P1872™/D2
Draft Standard for Standard for Ontologies for Robotics and Automation

Sponsor

Standing Committee for Standards
of the
IEEE Robotics and Automation Society

Approved <Date Approved>

IEEE-SA Standards Board

Abstract: The growing requirements for robots to deal with complex situations is increasing the need for the standardization of how such artificial agents represent and communicate their knowledge about the world. This includes the standardization of the concepts, relations and axioms in the Robotics and Automation (R&A) domain. This standard defines a core ontology that specifies the main, most general concepts, relations and axioms of R&A. It is intended as a reference for knowledge representation and reasoning in robots, as well as a formal reference vocabulary for communicating knowledge about R&A between robots and humans. This standard is composed by a core ontology about R&A and a methodology for instantiating it and/or extending it with new concepts, relations and axioms.

Keywords: Ontology, Robotics, Core Ontology, Methodology, Automation.

The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA

PDF: ISBN 978-0-XXXX-XXXX-X STDXXXXX
Print: ISBN 978-0-XXXX-XXXX-X STDPDXXXXX

Important Notices and Disclaimers Concerning IEEE Standards Documents

IEEE documents are made available for use subject to important notices and legal disclaimers. These notices and disclaimers, or a reference to this page, appear in all standards and may be found under the heading "Important Notice" or "Important Notices and Disclaimers Concerning IEEE Standards Documents."

Notice and Disclaimer of Liability Concerning the Use of IEEE Standards Documents

IEEE Standards documents (standards, recommended practices, and guides), both full-use and trial-use, are developed within IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association ("IEEE-SA") Standards Board. IEEE ("the Institute") develops its standards through a consensus development process, approved by the American National Standards Institute ("ANSI"), which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of the Institute and participate without compensation from IEEE. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, IEEE does not independently evaluate, test, or verify the accuracy of any of the information or the soundness of any judgments contained in its standards.

IEEE does not warrant or represent the accuracy or content of the material contained in its standards, and expressly disclaims all warranties (express, implied and statutory) not included in this or any other document relating to the standard, including, but not limited to, the warranties of: merchantability; fitness for a particular purpose; non-infringement; and quality, accuracy, effectiveness, currency, or completeness of material. In addition, IEEE disclaims any and all conditions relating to: results; and workmanlike effort. IEEE standards documents are supplied "AS IS" and "WITH ALL FAULTS."

Use of an IEEE standard is wholly voluntary. The existence of an IEEE standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard.

In publishing and making its standards available, IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity nor is IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing any IEEE Standards document, should rely upon his or her own independent judgment in the exercise of reasonable care in any given circumstances or, as appropriate, seek the advice of a competent professional in determining the appropriateness of a given IEEE standard.

IN NO EVENT SHALL IEEE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO: PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE PUBLICATION, USE OF, OR RELIANCE UPON ANY STANDARD, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE AND REGARDLESS OF WHETHER SUCH DAMAGE WAS FORESEEABLE.

Translations

The IEEE consensus development process involves the review of documents in English only. In the event that an IEEE standard is translated, only the English version published by IEEE should be considered the approved IEEE standard.

Official statements

A statement, written or oral, that is not processed in accordance with the IEEE-SA Standards Board Operations Manual shall not be considered or inferred to be the official position of IEEE or any of its committees and shall not be considered to be, or be relied upon as, a formal position of IEEE. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that his or her views should be considered the personal views of that individual rather than the formal position of IEEE.

Comments on standards

Comments for revision of IEEE Standards documents are welcome from any interested party, regardless of membership affiliation with IEEE. However, IEEE does not provide consulting information or advice pertaining to IEEE Standards documents. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Since IEEE standards represent a consensus of concerned interests, it is important that any responses to comments and questions also receive the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to comments or questions except in those cases where the matter has previously been addressed. For the same reason, IEEE does not respond to interpretation requests. Any person who would like to participate in revisions to an IEEE standard is welcome to join the relevant IEEE working group.

Comments on standards should be submitted to the following address:

Secretary, IEEE-SA Standards Board
445 Hoes Lane
Piscataway, NJ 08854 USA

Laws and regulations

Users of IEEE Standards documents should consult all applicable laws and regulations. Compliance with the provisions of any IEEE Standards document does not imply compliance to any applicable regulatory requirements. Implementers of the standard are responsible for observing or referring to the applicable regulatory requirements. IEEE does not, by the publication of its standards, intend to urge action that is not in compliance with applicable laws, and these documents may not be construed as doing so.

Copyrights

Photocopies

Updating of IEEE Standards documents

Users of IEEE Standards documents should be aware that these documents may be superseded at any time by the issuance of new editions or may be amended from time to time through the issuance of amendments, corrigenda, or errata. An official IEEE document at any point in time consists of the current edition of the document together with any amendments, corrigenda, or errata then in effect.

Every IEEE standard is subjected to review at least every ten years. When a document is more than ten years old and has not undergone a revision process, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE standard.

In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit the IEEE-SA Website at http://ieeexplore.ieee.org/xpl/standards.jsp or contact IEEE at the address listed previously. For more information about the IEEE-SA or IEEE's standards development process, visit the IEEE-SA Website at http://standards.ieee.org.

Errata

Errata, if any, for all IEEE standards can be accessed on the IEEE-SA Website at the following URL: http://standards.ieee.org/findstds/errata/index.html. Users are encouraged to check this URL for errata periodically.

Patents

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken by the IEEE with respect to the existence or validity of any patent rights in connection therewith. If a patent holder or patent applicant has filed a statement of assurance via an Accepted Letter of Assurance, then the statement is listed on the IEEE-SA Website at http://standards.ieee.org/about/sasb/patcom/patents.html. Letters of Assurance may indicate whether the Submitter is willing or unwilling to grant licenses under patent rights without compensation or under reasonable rates, with reasonable terms and conditions that are demonstrably free of any unfair discrimination to applicants desiring to obtain such licenses.

Essential Patent Claims may exist for which a Letter of Assurance has not been received. The IEEE is not responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patents Claims, or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility. Further information may be obtained from the IEEE Standards Association.

Participants

At the time this draft standard was completed, the ORA - Ontologies for Robotics and Automation Working Group had the following membership:

Craig Schlenoff, *Chair*
Edson Prestes, *Vice Chair*
Paulo Jorge Sequeira Gonçalves, *Secretary*

Joel Luis Carbonera	Veera Ragavan Sampath Kumar	Sébastien Gérard
Sandro Rama Fiorini	Marcos Ennes Barreto	Yacine Amirat
Vitor Augusto Machado Jorge	Raj Madhavan	Vitor Fortes Rey
Mara Abel	Angela Locoro	Stephen Balakirsky
Tamás Haidegger	Maki K. Habib	Signe Redfield
Howard Li	Abdelghani Chibani	

The following members of the <individual/entity> balloting committee voted on this standard. Balloters may have voted for approval, disapproval, or abstention.

[To be supplied by IEEE]

Balloter1	Balloter4	Balloter7
Balloter2	Balloter5	Balloter8
Balloter3	Balloter6	Balloter9

When the IEEE-SA Standards Board approved this standard on <Date Approved>, it had the following membership:

[To be supplied by IEEE]

<Name>, *Chair*
<Name>, *Vice Chair*
<Name>, *Past Chair*
Konstantinos Karachalios, *Secretary*

SBMember1	SBMember4	SBMember7
SBMember2	SBMember5	SBMember8
SBMember3	SBMember6	SBMember9

*Member Emeritus

Also included are the following nonvoting IEEE-SA Standards Board liaisons:

<Name>, *DOE Representative*
<Name>, *NIST Representative*

IEEE Standards Program Manager, Document Development

<Name>
IEEE Standards Program Manager, Technical Program Development

Introduction

This introduction is not part of P1872/D2, Draft Standard for Standard for Ontologies for Robotics and Automation.

Seamless and unambiguous communication between people in any kind of group demands a common, well-defined vocabulary. Otherwise, misinterpretations can happen and no information or, even worse, incorrect information can be exchanged between the participants, often with very negative consequences. This could happen when two people who do not speak the same language try to communicate. The same applies to human/robot and robot/robot communication, where an intermediate, standard language with clear and well-defined terms is a *sine qua non* condition for common understanding.

The growing complexity of behaviors that robots are expected to perform naturally entails the use of increasingly complex knowledge, as well as the need for multi-robots and human-robot collaboration. In this context, the need for a standard and well-defined model for capturing this knowledge is becoming evident. The existence of such a standard knowledge model, precisely defining the concepts of the robotics domain, will ensure common understanding among various stakeholders involved in the lifecycle of robotics systems; enabling efficient and reliable data integration and information exchange among them.

Ontology plays a fundamental role in this context. It formally specifies the key concepts, properties, relationships and axioms of a given domain. Unlike taxonomies, which provide only a set of vocabulary and a single type of relationship between terms, an ontology provides a richer set of relationships, constraints and rules. In general, ontologies make the relevant knowledge about a domain explicit in a computer-interpretable format, allowing software to reason over that knowledge to infer new information. Furthermore, ontologies are a great tool for diminishing the ambiguity in knowledge transfer among groups of humans, robots, and other artificial systems that share the same conceptualization.

In this sense, the Ontologies for Robotics and Automation Working Group (ORA WG) is actively working with industry, academia, and government organizations to develop a set of ontologies and an associated modeling methodology to be used as a standard in Robotics and Automation (R&A). As it is extremely difficult to develop a single ontology that covers the entire scope of R&A, the ORA WG decided to focus initially on a reduced set of subdomains: industrial robotics, service robotics, and autonomous robotics. This decision was based on the prevalence of robots in these markets and the standardization necessities accompanying them. Therefore, ORA WG comprises four subgroups, three of them associated with each of the above subdomains. The fourth subgroup, called Upper Ontology/Methodology (UpOM), is in charge of developing of a more general ontology to bring all of the subdomain ontologies together. *This document is the result of the work done by the UpOM and presents a core ontology for R&A called CORA, which specifies the general notions behind R&A and aims to provide clear definitions of the common concepts that will permeate all sub-ontologies to be developed within the ORA WG*. Thus, CORA focuses on defining what a robot is, along with the specification of other related entities. As important as the ontology itself, this document also includes a description of the ontology engineering process that directed the development of CORA, focusing on the methodological difficulties involved in the alignment of the definitions of very broad and overloaded terms, such as device, robot and robotic system.

Contents

1. Overview 9
1.1 Scope 9
1.2 Purpose 9

2. Normative references 10

3. Definitions 10

4. Core Ontology for Robotics and Automation : Axioms 13
4.1 SUMO 13
4.2 CORAX Axioms 14
4.3 CORA axioms 19

5. RPARTS axioms 26
5.1 RPARTS:Sensor 27
5.2 RPARTS:LocomotionDevice 27
5.3 RPARTS:Actuator 28
5.4 RPARTS:EndEffector 29
5.5 RPARTS:ElectronicDevice 29
5.6 RPARTS:MechanicalJoint 30
5.7 Other parts 31

6. POS axioms 32
6.1 POS:PositionMeasure 32
6.2 POS:PositionCoordinateSystem 32
6.3 POS:PositionTransformation 33
6.4 POS:PositionPoint 34
6.5 POS:PositionRegion 35
6.6 POS:OrientationMeasure 37
6.7 POS:OrientationCoordinateSystem 37
6.8 POS:OrientationTransformation 38
6.9 POS:OrientationPoint 39
6.10 POS:OrientationRegion 40
6.11 POS:Pose 41

7. Specialization guidelines 41
7.1 Robot structure pattern 41

Draft Standard for Standard for Ontologies for Robotics and Automation

1. Overview

1.1 Scope

This standard defines a methodology for the representation of, reasoning about, and communication of knowledge in the robotics domain. This includes a linguistic framework and generic concepts represented by a core ontology that can be specialized to capture the detailed semantics of glossary terms and their definitions.

This standard contains the Core Ontology for Robotics and Automation (CORA) with the representation of fundamental concepts from which the more detailed concepts belonging to other ORA WG ontologies are constructed. This standard also defines the ontology engineering methodology used to construct the ORA ontologies.

1.2 Purpose

The purpose of this standard is to provide a methodology for knowledge representation and reasoning in robotics and automation, together with the core ontology for the robotics and automation domain. The standard provides a unified way of representing knowledge and provides a common set of term definitions, allowing for unambiguous knowledge transfer among any group of human, robots, and other artificial systems.

It aims to provide a common vocabulary along with clear and concise definitions from the robotics and automation domain. With the growing complexity of behaviors that robots are expected to perform as well as the need for multi-robot and human-robot collaboration, the need for a standard and well-defined knowledge representation is becoming more evident. The standard knowledge representation methodology and terminology: 1) more precisely defines the concepts in the robot's knowledge representation, 2) ensures common understanding among members of the community, and 3) facilitates more efficient data integration and transfer of information among robotic systems. Information included in this knowledge representation encompasses, but is not limited to, robot hardware and software; activities and goals; environment; cause and effects of performing actions; and relationship among other robots and people.

The intended audience for this standard is robot manufacturers, system integrators, robot end-users (part manufacturers, automotive industry, construction industry, service and solution providers, etc.), robot equipment suppliers, robot software developers, and researchers/developers.

2. Normative references

The following referenced documents are indispensable for the application of this document (i.e., they must be understood and used, so each referenced document is cited in text and its relationship to this document is explained). For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments or corrigenda) applies.

[N1] Huang, H.-M, Albus, J., and Messina, E.,"Toward a generic model for autonomy levels for unmanned systems (ALFUS)," *Performance Metrics for Intelligent Systems (PerMIS) Workshop*, Gaithersburg, MD, USA, 2003.

[N2] ISO/FDIS 8373, "Robots and robotic devices" – Vocabulary, 1996.

[N3] *SUO-KIF*, http://suo.ieee.org/SUO/KIF/suo-kif.html. Acessed on 29 October 2013.

[N4] *SUMO*. http://www.ontologyportal.org. Acessed on 29 October 2013.

[N5] Niles, I., and Pease, A., "Towards a standard upper ontology," *International Conference on Formal Ontology in Information Systems*, Ogunquit, ME, USA, pp. 2—9, Oct. 2001.

3. Definitions

For the purposes of this document, the following terms and definitions apply. The *IEEE Standards Dictionary Online* should be consulted for terms not defined in this clause. [1]

Some definitions refer to concepts and relations from SUMO ontology [N4][N5]. These references are marked *itallic*.

artificial system: An artifact (*Artifact*, in SUMO) formed by various interacting devices (*Device*, in SUMO) and other objects (*Object*, in SUMO) in order to execute a function.

automated robot: It is a role of a robot in a given task in which the robot acts as an automaton, not adapting to changes in the environment and/or following scripted plans. Contrast: fully autonomous; semi-autonomous robot; teleoperated robot; remote controlled robot.

collective robotic system: A robotic system having a robot group as part. *See also:* **robotic system**; **robot group**. **Contrast: simple robotic system**.

