical and abstract objects based on the human common sense perception of these categories. It is common to consider that a physical object is a category of things which have tangible physical properties, can be sensed, are extended in space and time. On the contrary, an abstract object does not possess the abovementioned properties. A revisionary ontology is committed to capture the intrinsic nature of the world. As a consequence, such a commitment may impose that only entities extended in space and time exist.

Though we refrain from modeling abstract things in PSI as much as possible⁴, we still have to model immaterial things which are not made of matter, do not possess spatial properties, etc. Therefore, revisionarism would have been a wrong choice for PSI. PSI Upper-Level ontology is a descriptive ontology and has to be based on a descriptive foundational ontology like DOLCE, OpenCYC, or SUMO.

Multiplicativism vs. Reductionism. A multiplicative ontology allows different entities to be co-localized in the same space-time. The difference of these entities means that they have different essential properties. For example, a silicon wafer of a chip (a material object) and a definite amount of silicon this wafer is made of (an amount of matter) are co-localized in space-time for the whole life of this particular chip. A reductionist ontology postulates that each space-time location contains at most one object. Differences in essential properties are regarded as being linked to different points of view from which one can look at the same spatio-temporal entity. Reductionist approach therefore extracts all essential properties different from spatio-temporal ones from entities and places them to the views on these entities.

In PSI it is considered that an entity possesses all its essential properties and the views on an entity may reveal different subsets of these properties depending on the point of view. For example, an agent may be (i) a model of one physical person – a designer; (ii) a model of a group of designers working on one design project – a development team. PSI Upper-Level ontology should therefore be a multiplicative ontology – like DOLCE or SUMO.

Possibilism vs. Actualism. An actualistic ontology postulates that everything that exists is actual. Things that are not actual and, therefore, do not exist may be withdrawn from consideration. Different forms of possibilism are based on different ways of the denial of this postulate. For example our beliefs, which are hypotheses based on incomplete, partial knowledge about the world, are very often roughly equally believed possible alternatives. Considering such alternatives is characteristic to human common sense and cognition. Committing to possibilism means being able to represent possibilia – possible alternative entities in a domain corresponding to different modalities in different possible worlds. Possibilism is particularly useful in reasoning about future courses of processes and about actions [17].

PSI Upper-Level ontology has to be capable of modeling possibilia. For example, a design process depending on the future events in its environment may take one of the possible alternative courses. These alternative courses should all be considered and analysed for choosing the best possible one to follow. Hence, we have to commit to possibilism of a foundational ontology like DOLCE or OCHRE.

Endurantism vs. Perdurantism. A fundamental ontological choice is the commitment to a way of modeling changes of things in time. Endurantism (also called 3D paradigm) postulates that all things do not change in time in the sense that all the proper parts of an entity (a whole) are present in this whole at any moment of the existence of this whole. Differently to that, perdurantism (also called 4D paradigm) assumes that entities may have different proper parts at different moments of their existence – meaning that entities have both spatial and temporal parts.

PSI Upper-Level ontology needs to model both endurants and perdurants. Indeed, many of the concepts characteristic to engineering design always contain all of their proper parts, but many other of them are composed of temporally

⁴ As one may find out in Section 4 all concepts of PSI Upper-Level ontology are not abstract.

different parts – like phases in a design process. Therefore, a reference foundational ontology for PSI Upper-Level ontology should be based on 4D paradigm, comprising 3D as a particular case. Such ontologies are BFO, DOLCE, and SUMO.

The requirements analysed above reveal that only DOLCE commits to all the ontological choices required by PSI. SUMO does all except possibilism. This is why we fully commit to the foundational framework of the upper taxonomical level of DOLCE in our design of PSI Upper-Level ontology. We also use SUMO extended by WordNet as a target for mapping the concepts of PSI Upper-Level ontology and PSI Core Ontologies because SUMO+WordNet is probably the most prominent linguistic resource describing the semantics of human common sense.

2.2 Overview of the PSI Modeling Approach

This section very briefly presents the main postulates and assumptions of our approach to modeling Dynamic Engineering Design Processes and the Environments these processes flow through. This approach is presented in more detail in PSI Theoretical Framework v.2.3 [17].

In PSI an engineering design process is understood as a goal-directed process of transforming the representations of a design artifact, which is considered the goal of such a process. We use a state-based model for representing engineering design process and denote process states by characteristic sets of design artifact representations. The model of a state in an engineering design process allows making states sensitive to static as well as dynamically changing requirements. Transformation actions are associated with these states and the classification of these actions is provided. An engineering design process is further on described as a problem solving process. Therefore, we associate a decision taking procedure with each process state. The function of this decision taking is choosing the course of the process in the state space. This dynamically developed course of an engineering design process is called a transformation path. We are interested in creative processes only. A creative process is a process of finding a transformation path through the possible set of the states of the environment bringing up the state in which the goal, stimulating the beginning of the process, is achieved.

The key point of PSI modeling approach is stressing the social character of a design process and the autonomy and pro-activity of the members of design teams. It is considered that a proper balance of rationality (self-interest) and socially oriented behaviour (benevolence) should be dynamically reached in design teams for reaching the highest possible degree of coherence in their collaborative actions. It is outlined that such sub-optimal coherence in a team may be only obtained through reaching agreements in different kinds of multi-issue negotiations. These agreements form the basics of the decisions to be taken about the choice of the course of action in each engineering design process state.

The formalism for representing actions is devised through answering important questions like: (i) Is an Action simple or compound? (ii) Does an Action transit the process to a different state? (iii) What are the changes applied to the Design Artifact? (iv) What are the dependencies among actions? (v) Do actions depend on the environment of the process? etc. A distinguishing feature of PSI approach is allowing different possible views on the same action by different actors. An action may be treated as an atomic action or a compound one – a stateful process. Further on, different types of actions may have different relationships with the environment, process states, requirements to design artifact representations; the influences caused by external or internal events perceived as happenings by the members of the design team. Actions may have different kinds of dependencies to other actions. By introducing a formal model of these dependencies the mechanism for modeling concurrency and co-execution of actions is provided.

One of the central pieces of PSI modeling framework is the model of an Environment. Processes of engineering design do not occur in vacuum, but flow through environments. Designers, who develop design artifacts in engineering design processes, are also not fully autonomous entities. They are influenced by their environments. An environment is a temporal aggregation of different kinds of objects which surround the process or the object in question. It is assumed that a process or an object surrounded by its environment is situated in the environment. By “surrounding” several distinct things are meant: (i) an object situated in the environment may be changed by the objects constituting this environment; (ii) a process is always situated in one or more environments because it connects the states of its environment(s); (iii) an environment may be changed by the objects of this environment or by the objects external to this environment (situated in other environments) in events.

An environment influences an object or a process situated in it. For example, the same person situated in different environments may have different properties: play different roles, have different beliefs, have different availabilities, execute different atomic actions, etc.

The environment(s), through which a process flow, are transformed because of the execution of atomic actions in the process. It should be noticed that the components of the environment are changed in actions (not by actions) by those agents who execute these actions. The kinds of these changes are: (i) resources are changed because of consumption; (ii) agents are changed because they execute actions – become more skilled, become more or less available, receive utility (as incentives) or lose utility (in penalties); (iii) artifacts change because they are developed in the process.

2.3 Vertical and Horizontal Structure of the Suite of Ontologies

PSI Upper-Level ontology, as already mentioned before, is the upper level theory for the Core and Extension ontologies of the PSI Suite of Ontologies. PSI Suite is an interlinked modular library of ontologies describing the Domain of Dynamic Engineering Design Processes and Design Systems in Microelectronics and Integrated Circuits. PSI Upper-Level ontology is more domain-independent. It formalises an upper-level theory of stateful creative dynamic processes, pro-active agents, and objects situated in nested dynamic environments based on the formal representation of time and events. This theory may be used as the upper-level representation for domain ontologies in different application domains having common features. PSI Upper-Level ontology is designed as a semantic bridge formalizing the mappings of PSI Domain ontologies to abstract ontological foundations and common sense. It is also used as the semantic “glue” for bridging PSI domain theory with other theories widely accepted in the domains where processes, states, and participating objects are the major entities. These mappings and semantic bridges are supposed to ease the commitment of potential users to PSI Suite. PSI Upper-Level ontology is also used as a “proxy” for different kinds of evaluation of PSI ontologies – please refer to Section 3 for more details.

2.4 The Overview of the PSI Upper-Level Ontology

The overview of the ontology is given by the presentation of the semantic contexts of its key concepts: a Process, an AtomicAction, an Environment, a State, an Object, an Agent, a Rule, a Characteristic, a Value, and a Measure.

A Process (Fig. 2.1) is a specialization of an Event⁵ that is stateful and possesses pro-active character. A Process has its Environment – the part of the world which may influence the course of the Process or may be changed in the course of the Process. A Process is pro-actively directed by the Agent who manages it. Pro-activeness of the Agent is understood in the sense that the Agent pursues a particular Goal in the managed Process.

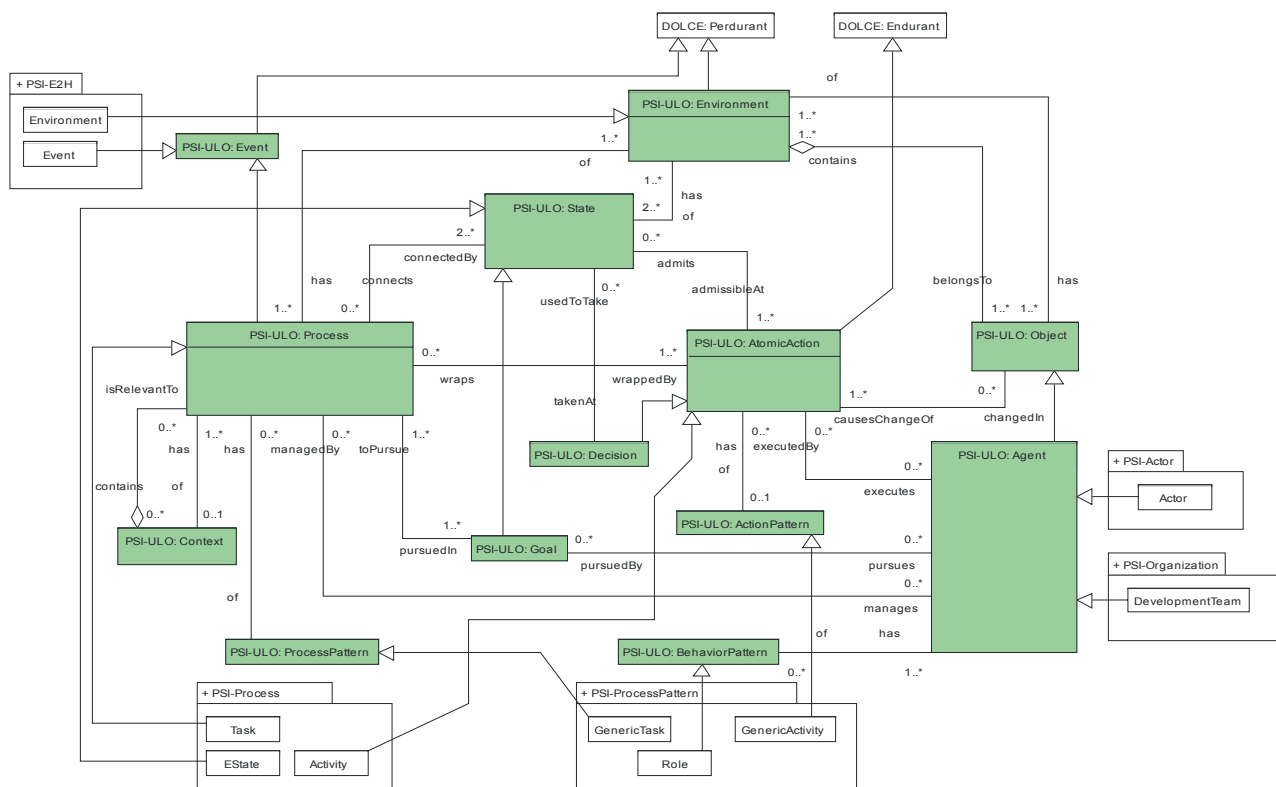


Fig. 2.1: The semantic context of the concept of a Process.

This Goal is the State of the Environment, which the Agent desires to reach. It should also be mentioned that the change in the Environment is not produced by the Process, but by the entities who act in this process – those Agents who execute AtomicActions wrapped by the Process. In general, it is considered that changes may only be applied by Agents

⁵ Detailed discussion of PSI approach to modelling Environments, Events, and event Happenings subjectively perceived by Agents is presented in [18] and [31].

through execution of Atomic Actions. For example, it is wrong to say that a multimedia controller layout has been designed by the process of logical design. In fact the appearance of the layout for the multimedia controller in a certain state of the Environment (the measurable change in the Environment) has been achieved by the team of Agents who executed a particular sequence of AtomicActions. By that the Agents applied the sequence of particular changes to the Environment and guided the environment through the sequence of States towards the Goal. Processes in an engineering Environment can not connect any arbitrary State to any other arbitrary State because it is senseless with respect to the technology or the methodology. Some sequences of States may therefore be withdrawn from the engineering design routine and some other sequences of States may be suggested or prescribed by an industrial standard or a company policy. These prescriptions are ProcessPatterns. From the point of view of an Agent a Process could be relevant to a particular working Context as well as it may have its Context.

The model of an Environment is further refined in the PSI Core Environment, Event, and Happening (E2H) ontology [30, Section 5.2]. ProcessPatterns are further elaborated as GenericTasks, BehaviourPatterns – as Roles, ActionPatterns – as GenericActivities in the PSI Core ProcessPattern ontology [30, Section 5.5]. The concepts of a Process and an AtomicAction are further refined in the PSI Core Process ontology [30, Section 5.5.2].

Any Process, as a pro-active stateful manifestation of a change in the Environment, is managed by an Agent for reaching the State of affairs in which the constituents of the Environment possess the properties partially or fully matching the Goal of that Agent. It is considered that a Process has reached its target State if such a state of affairs is reached. Otherwise the Process fails to reach its target State.

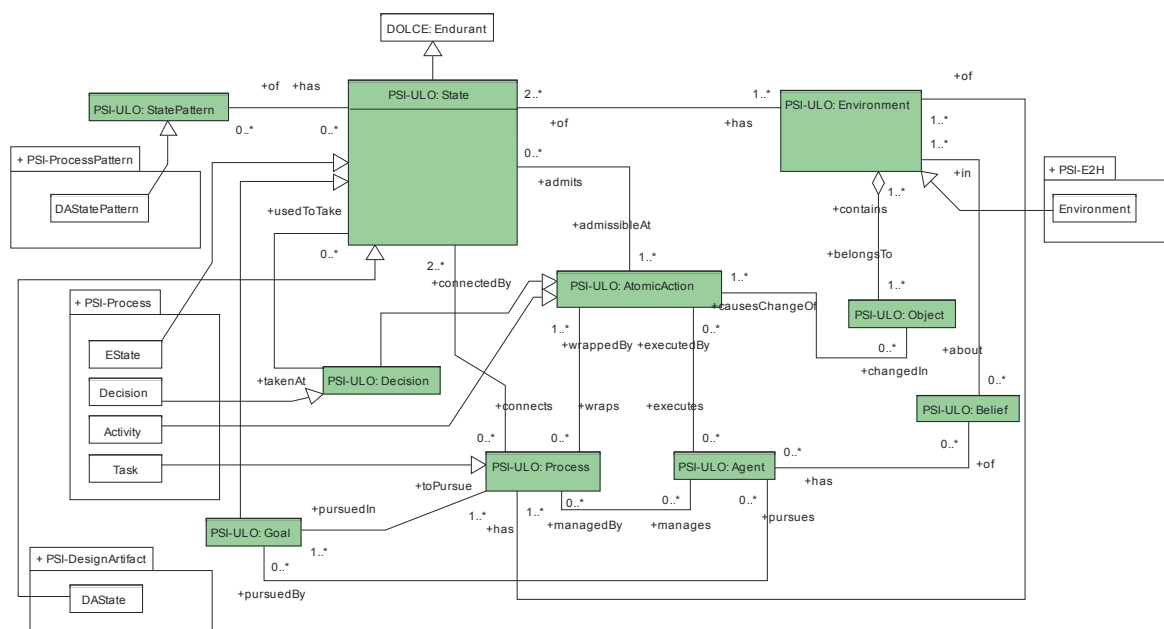


Fig. 2.2: The semantic context of the concept of a State.

A Goal, if complex, can be decomposed to simpler partial Goals as often done in problem solving. Such partial Goals are in fact the states of affairs that should be reached before the overall compound Goal can be attacked. States (Fig. 2.2) are the configurations of the constituents of an Environment. It is considered that a State is reached when the Characteristics of the constituents of the Environment have Values in the ranges satisfactory matching the corresponding Goal or partial-Goal of an Agent. In engineering design the mentioned Goals are technologically controlled. For example, a technology of digital front-end design in microelectronics and integrated circuits prescribes that an overall Goal of a digital back-end design is the development of a design artefact in the GDSII layout representation. At the same time the technology suggests that the netlist, floorplan, placement and routing representations should be developed before the overall Goal can be reached. In these settings the States can be seen as technological milestones on the path through the problem solution space leading to the overall Goal. The requirements to the ranges of the property values of the constituents of the Environment are denoted by StatePatterns. StatePatterns are controlled by the Policies of a company that are normally be based on the standards of the particular industrial sector. Goals and corresponding partial Goals may be pursued by taking different alternative paths going through different States. If a problem solution space is represented as a directed graph, a State may have several alternative outgoing edges. These edges correspond to different admissible AtomicActions applying different changes to the Environment. A Decision on the choice of an admissible AtomicAction should be taken for choosing the continuation of the path at any State. In particular, a Decision in the target State chooses among the alternative to terminate the

process in success and the alternative to refine the values of the properties of the constituents of the Environment heading to the same target State. Hence, a Decision is a specific Atomic Action which applies changes to an Environment indirectly – by choosing the alternative on the solution path. A Decision is also a mechanism to alter the course of the Process when the Goal or the sub-Goals are dynamically changed. In difference to an Environment, which is a Perdurant, a State is an Endurant because all its parts should be present at any instant of time of the presence of a State.

The model of a State is further refined in the PSI Core Process ontology [30, Section 5.5.2] and DesignArtifact ontology [30, Section 5.6].

An Object (Fig. 2.3) is a Holon⁶ that is situated in an Environment and may be changed by an Agent. An Object has its Environment and belongs to an Environment as a part – is situated in the Environment by other words.

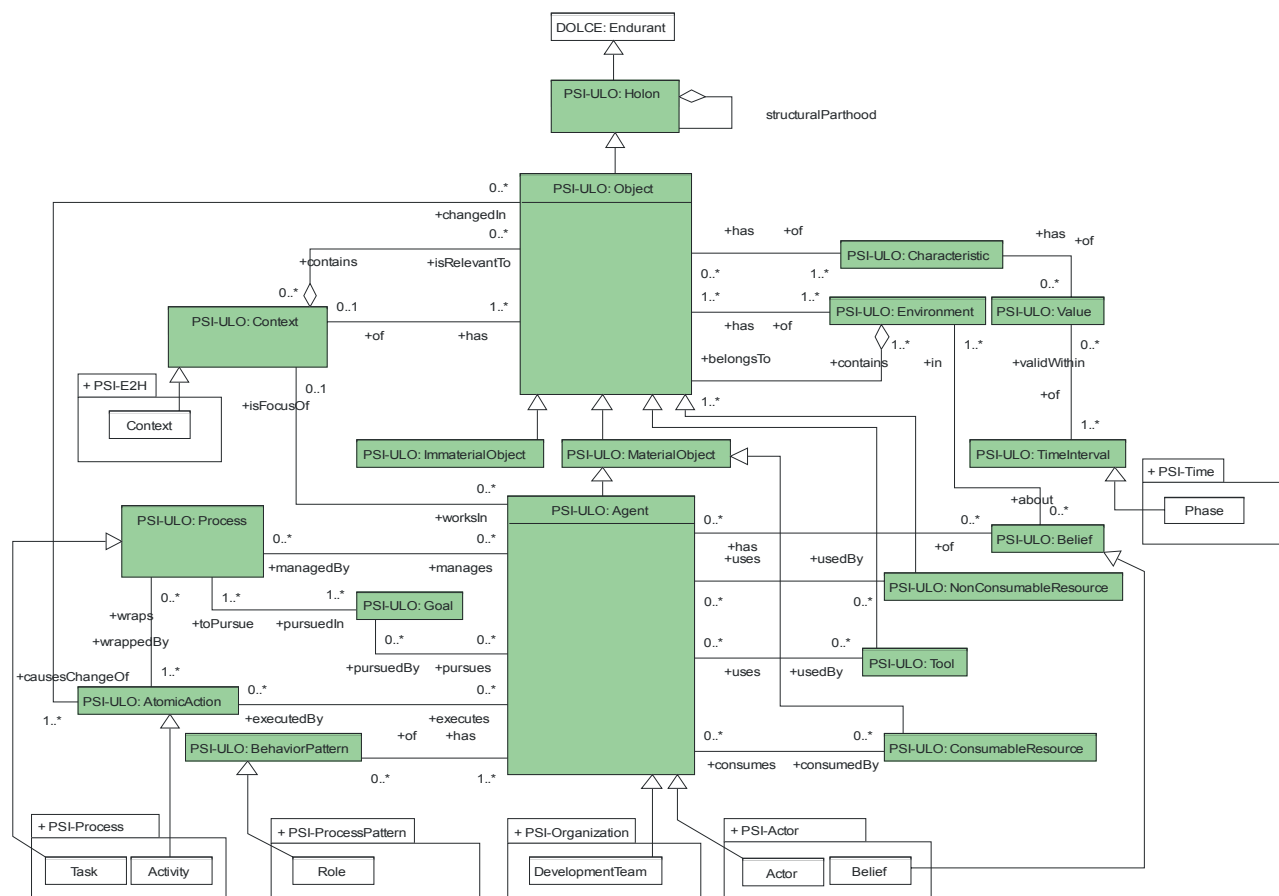


Fig. 2.3: The semantic contexts of the concepts of an Object and an Agent.

An Object may be changed in the course of an AtomicAction executed by an Agent. An Object may have Characteristics. The relationship between an Object and a Characteristic is further refined in the PSI Core ontologies by the relationships between the subclasses of and Object from one side and a Characteristic from the other side. An Object could be either material or immaterial. MaterialObjects are those Objects which are physically or legally substantial in the sense that they possess tangible physical non-temporal properties like mass, color, shape, size, speed, usage right and can not be copied or duplicated without borrowing a definite amount of physical or legal⁷ substance for it. The law of conservation of matter is applicable to material objects. MaterialObject subclasses are an Agent, a MaterialArtefact, a ConsumableResource, and a Tool. In difference to a ConsumableResource that is always material, a NonConsumableResource subsumes directly to an Object because it could be either material or immaterial. ImmaterialObjects in contrast to material ones are not substantial in a physical or legal sense. Hence, they can be copied

⁶ As specified in Section 4.10 a Holon is a compound entity which is a whole and, simultaneously, a part of a larger whole. A philosophical definition of a Holon implicitly assumes that all parts of a Holon are present at any instant of time of the existence of this Holon. In PSI Upper-Level ontology this assumption is made explicit by introducing a structural parthood relationship of a Holon to self and making a Holon a subclass of a DOLCE: Endurant.

⁷ By an odd term of “legal substance” we mean a legal permission to have an extra copy of an Object which is not a physical object in the sense of SUMO [11] or DOLCE [14]. A good example of such an Object is a software program with a license (legal substance).

or duplicated without consuming physical or legal substance. ImmaterialObject subclasses are an ImmaterialArtefact, a Rule, a Plan, a Fact, a Belief. From the point of view of an Agent an Object could be relevant to a particular working Context as well as it may have its Context. The model of an Object is further refined for its subclasses in several PSI Core ontologies. For instance the model of an Agent for individuals is refined in the PSI Core Actor ontology [30, Section 5.3] and for teams – in the PSI Core Organization ontology [30, Section 5.4]. The representations of different sorts of resources and Tools are further elaborated in the PSI Extension ontologies [30, Section 5.2]. The model of a Context is further elaborated in the PSI Core E2H ontology [30, Section 5.2].

