ISO 15926

An introduction by Hans Teijgeler
February 2012

A bit of history

1991 – EU ESPRIT project called 'ProcessBase'
target: develop a data model for lifecycle information of a facility

1992 – EPISTLE was founded
with consortia from UK, Netherlands and Norway

1992 – 2007 – Development of ISO 10303-221
Functional data and their schematic representation for process plants

1996 – 2003 – Development of ISO 15926-2
Data Model

1994 – 2007 – Development of ISO 15926-4
Initial Reference Data

1999 – 2011 – Development of ISO 15926-7 & -8
Implementation methods for the integration of distributed systems:
Template methodology & OWL implementation

See also:
http://en.wikipedia.org/wiki/ISO_15926

Why ISO 15926? [1]

- **Globalization** – we no longer operate in our safe little world, we now have to exchange information globally with many partners
- These partners use a **plethora of software**
- Each system has its **proprietary** internal schema, vocabulary, and data format
- **Re-keying** is costly and error-prone
- The number of point-to-point interfaces **increases exponentially** with the number of interfaced systems

Why ISO 15926? [2]

- There are about **5000 natural languages** spoken in the world, not counting the dialects
- Different languages have **different words** for the same thing (e.g. English: ‘vacuum cleaner’ but in Dutch ‘stofzuiger’, translated: ‘dust sucker’), so translating will not exceed the Google translation quality
- Standards invariably cover a **limited domain**, where **ISO 15926** can cover the universe

Why ISO 15926? [3]

Computers are dumb. Normally computer cannot interpret this “gadget” on my front page:

A human being can interpret this as:

At this moment the outside temperature in the town of Sangju in Korea is -7 degrees Celsius and that is a sunny day.

A computer has no clue…..

Why ISO 15926? [4]

Normally computers don't understand their own data. A computer does not blush when handling dirty words.

Computer systems exchange **data** and not **information** .

When mapping your *data* to the format of **ISO 15926** you turn that data into an ISO-standard representation of ***information***.

Why ISO 15926? [5]

Until now the Internet primarily is a

Web of Documents (HTML)

ISO 15926, with the help of **Semantic Web** technologies, can create a

Web of Information (RDF)

or rather a

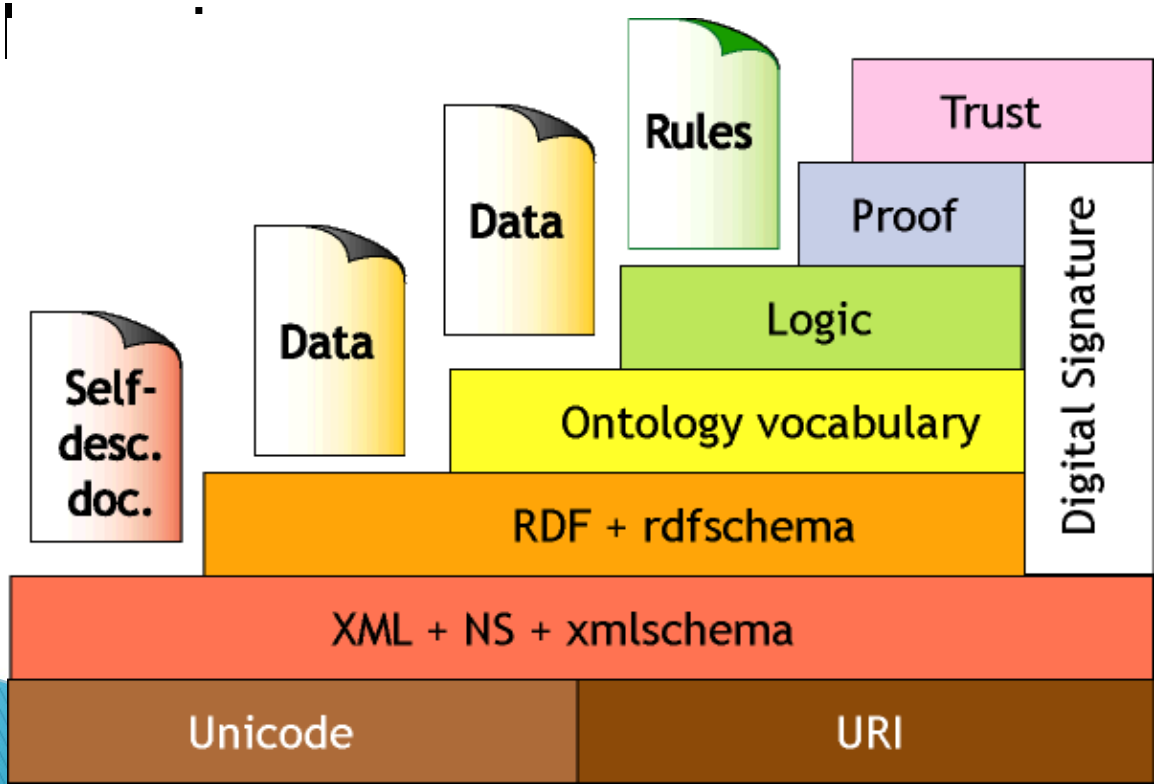
Hybrid Web of Information & Documents

Why ISO 15926? [6]

Why not use the Semantic Web as-is?

Because those semantics are too 'loose' to be reliable enough for engineering work.

ISO 15926 fills the "Ontology vocabulary" and the "Logic"



Objectives of ISO 15926

To build the Semantic Web for the Process Industries for the:

- Integration,
- Sharing,
- Exchange, and
- Hand-over

of distributed Plant Lifecycle Information

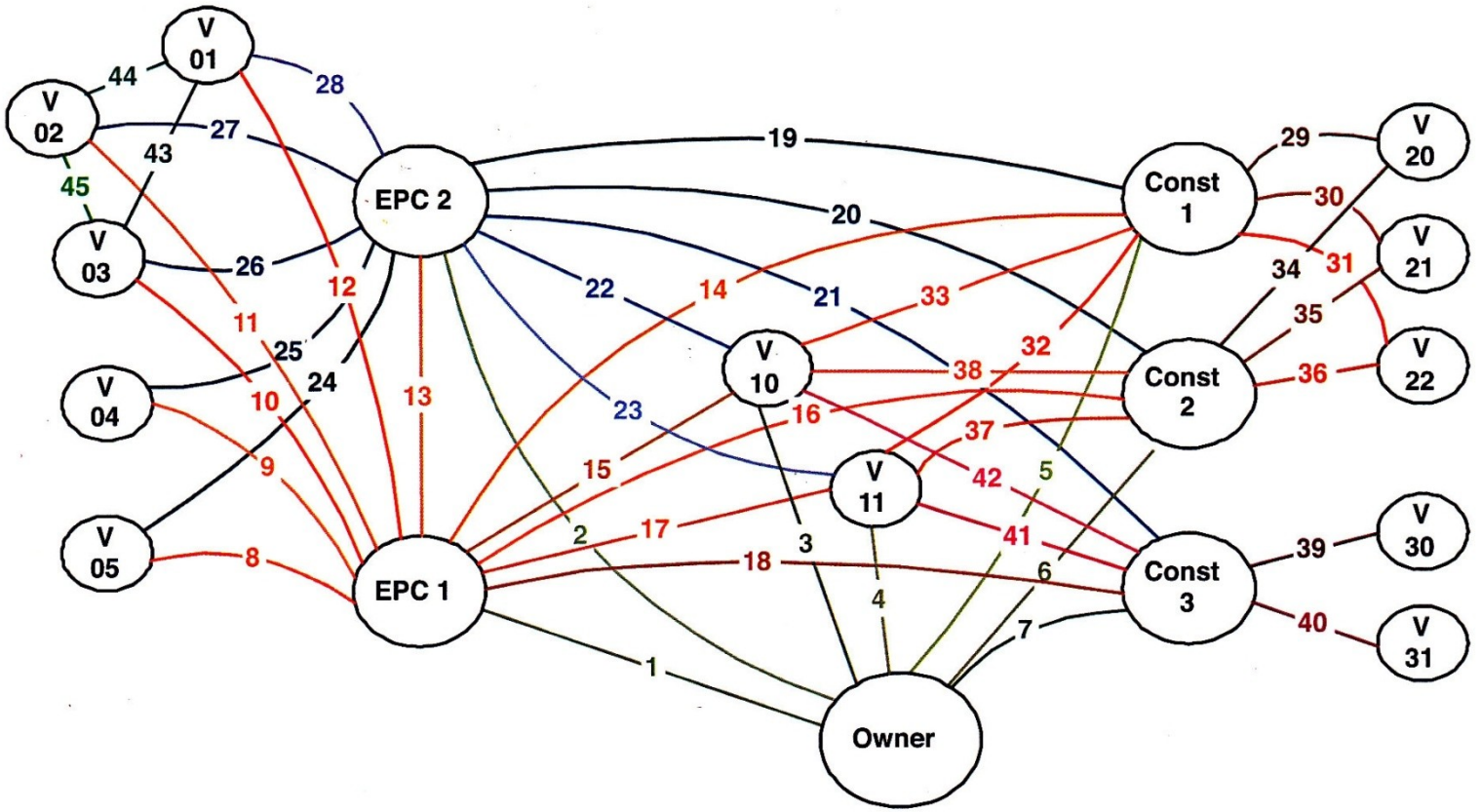
Present and future situation

Presently we have to build **many** point-to-point interfaces.

