

An Upper Ontology based on ISO 15926

Rafael Batres ^{a*}, Matthew West^b, David Leal^c, David Price^d,
Katsube Masaki^a, Yukiyasu Shimada^e, Tetsuo Fuchino^f, Yuji Naka^g

^aToyohashi University of Technology

Hibarigaoka 1-1, Tempakucho, Toyohashi 441-8580, Japan

^bShell International Petroleum Company Limited
London SE1 7NA, UK

^cCAESAR Systems Limited

29 Somertrees Avenue, Lee London SE12 0BS, UK

^dEurostep Ltd.

Cwttir Lane, St. Asaph, Denbighshire LL17 OLQ, UK

^eNational Institute of Industrial Safety

1-4-6, Umezono, Kiyose, Tokyo 204-0024, Japan

^fChemical Engineering Department, Tokyo Institute of Technology
2-12-1 S1-10 O-okayama

Meguro-ku, Tokyo 152-8552, Japan

^gTokyo Institute of Technology
4259 R1-19 Nagatsuta Midori-ku Yokohama 226-8503, Japan

Abstract

Ontologies reflect our view of what exists, and developing ontologies for a given domain requires a common context. This context can be characterized explicitly by means of an upper ontology. Upper ontologies define top-level concepts such as physical objects, activities, mereological and topological relations from which more specific classes and relations can be defined. As an effort to support the development of domain ontologies, we are developing an OWL ontology based on the ISO 15926 standard. This paper introduces the key aspects of the ontology, describes some of its main classes and properties and discusses its benefits and applications in the process engineering domain.

Keywords: ontologies, ISO 15926, temporal parts, four dimensionalism, causality, OWL

* Author to whom correspondence should be addressed. The corresponding author's address is:
Toyohashi University of Technology, Faculty of Engineering, Hibarigaoka 1-1, Tempaku-cho, Toyohashi 441-8580, Japan.
E-mail: rpb@pse.tut.ac.jp

1. Introduction

The number of computer applications in the chemical process industries has grown quickly in the past 20 years. Traditionally, these applications have operated independently functioning as separate islands of information that limit the enterprise's ability to share and exchange information inside and outside the enterprise. Integration between these applications arguably permits a lower cost of ownership, better accessibility to end-users and partners, greater efficiency, effectiveness and competitiveness in organisations (Wainwright, 2004). Typically, existing integration solutions are defined in a top-down approach by using shared information models in formats such as XML. Developing common XML interfaces is complicated by the multiplicity of views of information and by the fact that information models developed from scratch tend to be prescriptive (what will be) because such models are developed so as to meet integration requirements imposed by either existing software or by functions to be carried out by software components. The more applications that are included in the integration architecture the higher is the probability of having islands of information.

To support integration that focuses on the longer-term solution effectiveness, an unambiguous description of the worldview is needed that both computers and humans can understand. The specification of such a worldview is referred to as an ontology.

Ontologies describe a shared and common understanding of a domain that can be communicated between people and heterogeneous software tools. We construct an ontology by defining classes of things, their taxonomy, the possible relations between things and axioms for those relations. A class represents a category of things that share a set of properties. A relation is a function that maps its arguments to a Boolean value of true or false. Examples of relations are `less_than`, `connected_to`, and `part_of`. Class taxonomies are defined with the use of the subclass relation. A class is a subclass of another class if every member of the subclass is also a member of the superclass.

Loosely speaking, collections of classes of objects such as entity-relationship models from the database community or object-oriented class definitions can be considered as ontologies (Ding, 2001). These "loose

ontologies” are usually developed in an ad-hoc manner and tend to focus on data rather than on what exists in reality. On the other hand, ontologies are a theory of reality: “describe the kinds and structures of objects, properties, events, processes and relations in every area of reality” (Smith 2003).

Ontologies can be developed using top-down or bottom-up approaches. The bottom-up approach starts with the most specific concepts in a domain of application. A bottom-up approach results in ontologies that are difficult to modify and integrate with ontologies developed for other domains or applications (Uschold and Gruninger, 1996). Top-down approaches start with high-level concepts that are assumed to be common to many application areas. The top-down approach facilitates integration of applications with ontologies that are easier to maintain. Unfortunately, engineers using the top-down approach are susceptible to imposing arbitrary high-level categories which are prescriptive (what will be), not meeting the user’s requirements. These problems can be avoided with an upper ontology.

Upper ontologies define top-level classes such as physical objects, activities, mereological and topological relations from which more specific classes and relations can be defined. Examples of upper ontologies are SUMO (Niles and Pease, 2001), Sowa upper ontology (Sowa, 2000), Dolce (Gangemi et al. 2002), CliP (Bayer, 2003), and ISO 15926-2 (ISO 15926-2, 2003). Engineers using an upper ontology to develop a more specific domain ontology can start by identifying key concepts by means of activity modeling (Marca and McGowanm, 2005), use cases (Missikoff, 2005) and competency questions (Uschold and Gruninger, 1996). This concepts are then defined based on the more general concepts provided by the upper ontology. This avoids reinventing the wheel while having better integration and maintenance.