An Agent (Fig. 2.3) is a MaterialObject that possesses pro-activity, is able to execute AtomicActions and to manage Processes. The pro-activity of an Agent is revealed in pursuing the Goals of changing an Environment to a desired State. An Agent is the only entity which can change an Environment by executing AtomicActions applied to the Objects in the Environment. An Agent has Beliefs about the Environment(s) that are the hypotheses believed to be true. These Beliefs may further become facts if confirmed by the happenings [18], [31] perceived by the observers. Beliefs together with desires and intentions are important basic elements forming the behaviour of an Agent. This behaviour is regulated by BehaviourPatterns specified as Rules. An Agent is an abstract entity which is a generic model for an individual person (a manager, a designer), a group of persons or artificial agents acting on behalf of physical persons (a team or an organizational unit), or an external pro-active entity influencing the Environment of an observed Process in a definite way.

A Rule (Fig. 2.4) is an ImmaterialObject which is a principle, a condition, a procedure, a generic pattern, or a norm shaping out a possible Process, Action, behaviour, or State. A Rule may be an atomic proposition or a more complex composition of other Rules due to the inherited structural parthood relationship to self. As far as a Rule is an Endurant no temporal parthood relationships are allowed for its proper parts – the composition of a rule can not be changed in time. A Rule itself still has a temporal property of validity – it is valid within a particular TimeInterval or several TimeIntervals. In PSI a Pattern, a Policy, and a Metric subsume to a Rule.

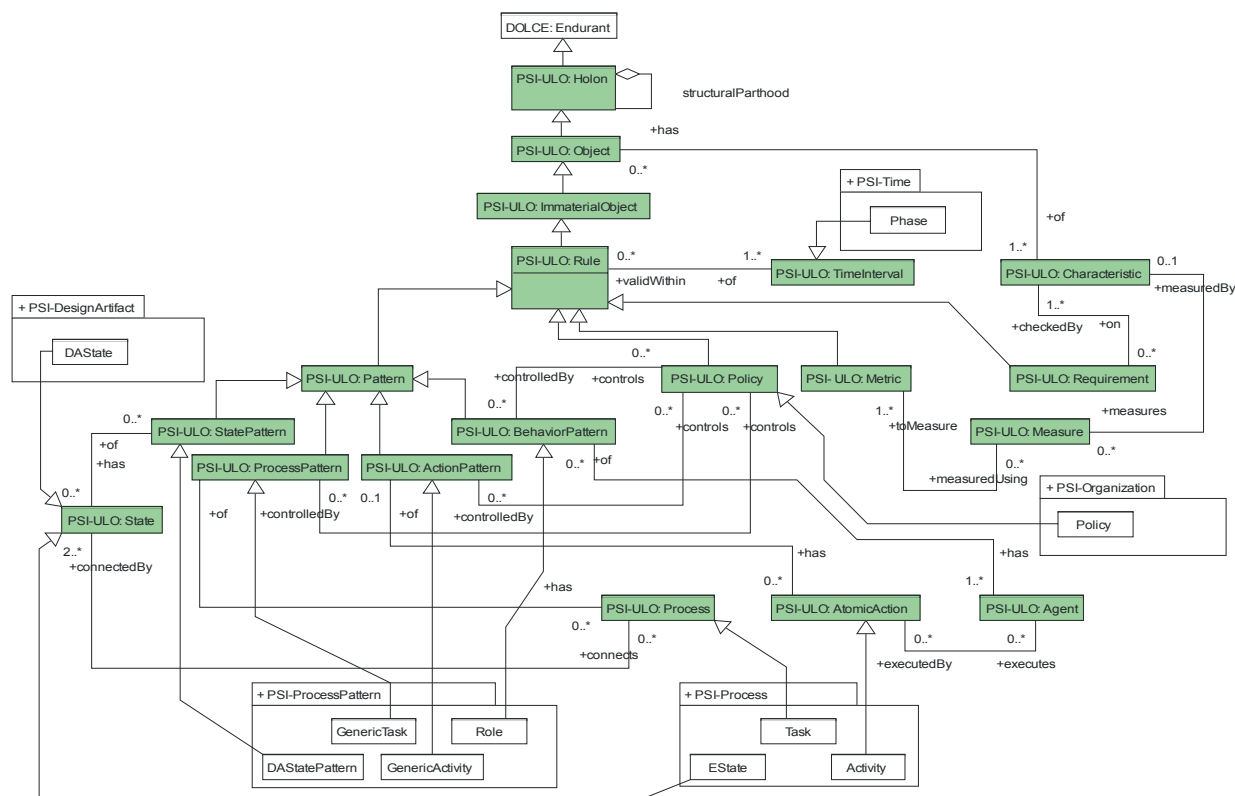


Fig. 2.4: The semantic context of the concept of a Rule.

Pattern models are further elaborated in the PSI Core ProcessPattern ontology [30, Section 5.5]. The refined model of a Policy is the part of the PSI Core Organization ontology [30, Section 5.4]. The model of time is further elaborated in PSI Core Time ontology [30, Section 5.1].

A Characteristic (Fig. 2.5) is a distinguishing property that has Values and may be checked by one or more Requirements. A characteristic is not an attribute of a concept because the latter inheres to a particular concept, but the former may have different semantic scents for different concepts. For example, a Dependency (a subclass of a Characteristic) may mean a particular form of co-execution for AtomicActions, a causal relationship for Events, or a commitment for Agents. Associations of a Characteristic with the other concepts of the upper-level model are not defined due to this very reason. These relationships may have different semantics for different counterparts or even in

different ontological contexts. Therefore, these associations are defined lower on in the knowledge hierarchy of PSI Suite of Ontologies – either in the Core or in the Extension ontologies where more appropriate.

The following specializations of a Characteristic are defined at the upper-level because of their generic character and multiple semantic scents.

A Dependency is a Characteristic that defines the qualities of a relationship among different entities of the same type. For example, we shall say that: one Event depends on the other Event if the latter causes the occurrence of the former [18], [31]; one AtomicAction depends on the other AtomicAction if the former can not be executed without the latter⁸; one Agent depends on the other Agent if the former committed to execute an AtomicAction in the process managed by the latter and so on.

A Consumption is a Characteristic that defines either the speed or the volume of the consumption of something – normally a consumable resource.

An Ability is a Characteristic that defines the ability of an Agent to perform actions.

A Location is a Characteristic that defines spatial or affiliation properties of an Object or a Process. A Process may be located within one organization, but also may involve several organizational entities. A Policy may have a local sphere of power within a team, but also may have a more widespread character – a National regulatory policy is obligatory for all Organizations in a country, etc.

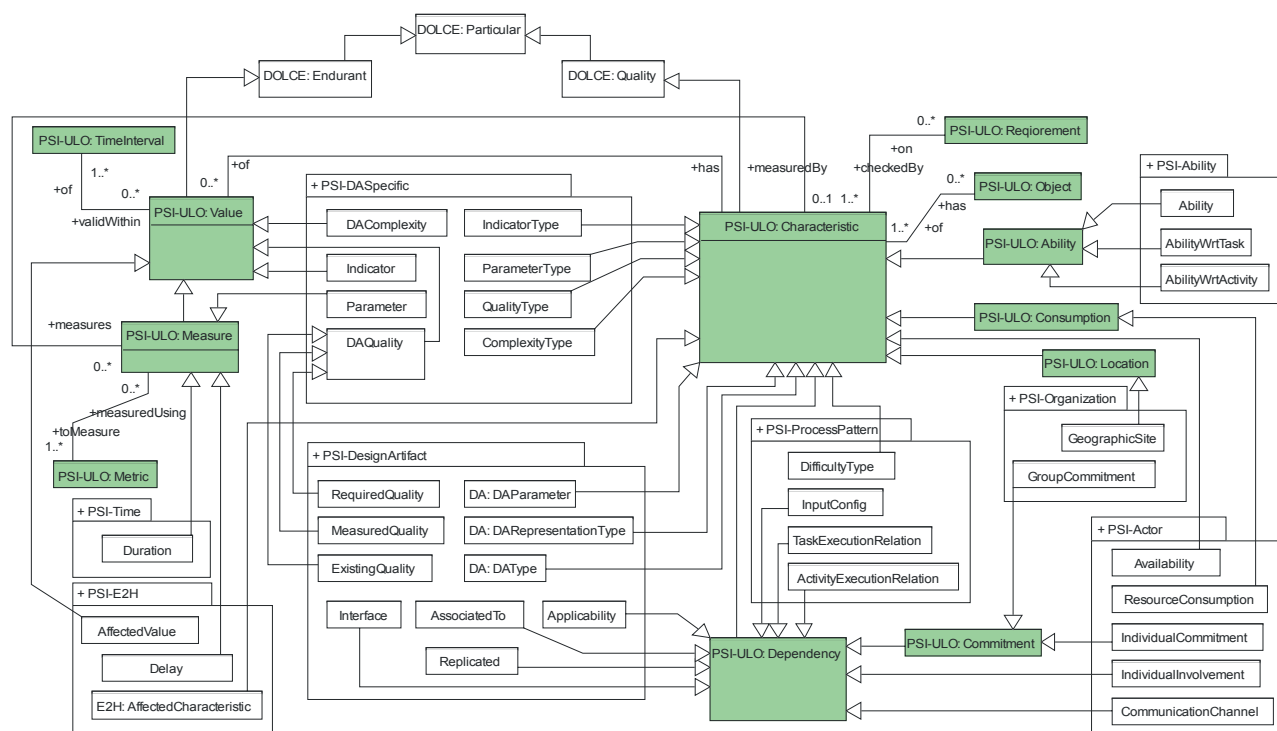


Fig. 2.5: The semantic context of the concepts of a Characteristic, a Value, and a Measure.

A Value is the mapping of an individual Characteristic to the corresponding “quality space” [14] pointing to the position of this individual Characteristic in this space. A Value is semantically equivalent to DOLCE’s concept of a Quale [14]. For example, the Value of an Agent’s Characteristic of Utility may be the quantity of the Units of Welfare [6] collected by the individual Agent in his life time. As a Value maps a particular Characteristic to a specific value space, we are interested in those Characteristics that may have Values. For a Characteristic “having a Value” may mean that: (i) this value is measured; or (ii) this value is assigned; or (iii) this value is computed (e.g. inferred). A Value may have the temporal property of validity within a particular TimeInterval. A Measure is the subclass of a Value. Measures are Values that are measured using an appropriate mechanism and a unit of measurement.

⁸ Different aspects of action to action dependencies are analysed in [17] and further developed in PSI ProcessPattern and Process Ontology

The models of a Characteristic and its upper-level subclasses are further elaborated in the PSI Core DASpecific, DesignArtifact, ProcessPattern, Actor ontologies [30, Section 5] and in the PSI Extension Ability ontology [30, Section 6.5].

3. Ontology Engineering Methodology

The methodology used for the development of the PSI Upper-Level ontology v.2.3 as the part of the PSI Suite of Ontologies v.2.3 is the Shaker Modeling Methodology.

Shaker Modeling Methodology– is the combination of bottom-up and top-down modeling and knowledge engineering techniques exercised in subsequent design iterations. The source for the top-down activity is the PSI Modeling Framework [17] – the document that specifies the approach and the method for modeling the domain of microelectronic and IC design at rather a high level of detail. The sources for the bottom-up phase are the user evaluation results and the requirements by the subject experts with respect to the previous revision of the Core and Extensions of the PSI Suite of Ontologies [25]. The intermediate level of knowledge representation is the PSI Upper-Level ontology (this document). The Upper-Level ontology serves as the reference descriptive theory of the domain. More specifically, it is used as:

(i) the semantic bridge facilitating to mapping domain ontologies to abstract ontological foundations and common sense; (ii) the semantic “glue” for bridging the domain ontologies to other theories describing the same or similar domains; (iii) a “proxy” for different kinds of evaluation of the domain ontologies. The use of the Upper-Level ontology as a referential theory eases reaching the commitment of potential users to the PSI Suite of Ontologies.

Shaker Modeling Methodology ensures that all mentioned knowledge representations are refined in a way that:

- (i) The requirements collected at the evaluation of the previous revision are met
- (ii) The taxonomy of the resulting ontology is formally correct;
- (iii) The resulting ontology is aligned with the chosen foundational ontology and commonsense reference ontology.

In addition, the resulting ontology may be mapped to a different ontology if such a mapping is needed.

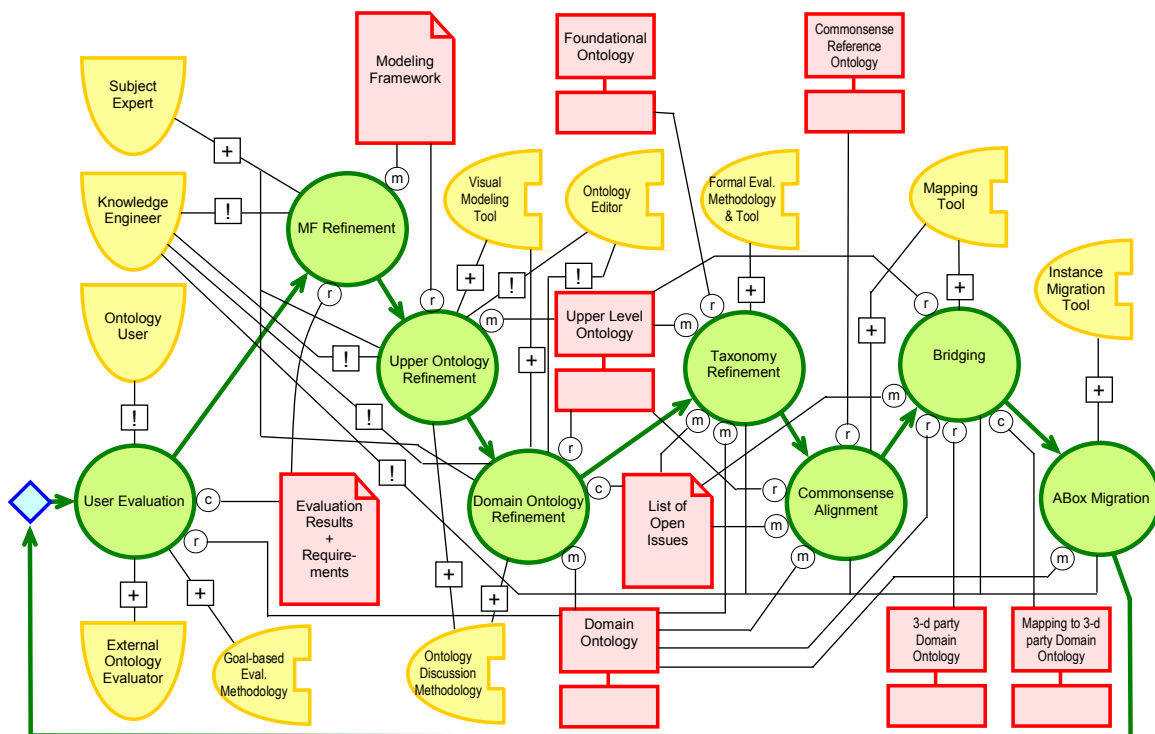


Fig. 3.1: The phases of ontology refinement iteration in Shaker Modeling process.

The phases of one refinement iteration are pictured in Fig. 3.1. The notation proposed for ISO/IEC 24744 [32] has been used for the graphical representation of the methodology. The phases are:

- (i) User Evaluation
- (ii) Modeling Framework Refinement
- (iii) Upper Ontology Refinement

- (iv) Domain Ontology Refinement;
- (v) Taxonomy Refinement
- (vi) Commonsense Alignment
- (vii) Bridging
- (viii) ABox Migration

This activity flow is not prescribed rigidly. Some phases may be merged, some phases may be skipped depending on the quality requirements, the complexity of the domain, and the competence of the users and subject experts. For example, the Modeling Framework Refinement, Upper Ontology Refinement may be merged with Domain Ontology Refinement phase for a small ontology development project with low complexity and narrow scope. Commonsense Alignment and Bridging may also be skipped if the scope narrow and the role and the future use of the ontology are well understood by the target group of users. A decision about expanding or shrinking the process may be made using cost estimation for ontology development as for example proposed in [33].

The methodological framework of Shaker Modeling does not also prescribe the use of particular tools in its phases, but outlines the goal and the activity to be performed for reaching the desired state of affairs. The adopters have freedom of using the tools they are accustomed to.

User Evaluation (UE) . This phase is the initial one in the iteration. Its goal is to find out if the refinement is required. More specifically, the objective of the user evaluation is to find out if the Domain Ontology fits the user requirements. An external evaluation by independent experts may also be done at this phase to ensure that evaluation results are unbiased and of good quality. Any appropriate goal-based evaluation routine may be used for this activity as a tool. User evaluation produces the report containing the list of requirements for ontology refinement. These requirements are used as an input for the refinement of the Modeling Framework. UE is a bottom-up activity because the input (Domain Ontology) is presented in a less abstract and less high-level form than the output (requirements list).

Modeling Framework Refinement (MFR) . In this phase the Modeling Framework specification is further elaborated to meet the requirements produced in the User Evaluation Phase. Though it is mainly the work for the knowledge engineer who is considered the owner of this specification, the involvement of the subject experts is recommended for better understanding and elaborating the requirements. The outcome of the phase is the next fixed revision of the Modeling Framework specification. This revision provides the requirements for the subsequent ontology refinement phases in a more formal and elaborated representation than it has been listed in the user requirements document. MFR is a top-down activity because the input is provided in a more high-level and abstract representation than the output.

Upper Ontology Refinement (UOR) . In this phase the changes of the Modeling Framework are filtered through the Upper Level Ontology in a top-down manner. Some of the required higher-level changes are reflected in the updates of the referential upper level model of the domain – the others are analyzed and planned for been elaborated at the lower level of the Domain Ontology within the subsequent phase. The refinements done at this level are: (i) *contextual* – the semantics of some upper level concepts and their contexts [34] may be changed; (ii) *taxonomical* – the subsumption hierarchy may be refined; (iii) *structural* – the modular structure of the underlying Domain Ontology may require changes dictated by the refinement of the upper level model. UOR is also a verification and validation activity that checks: (i) if the Modeling Framework could be implemented in the ontology; and (ii) if the refined Modeling Framework is correct and unambiguous with respect to its implementation. Shaker Modeling methodology suggests that this phase is begun with performing a modeling activity resulting in a graphical representation of the upper level model, for example, a UML class diagram. This diagram may be used for discussing the model and reaching agreements on ontological choices with subject experts. The discussions may be guided by an ontology discussion methodology which, in fact, is the routine for reaching agreements and consensus on ontological choices. Once the consensus is reached and the upper level model is fixed in its graphical representation, the coding activity may be performed. At this step the model is transformed to the ontology in a chosen specification language using an appropriate ontology editor tool. A reference specification of the Upper Level Ontology could also be produced for documentation purposes especially for the projects with several partners and distributed development settings. UOR uses the Modeling Framework Specification and the previous revision of the Upper Level Ontology as its inputs. It produces the new revision of the Upper Level Ontology as the output.

Domain Ontology Refinement (DOR) . Within this phase the requirements that have been formalized in the Modeling Framework and have not been captured in the Upper Level Ontology are applied to the refinement of the Domain Ontology. It is therefore supposed that the concepts of the Domain Ontology are related to the concepts of the Upper Level Ontology by subsumptions only. The task of the phase is to fully elaborate the required changes in domain model semantics and harmonize them with the reference upper level theory using the joint subsumption hierarchy of the upper and domain levels. It is suggested to start with the *structural* refinements, continue with *taxonomical* refinements, and complete with *contextual* refinements in the Domain Ontology. Implementing the changes at the domain level may result in affecting the upper level model and the Modeling Framework. If so, the list of open issues is formed and used in subsequent phases. Modeling and coding activities are performed in a manner similar to that of the UOR phase. DOR

uses the Modeling Framework Specification, the Upper Level Ontology, and the previous revision of the Domain Ontology as inputs. It produces the next revision of the Domain Ontology and the List of Open Issues as the outputs.

Taxonomy Refinement (TR) . This phase starts with the analysis of the List of Open Issues. The changes to the joint subsumption hierarchy of the domain and upper level theories are applied if there were the issues recorded at the domain level of abstraction. This step of TR is therefore a bottom-up activity. Next step is the formal evaluation of the joint taxonomy using an appropriate methodology (e.g., OntoClean [24]). The joint taxonomy is checked for the conformance to the ontological choices of the chosen Foundational Ontology in a top-down way. Evaluation results may point to the defects of the taxonomy which may then be immediately repaired. However, some defects found at this step may point to modeling mistakes. Such cases are the feedback for the follow-up iteration of Shaker Modeling and have to be added to the open issues list. TR uses the Domain Ontology, the Upper Level Ontology, and the chosen Foundational Ontology as the inputs. The choice of the Foundational Ontology is determined by the specific characteristics of the domain and corresponding ontological choices (see, e.g., [22]). TR produces the modifications of the Domain Ontology and the Upper Level Ontology.

Commonsense Alignment (CA) . Ensuring that the taxonomy is formally correct is not enough for building a good ontology. One more important aspect to be kept in mind is that “...every ontology is a treaty – a social agreement – among people with some common motive in sharing”⁹. Such an agreement may be reached easier if the subject is intuitively clear to the social group – i.e. aligned with their common sense. Therefore, Shaker Modeling framework suggests the phase for commonsense alignment. The Domain Ontology is mapped to the chosen Commonsense Reference Ontology using the Upper Level Ontology as the semantic “glue”. The upper level theory is used as a mediator because its level of abstraction is higher than the one of the domain ontology and is closer to the commonsense theory. These mappings allow checking if the Domain Ontology is sound enough to adequately conform to human beliefs about what the domain is. If the result of such verification is positive (all the mappings are easily built and their semantics is clear), then we may expect that the Domain Ontology will be accepted by the target social group of users without major difficulties. If the mapping reveals semantic gaps, mismatches, or hanging concepts then the ontology is either a novel extension of the chosen commonsense theory or, more probably, is not very well designed. These issues are a good feedback for the subsequent ontology refinement iterations and therefore have to be added to the List of Open Issues. One more utility of the commonsense mappings is their use as “referees” at the subsequent bridging phase. CA uses the chosen Commonsense Reference Ontology as the input and the Upper Level Ontology as the mediator creating the mappings of the concepts of the Domain Ontology to the concepts of the Commonsense Reference Ontology and, possibly, for modifying the Domain Ontology. It is performed by the knowledge engineer who may use an appropriate ontology mapping tool.

Bridging (B) . The objective of the Bridging phase is to evaluate the completeness and the expressiveness of the Domain Ontology by comparing it to the so called “Golden Standard” [35]. By a “Golden Standard” we mean a highly reputable ontology describing the theory of the same or a similar domain which has already gained broad commitment by domain experts. The evidence of such a commitment may be that “Golden Standard” ontology is the basics for a standard, a de-facto standard, or a standardization proposal. Similarly to CA the mappings of the concepts of the Domain Ontology to the chosen “Golden Standard” are built. If all the concepts of a “Golden Standard” are mapped by the concepts of the Domain Ontology then it may be estimated that the domain theory covers the domain equally to or better than a “Golden Standard”. Otherwise, the Domain Ontology is less complete than the “Golden Standard”. In the latter case the reasons of potential incompleteness should be analyzed. In a safe case it may be found out that the domain described by the “Golden Standard”, though similar to the one of the Domain Ontology, is broader. Otherwise, the domain theory is incomplete – the revealed defects need to be added to the List of Open Issues. The mappings in the opposite direction – from the concepts of a “Golden Standard” to the concepts of the evaluated Domain Ontology, may help assessing the level of the expressiveness of the target. For example, if all the concepts of the “Golden Standard” map to single concepts of the evaluated Domain Ontology then it may be the case that the Domain Ontology possesses at least the same level of expressiveness at the “Golden Standard”. A more difficult situation is encountered when there is no a “Golden Standard” for the domain. In this case it is suggested to use the ontology from a different domain which is built using the similar modeling principles and ontological choices. B is considered to be a top-down activity because it relies on the Upper Level Ontology as the starting reference point and the mediator for creating mappings among the two domain theories. B uses the Upper Level Ontology, the Domain ontology and the “Golden Standard” ontology as the inputs and creates the bi-directional mappings between the Domain Ontology and the “Golden Standard” as the output. The phase is performed by the knowledge engineer who may use an appropriate ontology mapping tool.