Assume N applications, then the maximum number of such interfaces is N*(N-1), so **exponentially growing** with N.

In the **future** each application requires **one** ISO 15926 interface, so the number equals N.

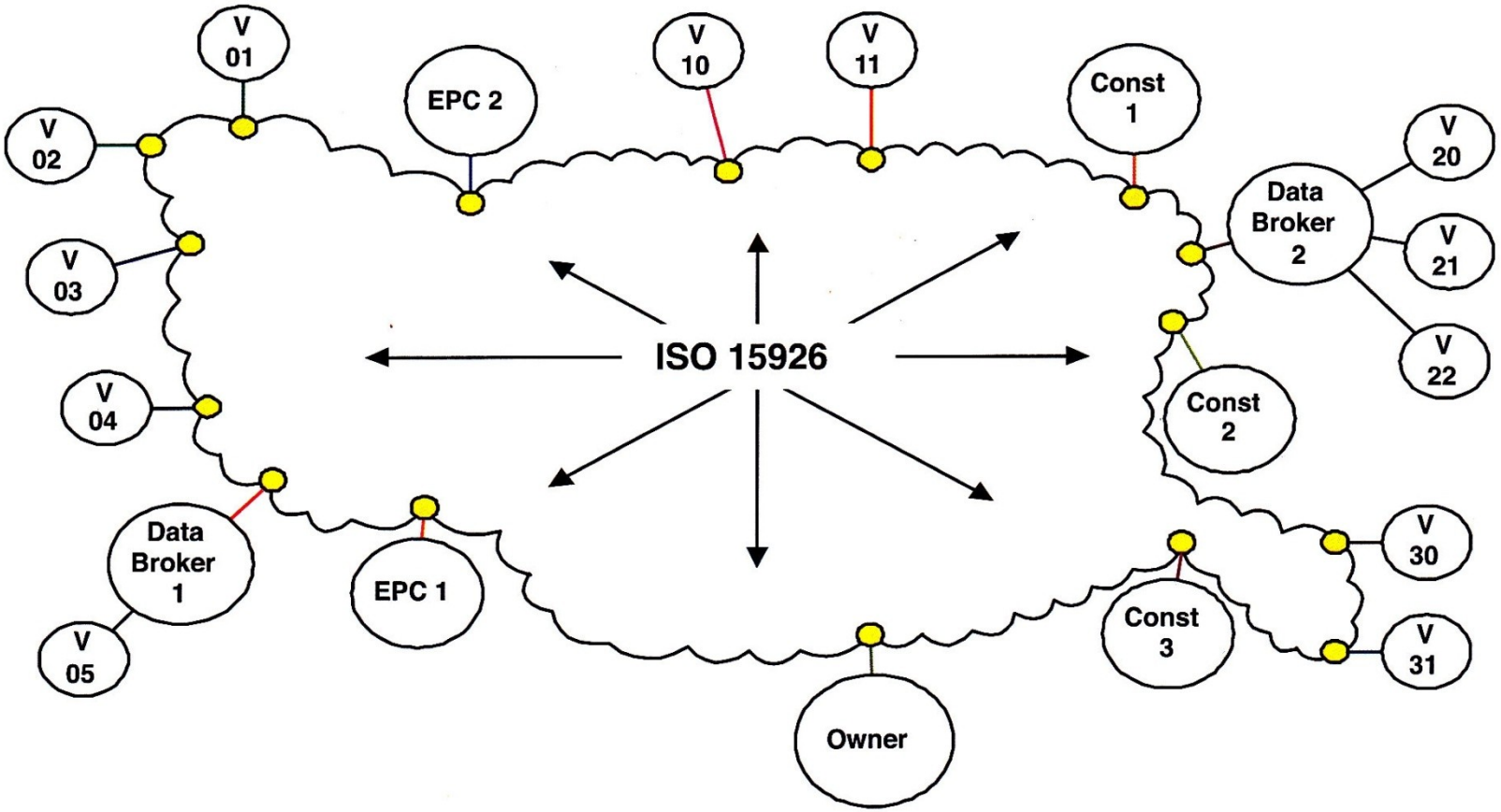
Interfaces without ISO 15926



Many interfaces required (max. N*(N-1))

(source: An Introduction to ISO 15926- by Fiatech)

Interfaces with ISO 15926



All applications have one interface

(source: An Introduction to ISO 15926- by Fiatech)

Information Integration [1]

Information about a plant is created and recorded in many different computer systems across the globe.

ISO 15926 ties all this distributed information together, using Semantic Web technologies, such as:

- **RDF** – Resource Description Framework
- **OWL** – Web Ontology Language
- **SPARQL** – SPARQL Protocol and RDF Query Language

Information Integration [2]

Each object, be it:

a class, a physical object, a property, a number, etc,

has its own, unique, identifier, such as:

http://xyz-corp.com/purl#PERS-1873245

Any time we have information about PERS1873245 reference is made to that **URI** (= Uniform Resource Identifier).

This reference can be fetched from any computer system on the Internet, using **SPARQL**.

Information Sharing [1]

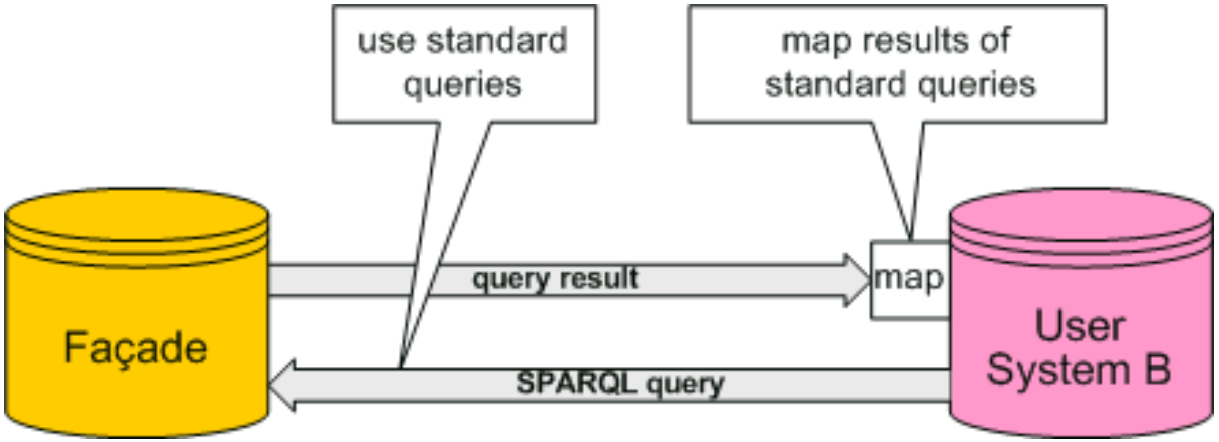
In **ISO 15926** representation of information is "owned" by **one** party at a given date-time.

Information is represented in the form of a "**Template**", as defined in Parts 7 and 8, such as:

```
<owl:Thing rdf:ID="T-54598549">
   <rdf:type rdf:resource="http://p7tpl.rdlfacade.org/data#AssemblyOfAnIndividual"/>
   <p7tpl:hasTemporalWholeOfWhole rdf:resource="http://www.xyz-corp.com/plant21#MPO-498439"/>
   <p7tpl:hasWhole rdf:resource="http://www.xyz-corp.com/plant21#MPO-498439_2012-02-14T14-37-28Z"/>
   <p7tpl:hasAssemblyType rdf:resource="http://rdl.rdlfacade.org/data#R560949833"/>
   <p7tpl:hasTemporalWholeOfPart rdf:resource="http://www.xyz-corp.com/plant21#MPO-430053"/>
   <p7tpl:hasPart rdf:resource="http://www.xyz-corp.com/plant21#MPO-430053_2012-02-14T14-37-28Z"/>
   <p7tpl:valStartTime rdf:datatype="&xsd;dateTime">2012-02-14T14:37:28Z</p7tpl:valStartTime>
</owl:Thing>
```

In case someone else wants to know which impeller is a part of MPO-498439 at a given date-time, he/she can launch a **SPARQL** query and find that that is MPO-430053. This is **information pulling**.

Information Sharing [2]



This **Information Sharing** is done without actually transferring data from the owner to the querying party.

The result of the query may be **mapped** to the internal proprietary format of the software used by the latter, if so required.

Information Exchange [1]

Where **Sharing** requires an initiative of the receiving party, **Exchange** is done upon an initiative of a sending party (**information pushing**)

A typical example of exchange is the sending of a specification (e.g. for a control valve) to three suppliers, with the request for a quotation.