As an effort to support the development of process engineering ontologies, we are developing an upper ontology in the OWL language (see below) based on the ISO 15926 standard. Specifically, ISO 15926 Part 2 (standardized as ISO 15926-2:2003) specifies an ontology for long-term data integration, access and exchange (ISO-TC184, 2003). It was developed in ISO TC184/SC4-Industrial Data¹ by the EPISTLE

¹ <http://www.tc184-sc4.org/>

consortium² (1993-2003) and designed to support the evolution of data through time. The upper ontology contains 200 concepts including a meta-model for extending the ontology through what is known as a Reference Data Library (about 20,000 concepts from the engineering domain).

We have translated the original EXPRESS code (ISO 10303-11, 1994) of ISO 15926-2 to the OWL language that can be used directly in a number of inference software packages (W3C, 2004). Axiomatic definitions are currently being added as a way to implement some semantics of the standard that are not represented in the EXPRESS schema.

This paper introduces the key aspects of the ontology, describes some of its main classes and properties and discusses its benefits and applications in the process engineering domain.

2. OWL³

OWL is an ontology language for the Web that provides modeling constructs to represent knowledge with a formal semantics (Lacy, 2005). OWL was developed by the World Wide Web Consortium (W3C) Web Ontology Working Group (Bechhofer, 2004) and is being used to encode knowledge and enable interoperability in distributed computer systems (Finin and Ding, 2006). OWL inherits features of frames (Minsky, 1975) such as basic constructs for defining classes and instances. The `subClassOf` relation is used to describe specializations of a more generic class. A class can be defined in terms of the properties that characterize it. For example, if we assert that every centrifugal pump is a device that contains an impeller the definition of centrifugal pump can be represented in OWL as follows:

```
(Class centrifugal_pump
  (subClassOf pump)
  (subClassOf
    (Restriction composition_of_individual (someValuesFrom impeller))))
```

which is equivalent to the following XML serialization

² <http://www.epistle.ws/>

³ OWL is an acronym for Web Ontology Language

```
<owl:Class rdf:ID="centrifugal_pump">
  <rdfs:subClassOf rdf:resource="#pump"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="iso15926&composition_of_individual"/>
      <owl:someValuesFrom rdf:resource="#impeller"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

OWL provides constructs for defining relations in terms of their domains and ranges. The domain definition specifies the class to which the property belongs. Range definitions specify either OWL classes or externally-defined data types such as strings or integers. OWL uses the term Property to refer to relations.

Cardinality restrictions can be used to specify the exact number of values which should be on a specific relation of a given class. For example, a centrifugal pump can be defined as a pump that has at least one impeller⁴.

A relation can be declared as transitive, symmetric, functional or inverse of another property. If a relation R is transitive, and R relates A to B , and B is related to C via R then A is related to C via R . For example, if the plate finned tube $I23$ is part of intercooler x and intercooler x is part of multi-stage compressor y then $I23$ is also part of y . A relation R is symmetric if when A is related to B then B is related to A in the same way.

FunctionalProperty is a special type of relation such that for each thing in its domain, there is a single thing in its range. If some FunctionalProperty relates A to B then its inverse relation will link B to A . For example, if the relation *contains* is defined as FunctionalProperty then (contains tank-1 batch-1) is equivalent to (contained_in batch-1 tank-1) when *contains* is declared as an inverse relation of *contained_in*.

⁴ To implement this centrifugal_pump definition it becomes necessary to use qualified cardinality restrictions using the owl:valuesFrom which is a recently proposed addition to the OWL language:

```
(Class centrifugal_pump
  (subClassOf pump)
  (subClassOf
    (Restriction composition_of_individual (cardinality 1) (valuesFrom impeller)))
```

OWL provides constructs to define individuals (members of a class) such as those for describing which objects belong to which classes, the specific property values and whether two objects are the same or distinct. The prefixes owl, rdf, and rdfs are used to denote the namespaces where the OWL, RDF, and RDFS modeling constructs are respectively defined. Similar prefixes are also used to avoid name clashes, allowing multiple uses of a term in different contexts. For example, mil:tank and equip:tank can be used in an ontology to refer to a military tank and an equipment tank respectively.

The ontology is encoded in OWL for the following reasons:

- * knowledge represented in OWL can be processed by a number of inference software packages
- * some concepts within ISO 15926-2:2003 correspond to specific concepts within OWL (e.g. class, relation)
- * support of the creation of reusable libraries
- * a variety of publicly available tools for editing and syntax checking

3. Four dimensionalism

ISO 15926-2:2003 is founded on an explicit metaphysical view of the world known as *four dimensionalism*. In four dimensionalism, objects are extended in time as well as space, rather than being wholly present at each point in time, and passing through time. In addition, ISO 15926 has an extensional basis for identity of individuals. Thus if what appears to be two objects have all of the same parts (in both space and time) then they are the same object. For example, if a steel bar is made into a pipe, then there is a temporal part (state) of the steel bar that is coincident with the pipe. Because they are coincident, they are the same object. In other words, the pipe is a state of the steel bar.

Information systems have to support the evolution of information over time. For example, let us assume that a pump was designed and identified as P-101. Some time later, a manufacturer delivers a pump with serial number 1234 that meets the design specifications of P-101. Pump 1234 is installed and after a period of

operation the pump fails. Therefore, maintenance decides to replace it with pump 9876. This situation can be easily modeled using the concept of temporal parts as shown in Figure 1.

Figure 1. An object (possible individual) and its temporal part (state)

ISO 15926-2:2003 includes the class *functional_physical_object* to define things such as pump P-101 which have functional, rather than material continuity as their basis for identity (Figure 2). In other words, members of *functional_physical_object* are replaceable parts (West, 2003) of some artefact. P101 remains the same pump with the same function in the plant, even though the equipment item installed there (Pump 1234) is replaced by Pump 9876 (it would be very inconvenient to have to update drawings every time a piece of equipment was replaced). The 4 dimensional analysis of this shows that P101 consists of the temporal parts of the equipment items whilst they were installed as P101. So if S1 is the state of Pump 1234 whilst it was installed as P101 and S2 is the state of P9876 when it was installed as P101, then P101 also has S1 and S2 as parts.

Figure 2. A pump and its temporal parts 1234 and 9876

4. Top level concepts

thing is the root concept in the ontology which in turn has *abstract_object* and *possible_individual* as subtypes. A *thing* is anything that is or may be thought about or perceived, including material and non-material objects, ideas, and activities. Every *thing* is either a *possible_individual*, or an *abstract_object*.

Members of *possible_individual* are entities that exist in space and time, including physical objects like a compressor or imaginary individuals like Sherlock Holmes (Figure 3). A *possible_individual* has a life cycle bounded by *beginning* and *ending events* (see Section 6).

Figure 3. A *possible_individual* is an entity that exist in space and time

Individuals that belong to *abstract_object* can be said to exist in the same sense as mathematical entities such as numbers or sets but they cannot exist at a particular place and time. *possible_individual* is has a number of overlapping subtypes: *arranged_individual*, *actual_individual*, *whole_life_individual*, *physical_object*, *activity*, *period_in_time* and *event* (Figure 4).

Figure 4. Subclasses of *possible_individual*

Instances of *arranged_individual* have parts each of which plays a distinct *role* with respect to the whole. For example, centrifugal pump with serial number 1423 is an instance of *arranged_individual*. A *role* indicates what some thing has to do with an *activity*⁵. The impeller and volute of the pump have distinct roles. The role of the impeller is to convert the energy of the motor into kinetic energy that is imparted to the fluid that passes through the suction opening of the pump. The role of the volute is to convert that kinetic energy into potential energy.

An *actual_individual* is a *possible_individual* that exists in the present, past, or future of our universe, as opposed to some imagined universe. The computer used to edit this paper is an *actual_individual*.

A *whole_life_individual* is a *possible_individual* that is a member of a subclass of *possible_individual* and is not a temporal part of any other *possible_individual* that is also a member of the same subclass of *possible_individual*. For example, if pump is a subclass of *possible_individual* and the pump with serial number 1423 is an instance of pump then pump 1423 is a *whole_life_individual* because pump 1423 is not a temporal part of any other pump. *Physical_object* is described in Section 5. The classes of *activity*, *period_in_time* and *event* are explained in Section 6.

5. Physical objects

A *physical_object* is a *possible_individual* that is a distribution of matter, energy, or both. Examples of *physical_object* are a table, a pump, a piece of metal, a laser beam.

Because subtyping in ontologies supports multiple inheritance, instances of *physical_object* can also be instances of *arranged_individual* as with the centrifugal pump in the previous section. *Physical_object* is divided into *functional_physical_object*, *materialized_physical_object*, *spatial_location* and *stream* (Figure 5).

Figure 5. Subclasses of *physical_object*

As explained in Section 3, a *functional_physical_object* is a *physical_object* that is a replaceable part. In contrast, a member of *materialized_physical_object* has matter and/or energy continuity as its basis for identity. Matter or energy continuity requires some matter or energy to be common to adjacent temporal parts of the *materialized_physical_object*. Replacement of some components from time to time does not create a new identity. A *spatial_location* is a *physical_object* that has continuity of relative position such as a construction area, a country, a storage area or a tridimensional plane. A *stream* is a *physical_object* that is material or energy moving along a path, where the path is the basis of identity and may be constrained. The stream consists of the temporal parts of those things that are in the stream whilst they are in it. The material moving along the pipe that connected the feed-tank and the furnace of a hydrodesulfurization unit is an instance of stream.

5.1 Phase

In the ontology, solid, liquid, and gas represent classes subsumed by *material_in_single_phase* which is a subclass of *arranged_individual*. A *material_in_single_phase* is thus an *arranged_individual* that is an

⁵ A role can also indicate what something has to do with two abstract objects (relationship and a multidimensional object). In this ontology, *relationship* corresponds to `rdf:Property`. A *multidimensional_object* is an *abstract_object* that is an ordered list of *thing*. For example, the list [A,B,C] is a *multidimensional_object* which is different from [B,C,A].

amount of matter where all the matter is in a single phase. Excluded would be amounts of matter where parts of the matter were in different phases, e.g. a glass of water with ice in it.

When we apply heat to melt an ice-cube, some ontologies represent the liquid water as the same individual that was solid before. Since, solids have shape but liquids lack this property, the resulting liquid water can be represented as a new individual.

6. Activities

An activity is a *possible_individual* that brings about change by causing an *event* (Figure 6). Activities can have temporal boundings linking *events* as well as *points_in_time* because *activity* is a subclass of *possible_individual* which has a life cycle bounded by *beginning* and *ending events*. An *event* is a *possible_individual* that has zero extent in time, which means that it occurs at an instant in time. A *point_in_time* is an *event* that is the whole space extension with zero extent in time. The class activity can have as members, possible individuals such as physicochemical processes, plant operations, and abnormal situations. A *period_in_time* is a *possible_individual* that is the whole space extension for part of time.

Figure 6. An activity and its relationships (*beginning*, *ending*, *cause_of_event*, *participation*)

Causality is described by means of the *cause_of_event* relation. An *activity* consists of the temporal parts of those members of *possible_individual* that participate in the activity. For example, the mixing activity shares the temporal parts of the tank and agitator.

The *participation* relation is used to express that a *possible_individual* is involved in an activity. This is illustrated with the following OWL code in which *pump-during-operating-period* (temporal part of pump-1) participates in the *loading-fluid-in-the-tank* activity. The activity starts when the temporal part of pump-1 is

on and finishes when it is off. The temporal boundings of *loading-fluid-in-the-tank* are the same as the temporal boundings of *pump-during-operating-period*.