The Bridging phase completes the sequence of TBox design and evaluation activities for refining the Domain Ontology. The instances of the previous revision of the Domain ontology can now be migrated to the new revision.

ABox Migration (AM) . This phase completes the iteration of Shaker Modeling Process. Its task is copying the instances of the Domain Ontology from its previous revision to the new revision. As the revisions are subsumed to be

⁹ Tom Gruber, the interview for the Official Quarterly Bulletin of AIS Special Interest Group on Semantic Web and Information Systems, Volume 1, Issue 3, 2004.

different the transformation rules for instance migration have also to be created. An ontology versioning and evolution analysis framework and an instance migration tool may be used for performing this activity.

To summarize, Shaker Modeling framework is a generic methodology suggesting an iterative process based on the combination of top-down and bottom-up activities. It comprises: (i) three development phases (MFR, UOR, DOR) that are directed by the requirements and the open issues collected at previous iterations; (ii) four evaluation phases (UE, TR, CA, B) that facilitate to the quality of the developed revision and provide feedbacks for subsequent iterations; (iii) one instance migration phase (AM). The process could be applied within the lifecycle (refinement phase) of a Domain Ontology. The particular refinement process may be considered as accomplished when all the requirements are met and all the open issues are resolved.

4 Detailed Specification of the PSI Upper-Level Ontology

This section provides a detailed specification of PSI Upper-Level ontology v.2.3. The sources for the development of this revision of the Meta-Ontology are PSI Theoretical Framework v.2.3 [17] and PSI Core Ontologies v.2.2 [25]. The UML diagram of the taxonomical structure of PSI Upper-Level ontology is given in Fig. 4.1. The UML diagram depicting other types of relationships among the concepts of PSI Upper-Level ontology is given in Fig 4.2. The filenames in the PSI documents repository are: UML – PSI-ULO-Taxonomy-v-2-3.zargo, PSI-ULO-v-2-3.zargo; OWL – PSI-ULO.owl; Protégé PPRJ – PSI-ULO.pprj.

PSI Upper-Level ontology v.2.3 commits to the ontological choices of DOLCE [14] and therefore is descriptive, possibilistic, multiplicative, and perduranistic. As far as the upper level of DOLCE taxonomy is used, PSI Upper-Level ontology is, as DOLCE, the ontology of particulars.

It should be noted that the concepts of DOLCE are not the part of the PSI Upper-Level ontology. Their descriptions in Sections 4.1 – 4.4 are given for readers' convenience.

4.1 DOLCE: Particular

DOLCE concept of a Particular [14] is the root concept in PSI Upper-Level ontology. As in DOLCE we consider that Particulars in our ontology can't have instances. However the reason for it is different. We consider all PSI Upper-Level ontology concepts to be Particulars because we do NOT NEED their instances. These instances will never appear in PSI knowledge base and will never be used in PSI software. In difference to the concepts of PSI Upper-Level ontology, the concepts of the Core and the Extension ontologies of PSI Suite have instances and are represented in PSI knowledge base by them.

4.2 DOLCE: Endurant

DOLCE concept of an Endurant is the sub-class of a Particular. As specified in [14], Endurants are wholly present (i.e., all their proper parts are present) at any time they are present. In PSI Upper-Level ontology the sub-classes of a DOLCE: Endurant are a **PSI-ULO: AtomicAction**, a **PSI-ULO: Holon**, a **PSI-ULO: State**, a **PSI-ULO: Value**, and a **PSI-ULO: TimeInstant**.

4.3 DOLCE: Perdurant

DOLCE concept of a Perdurant is the sub-class of a Particular. As specified in [14], Perdurants extend in time by accumulating different temporal parts, so that, at any instant in time they are present, they are only partially present, in the sense that some of their proper temporal parts (e.g., their previous or future phases) may be not present. In PSI Upper-Level ontology the sub-classes of a **DOLCE: Perdurant** are **PSI-ULO: Phenomenon**, a **PSI-ULO: Event**, a **PSI-ULO: Environment**, and a **PSI-ULO: Context**.

4.4 DOLCE: Quality

DOLCE concept of a Quality is the sub-class of a Particular. As specified in [14], a Quality is a basic entity we can perceive or measure: shapes, colors, sizes, sounds, smells, as well as weights, lengths, electrical charges, etc. Qualities are Particulars because they inhere to particulars and distinguish them from each other. Qualities are neither Endurants, nor Perdurants because the latter two may have Qualities, but a Quality can not have Qualities itself. In a sentence, Qualities are perceivable or measurable characteristics of Endurants and Perdurants which combination is unique for an entity. Qualities may change in time. In PSI Upper-Level ontology a **PSI-ULO: Characteristic** is the sub-class of **DOLCE: Quality**.

4.5 A Phenomenon

A Phenomenon according to [21] is a thing that is shown, or revealed, or manifest in experience. Phenomena in philosophy are for a long time [16] denoted as the objects of the senses.

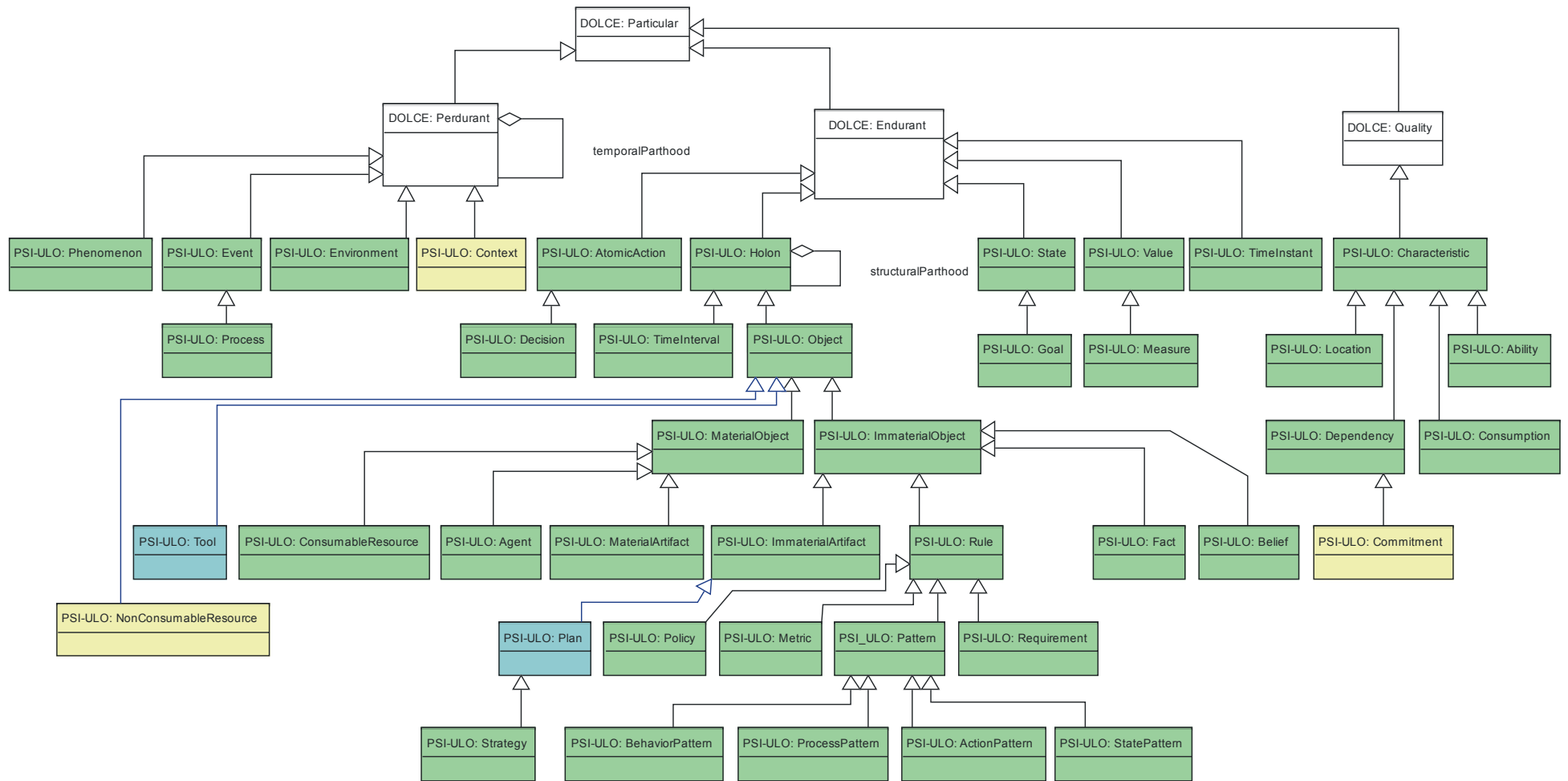


Fig. 4.1: UML diagram of the taxonomy in the PSI Upper-Level ontology.

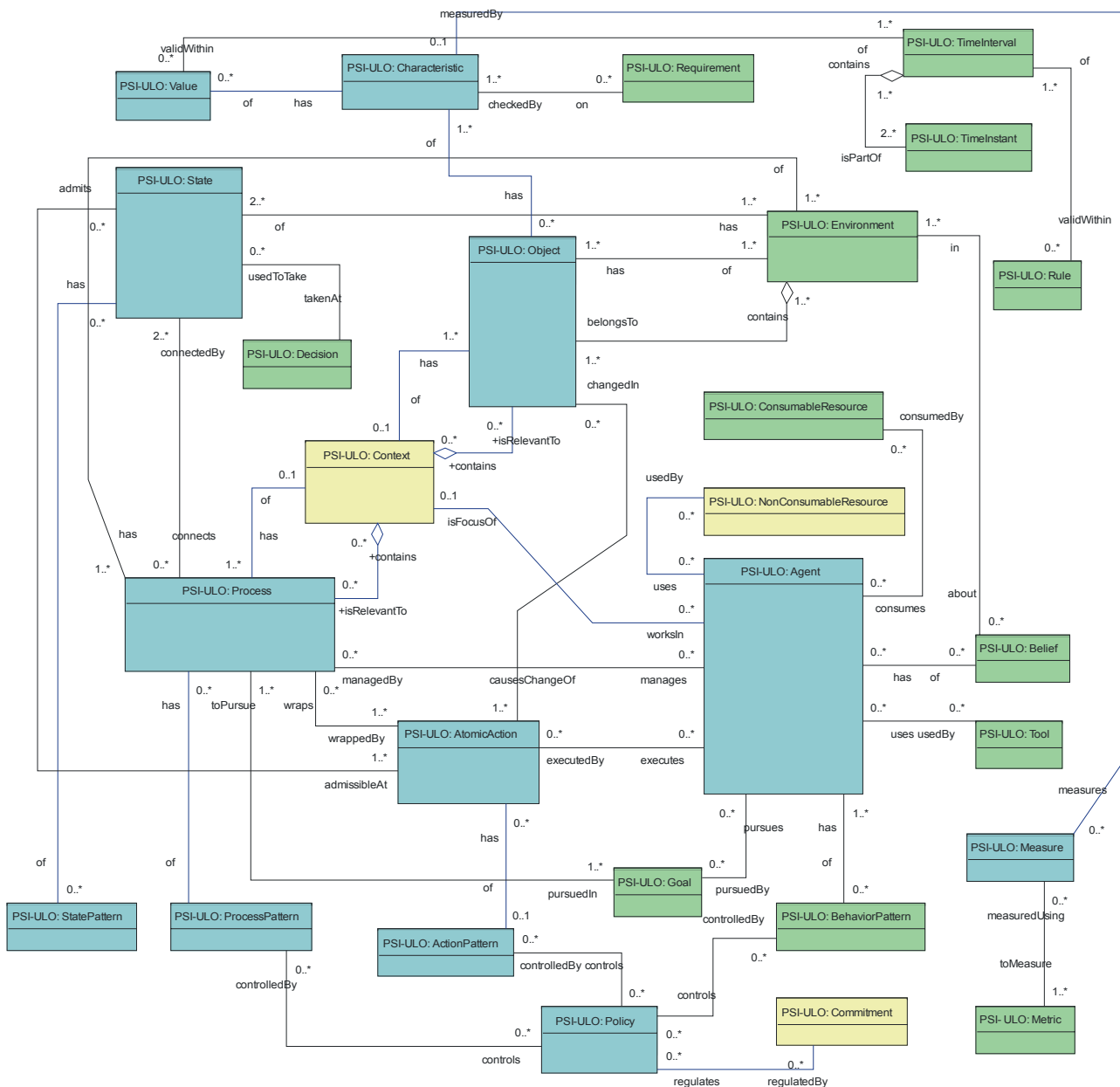


Fig. 4.2: UML diagram of the holonymy/meronymy relationships and associations in the PSI Upper-Level ontology.

In PSI Upper-Level ontology it is considered that an entity which can be sensed is the reflection of the change in the observable world. i.e. a Phenomenon is this kind of a change which can potentially be sensed or perceived. In PSI Theory of Time, Events, and Happenings [18] it is considered that Phenomena are manifested through Events. The concept of a Phenomenon is used in PSI Core Environment, Event, and Happening ontology.

Phenomena are Perdurants because not all of their proper parts may be present at any arbitrary TimeInstant of their presence. Therefore, Phenomena inherit temporalParthood relationship of a Perdurant. Proper parts of a Phenomenon are phases.

PSI-ULO: Phenomenon subsumes to WordNet: Phenomenon which is any state or process known through the senses rather than by intuition or reasoning. **PSI-ULO: Phenomenon** is not equivalent to **WordNet: Phenomenon** because it is not a state. **WordNet: Phenomenon** in turn subsumes to **SUMO: Physical** – an entity that has a location in space-time.

PSI-ULO: Phenomenon own properties:

Not defined

PSI-ULO: Phenomenon relationships:

Subsumption relationships:

PSI-ULO: Phenomenon is subclass of **DOLCE: Perdurant** (Section 4.3)

Meronymy relationships:

Own meronymy/holonymy relationships are not defined. **PSI-ULO: Phenomenon** inherits temporalParthood relationship of **DOLCE: Perdurant**

Associations:

Not defined

4.6 An Event

World, or a particular part of the World which is a Domain of Discourse, is not static. Different changes occur in it. These changes are Phenomena (Section 4.5). Phenomena are manifested as Events. If a Phenomenon is the change in the World which can, potentially be sensed, then an Event is a certain phase of the Phenomenon in which the change becomes more than a particular threshold – can be sensed and measured with available instruments. In that sense an Event is the manifestation or the occurrence of the phenomenon. Considering different Events as the manifestations of the phases of Phenomena is therefore a sort of a discretisation of Phenomena¹⁰.

Events could be both stateful and stateless. In the latter case we are not interested if an Event has its own phases which bring the world to distinct States of affairs or these States can not be distinguished because of imperfectness of our sensors. For example, the event of a sunrise is stateless. As observers, the overwhelming majority of people are not interested in extracting the phases of a sunrise. We simply do not care that such a wonderful view may be “prepared” as phase slices. From the other hand, those scientists who investigate the processes occurring in the corolla of the Sun, may distinguish the phase when only the corolla is seen over the horizon.

Events could also possess a pro-active character in the sense that there is an executor of this Event who or which pursues a definite Goal by executing or controlling the course of this Event.

Events are Perdurants because not all of their proper parts (if any) may be present at any arbitrary TimeInstant of their presense. Therefore, Events inherit temporalParthood relationship of a Perdurant. Proper parts of an Event are phases. In PSI it is considered that phases are characteristic to stateful Events. Hence, a phase is the development of an Event which brings the world¹¹ to a particular State of its affairs.

The notion of an Event is further elaborated in PSI Core Environment, Event, and Happening ontology [30].

PSI-ULO: Event is mapped to **SUMO: Process** using subsumption relationship

PSI-ULO: Event own properties:

Not defined

PSI-ULO: Event relationships:

Subsumption relationships:

PSI-ULO: Event is subclass of **DOLCE: Perdurant** (Section 4.3)

Meronymy relationships:

Own meronymy/holonymy relationships are not defined. PSI-ULO: Event inherits temporalParthood relationship of **DOLCE: Perdurant**

Associations:

Not defined

4.7 A Process

A Process is a specialization of an Event which is stateful and possesses pro-active character. A Process has its Environment – the part of the world which is changed in the course of the Process. A Process is pro-actively directed by

¹⁰ In functional terms a phenomenon may be represented as a fluent, while an event is a discrete function having the same changed feature as its basic variable.

¹¹ In PSI Upper-Level ontology it is considered that a phase brings not the whole world but its part – a certain Environment to a particular State. Please see Section 4.8 for the definition of an Environment.

the Agent who manages it. Pro-activeness of the Agent is understood in the sense that the Agent pursues a particular Goal in the managed Process. This Goal is the State of the Environment which the Agent desires to make reached. It should also be mentioned that the change in the Environment is not produced by the Process, but by the entities who act in this process – those Agents who execute AtomicActions wrapped by the Process. In general, it is considered that changes may only be achieved by Agents through execution of AtomicActions. For example, it is wrong to say that a multimedia controller layout has been designed by the process of logical design. In fact the appearance of the layout for the multimedia controller in a certain state of the Environment (the measurable change in the Environment) has been achieved by the team of Agents who executed a particular sequence of AtomicActions. By that the Agents introduced the sequence of changes in the Environment and guided the environment through the sequence of States towards the Goal.

Processes in an engineering Environment can not connect any arbitrary State to any other arbitrary State because it is senseless with respect to the technology or the methodology. Some sequences of States may therefore be withdrawn from the engineering routine and some other sequences of States may be suggested or prescribed by an industrial standard or a company policy. These prescriptions in terms of PSI Upper-Level ontology are ProcessPatterns.

The context of the concept of a Process in the PSI Upper-Level ontology is pictured in Fig. 4.3.

PSI-ULO: Process is mapped to **SUMO: Process** through **PSI-ULO: Event**

PSI-ULO: Process own properties:

Not defined

PSI-ULO: Process relationships:

Subsumption relationships:

PSI-ULO: Process is subclass of **PSI-ULO: Event** (Section 4.6)

Meronymy relationships:

PSI-ULO: Process inherits temporalParthood relationship to self of **DOLCE: Perdurant**.

New in v.2.3

PSI-ULO: Process (0...*) isRelevantTo – contains (0...*) **PSI-ULO: Context** (Section 4.37)

Associations:

PSI-ULO: Process (1...*) has – of (1...*) **PSI-ULO: Environment** (Section 4.8)

PSI-ULO: Process (0...*) connects – connectedBy (2...*) **PSI-ULO: State** (Section 4.11)

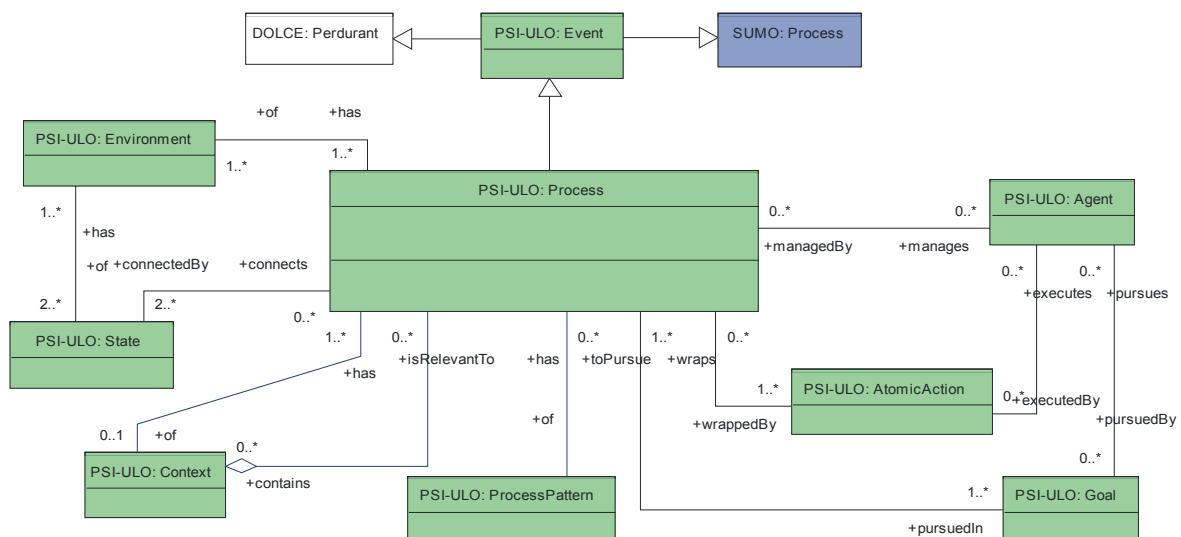


Fig. 4.3: The context of a **Process** in the PSI Upper-Level ontology.

PSI-ULO: Process (1...*) toPursue – pursuedIn (1...*) **PSI-ULO: Goal** (Section 4.12)

PSI-ULO: Process (0...*) wraps – wrappedBy (1...*) **PSI-ULO: AtomicAction** (Section 4.9)

PSI-ULO: Process (0...*) managedBy – manages (0...*) **PSI-ULO: Agent** (Section 4.19)

Changed in v.2.3

PSI-ULO: Process (40...*) has – of (1) **PSI-ULO: ProcessPattern** (Section 4.33)

New in v.2.3

PSI-ULO: Process (1...*) has – of (0..1) **PSI-ULO: Context** (Section 4.37)

4.8 An Environment

One of the basic postulates of PSI Upper-Level ontology is that any Process or Object has its Environments. An Environment is a temporal aggregation of different kinds of Objects which surround the Process or the Object in question. We shall also say that a Process or an Object surrounded by its Environment is situated in the Environment. By “surrounding” we understand several distinct things: (i) an Object situated in the Environment may be changed by the objects constituting this Environment; (ii) a Process is always situated in one or more Environments because it connects the States of its Environment(s); (iii) an Environment may be changed by the Objects of this Environment or by the Objects external to this Environment in Events.

An Environment influences an Object or a Process situated in it. For example, the same Agent (a subclass of an Object) situated in different Environments may have different properties: play different roles, have different beliefs, have different availabilities, execute different AtomicActions, etc.

An Environment, being a Perdurant, is a temporal aggregation of its proper parts. Indeed, the constituents of an Environment are not all present at every TimeInstant of the presense of this Environment. For example, different representations of a Design Artifact are accomplished at different time, different designers may become involved in a collaborative design process in its different phases, different ConsumableResources may become necessary at different stages of design work, etc. The concept of an Environment inherits temporalParthood relationship to self of **DOLCE: Perdurant**.

The context of the concept of an Environment in the PSI Upper-Level ontology is pictured in Fig. 4.4.

PSI-ULO: Environment is mapped to **SUMO: Entity**. No more specific mappings to SUMO ontology were found.

PSI-ULO: Environment own properties:

Not defined

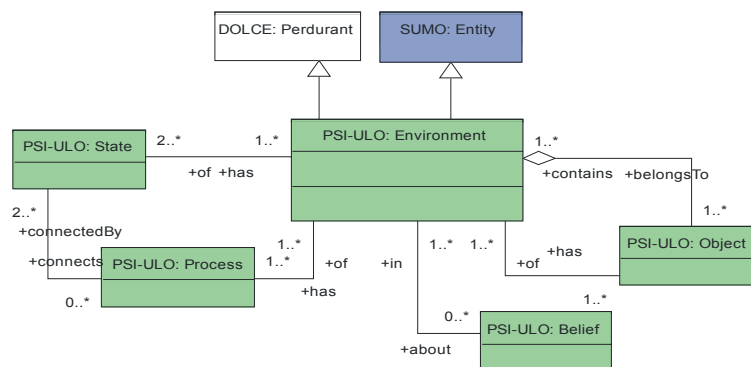


Fig. 4.4: The context of an **Environment** in the PSI Upper-Level ontology.

PSI-ULO: Environment relationships:

Subsumption relationships:

PSI-ULO: Environment is subclass of **DOLCE: Perdurant** (Section 4.3)

Holonymy relationships:

PSI-ULO: Environment inherits temporalParthood relationship to self of **DOLCE: Perdurant**

PSI-ULO: Environment (1...*) contains – belongsTo (1...*) **PSI-ULO: Object** (Section 4.17)

Comment: This parthood relationship also has temporal character. Different Objects belongTo a particular Environment within different TimeIntervals.