This aspect still requires a lot of study and testing.

Information Exchange [2]

My proposal is to:

- define an **XML Schema** for a type of "fill-in-the-blanks" document (e.g. data sheet)
- create an **XML** document, with **SPARQL queries** for the population of the blanks
- generate the document in **PDF** format
- send the document to the addressee, along with the credentials allowing him to fetch the queried information for (optional) mapping into his own system.

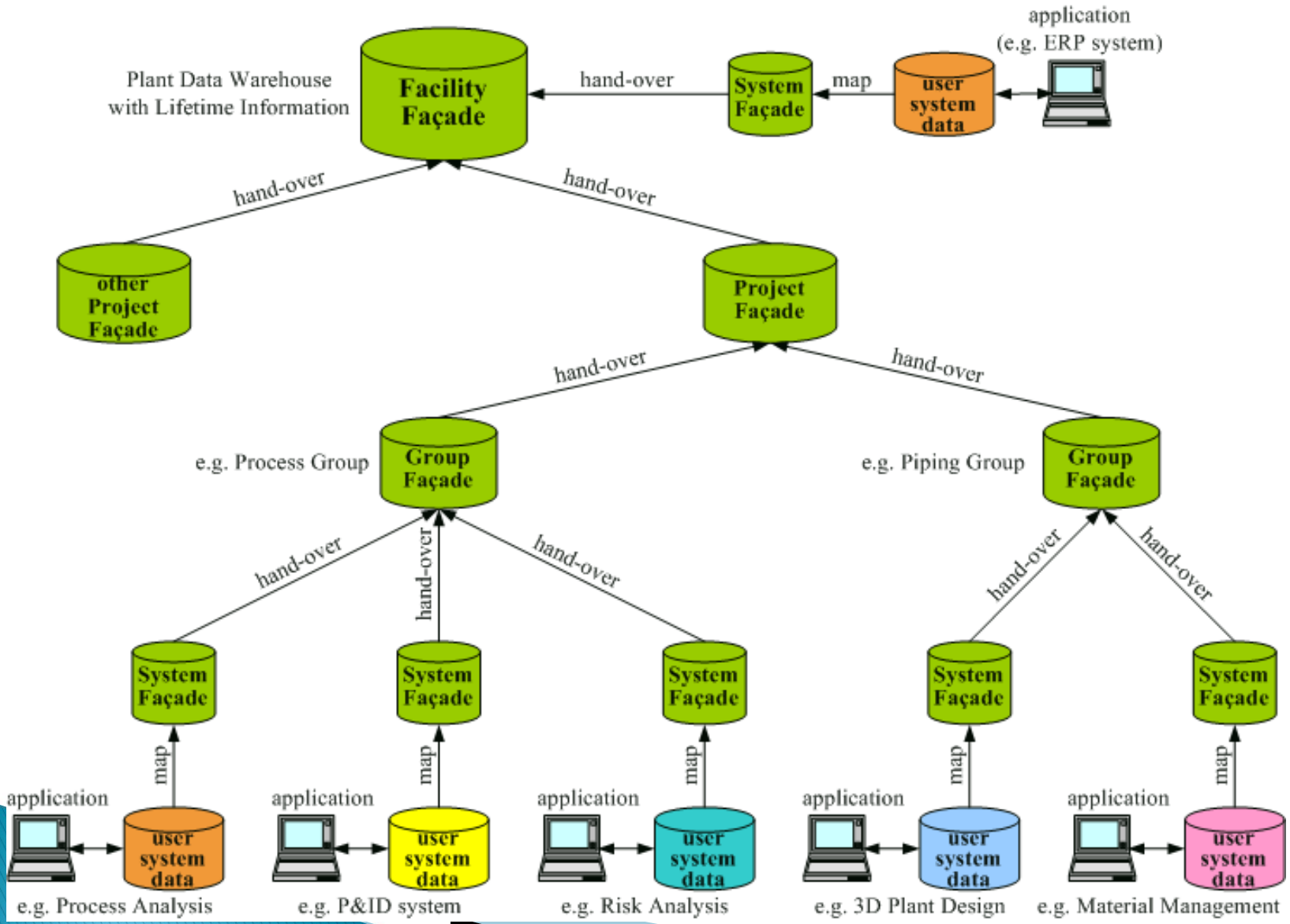
Information Hand-over [1]

Hand-over of information means that the records of that information are being moved from the façade of the previous owner to the façade of the next owner.

This will, in most cases, be done in many increments during a prolonged period of time. Only when the information is, in the responsibility of the owner, “final”, then it can be moved.

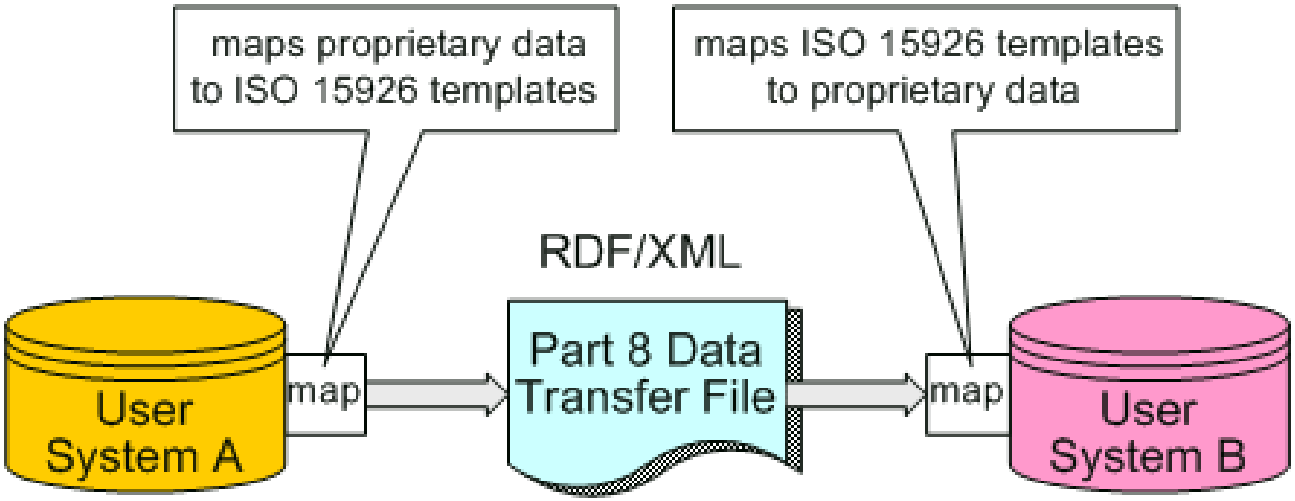
The configuration on the next slide is just a possibility. The project or company can set this up in line with the work processes.

Information Hand-over [2]



Direct interfacing

The standard can also be used for interfacing:



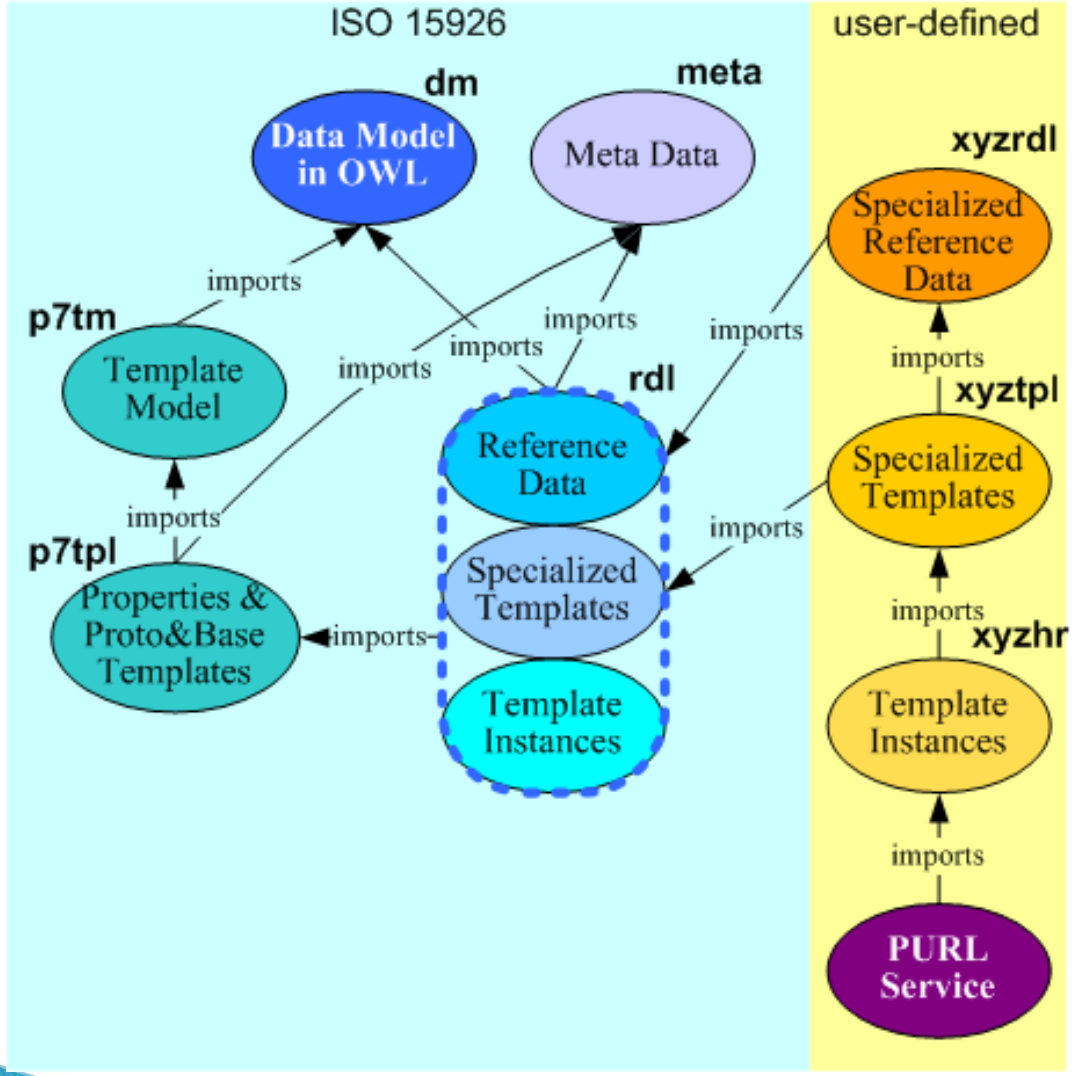
The disadvantage is that the same information is at more than one place and cannot be properly managed.