[1] *IEEE Standards Dictionary Online* subscription is available at: http://www.ieee.org/portal/innovate/products/standard/standards_dictionary.html.

coordinate system: An abstract entity (*Abstract*, in SUMO) which is defined in relation to a single reference object (*Object*, in SUMO). Coordinate systems are related through hierarchies (i.e. trees). For instance, the local coordinate system of a robot is referenced by the robot itself. The reference object is not necessarily the origin of the coordinate system. A coordinate system defines at least one dimension in which points get their coordinate values. Points in a given coordinate system can be mapped to other coordinate systems by means of a transformation. *See also:* **global coordinate system**; **local coordinate system**; **transformation**.

design: A design is a proposition (*Proposition*, in SUMO) that idealize the structure of one or more artifacts (*Artifact*, in SUMO). A design is particularly necessary to abstract information in contexts such as industrial robotics.

fully autonomous robot: It is a role of a robot in a given task in which the robot solves the task without human intervention while adapting to operational and environmental conditions. **Contrast: semi-autonomous robot, teleoperated robot; remote controlled robot; automated robot.**

global coordinate system: Usually, an agent chooses an arbitrary coordinate system as the global reference frame, which constitutes the global coordinate system for that agent. In a hierarchy of local coordinate systems, the global coordinate system is the one to which all local coordinate systems refer. It is the root of a tree of local coordinate systems. *See also:* **coordinate system**. **Contrast: local coordinate system**.

interaction: A process (*Process*, in SUMO) held between two agents (*Agent*, in SUMO). It is composed by two subprocesses defining action and reaction, such that a subprocess (e.g. action) initiated by agent x on a patient agent y causes a second process (e.g. reaction) having y as agent and x as patient.

local coordinate system: A coordinate system bound to a hierarchical structure of coordinate systems is called a local coordinate system. An agent (*Agent*, in SUMO) arbitrarily defines one of the local coordinate systems as the root of the hierarchy of coordinate systems. *See also:* **coordinate system**. **Contrast: global coordinate system**.

orientation measurement: Essentially a measure (*measure*, in SUMO) attributed to a (physical) object (*Object*, in SUMO) concerning the point or region at which the object is pointing/facing. *See also:* **pose**; **orientation point**; **orientation region**; **coordinate system**; **robot**.

orientation point: It is a point in a coordinate system denoting an orientation value. Orientation points in one coordinate system can be mapped to other coordinate systems. An example of use of orientation point is in "the robot is oriented 54 degrees in relation to the reference object". *See also:* **orientation measurement**. **Contrast: orientation region**.

orientation region: Defines an region or interval orientation in relation to a reference object (*Object*, in SUMO). For instance, the "south" interval of a compass constitutes an orientation region in the one-dimensional, circular coordinate system of the compass. Eventually, position regions and orientation regions are referred by similar words. For instance, it is valid to say that a robot is at the north position, facing north. The former relates to a position region; i.e., the north region of a given country; the later relates to an orientation region; i.e., the interval around north on the compass. *See also:* **orientation measurement**; **position region**. **Contrast: orientation point**.

physical environment: An object (*Object*, in SUMO) comprising at least one region (*Region*, in SUMO) and one object (*Object*, in SUMO) located in that region.

pose: A position and an orientation constitute a pose. The pose of an object is the description of any position and orientation bearing the same object. *See also:* **position measurement**; **orientation measurement**.

position measurement: Essentially a measure (*measure*, in SUMO) attributed to a (physical) object (*Object*, in SUMO) describing its position. A position can be described by a point or a region. For instance, one can describe a robot as positioned at coordinates (x, y) in the coordinate system, or at the front of the box, where "front" comprises a conical region centered on the box and pointed forward. *See also:* **pose**; **position point**; **position region**; **coordinate system**; **robot**.

position point: A position point refers to a point in a coordinate system. Position points are always defined in a single coordinate system. Two physical objects cannot have the exact same position position in the same coordinate system; i.e., they cannot be located at the same position point. *See also:* **position measurement**. **Contrast: position region**.

position region: A position region is an abstract region in a coordinate system. More specifically, a position region is defined by position points in a given coordinate system. It defines qualitative positions such as "left of", "in front of", "on top of", etc. These expressions define regions in relation to a reference object or in which other objects are placed. A position region is always generated by a given spatial operator applied on a reference object. *See also:* **position measurement**. **Contrast: position point**.

remote controlled robot: It is a role of a robot in a given task in which the human operator controls the robot on a continuous basis, from a location off the robot via only her/his direct observation. In this mode, the robot takes no initiative and relies on continuous or nearly continuous input from the human operator. Contrast: fully autonomous; semi-autonomous robot; teleoperated robot; automated robot.

robot group: A group (*Group*, in SUMO) of robots organized to achieve at least one common goal. *See also:* **robot**.

robot part: Devices (*Device*, in SUMO) that are parts of robots. *See also:* **robot**.

robot: For general purposes, robots are agentive devices (*Agent* and *Device*, in SUMO) in a broad sense, purposed to act in the physical world in order to accomplish one or more tasks. In some cases, the actions of a robot might be subordinated to actions of other agents (*Agent*, in SUMO), such as software agents (bots) or humans. A robot is composed of suitable mechanical and electronic parts. Robots might form social groups, where they interact to achieve a common goal. A robot (or a group of robots) can form robotic systems together with special environments geared to facilitate their work. *See also:* **autonomous robot**; **non-autonomous robot**; **semi-autonomous robot**; **robot group**; **robotic system**.

robotic environment: A physical environment equipped with a robotic system. *See also:* **physical environment**; **robotic system**.

robotic system: An artificial system formed by one or more robots (single robots or groups of robots) and at least one device supporting the operation of the robot(s). *See also:* **artificial system**; **robot**; **simple robotic system**; **collective robotic system**.

semi-autonomous robot: It is a role of a robot in a given task in which the robot and a human operator plan and conduct the task, requiring various levels of human interaction. Contrast: fully autonomous robot, teleoperated robot; remote controlled robot; automated robot.

simple robotic system: A robotic system having one and only one robot as part. *See also:* **robotic system**. **Contrast: collective robotic system**.

spatial operator: A mathematical function that can map reference objects (*Object*, in SUMO) to regions in a coordinate system. A position region is always generated by a given spatial operator applied on a reference object. *See also:* **coordinate system**.

teleoperated robot: It is a role of a robot in a given task in which an human operator, using sensory feedback, either directly controls the actuators or assigns incremental goals on a continuous basis, from a location off the robot. Contrasts: fully autonomous; semi-autonomous robot; remote controlled robot; automated robot.

transformation: Points in a coordinate system can be mapped to another coordinate system by means of a transformation. *See also:* **coordinate system**.

4. Core Ontology for Robotics and Automation : Axioms

The following sections present the formal definitions of the Core Ontology for Robotics and Automation (CORA) and additional ontologies.

The formal definitions are written as SUO-KIF [N3] formulas, such that:

a) Terms get a prefix determining in which ontology defines them. For instance, the predicate "CORA:Robot" determine that 'Robot" is defined in the "CORA" ontology.

b) Terms with no prefix are assumed to be entities defined in SUMO [N4] or primitives of SUO-KIF.

4.1 SUMO

The SUMO ontology [N4][N5] is a top-level ontology that defines the more basic ontological categories across all domains. This section gives only a brief overview of its main concepts, illustrated in Figure 1. Check the following normative references [N4][N5] for more information.

The main SUMO category is *Entity*, which is a disjoint partition of *Physical* and *Abstract* concepts. *Physical* represents entities that have space-temporal extension. *Abstract* describes entities that do not have or are chosen not to have space-temporal extensional.

Physical is further partitioned into *Object* and *Process*. *Object* exists in space, having spatial parts parallel in time. *Process* abstracts individuals present only in time.

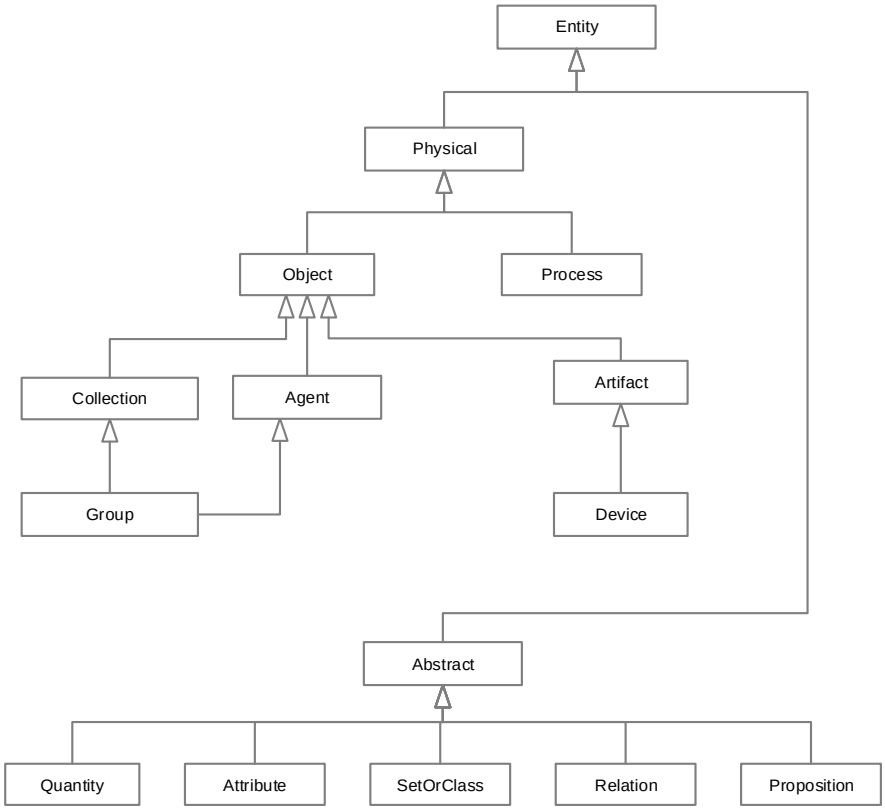


Figure 1– Basic SUMO Taxonomy

Physical things are separated in objects and processes, with processes being things that have temporal parts or stages. This means SUMO follows an endurantist perspective instead of a perdurantist one. For an endurantist, an object keeps its identity through time and so, while some processes might change things about it, every part that is essential to it is always present. On the other hand, for a perdurantist, an object is composed of every temporal part it has at all times and so all things about it are indexed in time.

An easy analogy is to think that perdurantists see things as regions in a 4D space while endurantists see them as 3D things that can change in processes.

Abstract is further partitioned into *Quantity*, *Attribute*, *SetOrClass*, *Relation* and *Proposition*. *Quantity* abstracts numeric and physical quantities. *Attribute* abstracts qualities which cannot or are chosen not to be reified as subclasses of Object. *SetOrClass* abstracts entities that have elements or instances. *Relation* abstracts n-ary relations, functions and lists. Finally, *Proposition* abstracts the notion of content of an entity, such as a formula or a drawing.

4.2 CORAX Axioms

The CORAX ontology defines concepts that are too general to be in the CORA ontology. These concepts cover aspects of reality that are necessary for modelling some aspects of the robotics, but are not explicitly or completely covered by SUMO.

4.2.1 CORAX:Design

The notion of product design is important across many domains. It is particularly necessary to abstract information in contexts such as industrial robotics.

A design is a *proposition*:

```
(subclass CORAX:Design Proposition)
```

A design idealizes the intended structure of one or more artifacts:

```
(instance CORAX:idealizes BinaryPredicate)
(instance CORAX:idealizes AsymmetricRelation)
(instance CORAX:idealizes IrreflexiveRelation)
(domain 1 CORAX:idealizes CORAX:Design)
(domain 2 CORAX:idealizes Artifact)
```

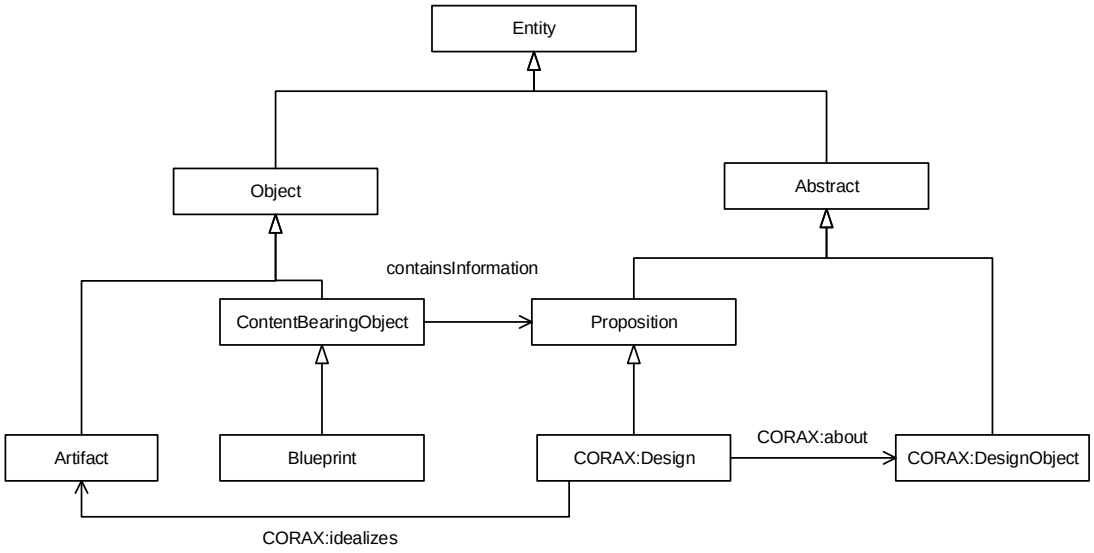


Figure 2 - Concepts related to CORAX:Design and CORAX:DesignObject.