Associations:

PSI-ULO: Environment (1...*) has – of (2...*) **PSI-ULO: State** (Section 4.11)

PSI-ULO: Environment (1...*) of – has (1...*) **PSI-ULO: Object** (Section 4.17)

PSI-ULO: Environment (1...*) in – about (0...*) **PSI-ULO: Belief** (Section 4.27)

PSI-ULO: Environment (1...*) of – has (1...*) **PSI-ULO: Process** (Section 4.7)

4.9 An AtomicAction and a Decision

An AtomicAction is a basic indivisible Action which is executed by an Agent and is applied to a transformed Object. Agents change Objects through AtomicActions. An AtomicAction is an Endurant because it is indivisible - it contains its only proper part, the AtomicAction, which is present at any TimeInstant of the presence of the AtomicAction. The atomicity of an AtomicAction does not allow considering it to be a Holon. Indeed, an AtomicAction can not be a proper part of a bigger AtomicAction because a bigger AtomicAction should be atomic as well.

AtomicActions are often executed in Processes. A Process is therefore a bigger structure that wraps one or several different AtomicActions. The execution of these AtomicActions is therefore arranged using the framework of a Process. AtomicActions can not be applied to the Objects in an Environment at an arbitrary State. In PSI it is assumed by [17] that a State has a set of admissible AtomicActions associated with it. By so saying, a State may or may not admit the execution of a particular AtomicAction.

AtomicActions analogously to Processes are executed based on generic engineering patterns. These ActionPatterns are formed according to the technology used in house or prescribed by a standard. ActionPatterns are further elaborated in PSI Core Process Pattern Ontology [30].

The context of the concept of an AtomicAction in the PSI Upper-Level ontology is pictured in Fig. 4.5.

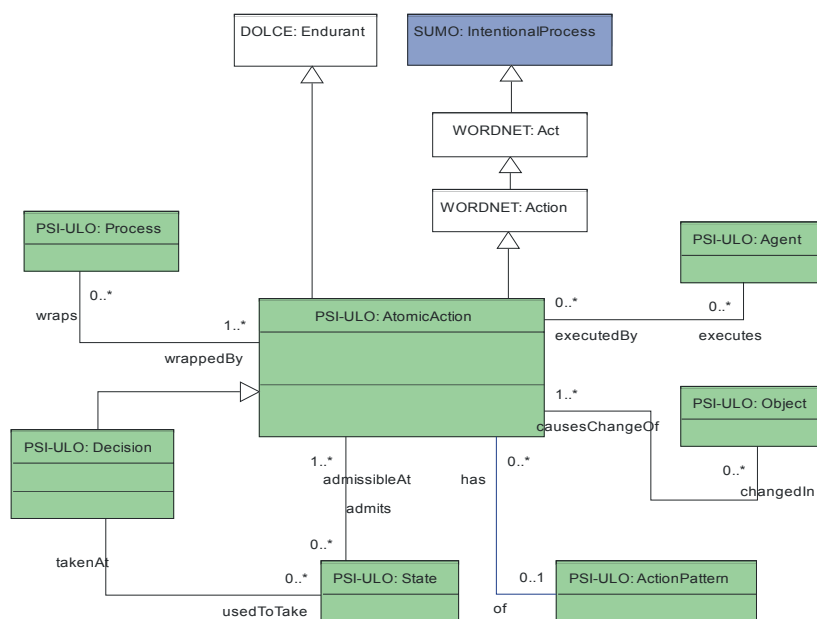


Fig. 4.5: The context of an **AtomicAction** in the PSI Upper-Level ontology.

PSI-ULO: AtomicAction is mapped to **WordNet: Action** and then to **SUMO: IntentionalProcess** using subsumption relationship. Though such a subsumption is in general semantically correct, neither **WordNet: Action** nor **SUMO: IntentionalProcess** do not consider atomicity/non-atomicity of an action.

PSI-ULO: AtomicAction own properties:

Not defined

PSI-ULO: AtomicAction relationships:

Subsumption relationships:

PSI-ULO: AtomicAction is subclass of **DOLCE: Endurant** (Section 4.2)

Meronymy relationships:

Not defined

Associations:

PSI-ULO: AtomicAction (1...*) wrappedBy – wraps (0...*) **PSI-ULO: Process** (Section 4.7)

PSI-ULO: AtomicAction (1...*) admissibleAt – admits (0...*) **PSI-ULO: State** (Section 4.11)

PSI-ULO: AtomicAction (1...*) causesChangeOf – changedIn (0...*) **PSI-ULO: Object** (Section 4.17)

PSI-ULO: AtomicAction (0...*) executedBy – executes (0...*) **PSI-ULO: Agent** (Section 4.19)

Changed in v.2.3

PSI-ULO: AtomicAction (~~1...*~~0...*) has – of (~~0...*~~0..1) **PSI-ULO: ActionPattern** (Section 4.34)

One specific kind of an AtomicAction is a Decision. A Decision like an AtomicAction applies a change on the Environment. However, it does it indirectly by choosing one AtomicAction among all admissible AtomicActions in a particular State. More details on admissible actions and states are provided in [17].

PSI-ULO: Decision own properties:

Not defined

PSI-ULO: Decision relationships:

Subsumption relationships:

PSI-ULO: Decision is subclass of **PSI-ULO: AtomicAction**

Meronymy relationships:

Not defined

Associations:

PSI-ULO: Decision (1) takenAt – usedToTake (0...*) **PSI-ULO: State** (Section 4.11)

4.10 A Holon

A Holon in an abstract sense is a compound entity which is a whole and, simultaneously, a part of a larger whole. A philosophical definition of a Holon implicitly assumes that all parts of a Holon are present at any TimeInstant of the existence of this Holon. In PSI Upper-Level ontology we make this assumption explicit by introducing a structural parthood relationship of a Holon to self and making a Holon a subclass of an Endurant. PSI-ULO: Holon has the following subclasses: a TimeInterval and an Object. These subclasses inherit mentioned structural parthood relationship.

Structural parthood differs from temporal parthood, characteristic to Perdurants. Temporal parts of a whole are not simultaneously present in a whole at any arbitrary TimeInstant of the existence of this whole, while structural parts all exist at any TimeInstant of the existence of the whole. Structural parthood relationship subsumes spatial parthood.

The context of the concept of a Holon in the PSI Upper-Level ontology is pictured in Fig. 4.6.

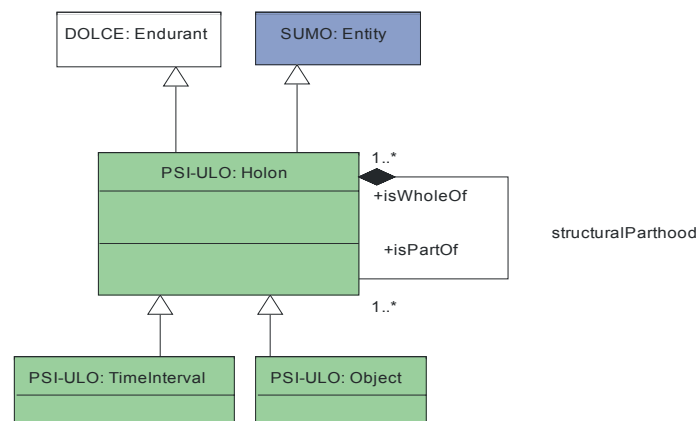


Fig. 4.6: The context of a **Holon** in the PSI Upper-Level ontology.

Given definition of a Holon does not specify if it is material or immaterial. This is why a Holon maps to SUMO: Entity and no more specific mappings are possible.

PSI-ULO: Holon own properties:

Not defined

PSI-ULO: Holon relationships:

Subsumption relationships:

PSI-ULO: Holon is subclass of **DOLCE: Endurant** (Section 4.2)

Holonymy/Meronymy relationships:

PSI-ULO: Holon (1...*) isWholeOf – isPartOf(1...*) **PSI-ULO: Holon**

Comment: This structuralParthood relationship says that a Holon always contains at least one part (possibly - itself) and is a part of at least one larger whole. The parts of the whole and the whole are of the same type as far as this relationship is composition.

Associations:

Not defined

4.11 A State

Any Process, as a pro-active stateful manifestation of a change of the Environment, is guided by its managing Agent to reach the state of affairs in which the constituents of the Environment possess the properties partially or fully matching the Goal of that Agent. We shall say that a Process has reached its target State if such a state of affairs is reached. Otherwise the Process fails to reach its target State. A Goal, if complex, can be decomposed to simpler partial-Goals as often done in problem solving. Such partial-Goals in fact are the states of affairs that should be reached before the overall compound Goal can be attacked. States in PSI Upper-Level ontology are the configurations of the constituents of an Environment. It is considered that a State is reached when the constituents of the Environment have properties with Values in the ranges satisfactory matching the corresponding Goal or partial-Goal of an Agent.

In engineering mentioned Goals are technologically controlled. Therefore States can be seen as technological milestones on the path through the problem solution space leading to the overall Goal. The requirements to the ranges of the property values of the constituents of the Environment are denoted by StatePatterns. StatePatterns are controlled by the Policies of a company which should be based on the standards of the particular industrial sector.

Goals and their partial-Goals may be pursued by taking different alternative paths going through different States. If a problem solution space is represented as a directed graph, a State may have several alternative outgoing edges. These edges correspond to different admissible AtomicActions applying different changes to the Environment. A Decision on the choice of an admissible AtomicAction should be taken to choose the continuation of the path in any State. Such a Decision in the target State chooses among the alternative to terminate the process in success and the alternative to refine the values of the properties of the constituents of the Environment heading to the same target State. Hence, a Decision is a specific AtomicAction which applies changes to an Environment indirectly – by choosing the alternative on the solution path. A Decision is also a mechanism to alter the course of the Process when the Goal or the sub-Goals are dynamically changed.

In difference to an Environment, a State is an Endurant because all its proper parts should be present at any TimeInstant of the presense of a State. A State is not a Holon because a State can not be a proper part of another State.

The context of the concept of a State in the PSI Upper-Level ontology is pictured in Fig. 4.7.

PSI-ULO: State is mapped to WordNet: State and then to **SUMO: Relation** using subsumption relationship.

PSI-ULO: State own properties:

Not defined

PSI-ULO: State relationships:

Subsumption relationships:

PSI-ULO: State is subclass of **DOLCE: Endurant** (Section 4.2)

Meronymy relationships:

Not defined

Associations:

PSI-ULO: State (2...*) **connectedBy** – connects (0...*) **PSI-ULO: Process** (Section 4.7)

Comment: A Process connects at least two different States: an initial State and a target State. Otherwise it is not a Process (a stateful Event).

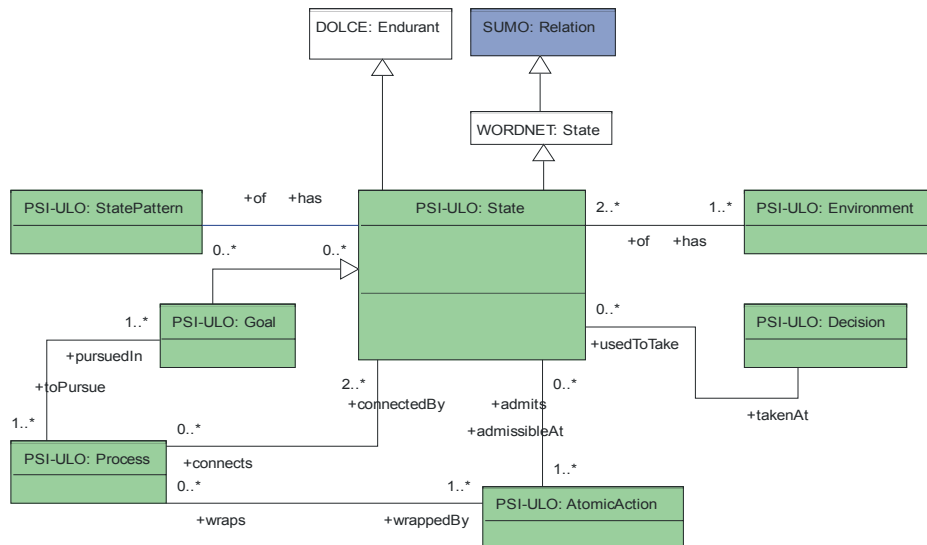


Fig. 4.7: The context of a **State** in the PSI Upper-Level ontology.

PSI-ULO: State (0...*) **admits** – **admissibleAt** (1...*) **PSI-ULO: AtomicAction** (Section 4.9)

Comment: Only AtomicActions which are admissible at at least one State are considered. Those AtomicActions that are not admissible even at one state are not interesting because they will never be applied.

PSI-ULO: State (0...*) **usedToTake** – **takenAt** (1) **PSI-ULO: Decision** (Section 4.9)

PSI-ULO: State (2...*) **of** – **has** (1...*) **PSI-ULO: Environment** (Section 4.8)

Changed in v.2.3

PSI-ULO: State (40...*) **has** – **of** (0...*) **PSI-ULO: StatePattern** (Section 4.35)

4.12 A Goal

A Goal is a subclass of a State (a configuration of the constituents of an Environment) which is desired to be reached by an Agent managing a Process. Please see also Section 4.11 for more details.

PSI-ULO: Goal is mapped through its **PSI-ULO: State** superclass to **WordNet: State** and then to **SUMO: Relation** using subsumption relationship.

PSI-ULO: Goal own properties:

Not defined

PSI-ULO: Goal relationships:

Subsumption relationships:

PSI-ULO: Goal is subclass of **PSI-ULO: State** (Section 4.11)

Meronymy relationships:

Not defined

Associations:

PSI-ULO: Goal (0...*) **pursuedBy** – pursues (0...*) **PSI-ULO: Agent** (Section 4.19)

PSI-ULO: Goal (1...*) **pursuedIn** – **toPursue** (1...*) **PSI-ULO: Process** (Section 4.7)

4.13 A Characteristic and its Specializations

A Characteristic in PSI is a distinguishing quality which has Values and may be checked by one or more Requirements. A characteristic is not an attribute of a concept because the latter inheres to a particular concept, but the former may have different semantic scents for different concepts. For example, a Dependency (a subclass of a Characteristic) may mean a particular form of co-execution for AtomicActions, a causal relationship for Events, or a commitment for Agents. Associations of a Characteristic with other concepts of PSI Upper-Level ontology are not defined due to this very reason. These relationships may have different semantics for different counterparts or even in different ontological contexts. Therefore, these associations are defined lower on in the knowledge hierarchy of PSI Suite of Ontologies – either in the Core or in the Extension ontologies where most appropriate.

PSI Upper-Level ontology defines several specializations of a Characteristic. These specializations appear at the meta-level because of their generic character and multiple semantic scents.

A **Dependency** is a Characteristic that defines the qualities of a relationship among different entities of the same type. For example, we shall say that: one Event depends on the other Event if the latter causes the occurrence of the former [18]; one AtomicAction depends on the other AtomicAction if the former can not be executed without the latter¹²; one Agent depends on the other Agent if the former committed to execute an AtomicAction in the process managed by the latter and so on.

A **Consumption** is a Characteristic that defines either the speed or the volume of the consumption of something – normally a consumable or a renewable resource.

An **Ability** is a Characteristic that defines the ability of an Agent to perform actions.

A **Location** is a Characteristic that defines spatial or affiliation properties of an Object or a Process. A Process may be located within one organization, but also may involve several organizational entities. A Policy may have a local sphere of power within a DevelopmentTeam, but also may have a more widespread character – a National regulatory policy is obligatory for all Organizations in a Country, etc.

The context of the concept of a Characteristic and its subclasses in the PSI Upper-Level ontology is pictured in Fig. 4.8.

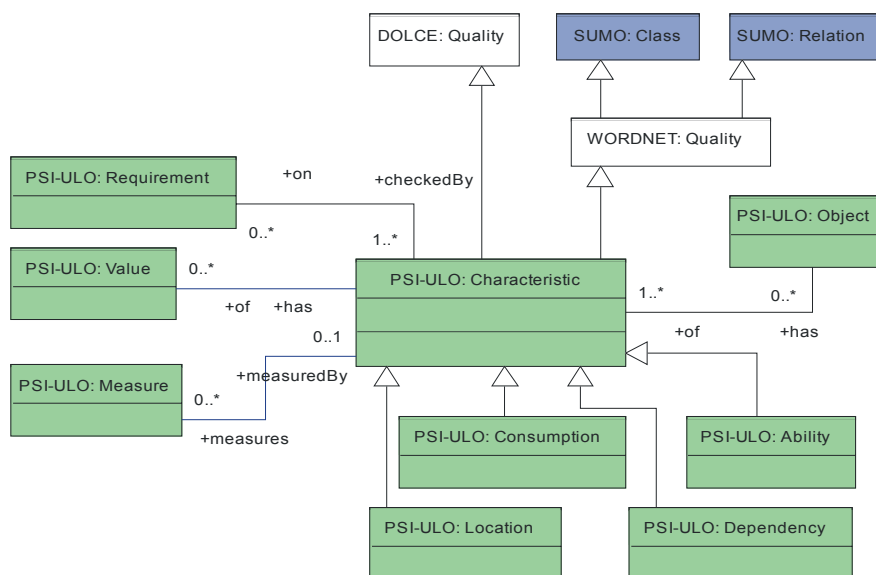


Fig. 4.8: The context of a Characteristic in the PSI Upper-Level ontology.

PSI-ULO: Characteristic subsumes to **DOLCE: Quality** and is mapped to **SUMO: Class** and to **SUMO: Relation** using subsumption.

PSI-ULO: Characteristic own properties:

¹² Different aspects of action to action dependencies are analysed in [17] and further developed in PSI Core Process Ontology [25]

Not defined

PSI-ULO: Characteristic relationships:

Subsumption relationships:

PSI-ULO: Characteristic is subclass of **DOLCE: Quality** (Section 4.4)

Meronymy relationships:

Not defined

Associations:

PSI-ULO: Characteristic (1...*) of – has (0...*) **PSI-ULO: Object** (Section 4.17)

PSI-ULO: Characteristic (1...*) checkedBy – checks (0...*) **PSI-ULO: Requirement** (Section 4.30)

Changed in v.2.3

PSI-ULO: Characteristic (1...*1) has – of (+0...*) **PSI-ULO: Value** (Section 4.14)

PSI-ULO: Characteristic (1...*0..1) verifiedBy – verifiesmeasures(0...*) **PSI-ULO: Measure** (Section 4.14)

PSI-ULO: Location own properties:

Not defined

PSI-ULO: Location relationships:

Subsumption relationships:

PSI-ULO: Location is subclass of **PSI-ULO: Characteristic**

Meronymy relationships:

Not defined

Associations:

Not defined

PSI-ULO: Dependency own properties:

Not defined

PSI-ULO: Dependency relationships:

Subsumption relationships:

PSI-ULO: Dependency is subclass of **PSI-ULO: Characteristic**

Meronymy relationships:

Not defined

Associations:

Not defined

PSI-ULO: Consumption own properties:

Not defined

PSI-ULO: Consumption relationships:

Subsumption relationships:

PSI-ULO: Consumption is subclass of **PSI-ULO: Characteristic**

Meronymy relationships:

Not defined

Associations:

Not defined

PSI-ULO: Ability own properties:

Not defined

PSI-ULO: Ability relationships:

Subsumption relationships:

PSI-ULO: Ability is subclass of **PSI-ULO: Characteristic**

Meronymy relationships:

Not defined

Associations:

Not defined

4.14 A Value and a Measure

A Value is the mapping of an individual Characteristic to the corresponding “quality space” [14] pointing to the position of this individual Characteristic in this space. A Value is semantically equivalent to DOLCE’s concept of a Quale [14]. For example, the Value of an Agent’s Characteristic of Utility may be the quantity of the Units of Welfare [5] collected by the individual Agent in his life time. As a Value maps a particular Characteristic to a specific value space, we are interested in those Characteristics that may have Values. For a Characteristic “having a Value” may mean that: (i) this value is measured; or (ii) this value is assigned; or (iii) this value is computed (e.g. inferred).

A Value in PSI Upper-Level ontology subsumes to **DOLCE: Endurant**. It may seem that a Value is an abstract thing and should subsume to **DOLCE: Abstract**, however it is not the case. A Value is concrete because it has a definite temporal property – it is valid only within a particular TimeInterval. Hence the Value of the validity property of a Value is the mapping to (a) particular TimeInterval(s).

PSI-ULO: Value subsumes to **WordNet: Value** and further on to **SUMO: Quantity**.

PSI-ULO: Value own properties:

Not defined

PSI-ULO: Value relationships:

Subsumption relationships:

PSI-ULO: Value is subclass of **DOLCE: Endurant** (Section 4.2)

Meronymy relationships:

Not defined

Associations:

PSI-ULO: Value (0...*) validWithin – of (1...*) **PSI-ULO: TimeInterval** (Section 4.16)

Changed in v.2.3

PSI-ULO: Value (±0...*) hasof – ofhas (±...*1) **PSI-ULO: Characteristic** (Section 4.13)

A Measure is the subclass of a Value. Measures are Values that are measured. PSI Upper-Level ontology however does not specify the mechanism and the units of measurement. It is elaborated at the lower levels of abstraction.

The notion of a Measure in SUMO+WordNet is polysemic. WordNet: Measure means: (i) the act or process of measuring; or (ii) measuring instrument; or (iii) a reference point against which other things can be evaluated; or (iv) the result of measuring; or even (v) musical notation for a repeating pattern of musical beats. SUMO: Measure is denoted as a generic predicate for asserting that a particular object is measured by a particular quantity. Such a rich polysemy makes direct mapping of PSI Core ontologies to the common sense reference ontology quite difficult and error prone. Our shaker modeling methodology (Section 3) helped in filtering out irrelevant semantic scents of the common sense concept of a Measure. The result is: **PSI-ULO: Measure** subsumes to **PSI-ULO: Value** and further on to **SUMO: Quantity**.

PSI-ULO: Measure own properties:

Not defined

PSI-ULO: Measure relationships:

Subsumption relationships:

PSI-ULO: Measure is subclass of **PSI-ULO: Value**

Meronymy relationships:

Not defined

Associations:

Not defined

4.15 A TimeInstant

A TimeInstant, as specified in [18], is a point in time having no duration. Further refinements of the model of a TimeInstant and its context are provided by PSI Core Time Ontology [30]. A TimeInstant is the subclass of a **DOLCE: Endurant** and is mapped to **SUMO: TimePoint**.

PSI-ULO: TimeInstant own properties:

Not defined

PSI-ULO: TimeInstant relationships:

Subsumption relationships:

PSI-ULO: TimeInstant is subclass of **DOLCE: Endurant** (Section 4.2)

Meronymy relationships:

PSI-ULO: TimeInstant (2...*) isPartOf – contains (1...*) **TimeInterval** (Section 4.16)

Associations:

Not defined

4.16 A TimeInterval

A TimeInterval is a segment of time. At the level of abstraction of PSI Upper-Level ontology no more specific properties of a TimeInterval and no specializations of a TimeInterval (like finite or infinite time intervals) are specified. More details are elaborated in [18]. A TimeInterval is a Holon in the sense that any TimeInterval could be a (structural) part of another TimeInterval and contain other TimeIntervals as its proper (structural) parts. No temporal properties of a TimeInterval exist as far as a TimeInterval is itself a basic entity¹³ for specifying temporal properties of other types of entities. The corollary is that a TimeInterval can not have a temporal parthood relationship with its parts.

In PSI Upper-Level ontology a TimeInterval is not an Object because co-location has no sense for TimeIntervals. No two or more temporally co-located time intervals are possible. For different Objects temporal or structural co-location is allowed.

PSI-ULO: TimeInterval subsumes to **SUMO: TimeInterval** in the sense that it has both an extent and a location on the universal timeline (which is also a TimeInterval in PSI Core Time Ontology). **PSI-ULO: TimeInterval** like **SUMO: TimeInterval** has no gaps.

PSI-ULO: TimeInterval own properties:

Not defined

PSI-ULO: TimeInterval relationships:

Subsumption relationships:

PSI-ULO: TimeInterval is subclass of **PSI-ULO: Holon** (Section 4.10)

¹³ Together with a TimeInstant.