Let’s have a break
in order to digest this material

The set-up of ISO 15926

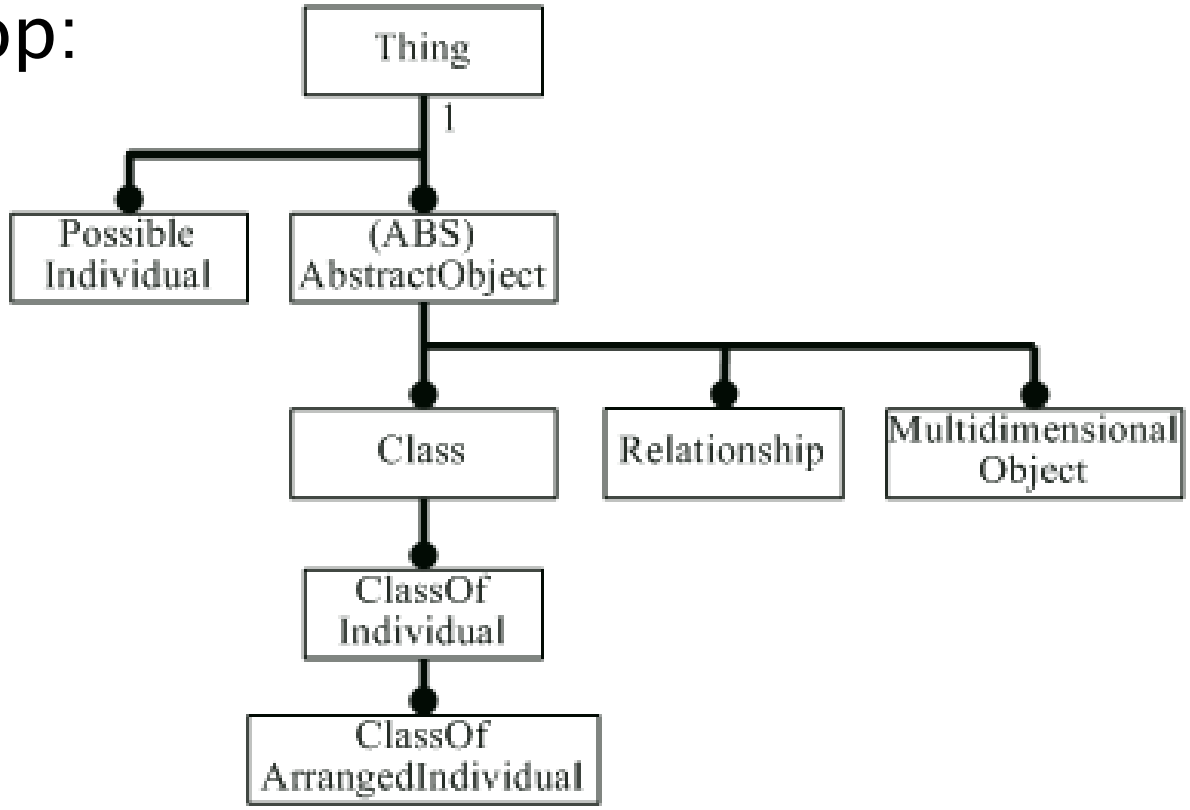
Part 1	Introduction	
Part 2	Data model	the “grammar”
Part 3	Reference geometry & topology	the “pictures”
Part 4	Reference classes & individuals	the “words”
Part 5	----	
Part 6	Reference data representation	
Part 7	Templates	the “sentences”
Part 8	OWL implementation	the “stories”
Part 9	Façade implementation	the “story books”

ISO 15926 configuration

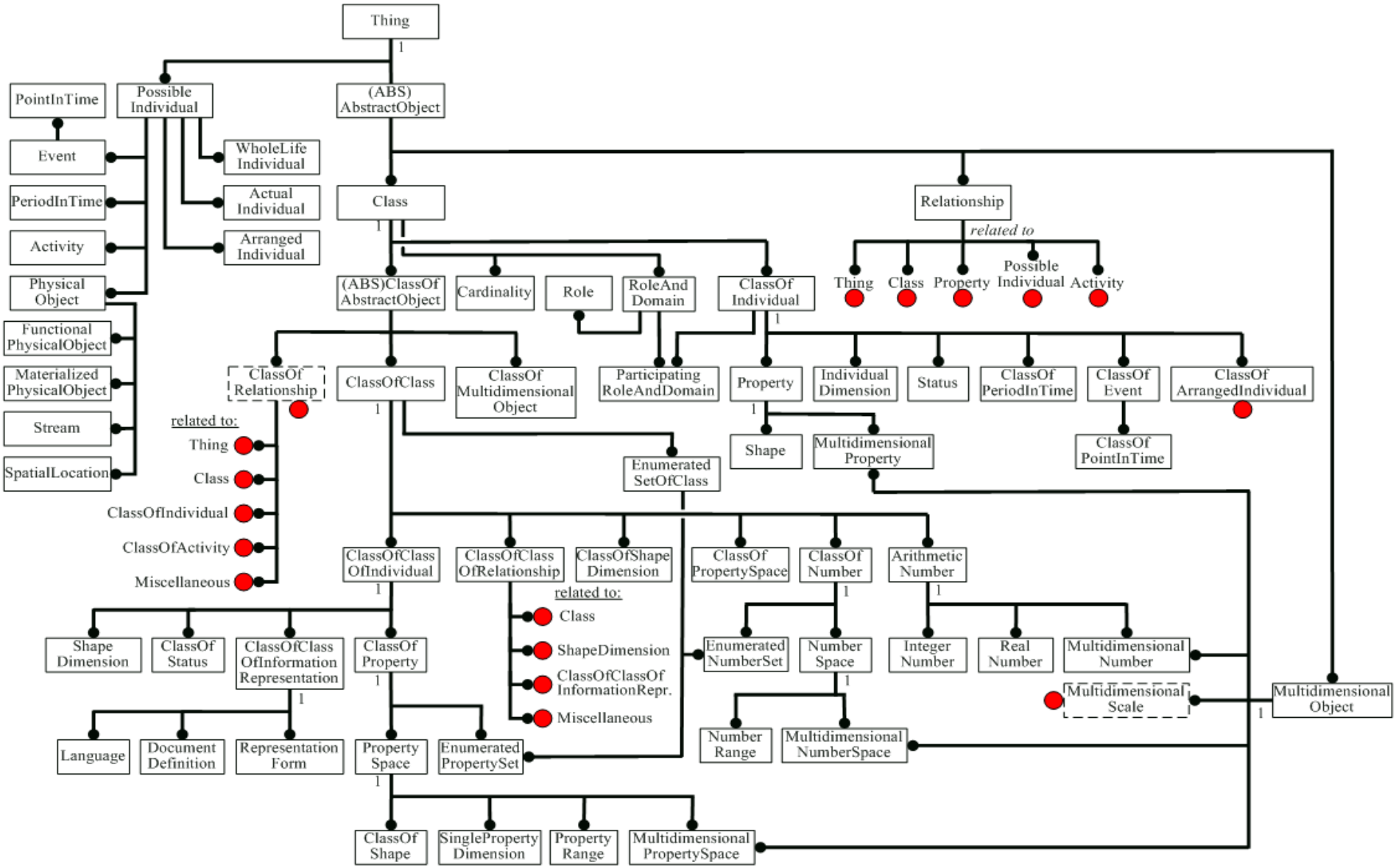


ISO 15926-2 Data Model [1]