Designs are *about* design objects. Design objects are abstract idealizations of the individual artifacts produced by the design:

```
(subclass CORAX:DesignObject Abstract)
```

The relation *about* captures the association between designs and design objects. More generally, the relation *about* states that a proposition states facts about some entity:

```
(instance CORAX:about BinaryRelation)
(instance CORAX:about AsymmetricRelation)
(instance CORAX:about IrreflexiveRelation)
(domain 1 Proposition)
(domain 2 Entity)

(=>
    (instance ?O CORAX:DesignObject)
```

```
    (exists (?D)
        (and
            (instance ?D CORAX:Design)
            (CORAX:about ?D ?O)
```

Design objects bear attributes and measurements that are expected to be found in the produced artifacts. These notions are represented by creating abstract counterparts of the relations attribute and measure:

```
(subrelation CORAX:designAttribute property)
(instance CORAX:designAttribute BinaryRelation)
(instance CORAX:designAttribute AsymmetricRelation)
(instance CORAX:designAttribute IrreflexiveRelation)
(domain CORAX:designAttribute 1 CORAX:DesignObject)
(domain CORAX:designAttribute 2 Attribute)


(instance CORAX:designMeasure BinaryRelation)
(instance CORAX:designMeasure AsymmetricRelation)
(domain CORAX:designMeasure 1 CORAX:DesignObject)
(domain CORAX:designMeasure 2 PhysicalQuantity)
```

Design objects may have other design objects as parts, e.g. reflecting the structure of the designed artifact. The relation must be represented as instances of *abstractPart*.

```
(=>
    (and
        (instance ?O CORAX:DesignObject)
        (abstractPart ?P ?O))
    (instance ?P CORAX:DesignObject)
```

SUMO states that propositions can be represented by *content bearing physicals*, such as a document, a diagram, or a computer CAD file. This standard does not put any restriction on the kinds of content bearing physicals that can represent a design.

The formula that defines a design object can be seen as one of the possible content bearing objects that describe a design, along other kinds of entities, such as CAD files or blueprints. It is possible to establish a formal link between the formula and the design that it represents by stating that the formula must entail the existence of at least one design object related to that design:

```
(=>
    (and
        (Formula ?DESC)
        (CORAX:Design ?DESIGN)
        (containsInformation ?DESC ?DESIGN))
    (entails ?DESC
        (exists ?DO
            (and
                (CORAX:DesignObject ?DO)
                (CORAX:about ?DESIGN ?DO)))))
```

4.2.2 CORAX:PhysicalEnvironment

A physical environment is an object comprising at least one region and one object located in that region.

```
(subclass CORAX:PhysicalEnvironment Object)
```

```
(=>
    (instance ?ENV CORAX:PhysicalEnvironment)
    (exists (?R ?O)
        (and
            (instance ?R Region)
            (part ?R ?ENV)
            (instance ?O Object)
            (part ?O ?ENV)
            (located ?O ?R))))
```

Physical environments have at least two parts: a region in which it is located and an object that serves as reference to form the environment. For instance, the environment of an empty room, includes the room and the minimal region in which the room is fully located.

All objects that are part of a physical environment are located in a region that is part of that environment.

```
(=>
    (instance ?ENV CORAX:PhysicalEnvironment)
    (exists (?R)
        (and
            (instance ?R Region)
            (part ?R ?ENV)
            (forall (?O)
                (=>
                    (part ?O ?ENV)
                    (located ?O ?R))))))
```

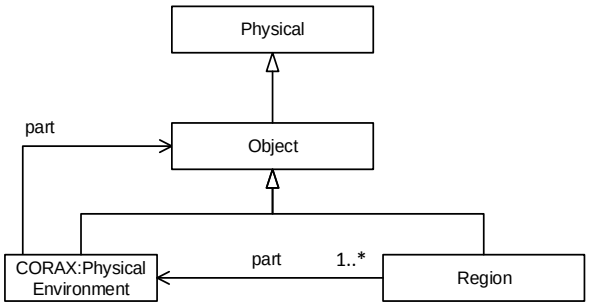


Figure 3 - Concepts related to CORAX:PhysicalEnvironment.

4.2.3 CORAX: Interaction

SUMO does not define the notion of interaction. An interaction is a process:

```
(subclass CORAX:Interaction Process)
```

An interaction is a process held between two agents. It is composed by two subprocesses defining action and reaction, such that a subprocess (e.g. action) initiated by agent *x* on a patient agent *y* causes a second process (e.g. reaction) having *y* as agent and *x* as patient.

```
 (=>
    (instance ?INT  CORAX:Interaction)
```

```
    (exists (?O1 ?O2 ?P1 ?P2)
        (and
            (instance ?P1 Process)
            (instance ?P2 Process)
            (subProcess ?P1 ?INT)
            (subProcess ?P2 ?INT)
            (instance ?O1 Object)
            (instance ?O2 Object)
            (agent ?O1 ?P1)
            (patient ?O2 ?P1)
            (agent ?O2 ?P2)
            (patient ?O1 ?P2)
            (causes ?P1 ?P2))))
```

An interaction defines an interaction relation:

```
(instance CORAX:interactsWith BinaryRelation)
(instance CORAX:interactsWith SymmetricRelation)
(instance CORAX:interactsWith IntransitiveRelation)
(domain CORAX:interactsWith 1 Object)
(domain CORAX:interactsWith 2 Object)

(=>
    (CORAX:interactsWith ?O1 ?O2)
    (exists (?INT ?C1 ?C2)
       (and
           (instance ?INT  CORAX:Interaction)
           (instance ?C1 CaseRole)
           (instance ?C2 CaseRole)
           (playsRoleInEvent ?O1 ?C1 ?INT)
           (playsRoleInEvent ?O2 ?C2 ?INT))))
```

4.2.4 CORAX:ArtificialSystem

An artificial system is an artifact formed by various interacting devices (and other objects) in order to execute a function.

```
(subclass CORAX:ArtificialSystem Artifact)
```

For any part in artificial system, there is at least one other part it interacts with:

```
(=>
    (instance ?S CORAX:ArtificialSystem)
    (forall (?P1)
        (=>
            (part ?P1 ?S)
            (exists (?P2)
               (and
                   (part ?P2 ?S)
                   (interactsWith ?P1 ?P2)
                   (not (equal ?P1 ?P2)))))))
```

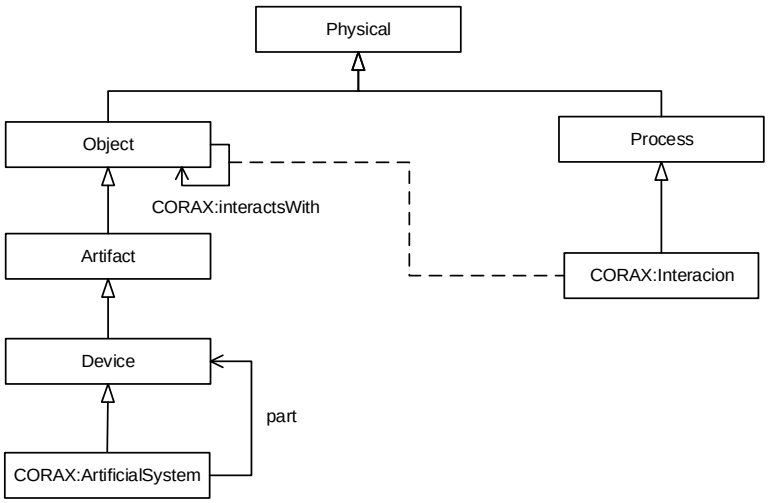


Figure 4 - Concepts related to CORAX:ArtificialSystem and CORAX:Interaction

4.3 CORA axioms

4.3.1 CORA:Robot

The main concept in CORA is robot. It relates most of the concepts in the ontology.

First, a robot is a Device (see Figure 5):

```
(subclass CORA:Robot Device)
```

SUMO defines that a device is an artifact with the purpose of serving as an instrument in a subclass of process. In certain cases, such as those involving multipurpose robots, the class of process may not be known a priori.

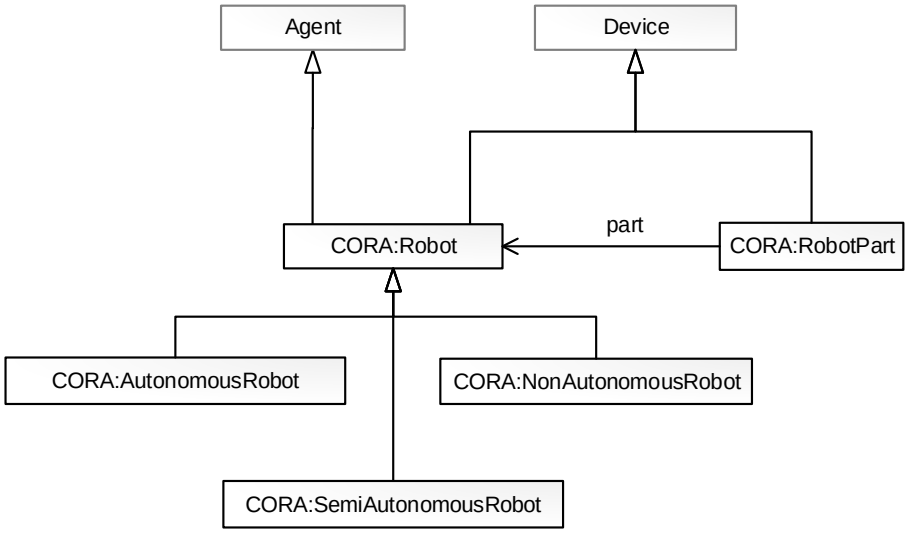


Figure 5 - Main concepts related to robot and robot part.

The instances of CORA:Robot can exhibit qualities through two inherited relations from Object. The attribute relation allows the robot to get qualitative qualities instantiating the abstract class Attribute. The measure relation allows the robot to get quantitative qualities instantiating the abstract class PhysicalMeasure.

A Robot is also an *agent* (see Figure 5);

```
(subclass CORA:Robot Agent)
```

Robots perform tasks by acting on the environment or themselves.

4.3.2 CORA:RobotPart

Robots have other devices as parts. There are a myriad of devices that can play the role of robot part, and it is not possible to determine in advance what devices can or cannot be robot parts.

```
(subclass CORA:RobotPart Device)

(=>
    (instance ?PART CORA:RobotPart)
    (exists (?ROBOT)
        (and
            (instance ?ROBOT CORA:Robot)
            (part ?PART ?ROBOT))))
```

Devices that are considered robot parts are not essentially robot parts, since they exist by themselves and, in most cases, they can be connected to other kinds of devices. For instance, a power source is essentially a device. However, a specific instance of power source can be dynamically classified as a robot part, during a specific time interval, while it is connected to a robot.

Considering a specific robot, when a specific instance of a particular subclass of Device is connected to the robot as a part, then this device is also an instance of a specific subclass of robot part.

```
(=>
    (and
            (instance ?ROBOT CORA:Robot)
```

```
            (instance ?CLASS1 SetOrClass)
            (subclass ?CLASS1 Device)
            (instance ?DEVICE ?CLASS1)
            (part ?DEVICE ?ROBOT))
    (exists (?CLASS2)
        (and
             (instance ?CLASS2 SetOrClass)
             (subclass ?CLASS2 CORA:RobotPart)
             (instance ?DEVICE ?CLASS2))))
```

4.3.3 CORA:RobotInterface

A robot interacts with the world surrounding it through at least one robot interface, which is part of the robot.

```
(subclass CORA:RobotInterface RobotPart)

(=>
    (instance ?ROBOT CORA:Robot)
    (exists (?INTER)
        (and
             (instance ?INTER CORA:RobotInterface)
             (part ?INTER ?ROBOT))))
```

Through the interface, the robot can sense and act on the environment as well as communicate with other agents. An interface is a device like any other that may be composed by other devices such as sensors, actuators or remote controls.

4.3.4 CORA:fullyAutonomousRobot , CORA:semiautonomousRobot , CORA:teleoperatedRobot, CORA:teleoperatedRobot and CORA:automatedRobot

The notion that a robot is an agent might raise some questions in situations where a robot requires a greater amount of input commands from an operator in order to execute tasks; e.g., teleoperated robots. This is one of the main points of ambiguity when differentiating robots from other machines. This issue is related to the problem of modelling autonomy.

The ALFUS [N1] standard defines how autonomy can be evaluated regarding robotic devices. CORA imports the notion of “Modes of Operation for Unmanned Systems” defined in ALFUS and redefines it in relation to the concept of process present in SUMO.

CORA assumes that autonomy relates to a particular occurrence of a process. A robot participating as an agent in a process might participate as a fully autonomous robot, as a semi-autonomous robot, as a teleoperated robot, as a remote controlled robot, or as an automated robot. CORA defines these different situations as case roles that a robot can assume in a process. So:

```
(subrelation CORA:agentRobot agent)
(domain 1 CORA:agentRobot CORA:Robot)
(domain 2 CORA:agentRobot Process)

(subrelation CORA:fullyautonomousRobot CORA:agentRobot)
(subrelation CORA:semiautonomousRobot CORA:agentRobot)
(subrelation CORA:teleoperatedRobot CORA:agentRobot)
(subrelation CORA:remotecontrolledRobot CORA:agentRobot)
```

```
(subrelation CORA:automatedRobot CORA:agentRobot)
```

The relations CORA:fullyAutonomousRobot, CORA:semiAutonomousRobot, CORA:teleoperatedRobot and CORA:remoteControlledRobot represent robots participating as agents in processes in the corresponding operation modes defined in ALFUS [N1], which are Fully Autonomous, Semi-autonomous, Teleoperation and Remote Control, respectively. CORA complements this list with the role CORA:automatedRobot attributed to robots acting as automatons in a process, such as a clockwork robots or fully programmed robots.

It is important to note that a given robot can play some of these roles in different processes at the same time. For example, a robotic rover exploring a planet can assume the role of semi-autonomous in the process of planet exploration, but it can, at the same time, be fully autonomous in the process of navigation.

At this stage, CORA does not define any structural or capability restrictions on the robots that assume these different roles.

4.3.5 CORA:RobotGroup

A robot is an agent and agents can form social groups. According to SUMO, a *group* is "a collection of agents", like a pack of animals, a society or an organization.

A robot group is a group in which the members are only robots (see Figure 6).

```
(subclass CORA:obotGroup Group)

(forall ?MEMBER
    (=>
        (and
            (instance ?GROUP CORA:RobotGroup)
            (member ?MEMBER ?GROUP))
        (instance ?MEMBER CORA:Robot)))
```

According to SUMO, a group is an agent, in the sense that it can act on its own. The agents that compose a group establish its agency.