```
<activity rdf:ID="loading-fluid-in-the-tank">
  <beginning>
    <event rdf:ID="pump-on"/>
  </beginning>
  <ending>
    <event rdf:ID="pump-off">
      <cause_of_event rdf:resource="#loading-fluid-in-the-tank"/>
    </event>
  </ending>
  <participation>
    <physical_object rdf:ID="pump-during-operating-period"/>
  </participation>
</activity>

<physical_object rdf:ID="pump-during-operating-period">
  <beginning rdf:resource="#pump-on"/>
  <ending rdf:resource="#pump-off"/>
  <temporal_whole_part rdf:resource="#pump-1"/>
</physical_object>

<physical_object rdf:ID="pump"/>
```

7. Mereology and topology

Mereology expresses the part-whole relations of an object (Simons, 2000). Part-whole relations are reflexive, antisymmetric, and transitive as described by the following axioms:

1. Everything is part of itself (x is a part of x)
2. Two distinct things cannot be part of each other (if x is a part of y, and y is a part of x then x=y)
3. Any part of any part of a thing is itself part of that thing (If x is a part of y and y is a part of z then x is a part of z)

Mereological descriptions are possible by means of *composition_of_individual* and its subproperties. *composition_of_individual*. Subproperties of *composition_of_individual* include *containment_of_individual* (used to represent things that are inside others) and *relative_location* (used to locate objects in a particular place).

Figure 7. Connected pipes and flanges

Topology refers to the connectivity between objects (Clarke, 1981). Topological descriptions are based on the use of the property *connection_of_individual* which is defined as symmetric and transitive. The symmetric character of the relation, allows us to infer that flange B is connected to flange A provided that A is connected to B (Figure 7). The definition of *connection_of_individual* and the topological description of the pipes are shown below:

```

<owl:ObjectProperty rdf:about="connection_of_individual">
  <rdfs:domain rdf:resource="#possible_individual"/>
  <rdfs:range rdf:resource="#possible_individual"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#SymmetricProperty"/>
</owl:ObjectProperty>

<pipe rdf:ID="A">
  <connection_of_individual rdf:resource="#F"/>
</pipe>

<pipe rdf:ID="B">
<connection_of_individual rdf:resource="#G"/>
</pipe>

<flange rdf:ID="F">
  <connection_of_individual rdf:resource="#G"/>
</flange>

<flange rdf:ID="G">
  ...
</flange>

```

Using the axiomatics for transitiveness, an inference engine can conclude that pipes A and B are connected because their flanges F and G are connected. For a treatment of four-dimensional mereotopology and its axiomatics, see Stell and Mathew (2004).

8. Physical quantities

In tandem with the ideas presented by Gruber and Olsen (1994) the upper ontology supports the idea that physical objects and activities should not be allowed to define physical quantities (3 kg, 5 m, etc.) as attributes because a physical quantity is not an inherent property of an object. For example, the setpoint of a temperature controller TIC_01 (a physical object) at 800 Kelvin should not be represented as an attribute (a relationship in ontology terms) because there is nothing intrinsic about 800 Kelvin that says it is the setpoint

of TIC_01. “800 Kelvin” is just the extent of the temperature quantity to which the temperature set point refers.

The mapping between a controller and a temperature quantity can be defined as an instance of *class_of_indirect_property*. The *class_of_indirect_property* is implemented as a subclass of owl:FunctionalProperty, whose domain is given by members of *class_of_individual* and whose range is given by members of *property_space*. *temperature_setpoint* is thus a relation whose range refers to instances of *temperature_quantity*. *temperature_quantity* is an instance of *property_space*, which makes it both a class and an instance. Furthermore, *property_space* is a subclass of *class_of_property*, which means that *temperature_quantity* is also an instance of *class_of_property* (Figure 8). The OWL code in Figure 8 also states that controller TIC_01 has temporal part whose setpoint is 800 Kelvin. The mapping between the value of 800 and the property is done by means of a *property_quantification*. A *property_quantification* is a functional mapping whose members map a *property* to an *arithmetic_number*.

In regards to units of measure, the approach in ISO 15926-2:2003 is to classify the *property_quantification*, in other words a classification relation is used to map an instance of *property_quantification* to an instance of *scale*. The approach used here defines *scale* as an OWL:property.