Meronymy relationships:

PSI-ULO: TimeInterval (1...*) contains – isPartOf (2...*) **PSI-ULO: TimeInstant** (Section 4.15)

Comment: By having assumed that a TimeInterval minimally contains two TimeInstants we commit to the discrete character of time.

Associations:

PSI-ULO: TimeInterval (1...*) of – validWithin (0...*) **PSI-ULO: Value** (Section 4.14)

PSI-ULO: TimeInterval (1...*) of – validWithin (0...*) **PSI-ULO: Rule** (Section 4.28)

4.17 An Object

An Object is a Holon which has Environment, belongs to an Environment, and may be changed in the course of the execution of an AtomicAction. An Object may have Characteristics. An Object could be either material or immaterial.

The context of the concept of an Object in the PSI Upper-Level ontology is pictured in Fig. 4.9.

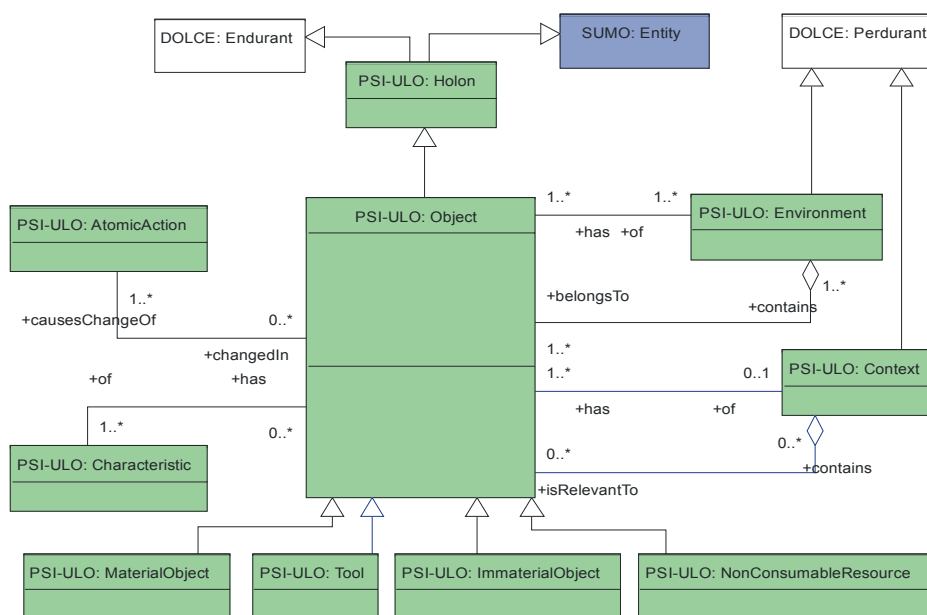


Fig. 4.9: The context of an **Object** in the PSI Upper-Level ontology.

PSI-ULO: Object subsumes to **PSI-ULO: Holon** and is mapped to **SUMO: Entity** through **PSI-ULO: Holon** using subsumption. More specific mapping to SUMO is not possible because an Object can be both material and immaterial.

PSI-ULO: Object own properties:

Not defined

PSI-ULO: Object relationships:

Subsumption relationships:

PSI-ULO: Object is subclass of **PSI-ULO: Holon** (Section 4.10)

Meronymy relationships:

PSI-ULO: Object (1...*) belongsTo – contains (1...*) **PSI-ULO: Environment** (Section 4.8)

New in v.2.3

PSI-ULO: Object (0...*) isRelevantTo – contains (0...*) **PSI-ULO: Context** (Section 4.37)

Associations:

PSI-ULO: Object (1...*) has – of (1...*) **PSI-ULO: Environment** (Section 4.8)

PSI-ULO: Object (0...*) changedIn – causesChangeof (1...*) **PSI-ULO: AtomicAction** (Section 4.9)

PSI-ULO: Object (0...*) has – of (1...*) **PSI-ULO: Characteristic** (Section 4.13)

New in v.2.3

PSI-ULO: Object (1...*) has – of (0..1) **PSI-ULO: Context** (Section 4.37)

4.18 A MaterialObject

MaterialObjects are those Objects which are physically or legally substantial in the sense that they possess tangible physical non-temporal properties like mass, color, shape, size, speed, usage right and can not be copied or duplicated without borrowing a definite amount of physical or legal¹⁴ substance for it. The law of conservation of matter is applicable to material objects. In PSI Upper-Level ontology MaterialObject subclasses are an Agent, a MaterialArtifact, a ConsumableResource.

The context of the concept of a MaterialObject in the PSI Upper-Level ontology is pictured in Fig. 4.10.

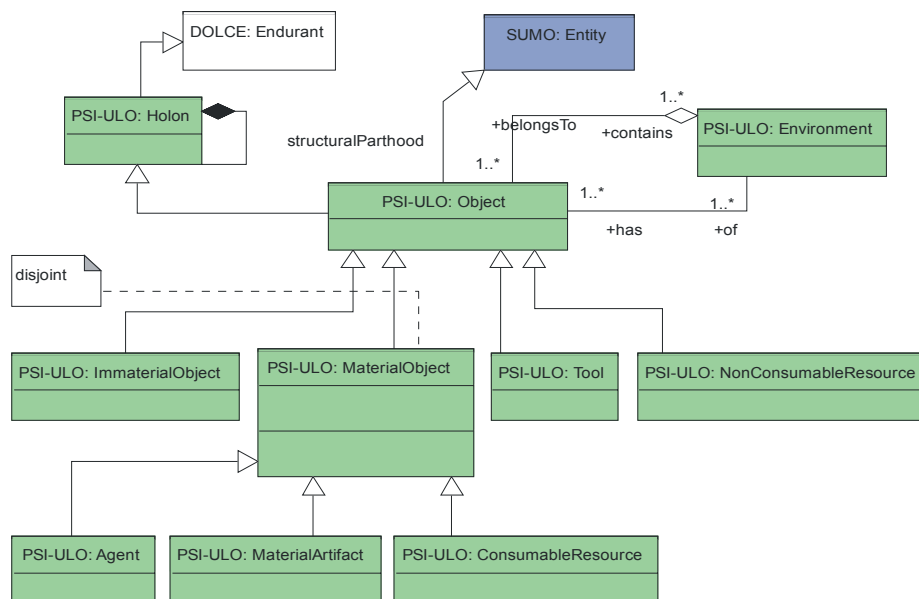


Fig. 4.10: The context of a **MaterialObject** in the PSI Upper-Level ontology.

PSI-ULO: MaterialObject subsumes to **PSI-ULO: Object** and is mapped to **SUMO: Entity** through **PSI-ULO: Holon** using subsumption. More specific mapping to SUMO is not possible because a MaterialObject can be both physical and abstract (i.e., having no spatial and temporal properties) in the sense of SUMO.

PSI-ULO: MaterialObject own properties:

Not defined

PSI-ULO: MaterialObject relationships:

Subsumption relationships:

PSI-ULO: MaterialObject is subclass of **PSI-ULO: Object** (Section 4.17)

New in v.2.3

¹⁴ By an odd term of “legal substance” we mean a legal permission to have an extra copy of an Object which is not a physical object in the sense of SUMO or DOLCE. A good example of such an Object is a software program having a license. Though such a program could be easily copied it is not legally allowed. One has to spend a consumable (i.e. material) resource to receive a legal permission for having an extra copy. This is why such a non-physical object is considered a MaterialObject in PSI Upper-Level ontology. Of course, all physical objects are MaterialObjects as well. From the other hand a software program which can be freely copied is an ImmaterialObject in PSI Upper-Level ontology.

Restriction: The sets of instances of a **PSI-ULO: MaterialObject** and a **PSI-ULO: ImmaterialObject** with respect to the subsumption to **PSI-ULO: Object** are disjoint – i.e. the intersection of these sets is empty. The union of these sets of instances is not complete because there is also the set of instances of the concept of a **PSI-ULO: Tool**, that could be either material or immaterial (Fig. 4.10).

Meronymy relationships:

Not defined

Associations:

Not defined

4.19 An Agent

An Agent is a MaterialObject that possesses pro-activity, is able to execute AtomicActions and to manage Processes. Pro-activity of an Agent is revealed in pursuing Goals of changing the Environment to a desired State. An Agent is the only entity which can change an Environment by executing AtomicActions applied to the Objects belonging to the Environment. An Agent has Beliefs about Environment(s) which are the hypotheses believed to be true by an Agent. These Beliefs may further become Facts if confirmed by the happenings perceived by the Agents (please see PSI Environment, Event, and Happening ontology [30] for more details). Beliefs together with desires and intentions are important basic elements forming the behavior of an Agent. This behavior is regulated by BehaviorPatterns specified as rules.

An Agent is an abstract entity which is a generic model for an individual person (a manager, a designer), a group of persons or artificial agents acting on behalf of physical persons (a team or an organizational unit), or an external proactive entity influencing the Environment(s) of an observed Process in a definite way. These aspects of an Agent are specialized and refined at the lower abstraction levels in PSI Core ontologies.

The context of the concept of an Agent in the PSI Upper-Level ontology is pictured in Fig. 4.11.

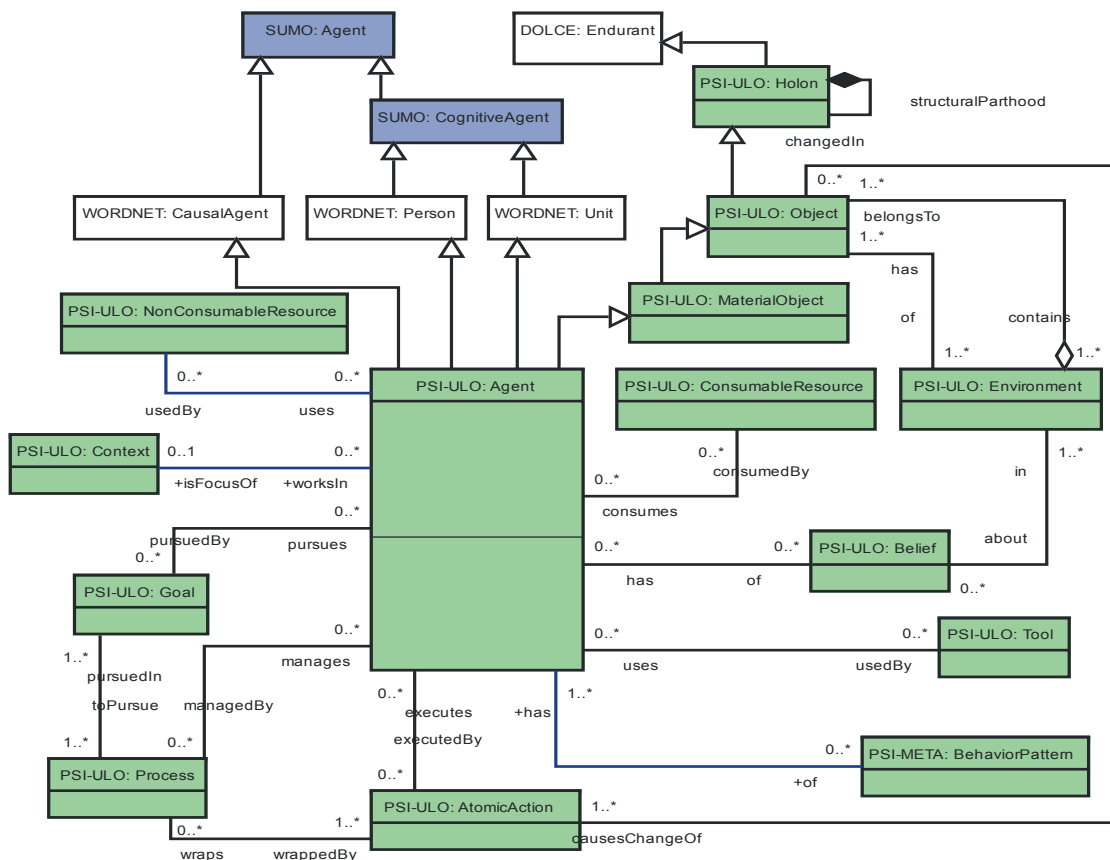


Fig. 4.11: The context of an Agent in the PSI Upper-Level ontology.

PSI-ULO: Agent subsumes to **PSI-ULO: MaterialObject** and is mapped to:

- **WordNet: Person** → **SUMO: CognitiveAgent** → **SUMO: Agent** if **PSI-ULO: Agent** is viewed as an individual agent
- **WordNet: Unit** → **SUMO: CognitiveAgent** → **SUMO: Agent** if **PSI-ULO: Agent** is viewed as a group of agents
- **WordNet: CausalAgent** → **SUMO: Agent** if it is viewed as an external pro-active cause of a change in the Environment

PSI-ULO: Agent own properties:

Not defined

PSI-ULO: Agent relationships:

Subsumption relationships:

PSI-ULO: Agent is subclass of **PSI-ULO: MaterialObject** (Section 4.18)

Meronymy relationships:

Not defined

Associations:

PSI-ULO: Agent (0...*) pursues – pursuedBy (0...*) **PSI-ULO: Goal** (Section 4.12)

PSI-ULO: Agent (0...*) manages – managedBy (0...*) **PSI-ULO: Process** (Section 4.7)

PSI-ULO: Agent (0...*) executes – executedBy (0...*) **PSI-ULO: AtomicAction** (Section 4.9)

PSI-ULO: Agent (0...*) uses – usedBy (0...*) **PSI-ULO: Tool** (Section 4.22)

PSI-ULO: Agent (0...*) has – of (0...*) **PSI-ULO: Belief** (Section 4.27)

PSI-ULO: Agent (0...*) consumes – consumedBy (0...*) **PSI-ULO: ConsumableResource** (Section 4.21)

PSI-ULO: Agent (1...*) has – of (0...*) **PSI-ULO: BehaviorPattern** (Section 4.32)

New in v.2.3

PSI-ULO: Agent (0...*) uses – usedBy (0...*) **PSI-ULO: NonConsumableResource** (Section 4.39)

PSI-ULO: Agent (0...*) worksIn – isFocusOf (0..1) **PSI-ULO: Context** (Section 4.37)

4.20 A MaterialArtifact

A **MaterialArtifact** is an artifact which is a **MaterialObject**. A **MaterialArtifact** differs from other **MaterialObjects** by the fact that it has been produced artificially and in the **Process** having the same **Environment** to which this **MaterialArtifact** belongs.

PSI-ULO: MaterialArtifact subsumes to **PSI-ULO: MaterialObject** and subsumes to **WordNet: Artifact** that is a man-made object taken as a whole. **WordNet: Artifact** further subsumes to **SUMO: Artifact** that is a corpuscular object that is the product of a making. More specific mapping of a **MaterialArtifact** to **WordNet** or **SUMO** is not possible because neither **WordNet** nor **SUMO** distinguish between material and immaterial artifacts.

PSI-ULO: MaterialArtifact own properties:

Not defined

PSI-ULO: MaterialArtifact relationships:

Subsumption relationships:

PSI-ULO: MaterialArtifact is subclass of **PSI-ULO: MaterialObject** (Section 4.18)

Meronymy relationships:

Not defined

Associations:

Not defined

4.21 A ConsumableResource

A ConsumableResource is a MaterialObject which is consumed by an Agent while executing an AtomicAction. A **PSI-ULO:ConsumableResource** maps to **SUMO: Resource** though it is erroneously considered in SUMO that a Resource is consumed by a process, but not by an agent. We consider that such a subsumption is correct because **SUMO: Resource** properties are changed in a Process – i.e. it is consumed.

PSI-ULO: ConsumableResource own properties:

Not defined

PSI-ULO: ConsumableResource relationships:

Subsumption relationships:

PSI-ULO: ConsumableResource is subclass of **PSI-ULO: MaterialObject** (Section 4.18)

Meronymy relationships:

Not defined

Associations:

PSI-ULO: ConsumableResource (0...*) consumedBy – consumes (0...*) **PSI-ULO: Agent** (Section 4.19)

4.22 A Tool

A Tool is an Object which is used as an instrument by an Agent to execute an AtomicAction. A Tool could either be material or immaterial. A Tool differs from a ConsumableResource by the fact that the properties of a Tool are not changed when it is used. For example, if a software tool is used by a designer to do logical verification of a functional block then this software tool is not consumed by the designer. However, the license of this software tool may be consumed by the designer. The license of this Tool is therefore the instance of a ConsumableResource. Mentioned software tool still remains material (having legal substance associated with it) because one can not copy it without borrowing a new license (a certain amount of legal substance). An example of an immaterial Tool is a copy of free software. A Tool, even if it is immaterial, differs from a NonConsumableResource (Section 4.39). A Tool is an instrument for executing the AtomicAction while a NonConsumableResource is an input for the AtomicAction that is not consumed in this action.

A **PSI-ULO: Tool** subsumes to **SUMO: Instrument**.

PSI-ULO: Tool own properties:

Not defined

PSI-ULO: Tool relationships:

Subsumption relationships:

Changed in v.2.3

PSI-ULO: Tool is subclass of **PSI-ULO: MaterialObject** (Section 4.17)

Meronymy relationships:

Not defined

Associations:

PSI-ULO: Tool (0...*) usedBy – uses (0...*) **PSI-ULO: Agent** (Section 4.19)

4.23 An ImmaterialObject

ImmaterialObjects in contract to MaterialObjects are not substantial in physical or legal sense. Hence, they can be copied or duplicated without consuming physical or legal substance for that. Please see Section 4.18 for more details. In PSI Upper-Level ontology the subclasses of an ImmaterialObject are an ImmaterialArtifact, a Rule, a Plan, a Fact, a Belief, a Representation.

The context of the concept of an ImmaterialObject in the PSI Upper-Level ontology is pictured in Fig. 4.12.

PSI-ULO: ImmaterialObject subsumes to PSI-ULO: Object and is mapped to SUMO: Entity through PSI-ULO: Holon using subsumption. More specific mapping to SUMO is not possible because an ImmaterialObject can be both concrete (being perceived by the senses, not abstract or imaginary) and abstract (i.e., having no spatial and temporal properties) in the sense of SUMO. For example, as an ImmaterialObject may be a part of an Environment at a particular TimeInstant of the Existence of this Environment, this ImmaterialObject may be perceived as a part of this Environment.

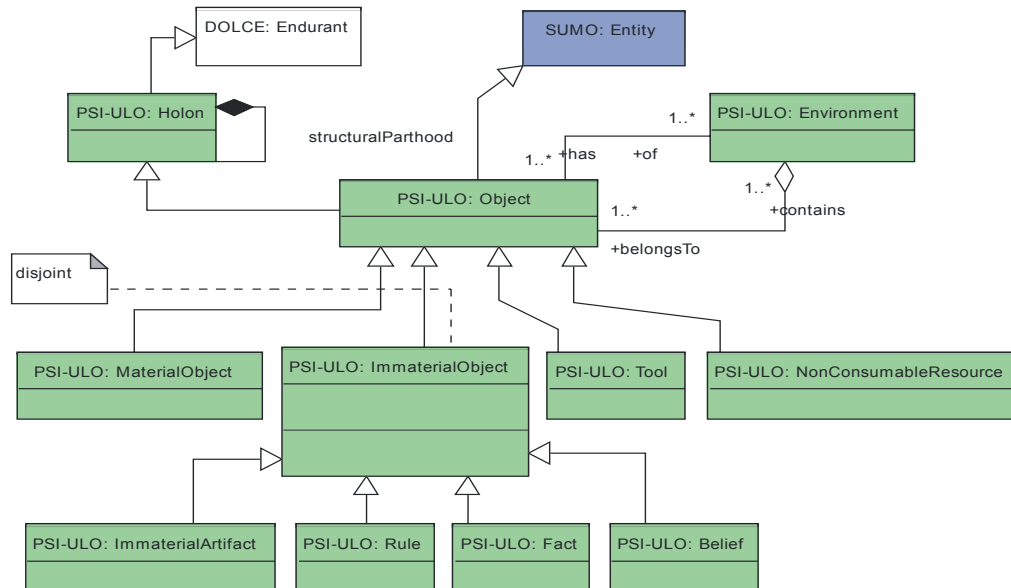


Fig. 4.12: The context of an **ImmaterialObject** in the PSI Upper-Level ontology.

PSI-ULO: ImmaterialObject own properties:

Not defined

PSI-ULO: ImmaterialObject relationships:

Subsumption relationships:

PSI-ULO: ImmaterialObject is subclass of **PSI-ULO: Object** (Section 4.17)

Meronymy relationships:

Not defined

Associations:

Not defined

4.24 An ImmaterialArtifact

An ImmaterialArtifact is an artifact that is not substantial in physical or legal sense. An ImmaterialArtifact differs from other ImmaterialObjects by the fact that it has been produced artificially and in the Process having the same Environment to which this ImmaterialArtifact belongs.

PSI-ULO: ImmaterialArtifact subsumes to **PSI-ULO: ImmaterialObject**. **PSI-ULO: ImmaterialArtifact** is mapped (using subsumption) to a **WordNet: Artifact** that is a man-made object taken as a whole. **WordNet: Artifact** further subsumes to **SUMO: Artifact** that is a corpuscular object been the product of a making. More specific mapping of an ImmaterialArtifact to WordNet or SUMO is not possible because neither WordNet nor SUMO distinguish between material and immaterial artifacts.

PSI-ULO: ImmaterialArtifact own properties:

Not defined

PSI-ULO: ImmaterialArtifact relationships:

Subsumption relationships:

PSI-ULO: ImmaterialArtifact is subclass of **PSI-ULO: ImmaterialObject** (Section 4.23)

Meronymy relationships:

Not defined

Associations:

Not defined

4.25 A Plan and a Strategy

A Plan is a specification of a Process. A Plan is used by an Agent having an intention of reaching a particular Goal at some future time. A Plan as a specification is an ImmaterialObject because it can be copied or duplicated without borrowing substance for that. The model of a Plan is further elaborated at the level of PSI Core. The semantics of **PSI-ULO: Plan** is equivalent to **SUMO: Plan**.

PSI-ULO: Plan own properties:

Not defined

PSI-ULO: Plan relationships:

Subsumption relationships:

Changed in v.2.3

PSI-ULO: Plan is subclass of **PSI-ULO: ~~ImmaterialObject~~ImmaterialArtifact** (Section 4.24)

Meronymy relationships:

Not defined

Associations:

Not defined

A Strategy is a subclass of a Plan. A Strategy is a high-level indicative Plan used to shape out organization-level objectives. The concept of a Strategy is refined in PRODUKTIV+ Performance Ontology [26].

PSI-ULO: Strategy own properties:

Not defined

PSI-ULO: Strategy relationships:

Subsumption relationships:

PSI-ULO: Strategy is subclass of **PSI-ULO: Plan**

Meronymy relationships:

Not defined

Associations:

Not defined

4.26 A Fact

A Fact is a statement or assertion of verified information about something that is the case or has happened. The semantics of **PSI-ULO: Fact** is equivalent to the semantics of **SUMO: Fact**.

Facts are ImmaterialObjects because they may belong to an Environment. Therefore the truth of Facts may change in time and Facts have life time in a particular Environment. As mentioned in Section 4.19, there is a relationship between Beliefs and Facts. Beliefs may become Facts if confirmed by the happenings perceived by the Agents in their Environment(s). This relationship is not explicitly specified at the level of abstraction of PSI Upper-Level ontology.

PSI-ULO: Fact own properties:

Not defined

PSI-ULO: Fact relationships:

Subsumption relationships:

PSI-ULO: Fact is subclass of **PSI-ULO: ImmaterialObject** (Section 4.23)

Meronymy relationships:

Not defined

Associations:

Not defined

4.27 A Belief

Beliefs are propositions or assertions about an Environment which are held to be true by an Agent for a certain TimeInterval. Beliefs are ImmaterialObjects because they may be parts of an Environment. Beliefs may change in time because of the changes in the Environment and the perception of these changes by the Agent. As mentioned in Section 4.19, there is a relationship between Beliefs and Facts. Beliefs may become Facts if confirmed by the happenings perceived by the Agents in their Environment(s). This relationship is not explicitly specified at the level of abstraction of PSI Upper-Level ontology.

PSI-ULO: Belief maps by subsumption to **WordNet: Belief**, that is any cognitive content held as true, and further to **SUMO: Proposition**.