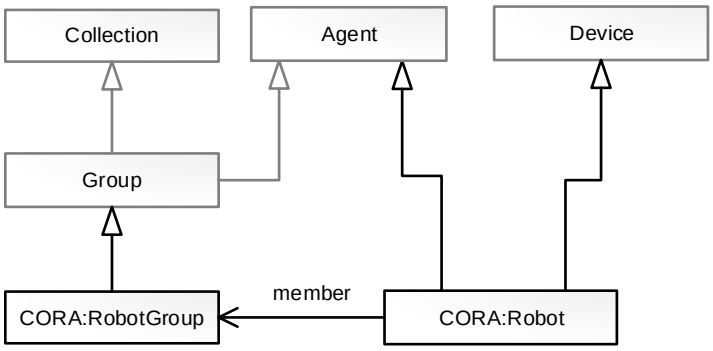


Figure 6 - Part of the ontology showing the main concepts related to CORA:RobotGroup

Examples of robot groups are robot teams, such as robot football teams and a team of soldering robots in a factory.

Robot group also encloses *complex robots*. These are embodied mechanisms formed by many agents attached to each other. An example is a robotic tank in which the hull and the turret are independent autonomous robots that can coordinate their actions to achieve a common goal. Another example is a robotic snake, composed of smaller, autonomous robots.

4.3.6 CORA:RoboticSystem

Robots and other devices can form robotic systems (Figure 7).

Robotic systems are artificial systems, formed by robots and a series of devices intended to support the robots to carry on their task (check ISO [N2] in normative references).

```
(subclass CORA:RoboticSystem CORAX:ArtificialSystem)

(instance CORA:support BinaryRelation)
(domain CORA:support 1 Device)
(domain CORA:support 2 CORA:RoboticSystem)


(=>
     (instance ?SYSTEM CORA:RoboticSystem)
     (exists (?ROBOT)
         (and
             (instance ?ROBOT Device)
             (part ?ROBOT ?SYSTEM)))))

(=>
     (instance ?SYSTEM CORA:RoboticSystem)
     (exists (?DEVICE)
         (and
             (instance ?DEVICE Device)
             (part ?DEVICE ?SYSTEM)
             (support ?DEVICE ?SYSTEM)
             (not (instance ?DEVICE CORA:Robot)))))
```

4.3.7 CORA:RoboticEnvironment

Robotic systems are usually located in environments.

```
(subclass CORA:RoboticEnvironment CORAX:PhysicalEnvironment)

(instance CORA:equippedWith BinaryRelation)
(instance CORA:equippedWith AsymmetricRelation)
(instance CORA:equippedWith ReflexiveRelation)
(domain CORA:equippedWith 1 CORA:RoboticEnvironment)
(domain CORA:equippedWith 2 CORA:RoboticSystem)


(=>
    (instance ?ENV CORA:RoboticEnvironment)
    (exists (?SYSTEM)
        (and
             (instance ?SYSTEM CORA:RoboticSystem)
             (CORA:equippedWith ?ENV ?SYSTEM))))
```

If a robotic environment is defined, then at least one part of the system must be part of the environment.

```
(=>
    (CORA:equippedWith ?ENV ?SYSTEM)
    (forall (?DEVICE)
            (=>
                (part ?DEVICE ?SYSTEM)
                (part ?DEVICE ?ENV))))
```

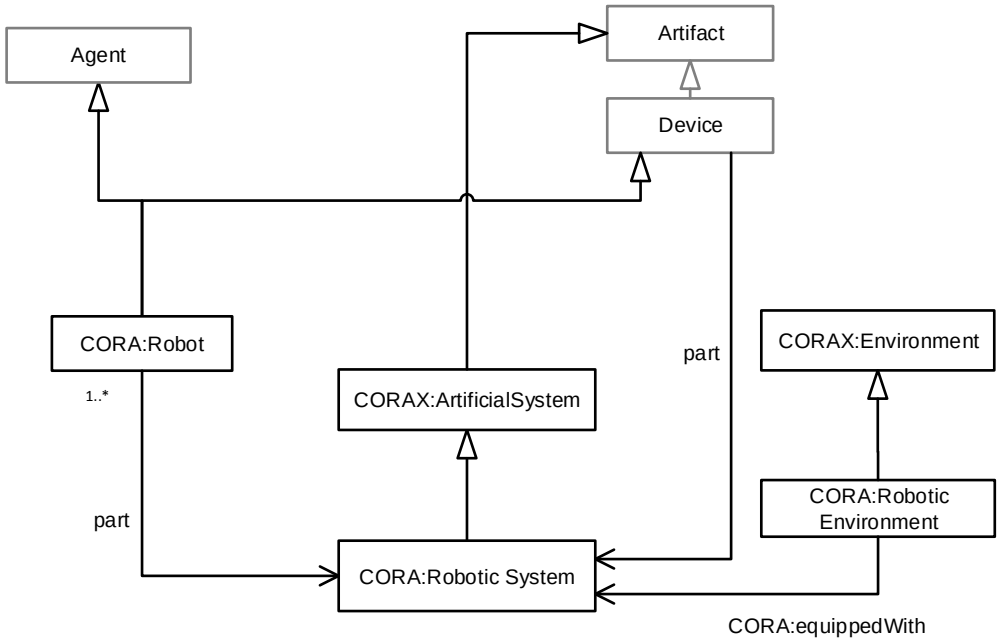


Figure 7- Robotic system and its relations with robot and robotic environment.

4.3.8 CORA:SingleRoboticSystem and CORA:CollectiveRoboticSystem

Robotic system might have one or more robots. Robotic systems are partitioned into single and collective robotic systems.

```
(subclass CORA:SingleRoboticSystem CORA:RoboticSystem)
(subclass CORA:CollectiveRoboticSystem CORA:RoboticSystem)
(partition CORA:RoboticSystem CORA:SingleRoboticSystem
                                    CORA:CollectiveRoboticSystem)
```

Single robotic systems have one and only one robot:

```
(=>
    (instance ?SYSTEM CORA:SingleRoboticSystem)
    (forall (?RA ?RB)
        (=>
            (and
                 (instance ?RA CORA:Robot)
                 (part ?RA ?SYSTEM)
                 (instance ?RB CORA:Robot)
                 (part ?RB ?SYSTEM))
```

```
            (equal ?RA ?RB))))
```

Collective robotic systems have two or more robots:

```
(=>
    (instance ?SYSTEM CORA:CollectiveRoboticSystem)
    (exists (?RA ?RB)
        (and
            (instance ?RA CORA:Robot)
            (part ?RA ?SYSTEM)
            (instance ?RB CORA:Robot)
            (part ?RB ?SYSTEM)
            (not (equal ?RA ?RB)))))
```

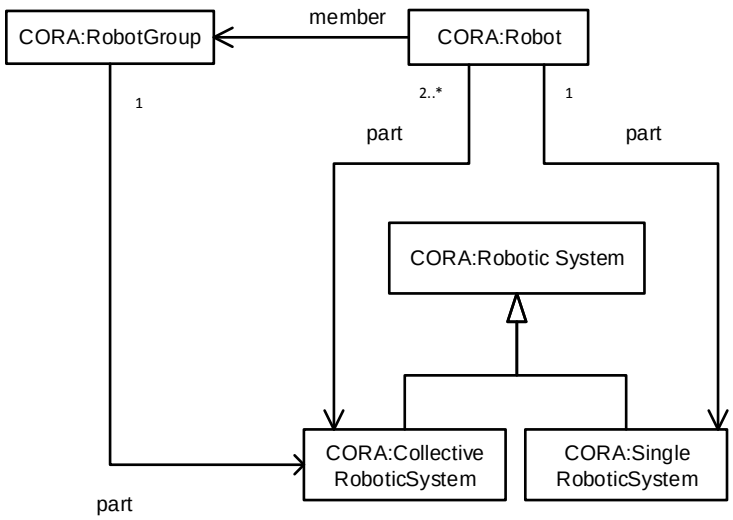


Figure 8– Different types of robotic systems.

It is assumed that all robots in a collective robotic system collaborate to achieve a common goal. As such, all robots in a collective robotic system must be members of a single robot group.

```
(=>
    (and
        (instance ?SYSTEM CORA:CollectiveRoboticSystem)
        (instance ?ROBOT CORA:Robot)
        (part ?ROBOT ?SYSTEM))
    (exists (?G1)
        (forall (?G2)
            (=>
                (and
                    (instance ?G1 CORA:RobotGroup)
                    (member ?ROBOT ?G1)
                    (instance ?G2 CORA:RobotGroup)
                    (member ?ROBOT ?G2))
                (equals ?G1 ?G2)))))
```

Such a robot group is also part of the robotic system.

```
(=>
    (and
```

```
        (instance ?SYSTEM CORA:CollectiveRoboticSystem)
        (instance ?ROBOT CORA:Robot)
        (part ?ROBOT ?SYSTEM))
    (forall (?G)
        (and
            (instance ?G CORA:RobotGroup)
            (member ?ROBOT ?G))
        (part ?G ?SYSTEM)))
```

Another example of robotic system is an automated home assistant system composed of a helper robot as well as by sensors and actuators to open doors. In particular, the notion of robotic system is motivated by the need to describe industrial robotic settings in other ORA ontologies.

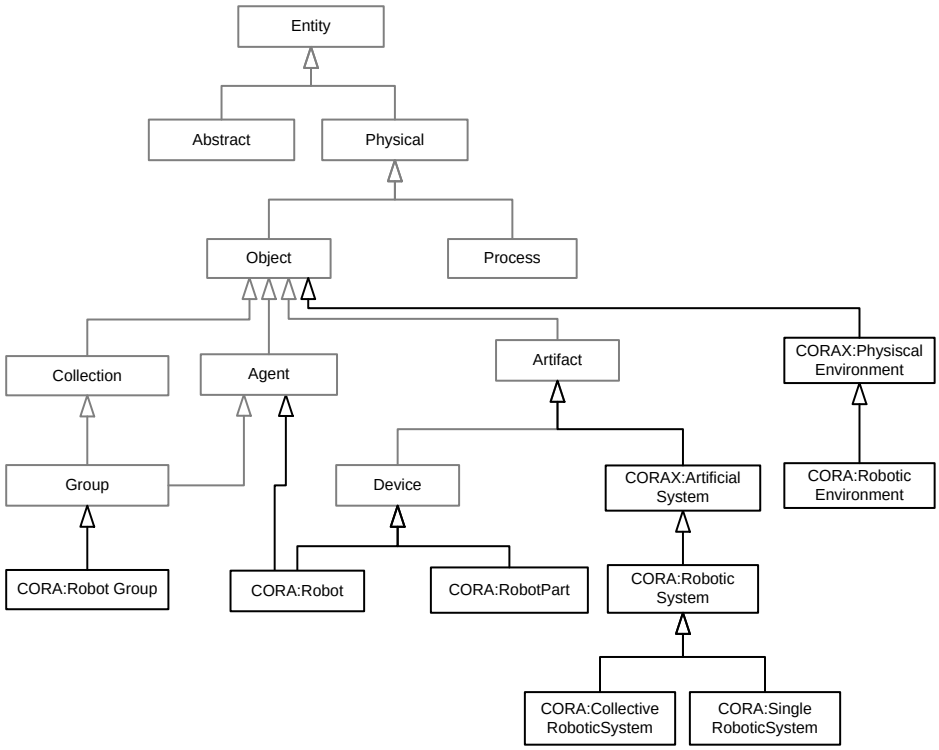


Figure 9 - Overview of taxonomy of concepts in CORA.

5. RPARTS axioms

CORA defines that any Device can be used as a robot part. Nevertheless, there are some devices that are more typically used as robot parts than others. The RPARTS ontology aggregates some of the most general kinds of robot parts, extending the ones already existing in SUMO.

RPARTS provides only the taxonomy of devices and their natural language definition. Nevertheless, these concepts can be extended in special applications.

Some of the devices are defined in SUMO [N4][N5] and in ISO/FDIS 8373 [N2].

5.1 RPARTS:Sensor

Sensors are measuring devices. They are divided into proprioceptive and exterioceptive.

```
(subclass RPARTS:Sensor MeasuringDevice)
(subclass RPARTS:Proprioceptive RPARTS:Sensor)
(subclass RPARTS:Exterioceptive RPARTS:Sensor)
```

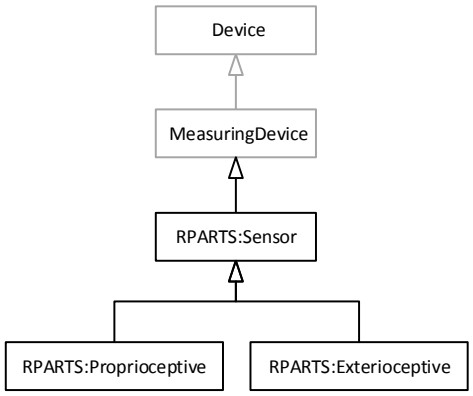


Figure 10 - Concepts related to Sensors

5.2 RPARTS:LocomotionDevice

Locomotion devices includes wheels, tracks, legs, wings and propellers. It is important to notice that this list is not exclusive.

```
(subclass RPARTS:LocomotionDevice Device)

(subclass RPARTS:TrackDevice RPARTS:LocomotionDevice)
(subclass RPARTS:LegDevice RPARTS:LocomotionDevice)
(subclass RPARTS:WingDevice RPARTS:LocomotionDevice)
(subclass RPARTS:PropellerDevice RPARTS:LocomotionDevice)

(subclass RPARTS:WheelLocomotionDevice RPARTS:LocomotionDevice)
(subclass RPARTS:WheelLocomotionDevice Wheel)

(subclass RPARTS:SwedishWheel RPARTS:WheelLocomotionDevice)
(subclass RPARTS:SphericalWheel RPARTS:WheelLocomotionDevice)
```