```
<owl:Class rdf:ID="temperature_quantity">
  <rdf:type rdf:resource="&lis;property_space"/>
  <rdfs:label>temperature_quantity</rdfs:label>
</owl:Class>

<owl:FunctionalProperty rdf:ID="temperature_setpoint">
  <rdf:type rdf:resource="&lis;class_of_indirect_property"/>
  <rdfs:domain rdf:resource="&ctrl;controller"/>
  <rdfs:range rdf:resource="#temperature_quantity"/>
</owl:FunctionalProperty>

<lis:physical_object rdf:ID="TIC_01">
  <rdfs:comment>Temperature Controller TIC-01</rdfs:comment>
</lis:physical_object>

<owl:ObjectProperty rdf:ID="kelvin">
  <rdf:type rdf:resource="#scale"/>
  <rdfs:domain rdf:resource="#temperature_quantity"/>
  <rdfs:range rdf:resource="#real"/>
</owl:ObjectProperty>

<lis:physical_object rdf:ID="temporal_part_of_TIC_01_at_800K">
  <lis:temporal_whole_part.whole rdf:resource="#TIC_01"/>
```

```

<temperature_setpoint>
  <rdf:Description>
    <rdf:type>
      <owl:Class rdf:about="#temperature_quantity"/>
    </rdf:type>
    <kelvin>
      <rdf:Description>
        <real>
          <content>
            <xsd:float rdf:value="800.0"/>6
          </content>
        </real>
      </rdf:Description>
    </kelvin>
  </rdf:Description>
</temperature_setpoint>
</lis:physical_object>

```

Figure 8. OWL Code illustrating the definition of physical quantities.

In the example of Figure 8, `temperature_set_point` is an instance of *class_of_indirect_property* defined so as to express that controllers can have a `temperature_setpoint` which accepts values of temperature but not pressure or any other *property_space*. Note that `kelvin` is defined in such a way that it would be possible to detect inconsistencies in the units of measure of temperature properties. The actual use of the *scale kelvin* contains the value of 800 K, meaning that controller TIC_01 had that value during a certain period of time.

An alternative representation (see Figure 9) is also possible that follows closer the original ISO 15926 specification with which one would represent the “800 K” as an instance of property which can be used when the upper ontology can be extended with additional classes but not with additional properties. Despite its simplicity, this second approach has no way to detect inconsistencies in the units of measure as indicated with `wrong_prop_652` which associates “800 bar” to the `temperature_setpoint`.

⁶ In this example, the element `<content>` is purely structural and relates the object `<real>` to the XML object `<xsd:float>`. More formally we could say:

```

<real>
  <decimal>800.0</decimal>
</real>

```

where `decimal` is the function from Real to Literal that is decimal encoding.

```
<lis:class_of_indirect_property rdf:ID="temperature_setpoint">
  <has_property_space rdf:resource="#temperature_quantity"/>
</lis:class_of_indirect_property>

<lis:scale rdf:ID="kelvin"/>

<lis:property_space rdf:ID="temperature_quantity">
</lis:property_space>

<lis:physical_object rdf:ID="temporal_part_of_TIC_01_at_800K">
  <lis:temporal_whole_part.whole rdf:resource="#TIC_01"/>
</lis:physical_object>

<indirect_property rdf:ID="ip_1023">
  <possessor rdf:resource="#temporal_part_of_TIC_01_at_800K"/>
  <has_property rdf:resource="#prop_651"/>
</indirect_property>

<indirect_property rdf:ID="ip_1024">
  <possessor rdf:resource="#temporal_part_of_TIC_01_at_800K"/>
  <has_property rdf:resource="#wrong_prop_652"/>
</indirect_property>

<lis:property rdf:ID="prop_651">
  <numeric_value rdf:resource="#nv_101"/>
  <unit_of_measure rdf:resource="#kelvin"/>
  <property_classified_as rdf:resource="#temperature_setpoint"/>
</lis:property>

<lis:property rdf:ID="wrong_prop_652">
  <numeric_value rdf:resource="#nv_101"/>
  <unit_of_measure rdf:resource="#bar"/>
  <property_classified_as rdf:resource="#temperature_setpoint"/>
</lis:property>