PSI-ULO: Belief own properties:

Not defined

PSI-ULO: Belief relationships:

Subsumption relationships:

PSI-ULO: Belief is subclass of **PSI-ULO: ImmaterialObject** (Section 4.23)

Meronymy relationships:

Not defined

Associations:

PSI-ULO: Belief (0...*) of – has (0...*) **PSI-ULO: Agent** (Section 4.19)

PSI-ULO: Belief (0...*) about – in (1...*) **PSI-ULO: Environment** (Section 4.8)

4.28 A Rule

The concept of a Rule in PSI Upper-Level ontology is a principle, a condition, a procedure, a generic pattern, or a norm regulating possible process, action, behavior, or a state of affairs. As far as a **PSI-ULO: Rule** subsumes to a **PSI-ULO: ImmaterialObject**, to a **PSI-ULO: Object** and to a **PSI-ULO: Holon**, it inherits the structural parthood relationship of a **PSI-ULO: Holon**. Hence, a Rule may be an atomic proposition or a more complex composition of other Rules. As far as a Rule is an Endurant no temporal parthood relationships are allowed for its proper parts – the composition of a rule can not be changed in time. A Rule itself still has a temporal property of validity – it is valid within a particular TimeInterval or several particular TimeIntervals.

If a Rule is a principle or condition that customarily governs behavior then it subsumes to **WordNet: Rule** and further on to **SUMO: Proposition**. If a Rule is a generalization that describes recurring facts or events then it subsumes to **WordNet: Law** and further on to **SUMO: Proposition**. If a Rule is something regarded as a norm constraining possible action or behavior then it subsumes to **WordNet: Regulation** and further on to **SUMO: Proposition**.

The context of the concept of a Rule in the PSI Upper-Level ontology is pictured in Fig. 4.13.

PSI-ULO: Rule own properties:

Not defined

PSI-ULO: Rule relationships:

Subsumption relationships:

PSI-ULO: Rule is subclass of **PSI-ULO: ImmaterialObject** (Section 4.23)

Meronymy relationships:

Not defined

Associations:

Changed in v.2.3

PSI-ULO: Rule (1...*0...*) validWithin – of (0...*1...*) **PSI-ULO: TimeInterval** (Section 4.16)

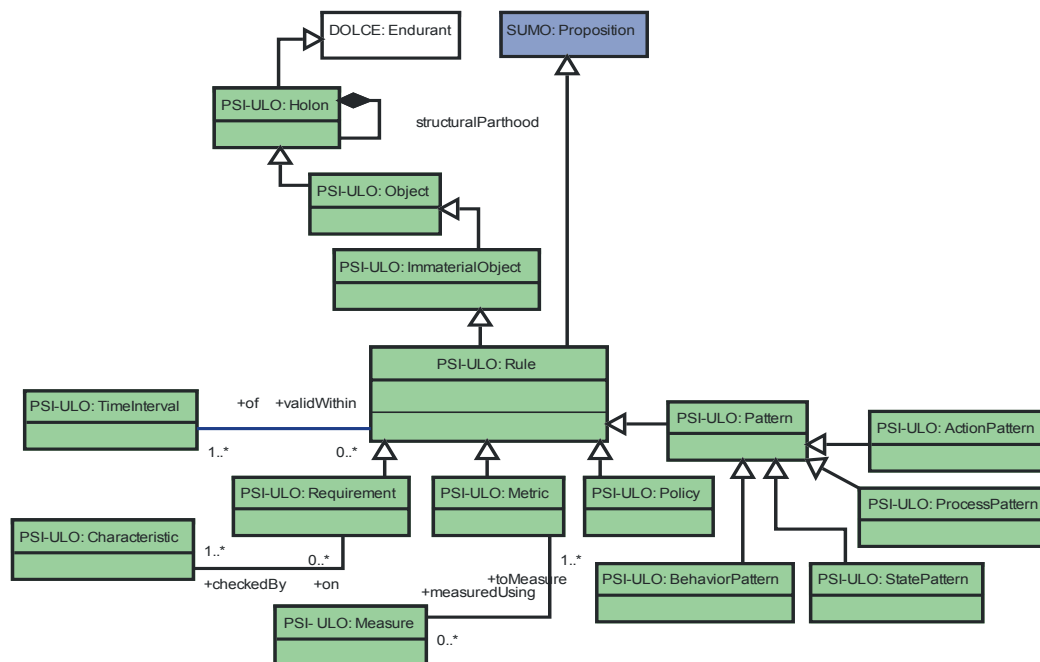


Fig. 4.13: The context of a **Rule** in the PSI Upper-Level ontology.

4.29 A Metric

A Metric is a Rule specifying the procedure of receiving the Measures of Characteristics.

PSI-ULO: Metric subsumes to a **PSI-ULO: Rule** and maps by subsumption to a **SUMO: Proposition**.

PSI-ULO: Metric own properties:

Not defined

PSI-ULO: Metric relationships:

Subsumption relationships:

PSI-ULO: Metric is subclass of **PSI-ULO: Rule** (Section 4.28)

Meronymy relationships:

Not defined

Associations:

PSI-ULO: Metric (1...*) toMeasure – measuredUsing (0...*) **PSI-ULO: Measure** (Section 4.14)

4.30 A Requirement

A Requirement is a Rule specifying the ranges of satisfactory Values of Characteristics.

PSI-ULO: Requirement subsumes to a **PSI-ULO: Rule** and maps by subsumption to a **SUMO: Proposition**.

PSI-ULO: Requirement own properties:

Not defined

PSI-ULO: Requirement relationships:

Subsumption relationships:

PSI-ULO: Requirement is subclass of **PSI-ULO: Rule** (Section 4.28)

Meronymy relationships:

Not defined

Associations:

PSI-ULO: Requirement (0...*) on – checkedBy (1...*) **PSI-ULO: Characteristic** (Section 4.13)

4.31 A Pattern

A Pattern is a Rule specifying normative or other rational constraints on potentially possible variants of behaviors, Processes, AtomicActions, and States.

PSI-ULO: Pattern subsumes to a **PSI-ULO: Rule** and maps by subsumption to a **SUMO: Proposition**.

PSI-ULO: Pattern own properties:

Not defined

PSI-ULO: Pattern relationships:

Subsumption relationships:

PSI-ULO: Pattern is subclass of **PSI-ULO: Rule** (Section 4.28)

Meronymy relationships:

Not defined

Associations:

Not defined

4.32 A BehaviorPattern

A BehaviorPattern is a Pattern of behavior. It is a specification of generic ways of acting required or expected of an Agent. When reasoning about choosing the most appropriate behavior in a particular situation an Agent searches for the best possible match among its BehaviorPatterns. For example, a BehaviorPattern for a bidder in a Vickrey auction is to bid using his or her true valuation of a lot¹⁵.

PSI-ULO: BehaviorPattern subsumes to a **PSI-ULO: Pattern** and maps by subsumption to a **SUMO: BehavioralModel** and, further to **SUMO: Proposition**. **SUMO: BehavioralModel** of a module characterizes only the module external interactions and their relationship constrained by the module. It ignores completely the module internal structure, e.g., it characterizes the module just as a ‘black-box’.

PSI-ULO: BehaviorPattern own properties:

Not defined

PSI-ULO: BehaviorPattern relationships:

Subsumption relationships:

PSI-ULO: BehaviorPattern is subclass of **PSI-ULO: Pattern** (Section 4.31)

Meronymy relationships:

Not defined

Associations:

PSI-ULO: BehaviorPattern (0...*) controlledBy – controls (0...*) **PSI-ULO: Policy** (Section 4.36)

Changed in v.2.3

¹⁵ Bidding true valuations is the dominant strategy in Vickrey auction – <http://www.gametheory.net/dictionary/Auctions/VickreyAuction.html>

PSI-ULO: BehaviorPattern (0...*) of – has (1...*0...*) PSI-ULO: Agent (Section 4.19)

4.33 A ProcessPattern

A ProcessPattern is a Pattern for a type of Processes describing (minimally) the common or the generic pre-conditions allowing the Process to be commenced, the stages characteristic to this type of Processes and corresponding typical States in which these stages have to be accomplished. ProcessPatterns are further elaborated in PSI Task-Activity Core ontology.

PSI-ULO: ProcessPattern subsumes to a **PSI-ULO: Pattern** and maps by subsumption to a **SUMO: Procedure** that is a sequence-dependent specification.

PSI-ULO: ProcessPattern own properties:

Not defined

PSI-ULO: ProcessPattern relationships:

Subsumption relationships:

PSI-ULO: ProcessPattern is subclass of **PSI-ULO: Pattern** (Section 4.31)

Meronymy relationships:

Not defined

Associations:

PSI-ULO: ProcessPattern (0...*) controlledBy – controls (0...*) PSI-ULO: Policy (Section 4.36)

Changed in v.2.3

PSI-ULO: ProcessPattern (1) of – has (1...*0...*) PSI-ULO: Process (Section 4.7)

4.34 An ActionPattern

An ActionPattern is a Pattern for a type of AtomicActions describing (minimally) the input types, the procedure of execution, required resource types, Tool types, and the output type. Figurally speaking an ActionPattern is a sort of a “script” for executing AtomicActions of a particular type. ActionPatterns are further elaborated in PSI Task-Activity Core ontology.

PSI-ULO: ActionPattern subsumes to a **PSI-ULO: Pattern** and maps by subsumption to a **SUMO: Procedure** that is a sequence-dependent specification.

PSI-ULO: ActionPattern own properties:

Not defined

PSI-ULO: ActionPattern relationships:

Subsumption relationships:

PSI-ULO: ActionPattern is subclass of **PSI-ULO: Pattern** (Section 4.31)

Meronymy relationships:

Not defined

Associations:

PSI-ULO: ActionPattern (0...*) controlledBy – controls (0...*) PSI-ULO: Policy (Section 4.36)

Changed in v.2.3

PSI-ULO: ActionPattern (0...*0..1) of – has (1...*0...*) PSI-ULO: AtomicAction (Section 4.9)

4.35 A StatePattern

A StatePattern is a Pattern for a type of States describing generic structural properties of such States and generic (technological) requirements to the characteristics of the constituents of these States. Normally a StatePattern is a

generic description of all the States which are reached after a particular technological phase is accomplished. For example, a **StatePattern** describing all possible States in which a designed microelectronic device is available in the forms (representations) of a logical layout and netlist is the **Pattern** for the final logical design State. **StatePatterns** are further elaborated in **PSI Task-Activity** and **Design Artifact Core** ontologies.

PSI-ULO: StatePattern subsumes to a **PSI-ULO: Pattern** and maps by subsumption to a **SUMO: Proposition**.

PSI-ULO: StatePattern own properties:

Not defined

PSI-ULO: StatePattern relationships:

Subsumption relationships:

PSI-ULO: StatePattern is subclass of **PSI-ULO: Pattern** (Section 4.31)

Meronymy relationships:

Not defined

Associations:

Changed in v.2.3

PSI-ULO: StatePattern (0...*) of – has (1...*0...*) **PSI-ULO: State** (Section 4.11)

4.36 A Policy

A **Policy** is a basic regulatory **Rule** (a norm) used in specifying **Patterns**.

PSI-ULO: Policy subsumes to a **PSI-ULO: Rule** and maps by subsumption to a **SUMO: Proposition**.

PSI-ULO: Policy own properties:

Not defined

PSI-ULO: Policy relationships:

Subsumption relationships:

PSI-ULO: Policy is subclass of **PSI-ULO: Rule** (Section 4.28)

Meronymy relationships:

Not defined

Associations:

PSI-ULO: Policy (0...*) controls – controlledBy (0...*) **PSI-ULO: BehaviorPattern** (Section 4.32)

PSI-ULO: Policy (0...*) controls – controlledBy (0...*) **PSI-ULO: ProcessPattern** (Section 4.33)

PSI-ULO: Policy (0...*) controls – controlledBy (0...*) **PSI-ULO: ActionPattern** (Section 4.34)

New in v.2.3

PSI-ULO: Policy (0...*) regulates – regulatedBy (0...*) **PSI-ULO: Commitment** (Section 4.38)

4.37 A Context

A **Context** is the new concept in the **PSI Upper-Level ontology v.2.3**.

A **Context** of an entity (that has the context) is the selection of related things that facilitate interpreting, using, or performing the entity having this context in a pragmatic way. For denoting a **Context** it is therefore required to answer:

(i) The **Context** of what is specified?

A **PSI-ULO: Context** could be either of a **PSI-ULO: Process** or of a **PSI-ULO: Object**. Examples are: the context of a development team affiliated to an organization, the context of a project, the context of the development process of the configurable multimedia controller.

(ii) What is relevant for the inclusion in the **Context**?

A **PSI-ULO: Context** may contain the instances of a **PSI-ULO: Process** or of a **PSI-ULO: Object** as relevant components. Examples are: (a) the context of the process of engineering design may contain the members of the development team, the manager, the resources used or consumed, the tools used, the design artifact under development; (b) the context of the development team (subclass of an object) may contain the design processes performed by the team, the organization to which the team is affiliated, the tools and the resources, etc.

(iii) Who uses the Context?

A **PSI-ULO: Context** is used by a **PSI-ULO: Agent** to define the current working focus and determine working priorities. For example if a manager supervises several design projects he has to concentrate on each of them at different time. When he is focused on a particular project the things relevant to the context of this project become more important. Therefore we may say that the manager has switched his work to the context of this project and considers that his actions applied to the items relevant to the chosen context are of the higher priority than the actions applied to the other contexts.

The composition of a **PSI-ULO: Context** may be changed in time. Therefore a **PSI-ULO: Context**, like a **PSI-ULO: Event** or a **PSI-ULO: Environment**, subsumes to a **DOLCE: Perdurant**. A **PSI-ULO: Context** is mapped to a **SUMO: Entity** using subsumption.

The semantic context of the concept of a Context is pictured in Fig. 4.14.

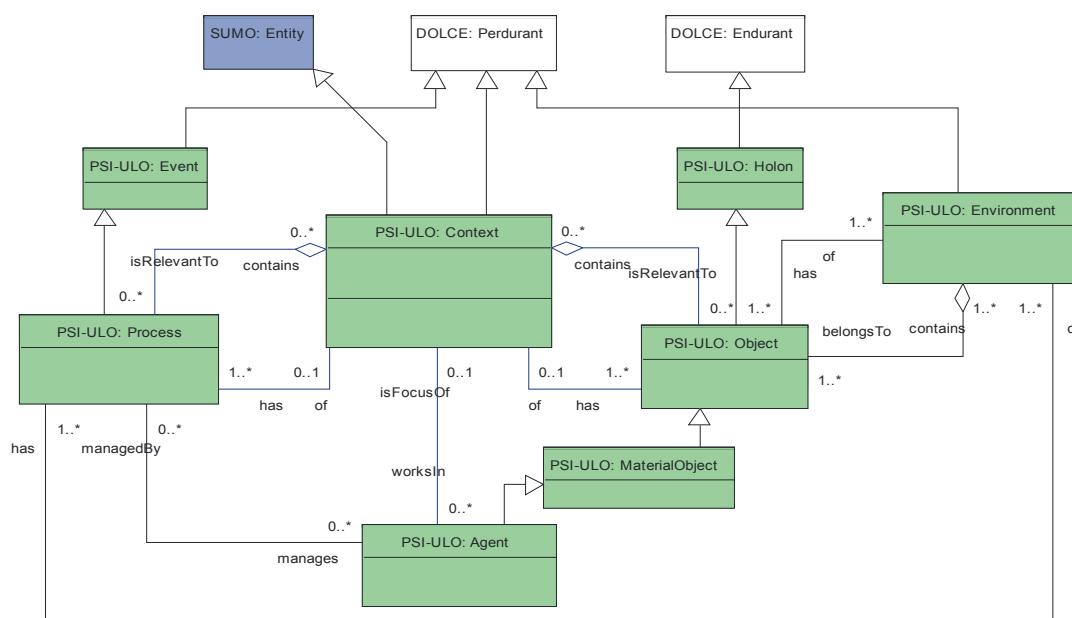


Fig. 4.14: The context of a **Context** in the PSI Upper-Level ontology.

PSI-ULO: Context own properties:

Not defined

PSI-ULO: Context relationships

Subsumption relationships:

PSI-ULO: Context is subclass of **DOLCE: Perdurant** (Section 4.3)

Holonymy relationships:

PSI-ULO: Context (0..*) contains – isRelevantTo (0..*) **PSI-ULO: Process** (Section 4.7)

PSI-ULO: Context (0..*) contains – isRelevantTo (0..*) **PSI-ULO: Object** (Section 4.17)

Associations:

PSI-ULO: Context (0..1) of – has (1..*) **PSI-ULO: Process** (Section 4.7)

PSI-ULO: Context (0..1) of – has (1..*) **PSI-ULO: Object** (Section 4.17)

PSI-ULO: Context (0..1) isFocusOf – worksIn (0..*) **PSI-ULO: Agent** (Section 4.19)

4.38 A Commitment

A Commitment is the new concept in the PSI Upper-Level ontology v.2.3

A Commitment of an Agent is the binding the Agent to the course of action, a plan, a pattern of behavior, an organizational structure. The semantically close SUMO concept to a Commitment is a **SUMO: Cooperation**. However it needs to be noted that a **SUMO: Cooperation** is a process but a **PSI-ULO: Commitment** is a binding self-obligation that is made public and is regulated by a **PSI-ULO: Policy**. No more refinements to the model of a commitment are done in the PSI Upper-Level ontology. The model is further developed at the level of PSI Core.

A **PSI-ULO: Commitment** subsumes to a **PSI-ULO: Dependency** and maps through subsumption to a **WordNet: Obligation** and further to a **SUMO: Relation**.

PSI-ULO: Commitment own properties:

Not defined

PSI-ULO: Commitment relationships

Subsumption relationships:

PSI-ULO: Commitment is subclass of **PSI-ULO: Dependency** (Section 4.13)

Meronymy relationships:

Not defined

Associations:

PSI-ULO: Commitment (0...*) regulatedBy – regulates (0...*) **PSI-ULO: Policy** (Section 4.36)

4.39 A NonConsumableResource

A **PSI-ULO: NonConsumableResource** is the new concept in the PSI Upper-Level ontology v.2.3.

A NonConsumableResource is an Object which is used by an Agent while executing an AtomicAction. In difference to a ConsumableResource that is material in physical or legal sense and is consumed in an AtomicAction by an Agent. A NonConsumableResource:

- Could be either material or immaterial
- Is not consumed in an AtomicAction, but is used by an Agent as an input for an AtomicAction.

A **PSI-ULO: NonConsumableResource** subsumes to a **PSI-ULO: Object** and maps through subsumption to a **SUMO: Entity**.

PSI-ULO: NonConsumableResource own properties:

Not defined

PSI-ULO: NonConsumableResource relationships:

Subsumption relationships:

PSI-ULO: NonConsumableResource is subclass of **PSI-ULO: Object** (Section 4.17)

Meronymy relationships:

Not defined

Associations:

PSI-ULO: NonConsumableResource (0...*) usedBy – uses (0...*) **PSI-ULO: Agent** (Section 4.19)

5 Mapping PSI Core and Extension Ontologies to Foundational Theories

PSI Suite of Ontologies v.2.3 [30] comprises the Core set of ontologies and the Extensions. The Core the Time ontology, the Environment, Event, and Happening ontology, the Actor ontology, the Organization ontology, the Process Pattern ontology, the Process ontology, the Design Artifact ontology, and the Design Artifact Complexity and Quality ontology. The Extensions are the IMS Resource ontology, the IMS Library ontology, the IMS Tool ontology, the Ability ontology, the Generic Negotiation ontology, and the Software Tool Evaluation ontology.

This Section describes the results of the mapping of the concepts of these ontologies to the foundational ontologies. The mappings are done using subsumption relationships. The rationale for the choice of the foundational ontologies (DOLCE [14] and SUMO [11] plus WordNet [12]) is described in Section 2. These mappings have been produced in the Commonsense Alignment phase of the Shaker Modeling Methodology (Section 3)

5.1 The Mappings of the PSI Core Ontologies

PSI Time ontology v.2.3 [30, Section 5.1] is based on the medium expressiveness subset of PSI Theory of Time [18]. It comprises the description of the relationships of the Allen's Time Interval Calculus, the means to cope with durations and periodic intervals of time. The definitions of all concepts in the PSI Time ontology are based on the concepts of a TimeInstant and a TimeInterval of the PSI Upper-Level ontology as pictured in Fig. 5.1. The mappings by subsumption relationships of the top-level concepts of the Time ontology to the concepts of DOLCE and SUMO are also given in Fig. 5.1.

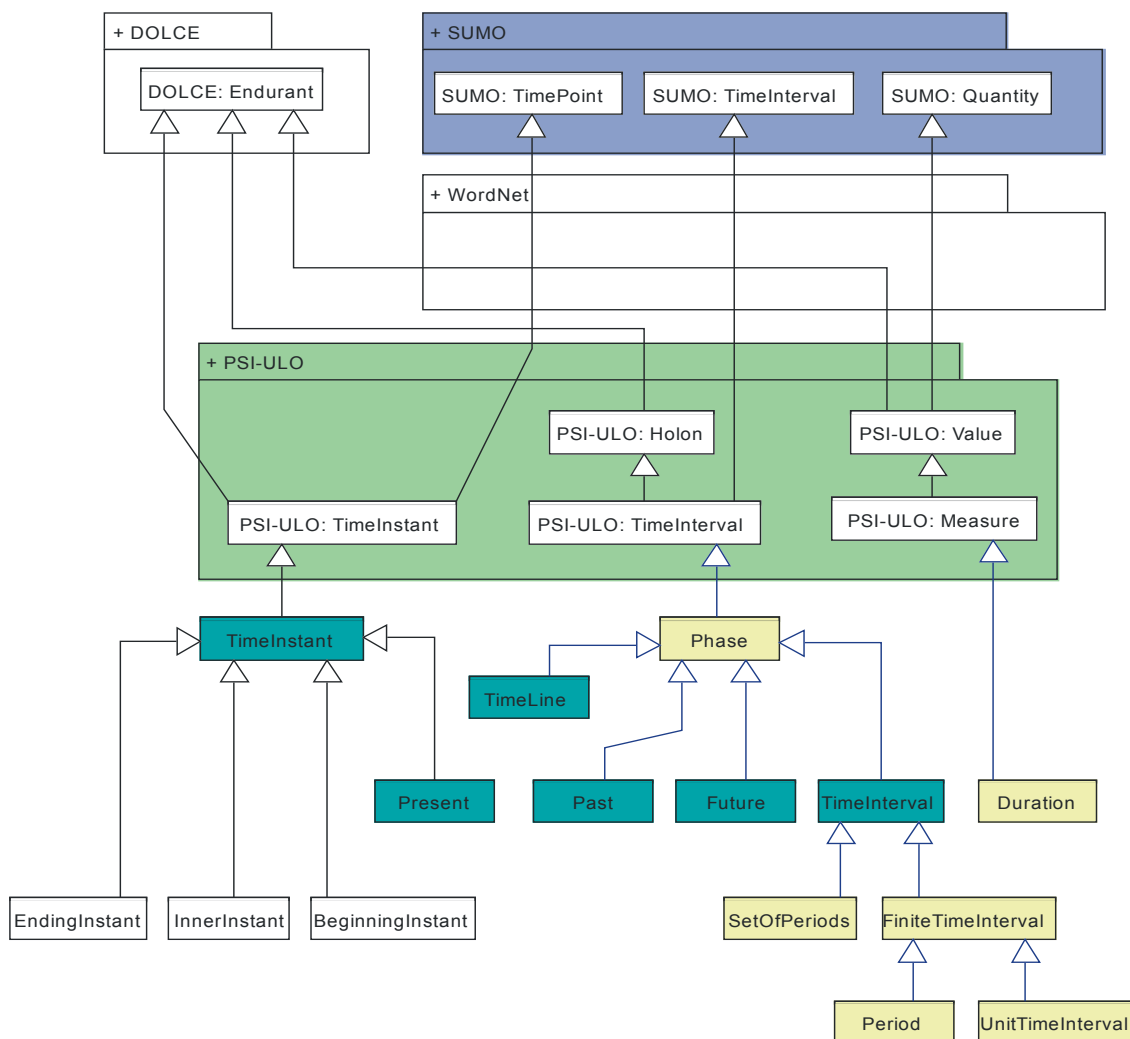


Fig. 5.1: The taxonomy of the PSI Core Time ontology and the mappings of its concepts to DOLCE and SUMO through the PSI Upper-Level ontology.