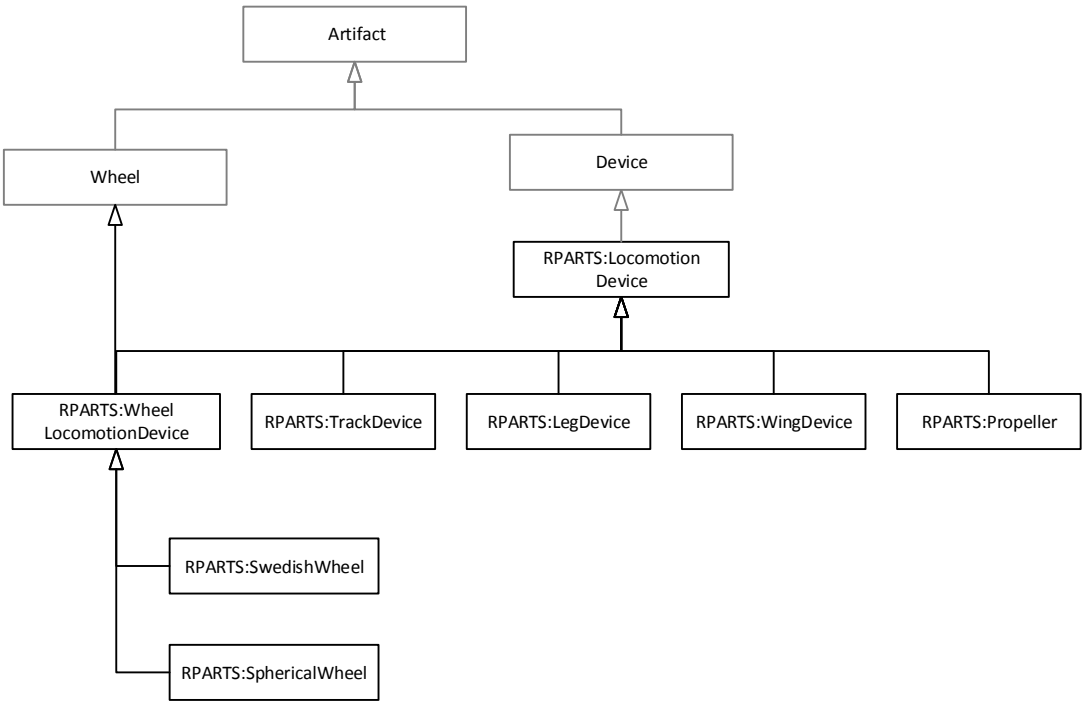


Figure 11 - Concepts related to Locomotion Devices

5.3 RPARTS:Actuator

Actuators can be hydraulic, pneumatic and electric. It is important to notice that other subclasses of actuator can be included.

```
(subclass RPARTS:Actuator Device)

(subclass RPARTS:HydraulicActuator RPARTS:Actuator)
(subclass RPARTS:PneumaticActuator RPARTS:Actuator)
(subclass RPARTS:ElectricActuator  RPARTS:Actuator)
```

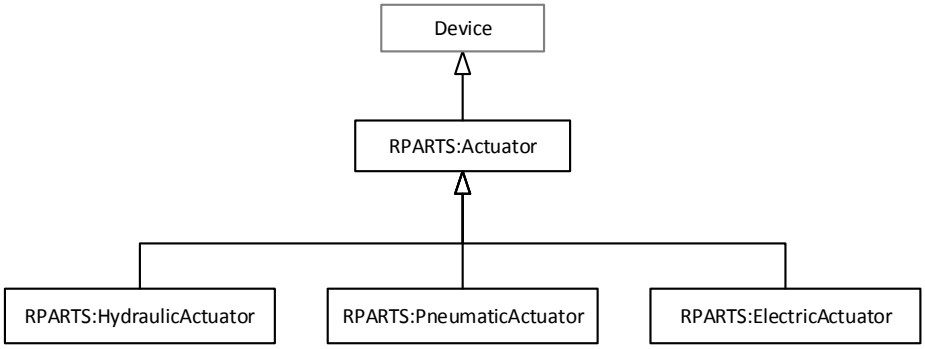


Figure 12 – Concepts related to Actuators

5.4 RPARTS:EndEffector

End effectors can be Grippers, Vacuum Effectors and Multi fingered hand effectors. Other subclasses of end effector can be included

```
(subclass RPARTS:EndEffector Device)

(subclass RPARTS:GripperEffector RPARTS:EndEffector)
(subclass RPARTS:VacuumEffector RPARTS:EndEffector)
(subclass RPARTS:MultifingeredHandEffector RPARTS:EndEffector)
```

There are two types of vacuum effector

```
(subclass RPARTS:VacuumEffectorSingleCup RPARTS:VacuumEffector)
(subclass RPARTS:VacuumEffectorMultiCup  RPARTS:VacuumEffector)
```

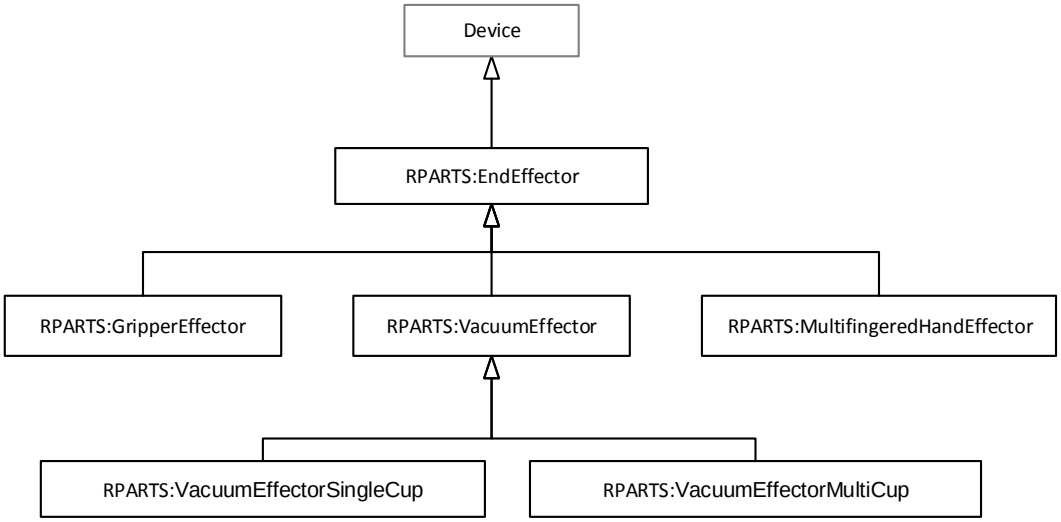


Figure 13 – Concepts related to EndEffectors

5.5 RPARTS:ElectronicDevice

Electronic devices comprise devices having electronic circuitry.

```
(subclass RPARTS:ElectronicDevice Device)

(subclass RPARTS:Microcontroller RPARTS:ElectronicDevice)
(subclass RPARTS:MotorController RPARTS:ElectronicDevice)

(subclass RPARTS:StepperMC   RPARTS:MotorController)
(subclass RPARTS:BrushedMC   RPARTS:MotorController)
(subclass RPARTS:ServoMC     RPARTS:MotorController)
(subclass RPARTS:BrushlessMC RPARTS:MotorController)
```

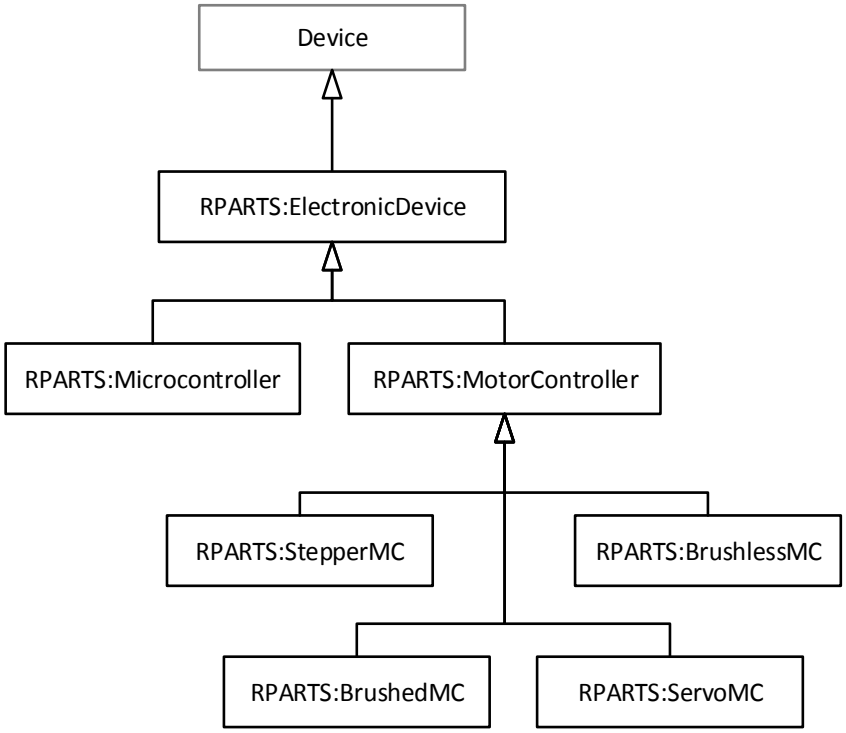


Figure 14 – Concepts related to Electronic Devices

5.6 RPARTS:MechanicalJoint

There are at least five types of mechanical joints. The universal joint is defined in SUMO.

```
(subclass RPARTS:SphericalJoint MechanicalJoint)
(subclass RPARTS:PrimasticJoint MechanicalJoint)
(subclass RPARTS:RevoluteJoint  MechanicalJoint)
(subclass RPARTS:SphericalJoint MechanicalJoint)
```

This list is not exclusive, and other types of joints can be conceived.

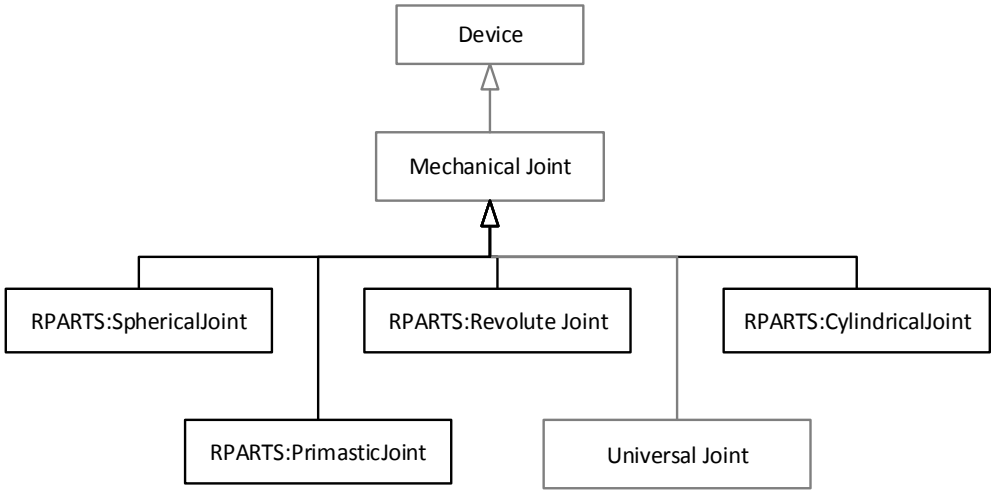


Figure 15 – Concepts related to Mechanical Joints

5.7 Other parts

Other important devices that play the role of robot parts include:

```
(subclass RPARTS:RoboticBase Device)
(subclass RPARTS:RobotChassis Device)
(subclass RPARTS:MechanicalLink Device)
(subclass RPARTS:ArmDevice Device)
(subclass RPARTS:WristDevice Device)
```

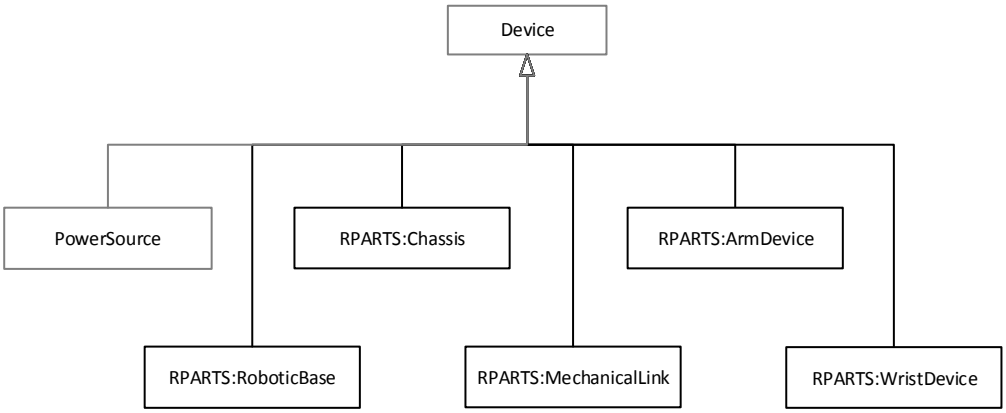


Figure 16 – Concepts related to other robot parts

6. POS axioms

POS is an ontology about position, orientation and pose.

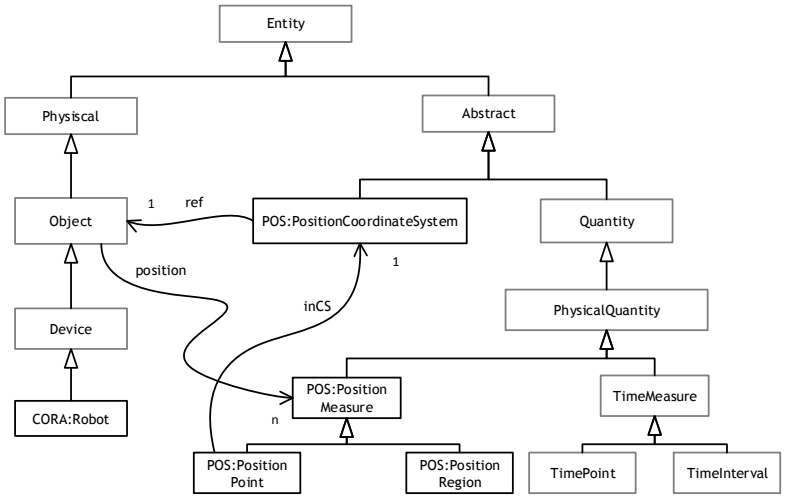


Figure 17 — Main concepts in POS ontology.