<lis:class_of_information_representation rdf:ID="nv_101">
  <content>
    <xsd:float rdf:value="800.0"/>
  </content>
</lis:class_of_information_representation >
```

Figure 9. An alternative approach for defining physical quantities

The first approach is preferred so as to ensure that units of measure are consistent with the physical quantities they represent. From the ontology, it would be always possible to recognize whether the wrong units of measure are specified for the temperature set-point.

Notice that it is also possible to specify temporal boundings (*beginning* and *ending*) to `temporal_part_of_TIC_01_at_800K` to indicate the time interval in which the setpoint of TIC-01 was at 800K.

9. Representing causality information

During the analysis of physical systems causality plays a very important role for many engineering problems. For example, causality is critical in HAZOP studies to identify potential latent design deficiencies (Kletz, 1999). This section discusses the benefits of using the upper ontology for representing and interpreting the information produced during a HAZOP study.

Typically, a team carrying out a HAZOP study formulates deviations by combining guidewords such as none, more, less and quantities (including process variables). The following is an example based on a case study described by Lawley (1974). Lawley's case study deals with a HAZOP study performed for the feed section of a proposed alkene dimerization plant. In that plant, a stream of alkene-alkane containing small amounts of water is transferred continuously from an intermediate storage tank using pumps J1 to a tank that is used for settling and buffering purposes. Water is removed from the settling tank at intervals to ensure an adequate conversion of the alkene. A fragment of the HAZOP studies is shown graphically in Figure 10. Note however that results of a HAZOP study are typically recorded in tabular form.

Figure 10. Fragment of a HAZOP study

The design deviation, the cause and one of its consequences in Figure 11 are part of the final results of a thinking process initiated by the experts in the HAZOP meeting. Because this thinking process is not recorded, details about the links between the results can be drawn by similar experts but cannot be interpreted by a computer program.

The deviation, its causes and consequences can be modeled in terms of causality relations of the upper ontology (Figure 11). The use of activities and intermediate events makes the HAZOP thinking process explicit resulting in what we call *causality networks*.

Figure 11. Causality network showing the causality between line blockage and loss of feed in the reaction section

Typically, there are cases where a cause can lead to multiple consequences. The explicit representation of the intermediate events and activities between the cause and a consequence of a given scenario can be used by the HAZOP team to predict alternative consequences derived from intermediate events or activities. For example, activity3 (Level of settling tank decreasing) can also lead to cavitation in the feed pumps J2 that transfer the hydrocarbons to the reaction section as shown in Figure 12.

Figure 12. Causality network showing the causality between line blockage and cavitation of pump J2

We have developed a causality network editor for creating graphical representations of the HAZOP thinking process or *causality networks* (Figure 13). A single causality network is constructed for each abnormal scenario.

Figure 13. Screenshot of a causality network editor.

The causality network editor can export the causality information as an OWL file for further processing. The editor is built using the JGraph⁷, and Jena libraries⁸. JGraph is a Java component for developing graph-related applications. Jena is used to generate OWL objects that are serialized to a file. The OWL code for the alkene dimerization case study is listed in the Appendix.

The exported code can be processed for knowledge extraction using an inference engine. For testing purposes, we have developed a querying tool based on the Java Theorem Prover known also as JTP (Figure 14). JTP is an object oriented Modular Reasoning System developed by the Knowledge Systems Laboratory of

⁷ Available online, <http://sourceforge.net/projects/jgraph/>

⁸ Available online, <http://jena.sourceforge.net/>

Computer Science Department in Stanford University JTP (Fikes, Jenkins and Gleb, 2003). By default, JTP is composed of a number of reasoners that implement algorithms such as generalized modus ponens, backward-chaining, and forward chaining and unification (Russell and Norvig, 1995). JTP also can process forward and backward chaining rules written in Knowledge Interchange Format better known as KIF⁹ (Sowa, 2000).

Figure 14. Screenshot of a querying tool based on the Java Theorem Prover

JTP translates each OWL statement into a KIF sentence of the form (PropertyValue Value Predicate Subject Object). Then it simplifies those KIF sentences using a series of axioms that define OWL semantics. OWL statements are finally converted to the form (Predicate Subject Object). The following forward chaining rules can be loaded by the querying tool to retrieve chains of events or activities:

```
(=>
  (and
    (|http://www.ompek.org/iso-15926#|::|cause_of_event| ?act1 ?event1)
    (|http://www.ompek.org/iso-15926#|::|beginning| ?act2 ?event1)
    (|http://www.ompek.org/iso-15926#|::|cause_of_event| ?act2 ?event2))
  (|http://www.ompek.org/iso-15926#|::|cause_of_event| ?act1 ?event2))

(=>
  (and
    (|http://www.ompek.org/iso-15926#|::|cause_of_event| ?act1 ?event1)
    (|http://www.ompek.org/iso-15926#|::|beginning| ?act2 ?event1)
    (|http://www.ompek.org/iso-15926#|::|ending| ?act2 ?event2))
  (|http://www.ompek.org/iso-15926#|::|cause_of_event| ?act1 ?event2))
```

Queries are formulated in KIF, where variables are preceded by a question mark. The following query retrieves the events initiated by line blockage:

Query:

```
(and (ecm:beginning ?act haz:event1) (ecm:cause_of_event ?act ?event))
```

Answer:

```
?act = http://www.ompek.org/causnet#activity1
?event = http://www.ompek.org/causnet#event2
?event = http://www.ompek.org/causnet#event3
?event = http://www.ompek.org/causnet#event4
?event = http://www.ompek.org/causnet#event5
```

⁹ KIF has been updated and simplified into a knowledge representation language known as CLIF (ISO-CL, 2006)

The query tool can be used to extract knowledge on causality of scenarios that were defined separately. For the cavitation scenario, the answer for the previous query includes event6 which is the beginning of the cavitation in J2.

10. Related work

10.1 Other ontology efforts

A number of ontologies have been developed in the process engineering domain. An early ontology project was the MDF ontology. This ontology was developed using a multi-dimensional theory in which physical, behavioral, and operational aspects or dimensions of the plant, process and product are explicitly represented (Batres and Naka, 2000). The objective was to facilitate integration between process simulators, CAD systems and operating procedure synthesis software. For example, the theory identified the need to create separate classes for equipment, physicochemical phenomena, and control activities at a time when data models assigned attributes of equipment and intended physicochemical phenomena to classes of unit operations. The ontology was developed in Ontolingua using the only ontology editor available at the time: the Stanford Ontology Editor. Reasoning was limited to the reasoner embedded in the ontology editor.

A more recent work has resulted in OntoCAPE which defines a comprehensive number of chemical engineering concepts implemented in DAML+OIL (Yang and Marquardt, 2004) based on CliP (Bayer, 2003) which is a conceptual data model founded on a systems-theoretic view of the world. OntoCAPE is structured into partial models which assemble classes related to a common topic. OntoCAPE includes classes and relations to describe process materials, process units, quantities, activities associated to conceptual process design. Generic definitions include temporal points, temporal durations, quantities, properties, coordinate system, and substance. Although much more extensive than the MDF ontology the focus is on the domain of process design and simulation. As ontologies are encoded in DAML+OIL (and more recently in OWL-DL) reasoning is supported by a wide-range of description-logics engines. Some limitations in the ontology are related to the commitment to keep compatibility with description-logics reasoners. For example, the ontology

lacks of safeguards to avoid inconsistencies such as kilograms being assigned instead of Kelvin in the reaction temperature property.

10.2 STEP

It is possible to view an ontology as a standard with which engineering activities and artefacts can be defined. This results in something similar to STEP (Standard for the Exchange of Product Model Data) (NIST, 1999) which is an ISO initiative. ISO 10303 is the official designation for this standard. In ISO 10303, Application Protocols (APs) are parts of the standard that define data models for a certain application domain. Each AP defines classes of objects, their taxonomy, and relations. From the point of view of information modeling, ontologies make a commitment to an unambiguous representation of the objects of a specific domain of discourse, while STEP APs make a commitment to a common data format. Furthermore, STEP APs capture object information as a snapshot in time and lack the ability to capture how the object changed through time. This was one of the motivations behind the development of ISO 15926.

11. Conclusions

Industries around the world recognize that some of the keys to compete in the ever-increasing global markets, as well as to meet increasingly tighter safety and environmental constraints lie in improved work flow processes and in the integration of information systems. However, many current information systems can be integrated only at great cost because of their incompatible proprietary representations of information. One approach to integration of information systems is by means of shared ontologies. In particular, upper ontologies define top-level concepts such as physical objects, activities, mereological and topological relations from which more specific classes and relations can be defined.

We have provided a brief overview of an upper ontology based on ISO 15926-2:2003 which has been implemented in OWL. Some applications of the ontology were also discussed. Specifically, the ontology is being used as an approach to represent and query knowledge generated during Hazards and Operability Studies, and it is also the upper ontology for defining and searching modeling services (Kraines et al., 2006).

It would be of great benefit to the process engineering community to explore the integration with other efforts such as the OntoCAPE ontology.

Acknowledgements

The authors would like to thank the reviewers of this paper for their useful comments and suggestions.

References

- Batres, R. & Naka, Y. (2000). Process Plant Ontologies based on a Multi-dimensional Framework. In: Malone, J. A., Trainham, J. A. and Carnahan, B. (Eds.). *AIChE Symposium Series*, 96 (323), 433-437.
- Bayer, B., 2003, Conceptual information modeling for computer aided support of chemical process design. VDI Verlag GmbH, Düsseldorf.
- Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F. and Stein, L. (2004). A. OWL Web Ontology Language Reference. <http://www.w3.org/TR/owl-ref/>
- Clarke, B. L. (1981). A Calculus of Individuals Based on 'Connection'. *Notre Dame Journal of Formal Logic*, 22, 204–218.
- Ding, Y. (2001). A review of ontologies with the Semantic Web in view. *Journal of Information Science*, 27(6), 377-384.