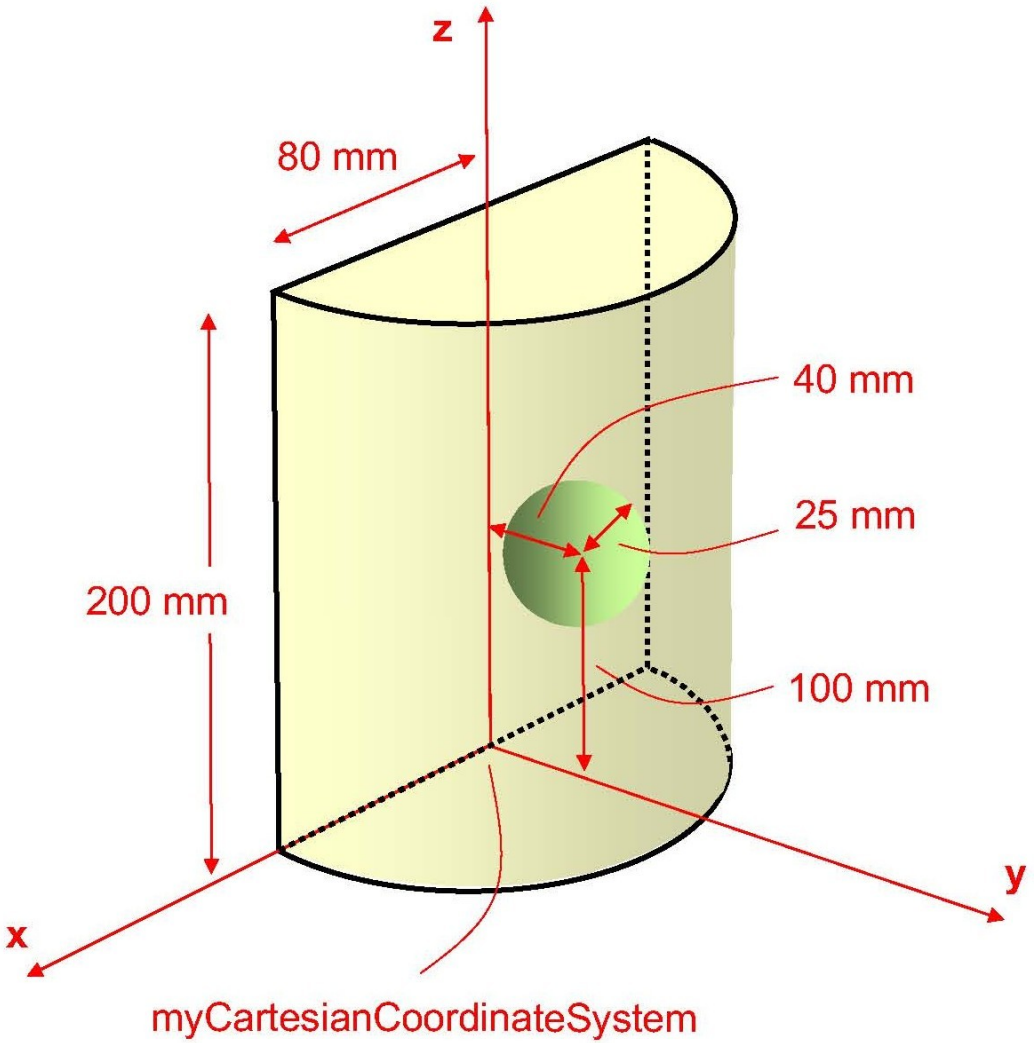
The core of ISO 15926 is the data model of **Part 2**.
It is a highly generic model with 201 entity types.
Here is the top:



ISO 15926-2 Data Model [2]



ISO 15926-3 Geometry & Topology



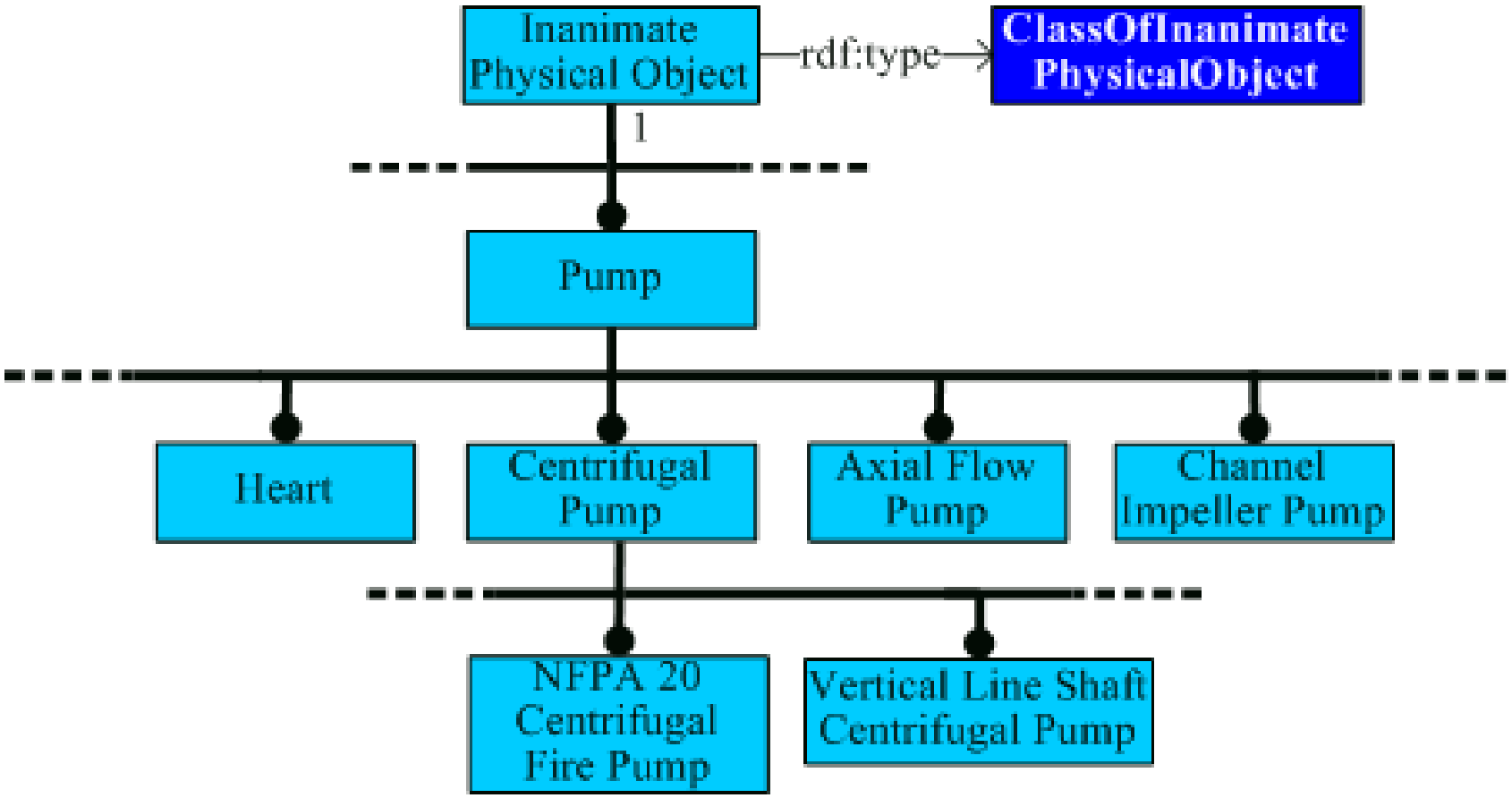
ISO 15926-4 Reference Data [1]

The Reference Data Library (RDL) and its user-defined extensions provide the “meat” on the “skeleton” of the data model.

It starts with a **taxonomy** (class hierarchy) for all applicable entity types. For example for ClassOfInanimatePhysicalObject: all pumps.