6.1 POS:PositionMeasure

A position is essentially a measure (or observation) attributed to a given (physical) object. The domain of values from which to take position values is a specialization of physical quantity, as defined by SUMO.

```
(subclass POS:PositionMeasure PhysicalQuantity)
```

A relation between objects and position measure values determines a position.

```
(subrelation POS:position measure)
(domain POS:position 1 Object)
(domain POS:position 2 POS:PositionMeasure)
```

A position point refers to a point in a coordinate system projected on the physical space. A position region is an abstract region in a coordinate system overlapping the physical spatial region occupied by the object. Both position point and position region are types of position measurement.

```
(subclass POS:PositionPoint POS:PositionMeasure)
(subclass POS:PositionRegion POS:PositionMeasure)

(partition POS:PositionMeasure POS:PositionPoint POS:PositionRegion)
```

6.2 POS:PositionCoordinateSystem

A coordinate system is an abstract entity.

```
(subclass POS:PositionCoodinateSystem Abstract)
```

A position coordinate system is defined in relation to a single reference object.

```
(instance POS:refPCS SingleValuedRelation)
(instance POS:refPCS BinaryPredicate)
(domain POS:refPCS 1 POS:PositionCoordinateSystem)
(domain POS:refPCS 2 Object)
```

6.3 POS:PositionTransformation

A transformation represents mappings between position points.

```
(subclass POS:PositionTransformationFn UnaryFunction)
(domain POS:PositionTransformationFn POS:PositionPoint)
(range  POS:PositionTransformationFn POS:PositionPoint)
```

Transformation define mapping between coordinate systems.

```
(instance POS:mapsPCS TernaryRelation)
(domain POS:mapsPCS 1 POS:PositionTransformationFn)
(domain POS:mapsPCS 2 POS:PositionCoordinateSystem)
(domain POS:mapsPCS 3 POS:PositionCoordinateSystem)
```

There is a mapping from one coordinate system to another iff there is a transformation that maps all points in one coordinate system to the another.

```
 (<=>
    (POS:mapsPCS ?T ?C1 ?C2)
    (forall (?P)
        (=>
            (and
                (instance ?P PositionMeasure)
                (POS:inPCS ?P ?C1))

            (exists (?PM)
                (and
                    (POS:inPCS ?PM ?C2)
                    (equal ?PM
                        (?T ?P)))))))
```

Transformations can be composed transitively by the function POS:pCompose.

```
(instance POS:pCompose SingleValuedRelation)
(instance POS:pCompose TernaryPredicate)
(domain POS:pCompose 1 POS:PositionTransformationFn)
(domain POS:pCompose 2 POS:PositionTransformationFn)
(domain POS:pCompose 3 POS:PositionTransformationFn)

(=>
    (POS:pCompose ?T1 ?T2 ?TT)
    (forall (?P ?PM ?PF)
        (=>
            (and
                (equal ?PM
                     (?T1 ?P))
                (equals ?PF
                     (?T2 ?PM)))
```

```
            (equals ?PF
                (?TT ?P))))))
```

In Robotics (as in other disciplines), coordinate systems are also related through hierarchies (i.e. trees). Usually, an agent chooses an arbitrary coordinate system as the global reference frame, which constitutes the global coordinate system (GCS) for that agent. Local coordinate systems (LCS) are defined in relation to GCS by hierarchical links. This notion of hierarchy is an arbitrary one defined by the agent and can be represented formally using

```
(instance POS:parentPCS BinaryPredicate)
(instance POS:parentPCS TransitiveRelation)
(domain POS:parentPCS 1 POS:CoordinateSystem)
(domain POS:parentPCS 2 POS:CoordinateSystem)
```

If two coordinate systems are hierarchically related, all points from one can be mapped to points in the other using transformations. This hierarchical link is represented as

```
(=>
    (POS:parentPCS ?C1 ?C2)
    (exists (?T1 ?T2)
        (and
            (instance ?T1 POS:PositionTransformationFn)
            (instance ?T2 POS:PositionTransformationFn)
            (POS:mapsPCS ?T1 ?C1 ?C2)
            (POS:mapsPCS ?T2 ?C2 ?C1))))
```

This means that, if two coordinate systems share a parent node in the hierarchy tree, there is a transformation between them. In fact, this transformation can be built in the following manner

```
(=>
    (and
        (POS:parentPCS ?C1 ?C2)
        (POS:parentPCS ?C2 ?C3))
    (exists (?T1 ?T2 ?T3)
        (and
          (instance ?T1 POS:PositionTransformationFn)
            (POS:mapsPCS ?T1 ?C1 ?C2)
          (instance ?T2 POS:PositionTransformationFn)
            (POS:mapsPCS ?T2 ?C2 ?C3)
            (POS:pCompose ?T1 ?T2 ?T3)
          (POS:mapsPCS ?T3 ?C1 ?C3))))
```

6.4 POS:PositionPoint

A position point denotes the quantitative position of an object in a coordinate system.

```
(instance POS:inPCS SingleValuedRelation)
(instance POS:inPCS BinaryPredicate)
(domain POS:inPCS 1 POS:PositionMeasure)
(domain POS:inPCS 2 POS:PositionCoordinateSystem)
```

Position points are always defined in a single coordinate system

```
(=>
    (instance ?P POS:PositionPoint)
```

```
    (exists (?C)
        (and
            (instance ?C  POS:PositionCoodinateSystem)
            (POS:inPCS ?P ?C))))
```

The position of an object in reference to another can be explicitly by means of a ternary predicate:

```
(instance POS:relPosition TernaryPredicate)
(domain POS:relPosition 1 Object)
(domain POS:relPosition 2 POS:PositionMeasure)
(domain POS:relPosition 3 Object)

(<=>
    (POS:relPosition ?O ?P ?OR)
    (and
        (POS:position ?O ?P)
        (exists (?CS)
            (and
                (instance ?CS POS:PositionCoordinateSystem)
                (POS:inPCS ?P ?CS)
                (POS:refPCS ?CS ?OR)))))
```

6.5 POS:PositionRegion

A position region is a qualitative position in a coordinate system:

```
(subclass POS:PositionRegion POS:PositionMeasurement)
```

Position points can be in position regions.

```
(instance POS:inPR BinaryRelation)
(domain POS:inPR 1 POS:PositionMeasure)
(domain POS:inPR 2 POS:PositionRegion)
```

A position region is related to one and only one coordinate system.

```
(=>
    (instance ?R POS:PositionRegion)
    (exists (?C)
        (and
            (instance ?C POS:PositionCoordinateSystem)
            (forall (?P)
                (=>
                   (POS:inPR ?P ?R)
                   (POS:inPCS ?P ?C))))))

(=>
    (and
        (instance ?R POS:PositionRegion)
        (POS:inPR ?P1 ?R)
        (POS:inPCS ?P1 ?C1)
        (POS:inPR ?P2 ?R)
        (POS:inPCS ?P2 ?C2))
    (equal ?C1 ?C2)
```

POS defines operators that act as generator of position regions, based on a single reference object. Operators can be of many types. For instance, a single operator can get the position point of an object and generate a position region that corresponds to the left side of that object.

```
(subclass POS:SpatialOperator Function)

(instance POS:pGenerated SingleValuedRelation)
(instance POS:pGenerated TernaryPredicate)
(domain POS:pGenerated 1 POS:SpatialOperator)
(domain POS:pGenerated 2 Object)
(domain POS:pGenerated 3 POS:PositionRegion)

(=>
    (instance ?R POS:PositionRegion)
    (exists (?OR ?G ?C)
        (and
            (instance ?G POS:spatialOperator)
            (POS:pGenerated ?G ?OR ?R)
            (POS:refPCS ?C ?OR)
            (forall (?P)
               (=>
                 (POS:inPR ?P ?R)
                 (POS:inPCS ?P ?C))))))
```

The position of an object in a position region is defined by the overlap between the object extension on a coordinate system and the position region.

The object extension in a coordinate system is the minimal region containing the object in a given coordinate system.

```
(instance POS:extensionPos TernaryPredicate)
(instance POS:extensionPos SingleValuedRelation)
(domain POS:extensionPos 1 Object)
(domain POS:extensionPos 2 POS:PositionCoordinageSystem)
(domain POS:extensionPos 3 POS:PositionRegion)
```

Position regions are related by special spatial relation. This standard defines at least one relation.

```
(instance POS:overlapsPosition BinaryRelation)
(instance POS:overlapsPosition ReflexiveRelation)
(instance POS:overlapsPosition SymmetricRelation)
(domain POS:overlapsPosition 1 POS:PositionRegion)
(domain POS:overlapsPosition 2 POS:PositionRegion)
```

These two notions allow the definition of position in terms of position regions as follows

```
(=>
    (and
        (instance ?O Object)
        (instance ?R POS:PositionRegion)
        (POS:position ?O ?R))
    (exists (?C)
        (and
          (instance ?C POS:PositionCoordinateSystem)
```

```
            (POS:extensionPos ?O ?C ?PR)
            (POS:overlapsPosition ?PR ?R)))
```

Relative position can also be defined for position regions.

```
(<=>
    (and
        (POS:relPosition ?O ?R ?OR)
        (instance ?R POS:PositionRegion))
    (exists (?G)
        (and
            (POS:position ?O ?R)
            (instance ?R POS:PositionRegion)
            (POS:pGenerated ?G ?OR ?R)
```

6.6 POS:OrientationMeasure

Analogous to a position measure, an orientation is essentially a measure (or observation) attributed to the orientation of a given (physical) object. The domain of values from which to take orientation values is a specialization of physical quantity.

```
(subclass POS:OrientationMeasure PhysicalQuantity)
```

A relation between objects and values of orientation measure determines the orientation between objects.

```
(subrelation POS:orientation measure)
(domain POS:orientation 1 Object)
(domain POS:orientation 2 POS:OrientationMeasure)
```

An orientation point refers to a point in an orientation coordinate system representing the exact orientation between two objects. An orientation region is an abstract region used to represent the qualitative orientation between objects. Both orientation point and orientation region are types of orientation measurement.

```
(subclass POS:OrientationPoint POS:OrientationMeasure)
(subclass POS:OrientationRegion POS:OrientationMeasure)

(partition          POS:OrientationMeasure          POS:OrientationPoint
POS:OrientationRegion)
```

6.7 POS:OrientationCoordinateSystem

An orientation coordinate system is an abstract entity.

```
(subclass POS:OrientationCoodinateSystem Abstract)
```

An orientation coordinate system is defined in relation of a single reference object.

```
(instance POS:refOCS SingleValuedRelation))
(instance POS:refOCS BinaryPredicate)

(domain POS:refOCS 1 POS:OrientationCoordinateSystem)
(domain POS:refOCS 2 Object)
```

6.8 POS:OrientationTransformation

A transformation represents a mapping of orientation points:

```
(subclass POS:OrientationTransformationFn UnaryFunction)
(domain POS:OrientationTransformationFn POS:OrientationPoint)
(range  POS:OrientationTransformationFn POS:OrientationPoint)
```

If a transformations maps all points of an orientation coordinate system to another, this is represented using POS:mapsOCS as follows

```
(instance POS:mapsOCS TernaryRelation)
(domain POS:mapsOCS 1 POS:OrientationTransformationFn)
(domain POS:mapsOCS 2 POS:OrientationCoordinateSystem)
(domain POS:mapsOCS 3 POS:OrientationCoordinateSystem)

 (<=>
    (POS:mapsOCS ?T ?C1 ?C2)
    (forall (?P)
        (=>
            (and
                (instance ?P OrientationMeasure)
                (POS:inOCS ?P ?C1))
             (exists (?PM)
                (and
                    (POS:inOCS ?PM ?C2)
                        (equal ?PM
                            (?T ?P))))))
```

Orientation transformations can be composed transitively by the function POS:oCompose.

```
(instance POS:oCompose SingleValuedRelation)
(instance POS:oCompose TernaryPredicate)))
(domain POS:oCompose 1 POS:OrientationTransformationFn)
(domain POS:oCompose 2 POS:OrientationTransformationFn)
(domain POS:oCompose 3 POS:OrientationTransformationFn)

(=>
    (POS:oCompose ?T1 ?T2 ?TT)
    (forall (?P ?PM ?PF)
        (=>
            (and
                (equal ?PM
                    (?T1 ?P))
                (equal ?PF
                    (?T2 ?PM))
            (equal ?PF
                (?TT ?P)))))
```

Orientation coordinate systems can be arbitrarily arranged in a hierarchy.

```
(instance POS:oParentCS TransitiveRelation)
(domain POS:oParentCS 1 POS:OrientationCoordinateSystem)
(domain POS:oParentCS 2 POS:OrientationCoordinateSystem)
```

As with position coordinate systems, if two orientation coordinate systems are hierarchically related, it is possible to map all points from one to the other using transformations. This hierarchical link is represented as

```
(=>
    (POS:oParentCS ?C1 ?C2)
    (exists (?T1 ?T2)
        (and
            (instance ?T1 POS:OrientationTransformationFn)
            (instance ?T2 POS:OrientationTransformationFn)
            (POS:mapsOCS ?T1 ?C1 ?C2)
            (POS:mapsOCS ?T2 ?C2 ?C1))))
```

This means that, if two coordinate systems share a parent node in the hierarchy tree, there is a transformation between them. In fact, to build this transformation, compositions can be built in the following manner

```
(=>
    (and
        (POS:oParentCS ?C1 ?C2)
        (POS:oParentCS ?C2 ?C3))
    (exists (?T1 ?T2 ?T3)
        (and
          (instance ?T1 POS:OrientationTransformationFn)
            (POS:mapsOCS ?T1 ?C1 ?C2)
          (instance ?T2 POS:OrientationTransformationFn)
            (POS:mapsOCS ?T2 ?C2 ?C3)
            (POS:oCompose ?T1 ?T2 ?T3)
          (POS:mapsOCS ?T3 ?C1 ?C2))))
```

6.9 POS:OrientationPoint

An orientation point denotes the quantitative orientation of an object in relation to the reference object of an orientation coordinate system. Orientation points are always defined in a single coordinate system

```
(instance POS:inOCS SingleValuedRelation))
(instance POS:inOCS BinaryPredicate)
(domain POS:inOCS 1 POS:OrientationPoint)
(domain POS:inOCS 2 POS:OrientationCoordinateSystem)

(=>
    (instance ?P POS:OrientationPoint)
    (exists (?C)
        (and
            (instance ?C  POS:OrientationCoodinateSystem)
            (POS:inOCS ?P ?C))))
```

The relative orientation of an object in reference to another can be defined by means of a ternary predicate

```
(instance POS:relOrientation TernaryPredicate)
(domain POS:relOrientation 1 Object)
(domain POS:relOrientation 2 POS:OrientationMeasure)
(domain POS:relOrientation 3 Object)

(<=>
    (POS:relOrientation ?O ?P ?OR)
```

```
    (and
        (POS:orientation ?O ?P)
        (exists (?CS)
          (and
             (instance ?CS POS:OrientationCoordinateSystem)
             (POS:inOCS ?P ?CS)
             (POS:refOCS ?CS ?OR)))))
```

6.10 POS:OrientationRegion

An orientation region is a qualitative orientation in an orientation coordinate system.

```
(subclass POS:OrientationRegion POS:OrientationMeasurement)
```

Orientation points can be located inside orientation regions.

```
(instance POS:inOPR BinaryRelation)
(domain POS:inOPR 1 POS:OrientationMeasure)
(domain POS:inOPR 2 POS:OrientationRegion)
```