- Fikes R., Jenkins, J. & Gleb, F. (2003). JTP: A System Architecture and Component Library for Hybrid Reasoning. *Proceedings of the Seventh World Multiconference on Systemics, Cybernetics, and Informatics*. Orlando, Florida, USA, July 27-30, 2003.
- Finin, T. & Ding, L. (2006). Search Engines for Semantic Web Knowledge. *Proceedings of XTech 2006: Building Web 2.0*, Amsterdam, May 16-19, 2006.
- Gangemi A., Guarino, N., Masolo, C., Oltramari, A., Schneider, L. (2002). Sweetening Ontologies with DOLCE. *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web*. Siguenza, Spain (pp. 166 - 181).
- Gruber, T. R. & Olsen, G. R. (1994). An ontology for engineering mathematics. In J. Doyle, P. Torasso, and E. Sandewall (Eds.), *Fourth International Conference on Principles of Knowledge Representation and Reasoning*, Gustav Stresemann Institut, Bonn, Germany, Morgan Kaufmann, 1994.
- ISO 10303-11 (1994). Industrial automation systems and integration – Product data representation and exchange – Part 11: Description methods: The EXPRESS language reference manual.
- ISO 15926-2 (2003). ISO-15926:2003 Integration of lifecycle data for process plant including oil and gas production facilities: Part 2 – Data model.
- ISO-TC184 (2003). ISO/FDIS 15926-2 - Lifecycle integration of process plant data including oil and gas production facilities: Data model: EXPRESS and EXPRESS-G listing. Available: [Online] http://www.tc184-sc4.org/wg3ndocs/wg3n1328/lifecycle_integration_schema.html
- ISO-CL (2006). An ISO effort towards an international standard for Common Logic. Available [Online] <http://philebus.tamu.edu/cl/>
- Kletz, T. (1999). *Hazop and Hazan: Identifying and Assessing Process Industry Hazards*. Institution of Chemical Engineers.
- Kraines S.B., Batres, R., Kemper, B., Koyama, M., Wolowski, V. (2006). Semantic searching based on ontologies and agent systems for knowledge discovery. *Journal of Industrial Ecology* (in press).
- Lacy, L. W. (2005). *Owl: Representing Information Using the Web Ontology Language*. Trafford Publishing.
- Lawley, H. G. (1974). Operability Studies and Hazard Analysis, *Chemical Engineering Progress*, 70 (4), 45-56.
- Marca, D. L., McGowan, C. L. (2005). *IDEF0 and SADT: A Modeler's Guide*. OpenProcess, Inc.
- Minsky, M. (1975). A framework for representing knowledge. In P. H. Winston (Ed.), *The Psychology of Computer Vision*. New York: McGraw-Hill (pp. 211-277).

- Missikoff, M., Navigli, R. (2005) Applying the Unified Process to Large-Scale Ontology Building. Proceedings of 16th IFAC World Congress (IFAC), Praha, Czech Republic, July 4-8, 2005.
- Niles, I. & Pease, A. (2001). Towards a Standard Upper Ontology. 2nd International Conference on Formal Ontology in Information Systems (FOIS), Ogunquit, Maine, October 17-19, 2001.
- Russell, S.J. & Norving, P. (1995). Artificial Intelligence: A modern Approach, Englewood Cliffs, NJ, USA, Prentice Hall
- Simons, P. (2000) Parts: A Study in Ontology. Oxford University Press, USA.
- Sowa, J. (2000). Knowledge Representation: logical, philosophical, and computational foundations. Brooks/Cole.
- Stell, J. G. & West, M. A 4-Dimensionalist Mereotopology. Proceedings of Formal Ontology in Information Systems, Varzi, A.C. Vieu, L. (eds), IOS Press, 2004 (pp 261-272)
- Uschold, M. & Gruninger, M. (1996). Ontologies: Principles, Methods and Applications Engineering Review, 11 (2), 93-113.
- West, M. (2003). Replaceable Parts: A Four Dimensional Analysis Proceedings of the Conference on Spatial Information Theory (COSIT), Ittingen, Switzerland, September 24-28, 2003.
- W3C (2004). OWL Web Ontology Language Overview, W3C Recommendation, [Online] Available: <http://www.w3.org/TR/owl-features/>
- Yang, A. & Marquardt, W. (2004). An Ontology-based Approach to Conceptual Process Modelling. Proceedings of ESCAPE-14, Portugal (pp. 1159-1164).

Appendix.

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns="http://www.ompek.org/causnet#"
  xmlns:lis="http://www.ompek.org/iso-15926#"
  xml:base="http://www.ompek.org/causnet">
  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://www.ompek.org/iso-15926"/>
  </owl:Ontology>
  <lis:activity rdf:ID="activity1">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Pressure decreasing in transfer line</rdfs:comment>
    <lis:beginning rdf:resource="#event1"/>
    <lis:cause_of_event rdf:resource="#event2"/>
  </lis:activity>
  <lis:activity rdf:ID="activity2">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Feed flow decreasing</rdfs:comment>
    <lis:beginning rdf:resource="#event2"/>
    <lis:ending rdf:resource="#event3"/>
  </lis:activity>
  <lis:activity rdf:ID="activity3">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Level of settling tank decreasing</rdfs:comment>
    <lis:beginning rdf:resource="#event3"/>
    <lis:ending rdf:resource="#event4"/>
  </lis:activity>
  <lis:activity rdf:ID="activity4">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Flow to reaction section decreasing</rdfs:comment>
    <lis:beginning rdf:resource="#event4"/>
    <lis:cause_of_event rdf:resource="#event5"/>
  </lis:activity>
  <lis:event rdf:ID="event1">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Line blockage</rdfs:comment>
  </lis:event>
  <lis:event rdf:ID="event2">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Low flow in feed to settling tank</rdfs:comment>
  </lis:event>
  <lis:event rdf:ID="event3">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >No flow at the upstream line of settling tank</rdfs:comment>
  </lis:event>
  <lis:event rdf:ID="event4">
```

```
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>No hydrocarbon in settling tank</rdfs:comment>
</lis:event>
<lis:event rdf:ID="event5">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Loss of feed to reaction section</rdfs:comment>
</lis:event>
</rdf:RDF>
```