PSI Environment, Event, and Happening ontology v.2.3 [30, Section 5.2] belongs to the Core of the PSI Suite of Ontologies. It is based on the medium expressiveness subset of the Theory of Time, Events, and Happenings [18].

The taxonomy of the **Environment, Event, and Happening ontology** and the mappings of its top-level concepts to DOLCE as well as to SUMO plus WordNet are pictured in Fig. 5.2.

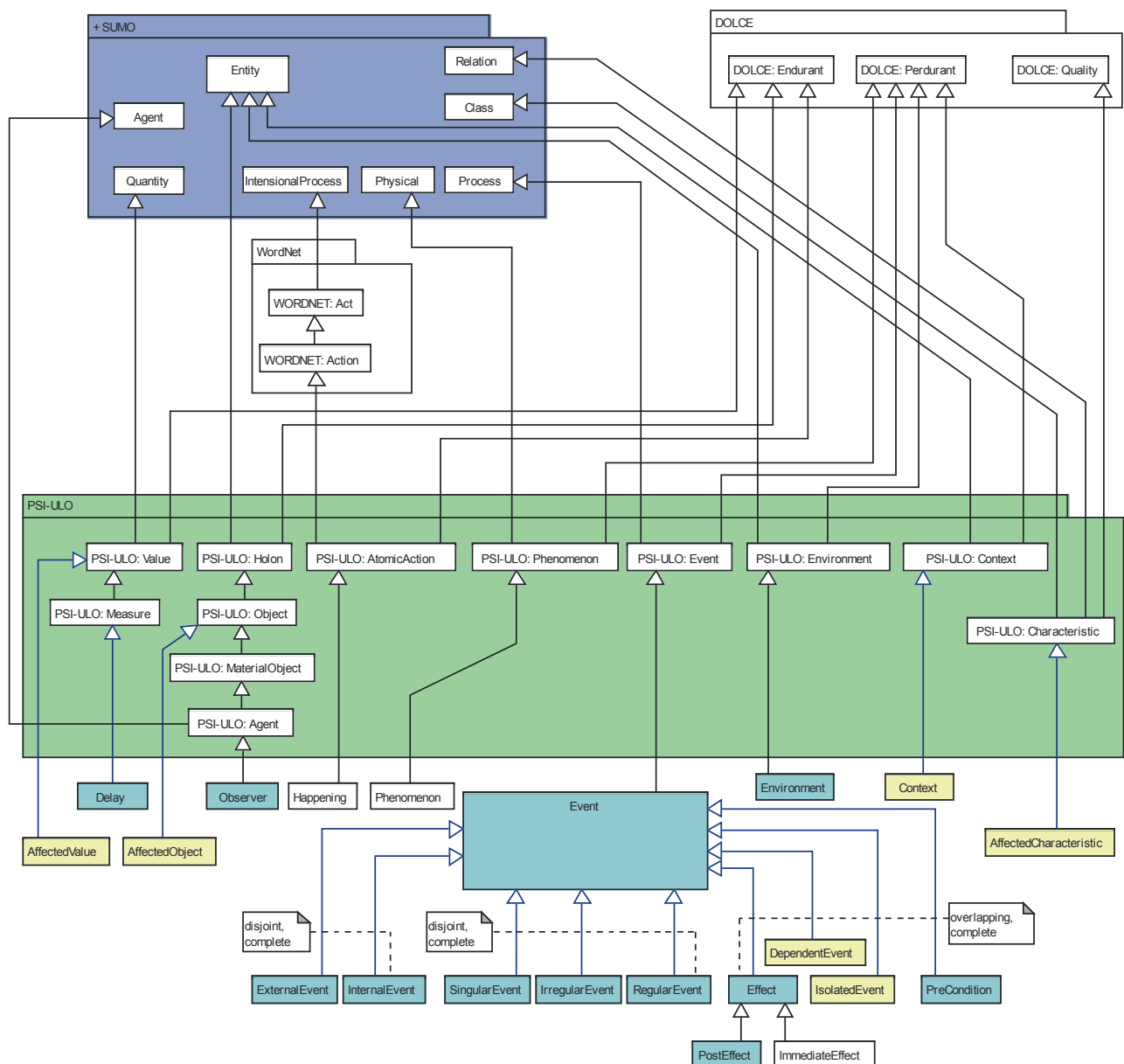


Fig. 5.2: The taxonomy of the PSI Core Environment, Event, and Happening ontology and the mappings of its concepts to DOLCE and SUMO+WordNet through the PSI Upper-Level ontology.

PSI Actor ontology v.2.3 [30, Section 5.3] belongs to the Core of the PSI Suite of Ontologies. The taxonomy of the **Actor ontology** and the mappings of its top-level concepts to DOLCE as well as to SUMO plus WordNet are pictured in Fig. 5.3.

PSI Organization ontology v.2.3 [30, Section 5.4] belongs to the Core of the PSI Suite of Ontologies. The taxonomy of the **Organization ontology** and the mappings of its top-level concepts to DOLCE as well as to SUMO plus WordNet are pictured in Fig. 5.4.

PSI Process Pattern and Process ontologies v.2.3 [30, Section 5.5] belong to the Core of the PSI Suite of Ontologies. The taxonomies of the **Process Pattern** and **Process ontologies** as well as the mappings of their top-level concepts to DOLCE as well as to SUMO plus WordNet are pictured in Fig. 5.5, 5.6.

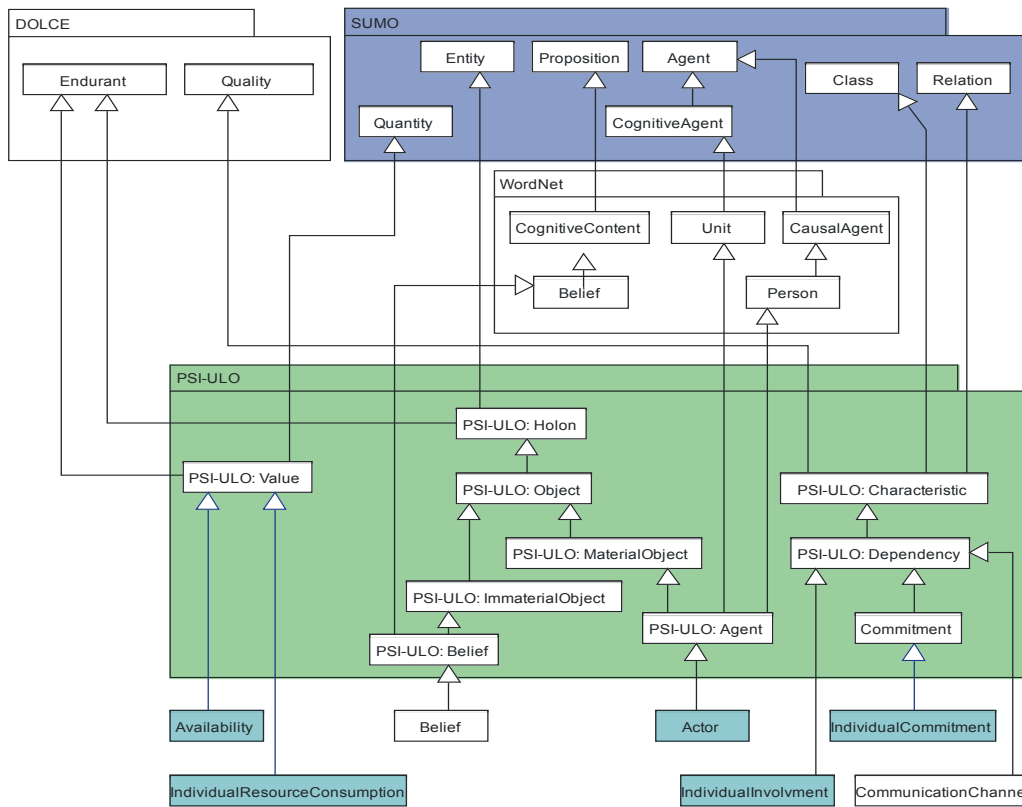


Fig. 5.3: The taxonomy of the PSI Core Actor ontology and the mappings of its concepts to DOLCE and SUMO+WordNet through the PSI Upper-Level ontology.

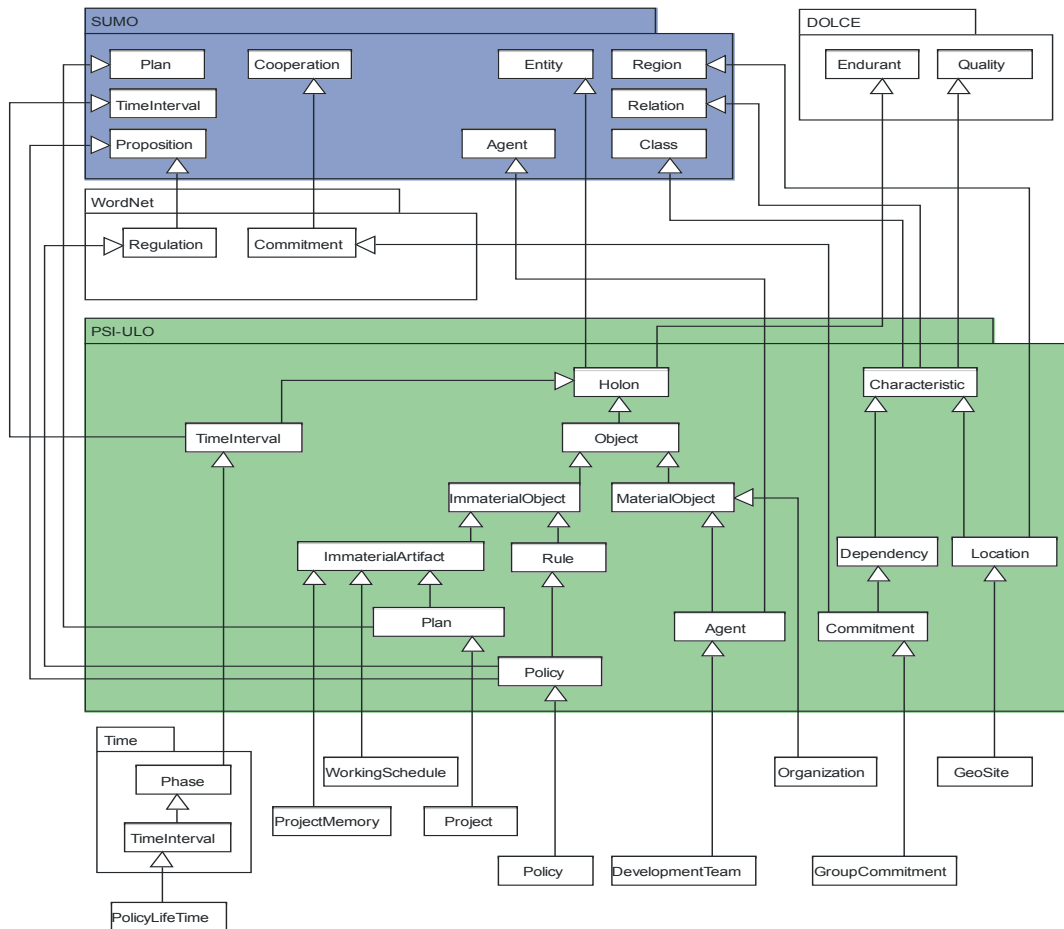


Fig. 5.4: The taxonomy of the PSI Core Organization ontology and the mappings of its concepts to DOLCE and SUMO+WordNet through the PSI Upper-Level ontology.

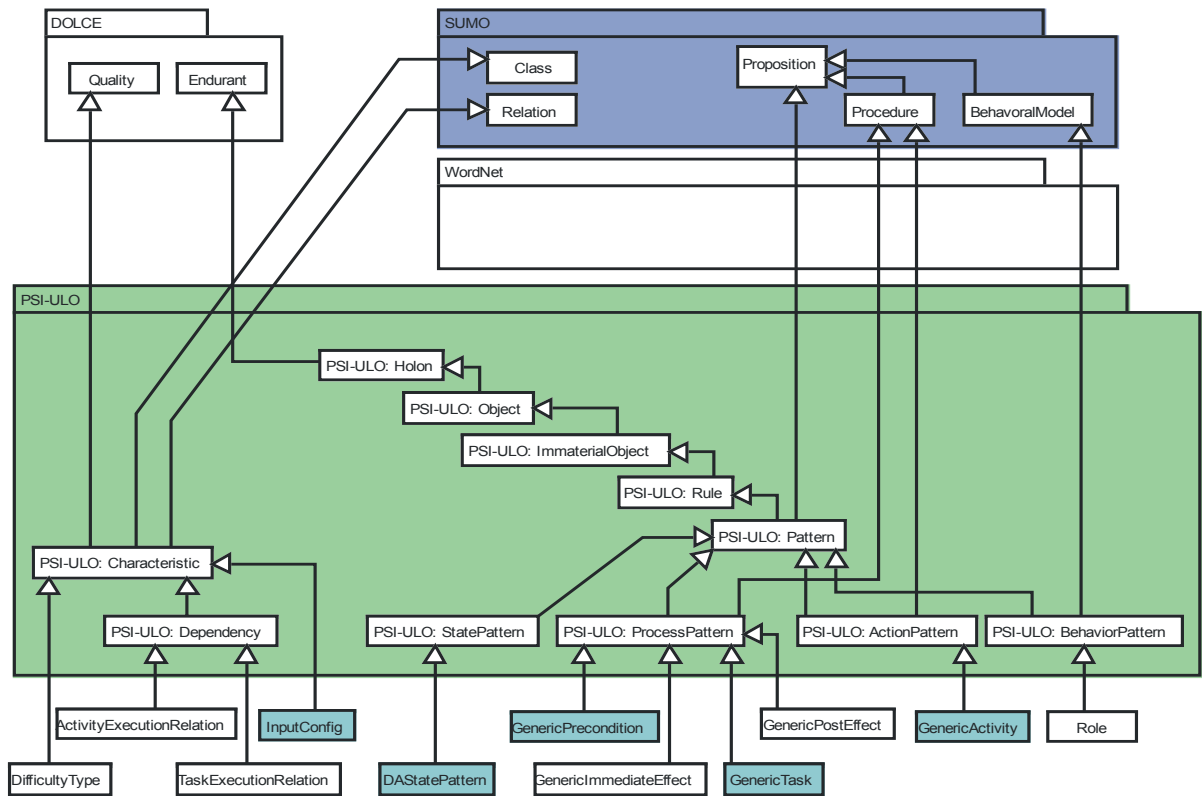


Fig. 5.5: The taxonomy of the PSI Core Process Pattern ontology and the mappings of its concepts to DOLCE and SUMO+WordNet through the PSI Upper-Level ontology.

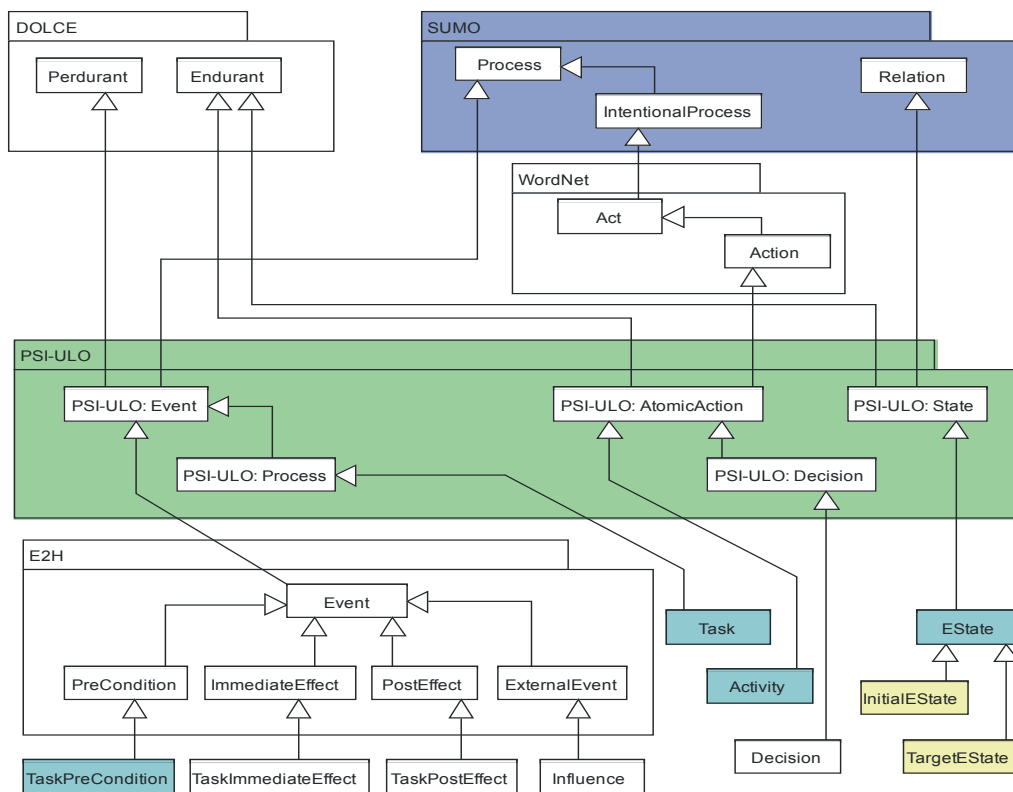


Fig. 5.6: The taxonomy of the PSI Core Process ontology and the mappings of its concepts to DOLCE and SUMO+WordNet through the PSI Upper-Level ontology.

PSI Design Artifact ontology v.2.3 [30, Section 5.4] belongs to the Core of the PSI Suite of Ontologies. The taxonomy of the **Design Artifact ontology** and the mappings of its top-level concepts to DOLCE as well as to SUMO plus WordNet are pictured in Fig. 5.7.

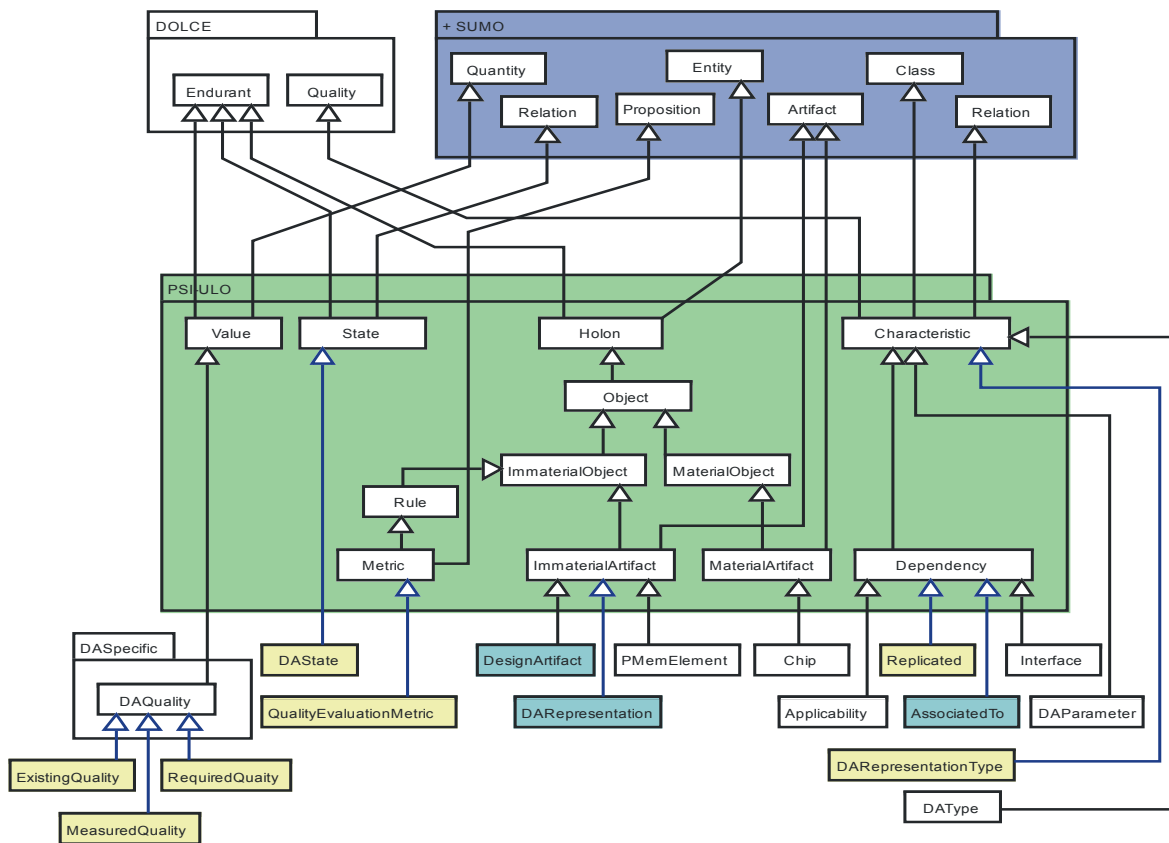


Fig. 5.7: The taxonomy of the PSI Core Design Artifact ontology and the mappings of its concepts to DOLCE and SUMO+WordNet through the PSI Upper-Level ontology.

PSI DASpecific ontology v.2.3 [30, Section 5.4] belongs to the Core of the PSI Suite of Ontologies. The taxonomy of the **DASpecific ontology** and the mappings of its top-level concepts to DOLCE as well as to SUMO plus WordNet are pictured in Fig. 5.8.

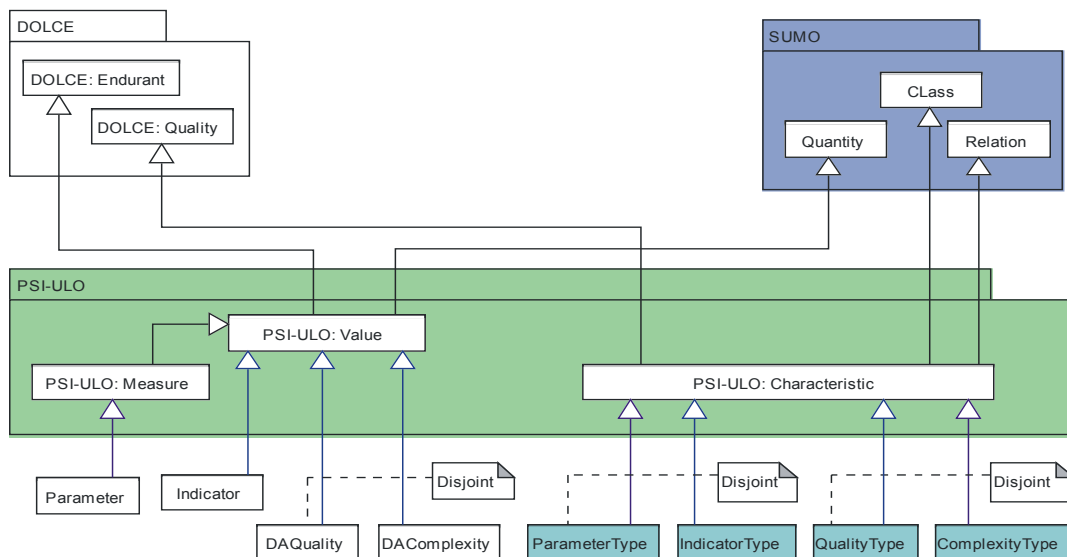
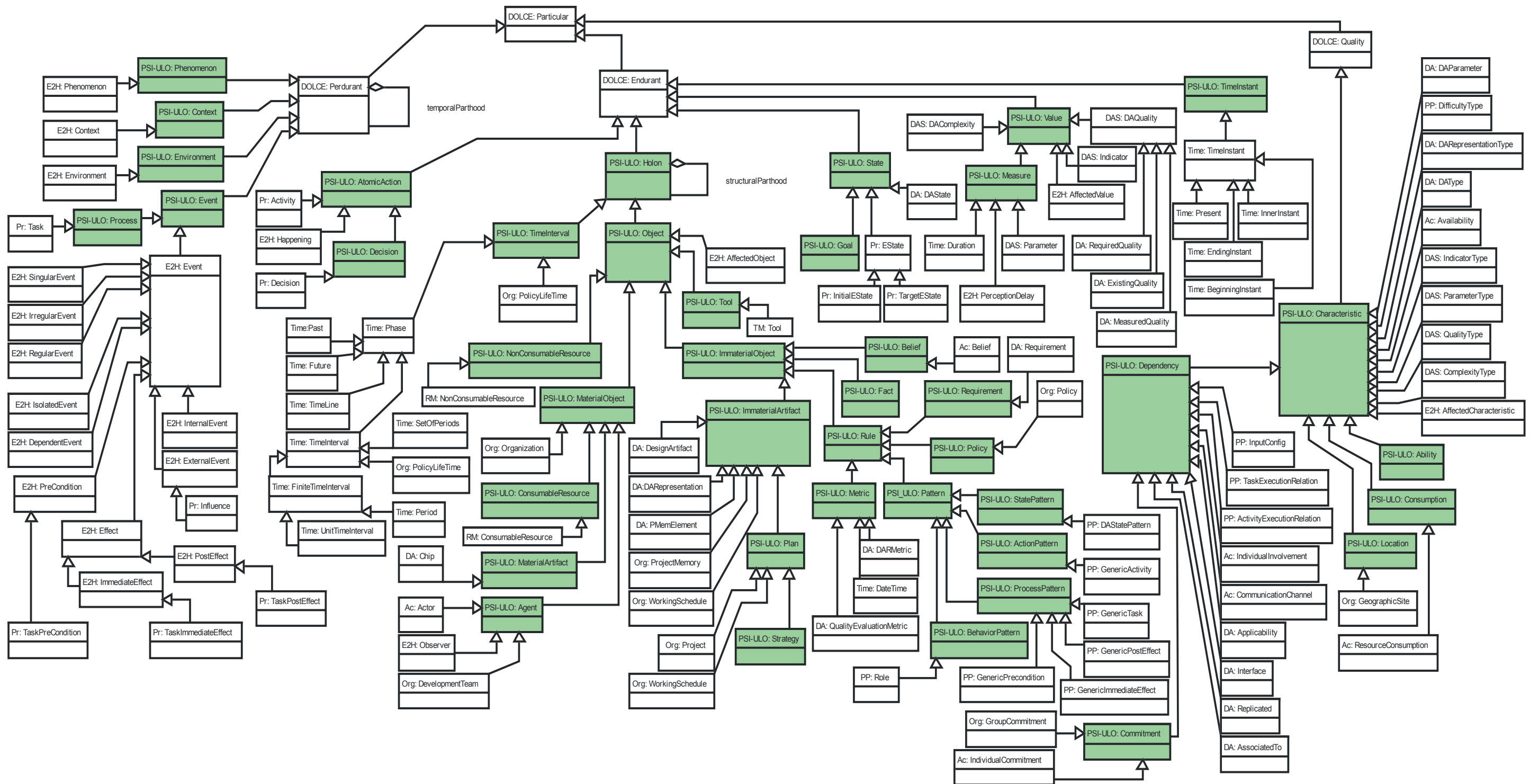


Fig. 5.8: The taxonomy of the PSI Core DASpecific ontology and the mappings of its concepts to DOLCE and SUMO+WordNet through the PSI Upper-Level ontology.