An orientation region is related to one and only one coordinate system.

```
(=>
    (instance ?R POS:OrientationRegion)
    (exists (?C)
        (and
           (instance ?C POS:OrientationCoordinateSystem)
           (forall (?P)
              (=>
                 (POS:inOPR ?P ?R)
                 (POS:inOCS ?P ?C))))))

(=>
    (and
        (instance ?R POS:OrientationRegion)
        (POS:inOPR ?P1 ?R)
        (POS:inOCS ?P1 ?C1)
        (POS:inOPR ?P2 ?R)
        (POS:inOCS ?P2 ?C2))
    (equal ?C1 ?C2)
```

POS defines operators that act as generator of orientation regions, based on a single reference object. Operators can be of many types. For instance, a single operator can get the orientation point of an object and generate a position region that corresponds to the north of that object.

```
(subclass POS:OrientationSpatialOperator Function)

(instance POS:oRGenerated SingleValuedRelation))
(instance POS:oRGenerated TernaryPredicate))
(domain POS:oRGenerated 1 POS:SpatialOperator)
(domain POS:oRGenerated 2 Object)
(domain POS:oRGenerated 3 POS:OrientationRegion)

(=>
    (instance ?R POS:OrientationRegion)
    (exists (?OR ?G ?C)
```

```
        (and
            (instance ?G POS:SpatialOperator)
            (POS:oRGenerated ?G ?OR ?R)
            (POS:refOCS ?C ?OR)
            (forall (?P)
                (=>
                  (POS:inOPR ?P ?R)
                  (POS:inOCS ?P ?C)))))))
```

Relative orientation can also be defined for orientation regions

```
(<=>
    (and
        (POS:relOrientation ?O ?R ?OR)
        (instance ?R POS:OrientationRegion))
    (exists (?G)
        (and
            (POS:orientation ?O ?R)
            (instance ?R POS:OrientationRegion)
            (POS:oRGenerated ?G ?OR ?R)))
```

6.11 POS:Pose

The pose of some object is a quantity composing an orientation and a position.

```
(subclass POS:PoseMeasure PhysicalQuantity)

(subrelation POS:pose measure)
(domain 1 POS:pose Object)
(domain 2 POS:pose POS:PoseMeasure)

(instance POS:posePosition SingleValuedRelation)
(instance POS:posePosition BinaryPredicate)
(domain POS:posePosition 1 POS:PoseMeasure)
(domain POS:posePosition 2 POS:PositionMeasure)

(instance POS:poseOrientation SingleValuedRelation)
(instance POS:poseOrientation BinaryPredicate)
(domain POS:poseOrientation 1 POS:PoseMeasure)
(domain POS:poseOrientation 2 POS:OrientationMeasure)
```

7. Specialization guidelines

CORA and related ontologies can be specialized to account for specific concepts, such as different kinds of robots, robot groups and robotic systems. This section defines guidelines on how to define such specializations.

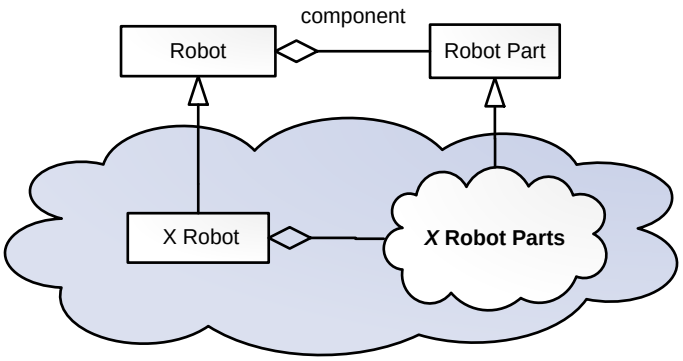
7.1 Robot structure pattern

The *robot structure pattern* allows one to define specific types of robots based on body structure. A *spine robot*, for example, is a robot that has an arm with two or more spherical joints (according do ISO/FDIS

8373). These partonomical relations are important when modeling the R&A domain since they capture the basic notion of compositionality necessary to project and build engineering devices.

In order to instantiate this pattern, one has to specialize the partonomic relation between *Robot* and *Robot Part* when specifying new types of robots, restricting the possible body parts that a particular kind of robot can have. Figure 18 a) shows how the pattern is instantiated.

(a)



(b)

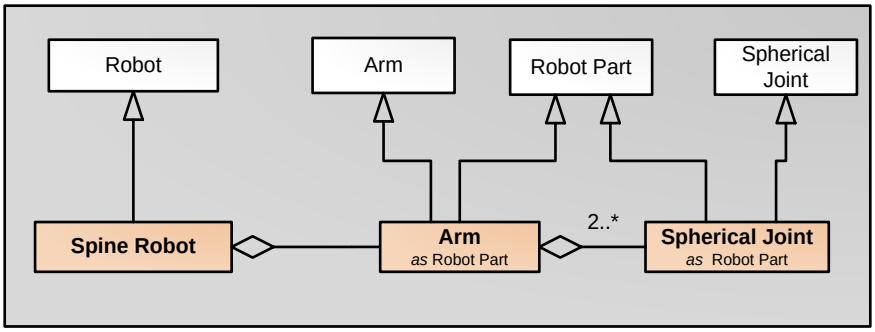


Figure 18 - Example of ontology pattern for specifying types of robots based on robot parts: (a) shows the general schema for extending the concept Robot with other kinds of robots based on robot parts; and (b) depicts pattern application example defining what a spine robot is.

In order to specify a given type of robot (e.g. X Robot) based on its necessary body parts, the Robot concept is extended by creating a new concept, and this concept is related to the concepts of the parts that compose its body (represented by the smaller cloud in Figure 18a). The Figure 18b shows how this is modeled in the case of spine robot. Spine robot has an arm as component that requires partonomic association with two actuators.

Annex A

(informative)

Ontology: General Aspects

In computer science, ontologies are formal tools that enable the description of objects, properties and relationships among such objects in a knowledge domain. In particular, two main definitions capture the essence, purpose and scope of an ontology. Studer et al. [B29] combines the definitions by Gruber [B10] and Borst [B1] and states that an ontology is "an explicit, formal specification of a shared conceptualization". On the other hand, Guarino [B11] stresses the formal aspects of a conceptualization and defines ontologies as "logical theories accounting for the intended meaning of a formal vocabulary". An ontology comprises at least a set of terms and their definitions as shared by a given community, formally specified in a machine-readable language, such as first-order logic. Ontologies are particularly important to provide machines with knowledge representation and reasoning capabilities to solve a task, as well as to allow high-level interoperability between systems.

The term ontology encompasses several ways of structuring its elements. From a mere list of terms and definitions to a formal theory specified, for instance, in first order logic; the structure of what has to be modeled changes dramatically at both extremes (see Uschold and Gruninger [B32]). Notwithstanding, the main elements of an ontology can be identified as [B9]:

a) Classes: stand for concepts at all granularities;

b) Relations: stand for associations between concepts;

c) Formal axioms: constrain and add consistency rules to the concept and relationship structures;

The range of activities concerning the ontology development process, the ontology life cycle, the methods and methodologies for building ontologies, and the tools and languages that support them is called *ontological engineering* (or *ontology engineering*) [B9].

Disparate classifications are available for systematizing different kinds of ontologies. This document adopts the classification of *levels of generality* by Guarino [B11], namely:

a) Top-level ontologies, which describe very general concepts, like space, time, matter, object, event, action, etc., which are independent of a particular problem or domain;

b) Domain ontologies, which describe concepts of a specific domain;

c) Task ontologies, which describe generic tasks or activities;

d) Application ontologies, which are strictly related to a specific application and used to describe concepts of a particular domain and task.

It is important to notice that there is a reusability-usability trade-off regarding the application of ontologies [B9], i.e., general ontologies are more reusable and less usable than specific ontologies. This trade-off implies that by going down, in the above classification, towards a greater specificity, there is an increase of usability and a decrease in reusability of the ontologies.

This document presents a *core ontology*. Not listed in the classification above, core ontologies can be viewed as mid-level ontologies, positioned between top-level ontologies and domain ontologies [B26]. Core ontologies reuse concepts defined by top-level ontologies and specify new concepts that can be used

in particular domains and tasks. Core ontologies specify concepts that are general in a large domain such as – in R&A domain – robot, device, and robotic system and their corresponding relationships. Below, some well-known ontologies are briefly presented.

Suggested Upper Merged Ontology (SUMO) [B23] [B24] is an upper level ontology that has been proposed as a starter document for *The Standard Upper Ontology Working Group* (SUO WG), an IEEE-sanctioned working group of collaborators from the fields of engineering, philosophy and information science. SUMO provides definitions for general-purpose terms and acts as a foundation for more specific domain ontologies. It was created by merging publicly available ontological contents into a single, comprehensive, and cohesive structure.

OpenCyc [B18] is the open source version of the Cyc technology, which corresponds to a large general knowledge base and commonsense reasoning engine. Cyc is an artificial intelligence project that attempts to assemble a comprehensive ontology and knowledge base of everyday common sense knowledge, with the goal of enabling AI applications to perform human-like reasoning.

Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) [B7] has been conceived by Nicola Guarino and his team at LOA (Laboratory for Applied Ontology) as part of a wider library of foundational ontologies developed in the WonderWeb Project (see http://wonderweb.semanticweb.org/). DOLCE is an ontology of particulars, that is, it does not describe types of universals (for more information about particulars and universals (see [B13]). In addition, the focus of the DOLCE ontology is to grasp the underlying categories of human cognitive tasks and the socio-cultural environment. This aspect is particularly important since it addresses the communication between the agents and/or their environment with specific emphasis on the social aspects of such communication (e.g. social plans, norms, institutions, and so on);

WordNet is a lexical database for English [B21] available on the web formed by sets of cognitive synonyms for nouns, verbs and adverbs. Each cognitive synonym set, called “synset,” expresses a different concept. Synsets are interlinked by conceptual-semantic and lexical relations. The full set can be seen as a network of words linked by conceptual relations to be used in reasoning based on computational human linguistics. This database is formed by 117,000 synsets and can be browsed and downloaded in the WordNet website [B34]. For example, the word “robot”, which is defined as “a mechanism that can move automatically”, is in the synset formed by “automaton”, “robot” and “golem”.

Other types of relevant ontologies are available in the web, e.g.: DBpedia ontology [B4]; Proton Ontology [B27]; and GFO Ontology [B8].

The previous presented ontologies were not specifically designed for the R&A community. The R&A domain should account, for example, for human-robot interaction (i.e. the capacity of robots to interface with humans for communication purposes), intelligent robots (i.e. robots capable of interacting with the environments and in general with external entities), collaborative robots (i.e. robots conceived to interact with human beings), and robot cooperation (i.e. robots that exchange information among them).

Annex B

(informative)

Ontology development

The development of CORA is supported by two well-known methodologies for building ontologies: METHONTOLOGY [B5] and OntoClean [B12]. METHONTOLOGY includes the identification of the ontology development process, a life cycle based on evolving prototypes, and particular techniques for carrying out each activity. The ontology development process refers to *which* activities are carried out when building ontologies. It includes three categories of activities that must be performed: project management activities, development-oriented activities and support activities.

METHONTOLOGY includes the following *project management activities*:

a) Planning, which identifies which tasks will be performed, how they will be arranged, how much time and what resources (such as other ontologies) are needed for their completion;
b) Control, which guarantees that planned tasks are completed in the manner that they were intended to be performed;
c) Quality Assurance, which assures that the quality of each and every resulting product (ontology, software and documentation) is satisfactory.

METHONTOLOGY also includes the following *development-oriented activities*:

a) Specification, which defines the purpose and scope of the ontology, its intended usage and target users. This activity also specifies some sources that could be used to acquire knowledge for the ontology development;
b) Conceptualization, which organizes and converts an informally perceived view of a domain into a semi-formal specification using a set of *intermediate representations* [B2] based on tabular and graph notations that can be understood by domain experts and ontology developers. The result of the conceptualization activity is the ontology conceptual model;
c) Formalization, which transforms the conceptual model into a formal or semi-computable model. This can be done, for example, by specifying a model using first order logic;
d) Implementation, which transforms the ontology previously formalized into a computable model, codified in an ontology representation language, such as OWL;
e) Maintenance, which updates and corrects the ontology.

Finally, METHONTOLOGY specifies the following *support activities*, which can be performed continuously along all the development process, and simultaneously to the *development-oriented activities*:

a) Knowledge acquisition, which directs the acquisition of domain knowledge from several sources, such as domain experts, domain literature (books and papers), other ontologies, thesaurus, domain glossaries, etc. Also, some techniques can be employed in this activity, such as: brainstorming, interviews, formal and informal analysis of texts, knowledge acquisition tools, etc.
b) Evaluation, which aims the technical judgment of the ontology, during and in between each activity. The ORA WG has adopted OntoClean [B12] as the main methodology of evaluation, which can be used along all the processes, checking for the quality of the taxonomies in the ontology. Furthermore, the ORA WG used the analysis of counter examples discussed by Sure et al. [B30] as a useful approach to evaluate the definitions of ontology concepts. It can help to determine when an expected instance of a class is not correctly classified due to problems in the definition of the class. Once the ontology is implemented in some codification language (such as OWL), several tools can be used to evaluate it. Among these tools, reasoners are important, since they support several types of evaluations, such as: consistency checking, subsumption checking,

equivalence checking and instantiation checking. In this activity, ORA WG considers other steps of evaluation in which a subgroup submits its partial results to other subgroups, as well as to the R&A community. Other useful ontology evaluation approaches, can be viewed in the work of Hartmann et al. [B14]. It is important to notice that an evaluation can raise several kinds of problems in the ontology and, therefore, it might force the return to any of previous activities where they can be solved.

c) Integration, whose goal is to consider the reuse of definitions already built into other ontologies. This is the case, for example, when definition given in an upper-level ontology (as DOLCE or SUMO) is adopted, or when a specific domain ontology extends a concept given in the core ontology.

d) Documentation, which ensures that the development process and the ontology itself are documented. This activity includes, in the broad sense, the maintenance and tracking of the sources of the terms and definitions of the ontology, as well as the clear definitions and examples embedded in the code of the implemented ontology.

e) Configuration Management, which records all versions of the documentation, software and ontology code to control the changes.

In METHONTOLOGY [B5], the *ontology life cycle* identifies the set of stages through which the ontology moves during its lifetime and describes what activities could be performed in each stage and how the stages are related (relation of precedence, return, etc.). In the evolving prototype life cycle, the ontology grows depending on the needs. Indeed, this model allows for modification, addition, and removal of definitions in the ontology at any time.