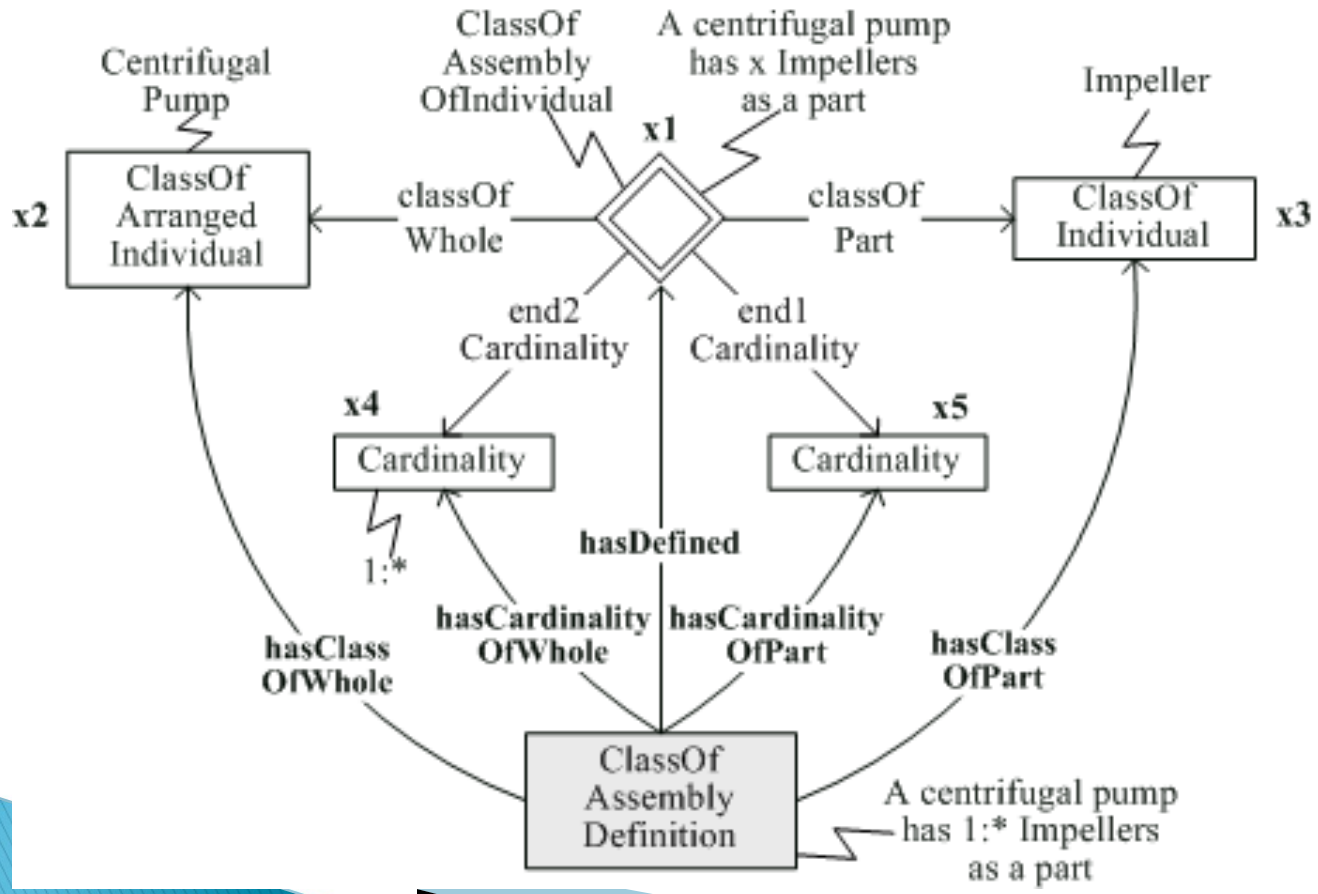
Then the interrelationships are added in the form of templates, thus creating an **ontology**

ISO 15926-4 Reference Data [2]

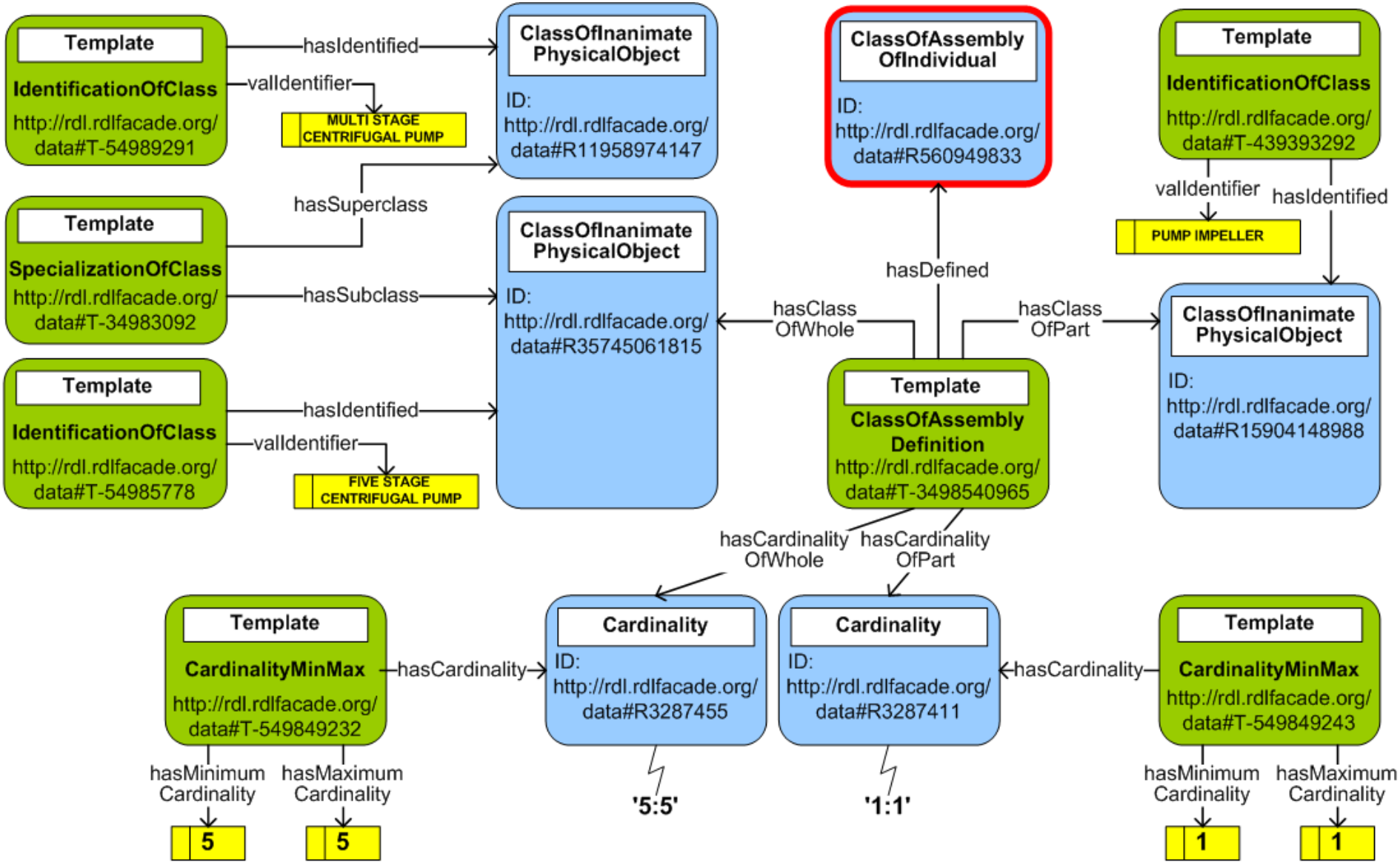


ISO 15926-4 Reference Data [3]

The interrelationships are added in the form of "templates", for example:

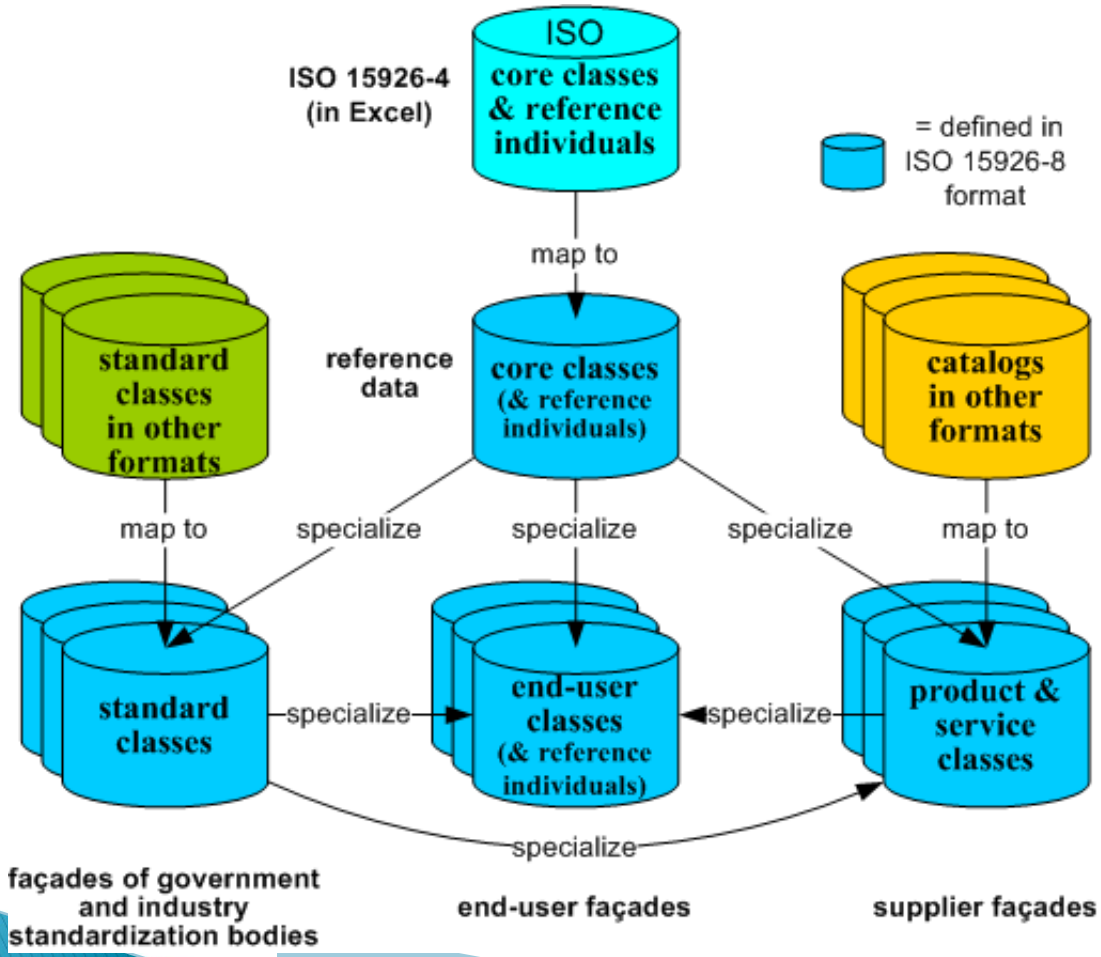


ISO 15926-4 Reference Data [4]



ISO 15926-4 Reference Data [5]

The extensions of the ISO 15926 RDL are to be configured as follows:



Object Information Models (OIM)

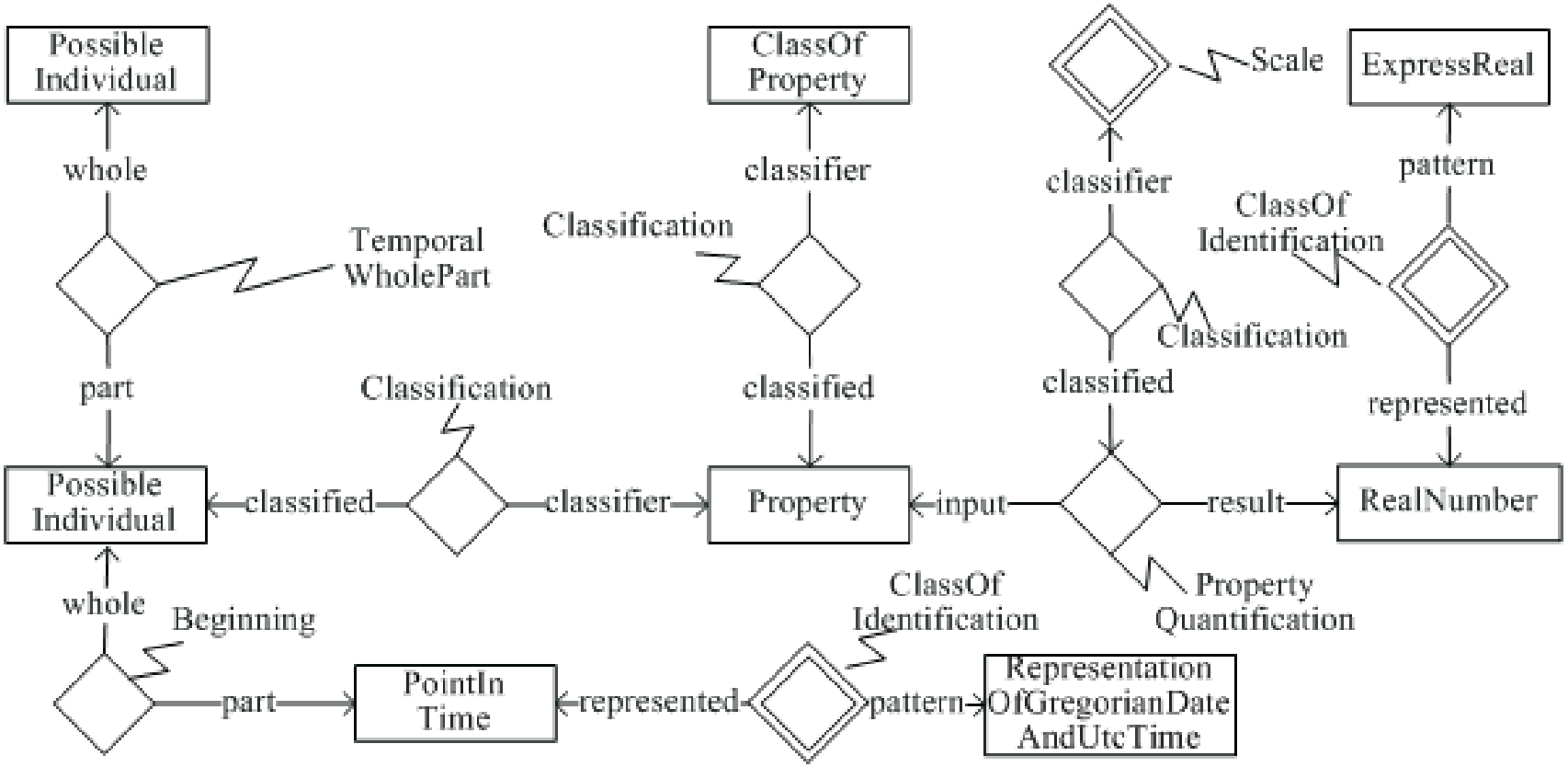
All templates, in which an **RDI** (Reference Data Item) **or a superclass of it** is involved, together form the **OIM** of that RDI.

OIMs are not stored as such, but can be dynamically created with a **SPARQL** query. That query result may include templates in other façades.

An OIM assists the person doing the mapping, because the target can be picked and used.

ISO 15926-7/8 Templates [1]

Here is the template 'PropertyWithValueOfTemporalPart'



ISO 15926-7/8 Templates [2]

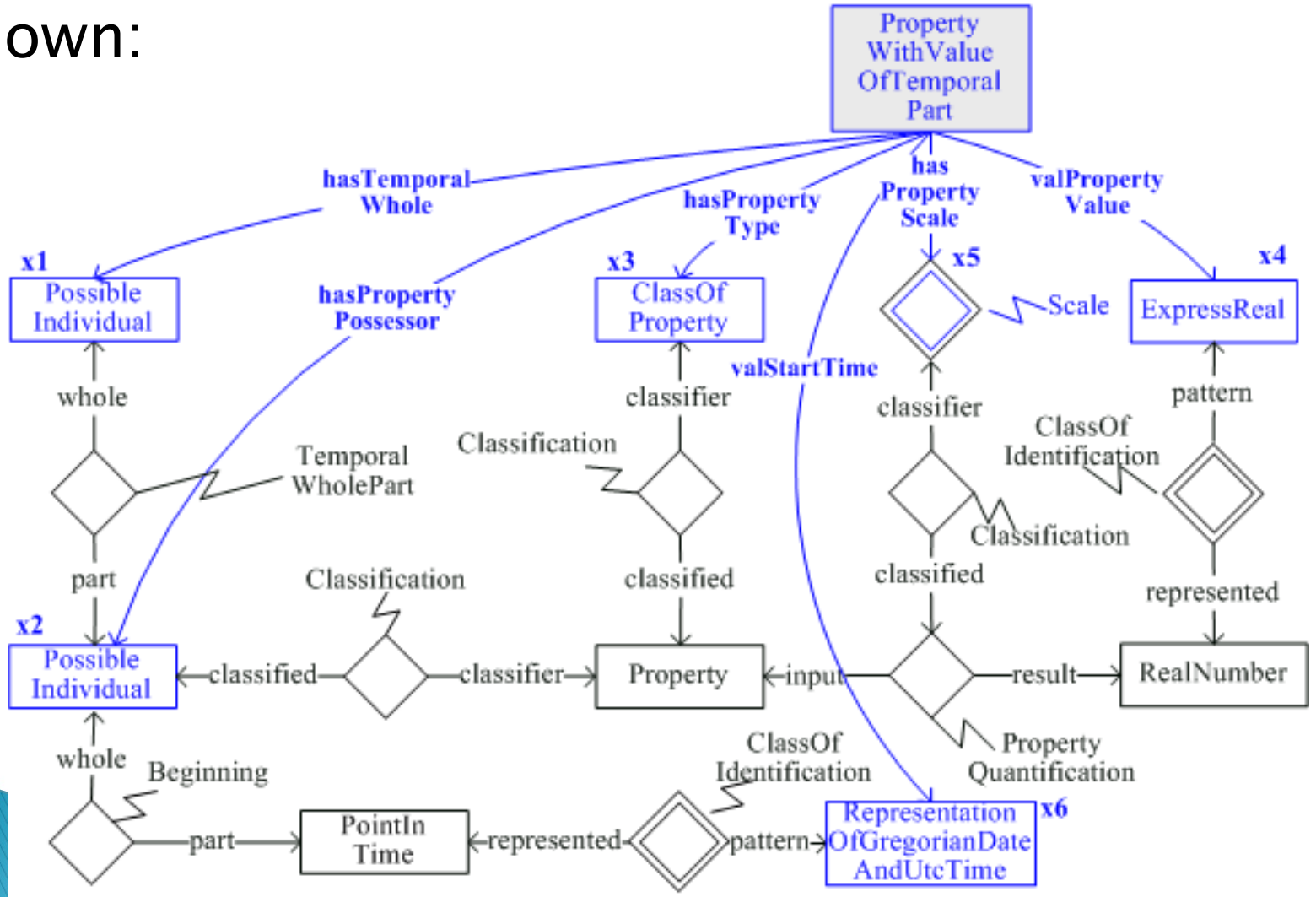
This is called a "**lifted template**", showing all the entity types and their relations that are required to express the semantics of the information.