The joint subsumption hierarchy of the PSI Core ontologies and the PSI Upper-Level ontology is pictured in Fig. 5.9.



Prefixes: DOLCE – DOLCE ontology; PSI-ULO – PSI Upper-Level ontology; Time – Time ontology; E2H – Environment, event, and Happening ontology; Ac – Actor ontology; Org: Organization ontology; PP – Process Pattern ontology; Pr – Process ontology; DA – Design Artifact ontology; DAS – DASpecific ontology

Fig. 5.9: Joint taxonomy joint taxonomy of the PSI Core ontologies and the PSI Upper-Level ontology.

5.2 The Mappings of the PSI Extension Ontologies

The Extension ontologies of the PSI Suite are of the following two categories:

- (i) The ontologies that were developed outside the PSI project and adopted in the PSI Suite

These extensions are the IMS Resource ontologyResource Mediator [30, Section 6.1], the IMS Library ontologyLibrary [30, Section 6.2], the IMS Tool ontologyTool Mediator [30, Section 6.3]. The IMS Resource and Tool ontologies require adjustments for the use in the PSI Suite. Therefore the mediators for these Extensions have been developed. The mappings of the concepts specified in these mediators are described in this Section. The Library ontology can be used without adjustment. The mappings of the concepts of the Library ontology that are used in the PSI Suite are described in this Section.

- (ii) The ontologies developed in the PSI project

These extensions are the Software Tool Evaluation ontologyST Evaluation [30, Section 6.4], the Ability ontologyAbility [30, Section 6.5], and the Generic Negotiation ontologyGeneric Negotiation [30, Section 6.6]. The mappings of all concepts of these ontologies are described in this Section.

The taxonomy of the **Mediators of IMS ontologies** and the mappings of their concepts to DOLCE as well as to SUMO plus WordNet are pictured in Fig. 5.10.

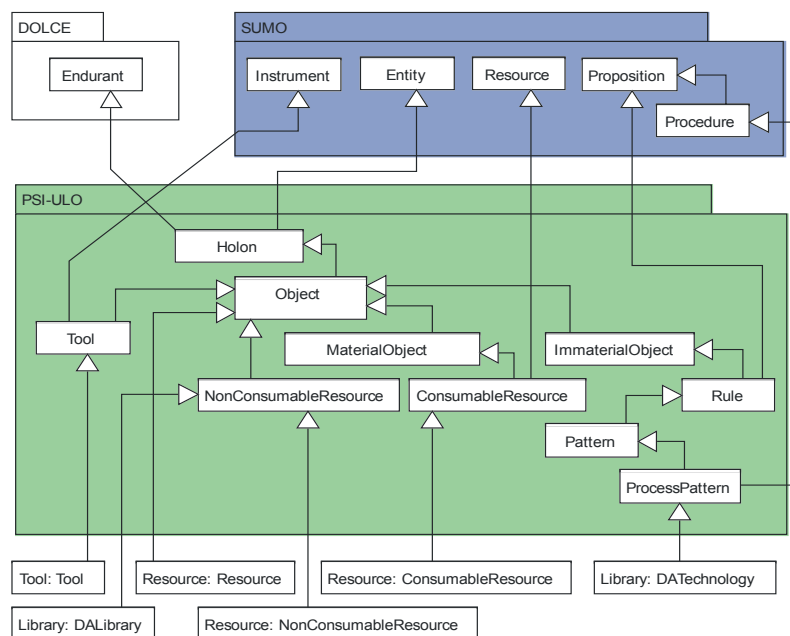


Fig. 5.10: The joint taxonomy of the Mediators of the IMS Resource, Tool ontologies and the used concepts of the IMS Library ontology. The mappings of these concepts to DOLCE and SUMO+WordNet through the PSI Upper-Level ontology.

The taxonomy of the **Software Tool Evaluation ontology** and the mappings of its concepts to DOLCE as well as to SUMO plus WordNet are pictured in Fig. 5.11.

The taxonomy of the **Ability ontology** and the mappings of its concepts to DOLCE as well as to SUMO plus WordNet are pictured in Fig. 5.12.

The taxonomy of the **Generic Negotiation ontology** and the mappings of its concepts to DOLCE as well as to SUMO plus WordNet are pictured in Fig. 5.13.

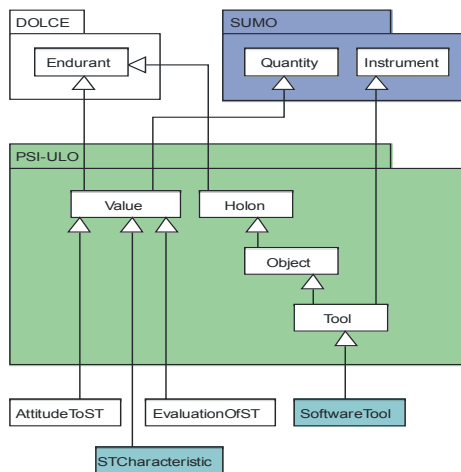


Fig. 5.11: The taxonomy of the Software Tool Evaluation ontology. The mappings of its concepts to DOLCE and SUMO+WordNet through the PSI Upper-Level ontology.

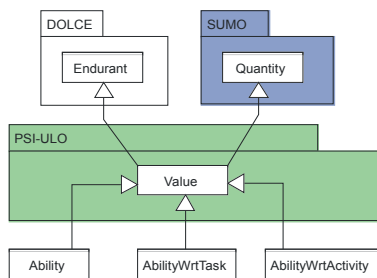


Fig. 5.12: The taxonomy of the Ability ontology. The mappings of its concepts to DOLCE and SUMO+WordNet through the PSI Upper-Level ontology.

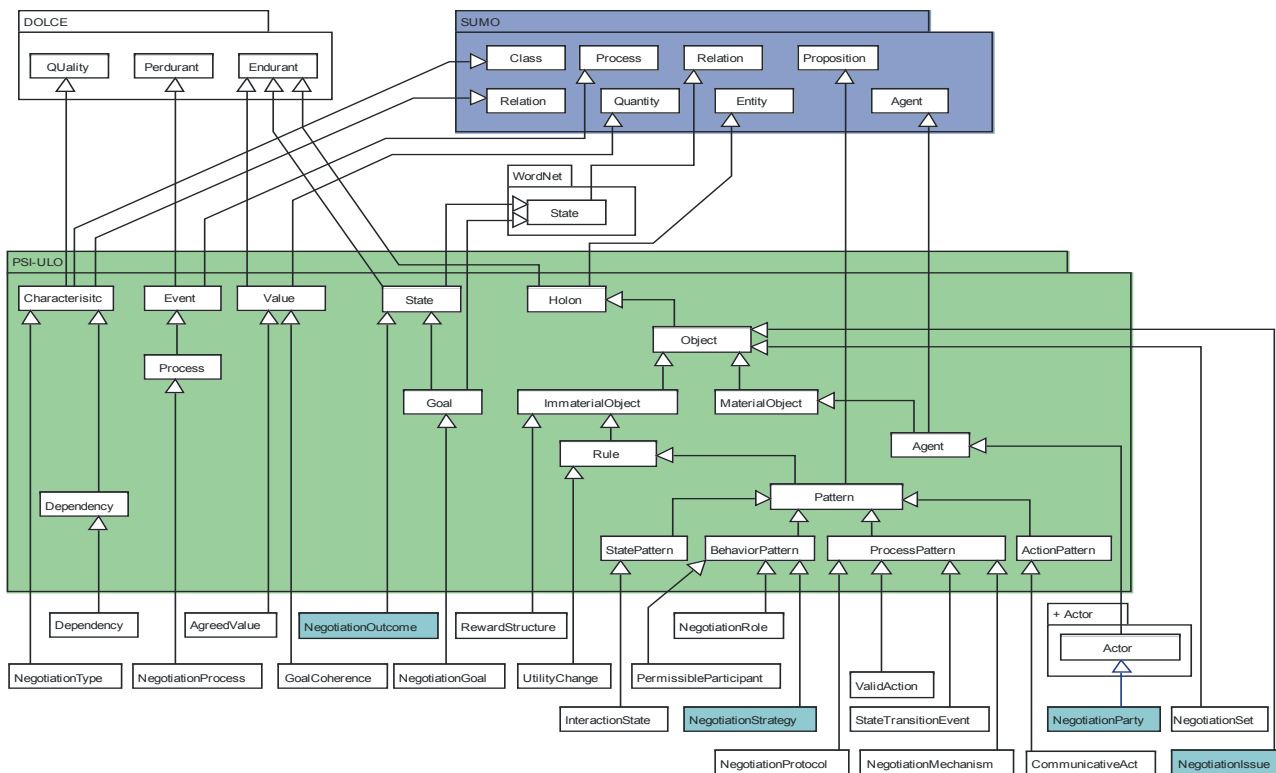


Fig. 5.13: The taxonomy of the Generic Negotiation ontology. The mappings of its concepts to DOLCE and SUMO+WordNet through the PSI Upper-Level ontology.

6 Future Work

Further development of PSI Upper-Level ontology is aligned by the ontology engineering methodology presented in Section 3. Consecutive refinements of the ontology will be undertaken each time new revisions of PSI Theoretical Framework and PSI Core ontologies are fixed.

7 The Index of Concepts, Key Properties, and Terms

Concepts and properties are **bold**. Instances are *italic*. Terms are normal.

A	
action	22
actualism	20
actualism	21
agent	23
agreement	22
availability	23
B	
belief	23
BFO	21
C	
CaMel notation	12
cardinality	13
co-execution	22
Colors in UML Diagrams	18
Change coloring	19
Structural coloring	7, 18
concurrency	22
consumption	23
D	
decision taking	22
DEDP	22
dependency	22
descriptivism	20
descriptivism	21
design artifact	22
DOLCE	20
Endurant	34
Particular	34
Perdurant	34
Quality	34
E	
endurantism	20
endurantism	22
Enterprise Ontology	20
environment	22
evaluation wrt Golden Standard	32
external evaluation	31
G	
goal	22
M	
Methodology	
ABox Migration	33
Bridging	32
Commonsense Alignment	32
Domain Ontology Refinement	31
Modeling Framework Refinement	31
Taxonomy Refinement	32
Upper Ontology Refinement	31
User Evaluation	31
Methodology – Shaker Modeling	30
minimal ontological commitment	11
multiplicativism	20
multiplicativism	21
multiplicity	12
N	
negotiation	22
O	
object	22
OCHRE	21
ontology engineering methodology	30
ontology imports	17
OpenCYC	21
P	
perdurantism	20
perdurantism	22
possibilism	20
possibilism	21
problem solving process	22
process	22
PSI Core	
joint taxonomy	68
PSI Core Ontology	
Actor	64
DASpecific	67
Design Artifact	67
Environment, Event, and Happening	64
Organization	64
Process	64
Process Pattern	64
Time	63
PSI Extension Ontology	
Ability	69
Generic Negotiation	69
Library	69
Resource Mediator	69
ST Evaluation	69
Tool Mediator	69
PSI Performance Ontology	54
PSI Suite of Ontologies	20
PSI- ULO	
Ability	44
ActionPattern	58
Agent	50
AtomicAction	40
BehaviorPattern	57
Belief	55
Characteristic	43
Commitment	61
ConsumableResource	52
Consumption	44
Context	60
Decision	40
Dependency	44
Environment	39

Event	37
Fact	55
Goal	43
Holon	41
ImmaterialArtifact	54
ImmaterialObject	53
Location	44
MaterialArtifact	52
MaterialObject	49
Measure	46
Metric	56
NonConsumableResource	62
Object	48
Pattern	57
Phenomenon	34
Plan	54
Policy	59, 60
Process	37
ProcessPattern	58
Requirement	57
Rule	56
State	42
StatePattern	59
Strategy	54
TimeInstant	47
TimeInterval	47
Tool	52
Value	46
PSI Upper-Level ontology	20, 34
PSL	20

R

reductionism	20
reductionism	21
resource	23
revisionarism	20
revisionarism	21
role	23

S

skill	23
soundness	32
state	22
SUMO	20

T

TOVE	20
Transformation	
UML association class to OWL	15
UML binary association to OWL	14
UML datatype to OWL	13
UML package to OWL	17
transformation path	22

U

UML class diagram	12
utility	23

W

WordNet	20
---------------	----

References

1. **Gruber, T.:** Towards principles for the design of ontologies used for knowledge sharing. *Int. J. Human-Computer Studies*, 43(5/6), 1995
2. **Miller R.:** Practical UML™: A Hands-On Introduction for Developers. <http://bdn.borland.com/article/0,1410,31863,00.html#classdiagrams>
3. **Schreiber G.** OWL Restrictions. <http://www.cs.vu.nl/~guus/public/owl-restrictions/>
4. **Falkovych K., Sabou M., and Stuckenschmidt H.:** UML for the Semantic Web: Transformation-Based Approaches. 2003, <http://citeseer.ist.psu.edu/falkovych03uml.html>
5. **Ermolayev, V., Jentzsch, E., Karsayev, O., Keberle, N., Matzke, W.-E., Samoylov, V., Sohnius, R.:** An Agent-Oriented Model of a Dynamic Engineering Design Proces. In: Kolp, M., Bresciani, P., Henderson-Sellers, B., and Winikoff, M. (Eds.): Agent-Oriented Information Systems III. 7th International Bi-Conference Workshop, AOIS 2005, Utrecht, Netherlands, July 26, 2005, and Klagenfurt, Austria, October 27, 2005. Revised Selected Papers, 2006, 168-183
6. **Ermolayev, V. (Ed.), Jentzsch, E., Matzke, W.-E., Schmidt, J., Schroeder, G., Weber, S. and Werner, J.:** Agent-Based Dynamic Engineering Design Process Modeling Framework. Technical Report. Cadence Design Systems, GmbH, 2004, 29 p.
7. **Ermolayev, V., Keberle, N.:** A Generic Ontology of Rational Negotiation. In: Karagiannis, D., Mayr, H.C. (Eds.): Information Systems Technology and its Applications. 5-th Int Conf ISTA'2006, May 30 – 31, 2006, Klagenfurt, Austria, 51-66, 2006
8. **Uschold, et al:** The Enterprise Ontology. *Knowledge Engineering Review*, 13(1), 1998
9. **Grueninger, M., Atefy, K., and Fox, M.:** Ontologies to Support Process Integration in Enterprise Engineering. *Computational & Mathematical Organization Theory* 6, 381–394, 2000.
10. **Bock, C. and Gruninger, M.:** PSL: A semantic domain for flow models. *Software Systems Modeling* (2005) 4: 209–231
11. **Niles, I. and Pease, A.:** Towards a Standard Upper Ontology. In: Proc. Int. Conf. on Formal Ontologies in Information Systems (FOIS'01), October 17-19, 2001, Ogunquit, Maine, USA.
12. **Fellbaum, C. (ed.):** *WordNet: An Electronic Lexical Database*, MIT Press, 1999.
13. **Keberle, N., Ermolayev, V., and Matzke, W.-E.:** Evaluating PSI Ontologies by Mapping to the Common Sense. In: Mayr, H. C, Karagiannis, D. (Eds.): Information Systems Technology and its Applications Proc. 6th Int. Conf. ISTA 2007, May 23-25, Kharkiv, Ukraine, 2007, GI LNI Vol 107 , 91-104
14. **Masolo, C., Borgo, S., Gangemi, A., Guarino, N., and Oltramari, A.:** WonderWeb Deliverable D18. Ontology Library (*final*). 31-12-2003
15. **Vrandečić, D., Pinto, S., Tempich, C. and Sure, Y.:** The DILIGENT knowledge processes. *J. of Knowledge Management*, 9(5), 2005, 85-96
16. **Aristotle:** Physics. Translated by R. P. Hardie and R. K. Gaye, eBooks@Adelaide, 2007
17. **Ermolayev, V., Jentzsch, E., Matzke, W.-E., Pěchouček, M., and Sohnius, R.:** Performance Simulation Initiative. Theoretical Framework v.2.3. Technical Report PSI-TF-TR-2009-1, 01.02.2009, VCAD EMEA Cadence Design Systems, GmbH, 44 p.
18. **Ermolayev, V., Jentzsch, E., Keberle, N. and Sohnius, R.:** Performance Simulation Initiative. Theory of Time, Happenings, and Events. Technical Report PSI-T-TR-2007-2, 30.10.2007, VCAD EMEA Cadence Design Systems, GmbH, 26 p.
19. **Simperl, E.:** Evaluation of the PSI Ontology Library. Technical Report, DERI Innsbruck, Austria, June, 2007, 13 p.
20. **Brank, J., Grobelnik, M., Mladenic, D.:** A Survey of Ontology Evaluation Techniques. In Proc. Conference on Data Mining and Data Warehouses (SiKDD 2005), October 17, 2005, Ljubljana, Slovenia.
21. The Oxford Dictionary of Philosophy. Oxford University Press, 1994, 1996, 2005. e-Edition: Answers.com 17 Nov. 2007. <http://www.answers.com/topic/phenomenon>
22. **D. Oberle, A. Ankolekar, P. Hitzler, P. Cimiano, M. Sintek, M. Kiesel, B. Mougouie, S. Vembu, S. Baumann, Massimo Romanelli, Paul Buitelaar, R. Engel, D. Sonntag, N. Reithinger, Berenike Loos, R. Porzel, H.-P. Zorn, V. Micelli, C Schmidt, Moritz Weiten, F. Burkhardt, J. Zhou:** DOLCE ergo SUMO: On Foundational and Domain Models in SWIntO (SmartWeb Integrated Ontology). Technical Report , AIFB, University of Karlsruhe. July 2006
23. **Peter Mika, Aldo Gangemi, Daniel Oberle, Marta Sabou:** Foundations for Service Ontologies: Aligning OWL-S to DOLCE. WWW2004, May 17–22, 2004, New York, NY USA. ACM 1-58113-844-X/04/0005, 563-572
24. **Guarino, N., Welty, C.:** Supporting ontological analysis of taxonomic relationships. *Data and Knowledge Engineering*, Vol. 39, No. 1, pp. 51-74, 2001
25. **Ermolayev, V., Jentzsch, E., Keberle, N., and Sohnius, R.:** Performance Simulation Initiative. The Suite of Ontologies v.2.2. Reference Specification. Technical Report PSI-ONTO-TR-2007-5, 28.12.2007, VCAD EMEA Cadence Design Systems, GmbH, 133 p.
26. **Ermolayev, V., Jentzsch, E., Keberle, N., Matzke, W.-E., Padeborg, C. and Sohnius, R.:** PRODUKTIV+ Performance Ontology v.1.0. Reference Specification. Technical Report PSI-P-ONTO-TR-2007-3, 30.09.2007,

- VCAD EMEA Cadence Design Systems, GmbH, 63 p.
27. **Guarino, N and Welty, C. A.:** Ontological Analysis of Taxonomic Relationships. In: Laender, A., and Storey, V. (eds.) Proc. 19th Int Conf on Conceptual Modeling (ER 2000), Salt Lake City, Utah, USA, Oct. 9-12, 2000, 210-224
 28. **Guarino, N and Welty, C. A.:** A Formal Ontology of Properties. In: Dieng, R. and Corby, O. (eds.) Proc. 12th Int Conf on Knowledge Acquisition, Modeling and Management, EKAW 2000, Juan-les-Pins, France, Oct. 2-6, 2000, 97-112
 29. **Guarino, N., and Welty, C.:** Towards a methodology for ontology-based model engineering. In: Bézivin, J. and Ernst, J. (eds.) Proc. ECOOP-2000 Int W-shop on Model Engineering (IWME 2000), Nice / Sophia Antipolis, France, Jun. 13, 2000
 30. **Ermolayev, V., Jentsch, E., Keberle, N., and Sohnius, R.:** Performance Simulation Initiative. The Suite of Ontologies v.2.3. Reference Specification. Technical Report PSI-ONTO-TR-2009-1, 23.09.2009, VCAD EMEA Cadence Design Systems, GmbH, 168 p.
 31. **Ermolayev, V., Keberle, N., and Matzke, W.-E.:** An Ontology of Environments, Events, and Happenings. In: Proc 31st IEEE Annual International Computer Software and Applications Conference ([COMPSAC 2008](#)), Turku, Finland, Jul. 28 - Aug. 1, 2008, 539-546
 32. **Henderson-Sellers, B. and Gonzalez-Perez, C.:** Standardizing Methodology Metamodelling and Notation: An ISO Exemplar. In: Kaschek, R., Kop, C., Steinberger, C., Fliedl, G. (eds.) UNISCON 2008. LNBIP, vol. 5, pp. 1–12. Springer, Berlin/Heidelberg (2008)
 33. **Simperl, E., Tempich, C., and Sure, Y.:** A Cost Estimation Model for Ontology Engineering. In: Cruz, I. F., et al. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 625–639. Springer, Berlin/Heidelberg (2006)
 34. **Ermolayev, V., Keberle, N., Matzke W.-E., and Vladimirov, V.:** A Strategy for Automated Meaning Negotiation in Distributed Information Retrieval. In: Y. Gil et al. (Eds.): ISWC 2005 Proc. 4th Int. Semantic Web Conf. (ISWC'05), 6-10 Nov., Galway, Ireland, LNCS 3729, pp. 201 – 215, 2005
 35. **Brank, J., Grobelnik, M., and Mladenić, D.:** A Survey of Ontology Evaluation Techniques. In: Grobelnik, M., Mladenić, D. (eds.) SiKDD-2005, pp. 166–169 (2005)