In the ontology life cycle proposed by METHONTOLOGY [B5], the project management activities are performed during the whole ontology development process. The ontology life cycle moves forward through the following states: specification, conceptualization, formalization, integration, implementation and maintenance. In any of these states, the process can move towards the first state (specification), restarting the cycle. The support activities (knowledge acquisition, evaluation of ontologies and documentation) are carried out during the whole life of the ontology. Figure B.1 presents a schematic representation of the ontology life cycle, according to METHONTOLOGY [B5], emphasizing which activities are performed and when they are performed within the whole process of ontology development.

B.1 Development activities in CORA

The UpOM group has begun its work by specifying the sources from which the domain knowledge could be acquired. The main sources are existing standards in the domain, textbooks, peer-reviewed papers, domain experts and other ontologies (including upper ontologies as DOLCE and SUMO). Moreover, UpOM group also decided to adopt a *middle-out approach* proposed by Uschold et al. [B33] for identifying concepts, namely, starting from the most relevant concepts and branching out both to the most abstract and to the most concrete ones.

Among the existing ontologies, the upper-level ontologies are very important, since they provide higher-level concepts that support the definition of concepts in CORA. On the other hand, the ontologies for R&A collected from literature encompass only a subset of terms of R&A, with specialized meaning for specific applications. Moreover, they have been developed by a small group of people, not representing the common shared knowledge of R&A community.

The UpOM group started working with the ISO/TC184/SC2 committee (Robotics and robotic devices) Working Group 1 (Vocabulary) [B16]. This working group has published the ISO/FDIS 8373 standard document which defines, in natural language, generic terms that are common in the R&A. This document was considered as a good starting point for developing the glossary of terms that would form the ontology, since the development moved toward an agreement of a broad community. The section “General Terms” was the most promising to the development of CORA, since it contains terms that describe the most general notions across the sub-domains of industrial robots, service robots, and autonomous robots. A subset of

terms has been chosen to be included in the CORA, such as robot, autonomy, robot system, robotic device, and so on.

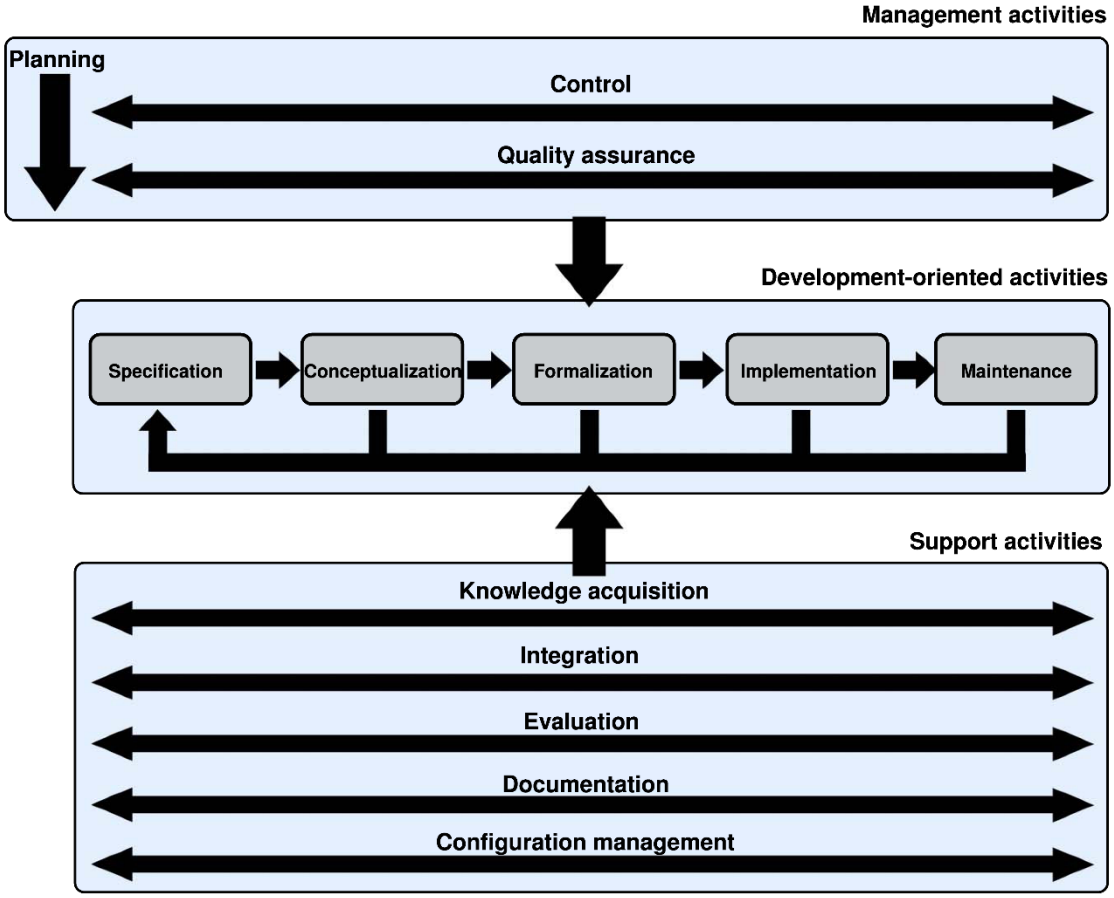


Figure B.1—Representation of the ontology life cycle, according to METHONTOLOGY [B5]. Figure adapted from Corcho et al. [B2]

From the terms identified in the last step, the conceptualization activity has been performed. In this step, the domain knowledge was organized using the intermediate representations discussed by Corcho et al. [B2], as suggested by METHONTOLOGY [B5], for representing the main concepts, taxonomies and relationships among the concepts. Furthermore, UML diagrams were used for representing this knowledge in a format that facilitates the overview of the main concepts, relations and taxonomies proposed by Gómez-Pérez et al. [B9]. The ISO document was remarkably valuable in this project for identifying some of the core terms of R&A and for becoming explicit the structure of the domain knowledge.

The results acquired in the previous step motivated further analysis of the literature in order to identify alternative conceptualizations of the core notions of the ontology. In this analysis, several alternative definitions for *robot* have been found, which emphasize different aspects, under different perspectives. Moreover, according to some works in the literature, some core notions in the R&A domain, such as *robot* [B3] and *automation* [B25] are surprisingly hard to define. Joseph Engelberger, a pioneer in R&A domain, expressed these difficulties in his comment [B20]: "I can't define a robot, but I know one when I see one". All these evidences indicate that accommodating all the alternative conceptualizations the community has about robots under a single broad definition is a challenging task.

From the alternative definitions of robot found in the literature, some important aspects of this concept are raised, such as: robot as an agent [B28], robot as a programmable machine with actuators and sensors [B6]

[B22][B19], and robot as having capability of performing a variety of tasks [B6] [B31] [B19]. From these aspects, the main terms of CORA were defined.

In the next step, ORA WG decided to integrate CORA with an upper-level ontology. Upper-level ontologies with clear ontological commitments can help the modeler to perform well-founded choices that are consistent with each other. Thus, the integration process of the distinct ontologies, which are being developed in the WG, can be facilitated, if all of them commit themselves to the same top-level ontology. Moreover, according to Jansen and Schulz [B17] the commitment to an upper-level ontology avoids wrong conclusions and mistakes typical of ad-hoc approaches for modeling ontologies; and forces the ontology developers to think about ambiguous terms, providing them a more precise definition. CORA ontology is aligned with SUMO, which is the most prominent proposal under consideration by the IEEE Standard Upper Ontology [B24]. SUMO provides a good description of the top most categories and includes the main notions and distinctions introduced in CORA ontology, such as agent, device and agent group; and allows a broader interpretation of these notions. Furthermore, the notion of autonomy in the domain of unmanned systems proposed *Autonomy Levels For Unmanned Systems* (ALFUS) framework [B15] is integrated into CORA. According to ALFUS, autonomy is determined by the contextual autonomous capability model, which specifies that autonomy depends on three axes, namely: mission complexity, environmental complexity and human independence.

Annex C

(informative)

Bibliography

Bibliographical references are resources that provide additional or helpful material but do not need to be understood or used to implement this standard. Reference to these resources is made for informational use only.

[B1] Borst, W. N., "Construction of Engineering Ontologies for Knowledge Sharing and Reuse." *Ph.D. diss.*, Universiteit Twente, 1997.

[B2] Corcho, O., Fernandez-López, M., Gómez-Pérez, A., and López-Cima, A. "Law and the Semantic Web: Legal Ontologies, Methodologies, Legal Information Retrieval, and Applications", *Lecture Notes in Computer Science*, Vol. 3369, Springer–Verlag, pp. 142–157.

[B3] Craig, J. J., *Introduction to robotics: Mechanics and Control* (Third Edition), Pearson Prentice Hall, 2005.

[B4] *DBpedia ontology*. http://wiki.dbpedia.org/Ontology/. Accessed on 26 June 2013.

[B5] Fernández, M., Gómez-Pérez, A., and Juristo, N., "METHONTOLOGY: from ontological art towards ontological engineering," *AAAI Spring Symposium*, California, USA, pp. 33–40, Mar. 1997.

[B6] Fu, K. S., Gonzalez, R. C., and Lee, C. S. G., *Robotics: control, sensing, vision and intelligence*, McGraw-Hill, 1987.

[B7] Gangemi, A., Guarino, N., Masolo, C., Oltramari, A., and Schneider, L., "Sweetening ontologies with DOLCE.", in: A. Gómez-Pérez, V. R. Benjamins (Eds.), *Proceedings of the 13th European Conference on Knowledge Engineering and Knowledge Management (EKAW2002)*, Volume 2473 of Lecture Notes in Computer Science, Springer, 2002, pp. 166–181.

[B8] *GFO Ontology*. http://www.onto-med.de/ontologies/. Accessed on 26 June 2013.

[B9] Gómez-Pérez.,A, and Benjamins, B. R., "Overview of Knowledge Sharing and Reuse Components: Ontologies and Problem-Solving Methods," *Workshop on ontologies and problem-solving methods*, Stockholm, Sweden, pp. 1–15, Aug. 1999.

[B10] Gruber, T. R., "Toward principles for the design of ontologies used for knowledge sharing," *Journal of Human-Computer Studies*, vol. 43, pp. 907—928, Nov. 1995.

[B11] Guarino, N., "Formal Ontology in Information Systems," *International Conference on Formal Ontology in Information Systems*, Trento, Italy, pp. 3–15, June 1998.

[B12] Guarino, N., and Welty, C. A., "An overview of OntoClean," Handbook on Ontologies, Berlin Heidelberg: Springer, 2009, pp. 201–220

[B13] Guizzardi, G, Herre, H., and Wagner, G. "On the general ontological foundations of conceptual modeling", *in: 1st International Conference on Ontologies*, volume 2519, pp. 1100–1117.

[B14] Hartmann, J., Spyns, P., Giboin, A., Maynard, D., Cuel, R., Surez-Figueroa, M. C., and Sure, Y., "D.1.2.3. methods for ontology evaluation," *Deliverable IST-2004-507482 KWEB D.1.2.3.*, Technical Report, EU-IST Network of Excellence (NoE) Knowledge Web Consortium, 2005.

[B15] Huang, H.-M, Albus, J., and Messina, E.,"Toward a generic model for autonomy levels for unmanned systems (ALFUS)," *Performance Metrics for Intelligent Systems (PerMIS) Workshop*, Gaithersburg, MD, USA, 2003.

[B16] ISO/FDIS 8373, "Robots and robotic devices" – Vocabulary.

[B17] Jansen, L. and Schulz, S., "The ten commandments of ontological engineering," *Workshop Ontologies in Biomedicine and Life Science*, Berlin, Germany, pp. 41—46, 2011. [{JANSEN:2011}]

[B18] Lenat, D.B., "CYC: a large-scale investment in knowledge infrastructure," *Communications of the ACM*, vol. 38, no. 11, pp.33—38, Nov. 1995.

[B19] Lund, H. H., "Modular robotics for playful physiotherapy," *IEEE International Conference on Rehabilitation Robotics*, Kyoto, Japan, pp. 571—575, 2009.

[B20] Macura, K. J., and Stoianovici, D., "Advancements in magnetic resonance-guided robotic interventions in the prostate", *Top. Magn. Reson. Imaging 19* (2008) 297–304.

[B21] Miller, G. A., "WordNet: a lexical database for English," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, Nov. 1995.

[B22] Munich, M. E., Ostrowski, J., and Pirjanian, P., "Ersp: a software platform and architecture for the service robotics industry," *International Conference on Robots and Systems*, Alberta, Canada, pp. 460—467, 2005.

[B23] Niles, I., and Pease, A., "Towards a standard upper ontology," *International Conference on Formal Ontology in Information Systems*, Ogunquit, ME, USA, pp. 2—9, Oct. 2001.

[B24] Niles, I., and Pease, A., "Origins of the IEEE Standard Upper Ontology," *Working Notes of the IJCAI-2001 Workshop on the IEEE Standard Upper Ontology*, pp. 4-10, 2001.

[B25] Nof, S. Y., *Springer Handbook of Automation*, Springer-Verlag, pp. 13–52.

[B26] Obrst, L., *Theory and Applications of Ontology: Computer Applications*, Dordrech:Springer, 2010, pp.27–66.

[B27] *Proton Ontology*. http://proton.semanticweb.org/. Accessed on 26 June 2013.

[B28] Steels, L. "When are robots intelligent autonomous agents", *Robotics and Autonomous Systems 15* (1995) 3–9.

[B29] Studer, R., Benjamins, V. R., and Fensel, D., "Knowledge Engineering: Principles and Methods," *Data and Knowledge Engineering*, vol. 25, no. 1-2, pp. 161–197, Mar. 1998.

[B30] Sure, Y., Gómez-Pérez, A., Daelemans, W., Reinberger, M.-L., Guarino, N., and Noy, N. F., "Why evaluate ontology technologies? because it works!," *IEEE Intelligent Systems 19* (2004) 74–81.

[B31] Thrun, S.,"Toward a framework for human-robot interaction," *Human-Computer Interaction*, Volume 19, no. 1, 9—24, June 2004.

[B32] Uschold, M., and Gruninger, M., "Ontologies and semantics for seamless connectivity," SIGMOD Record (2004) 58–64.

[B33] Uschold, M., and King, M., "Towards a methodology for building ontologies," *Workshop on Basic Ontological Issues in Knowledge Sharing held in conjunction with IJCAI*, http://www.aiai.ed.ac.uk/project/pub/documents/1995/95-ont-ijcai95-ont-method.ps, Accessed on 27 June 2013. .

[B34] *Wordnet*. http://wordnet.princeton.edu/. Accessed on 26 June 2013.