Shown is in fact a template class, the actual information is expressed with an instance of that template class, with instances of the entity types shown.

It is rather "verbose", and therefore we have a "**lowered template**", a kind of "shorthand":

ISO 15926-7/8 Templates [3]

In blue the "**signature**" of the lowered template is shown:



ISO 15926-7/8 Templates [4]

This signature is represented as follows:

Role No.	Role Name	Role Object Type
1	hasTemporalWhole	http://dm.rdlfacade.org/data#PossibleIndividual
2	hasPropertyPossessor	http://dm.rdlfacade.org/data#PossibleIndividual
3	hasPropertyType	http://dm.rdlfacade.org/data#ClassOfProperty
4	valPropertyValue	http://dm.rdlfacade.org/data#ExpressReal
5	hasPropetyScale	http://dm.rdlfacade.org/data#Scale
6	valStartTime	http://dm.rdlfacade.org/ data#RepresentationOfGregorianDateAndUtcTime

ISO 15926-7/8 Templates [5]

The listing of an instance of this template in **RDF/XML** format looks like this:

```
<owl:Thing rdf:ID="T340982">
    <rdf:type rdf:resource="&p7tpl;PropertyWithValueOfTemporalPart"/>
    <meta:annUniqueName rdf:datatype="&xsd;string">
        A temporal part of MPO349818 has a temperature of 67.7 Celsius since
        2011-08-14T17:46:00Z
    </meta:annUniqueName>
    <meta:annRule rdf:datatype="&xsd;string">#com1_4598292121</meta:annRule>
    <meta:annAccessCode rdf:datatype="&xsd;string">#com1_273872</meta:annAccessCode>
    <p7tpl:hasTemporalWhole rdf:resource="#MPO349818"/>
    <p7tpl:hasPropertyPossessor rdf:resource="#MPO349818_2011-08-14T17-46-00Z"/>
    <p7tpl:hasPropertyType rdf:resource="&rdl;R41192093771"/> <!-- Temperature -->
    <p7tpl:valPropertyValue rdf:datatype="&xsd;float">67.7</p7tpl:valPropertyValue>
    <p7tpl:hasPropertyScale rdf:resource="&rdl;R74877992703"/> <!-- degree Celsius -->
    <p7tpl:valStartTime rdf:datatype="&xsd;dateTime">
        2011-08-14T17:46:00Z
    </p7tpl:valStartTime>
</owl:Thing>
```

ISO 15926-7/8 Templates [6]

The grey part in the template shown on the previous slide contains the “meta data”.

metadata property	rdf:subPropertyOf
meta:annUniqueName	rdfs:label
meta:hasSource	rdfs:seeAlso
meta:annTextDefinition	rdfs:comment
meta:annNotes	rdfs:comment
meta:annAdministrativeNote	rdfs:comment
meta:annExplanatoryComment	rdfs:comment
meta:annChangeDescription	rdfs:comment
meta:annRule	A string that defines any rules, methods, or scripts that are applicable in any application software
meta:annAccessCode	A string giving the security code (resource group) of the information represented by an object. It is used as a parameter in the ACL (Access Control List)

ISO 15926-7/8 Templates [7]

Normally only the lowered templates will be instantiated.

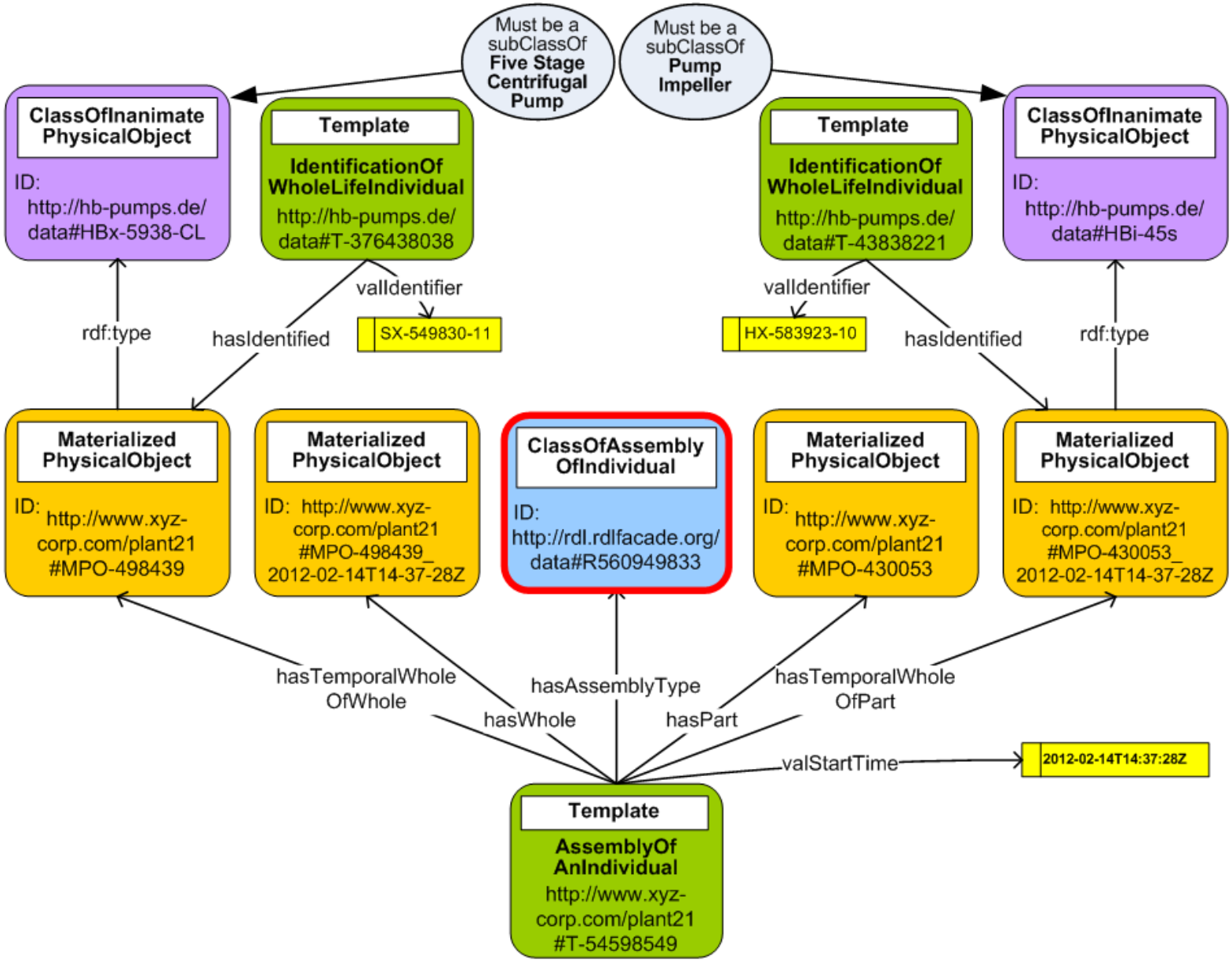
The lifted templates can be generated from the “FOL listing” (First Order Language), like the one shown for this template on the next slide.

With this FOL listing the complete model for an instance of the template can be generated in OWL and be used for validation, reasoning, etc.

ISO 15926-7/8 Templates [8]

PropertyWithValueOfTemporalPart(x1, x2, x3, x4, x5, x6) <->
PossibleIndividual(x1) &
PossibleIndividual(x2) &
ClassOfProperty(x3) &
ExpressReal(x4) &
ScaleTriple(x5) &
RepresentationOfGregorianDateAndUtcTime(x6) &
TemporalWholePartTemplate(x2, x1) &
exists u(PointInTime(u) &
    BeginningTemplate(u, x2) &
    ClassOfIdentificationTemplate(x6, u)) &
exists p(Property(p) &
    ClassificationTemplate(x2, p) &
    ClassificationTemplate(p, x3)) &
exists r(RealNumber(r) &
    ClassOfIdentification(x4, r)) &
exists q(PropertyQuantificationTriple(q, p, r) &
    ClassificationTemplate(q, x5)) .

Example of template usage



ISO 15926-7/8 Template Specs

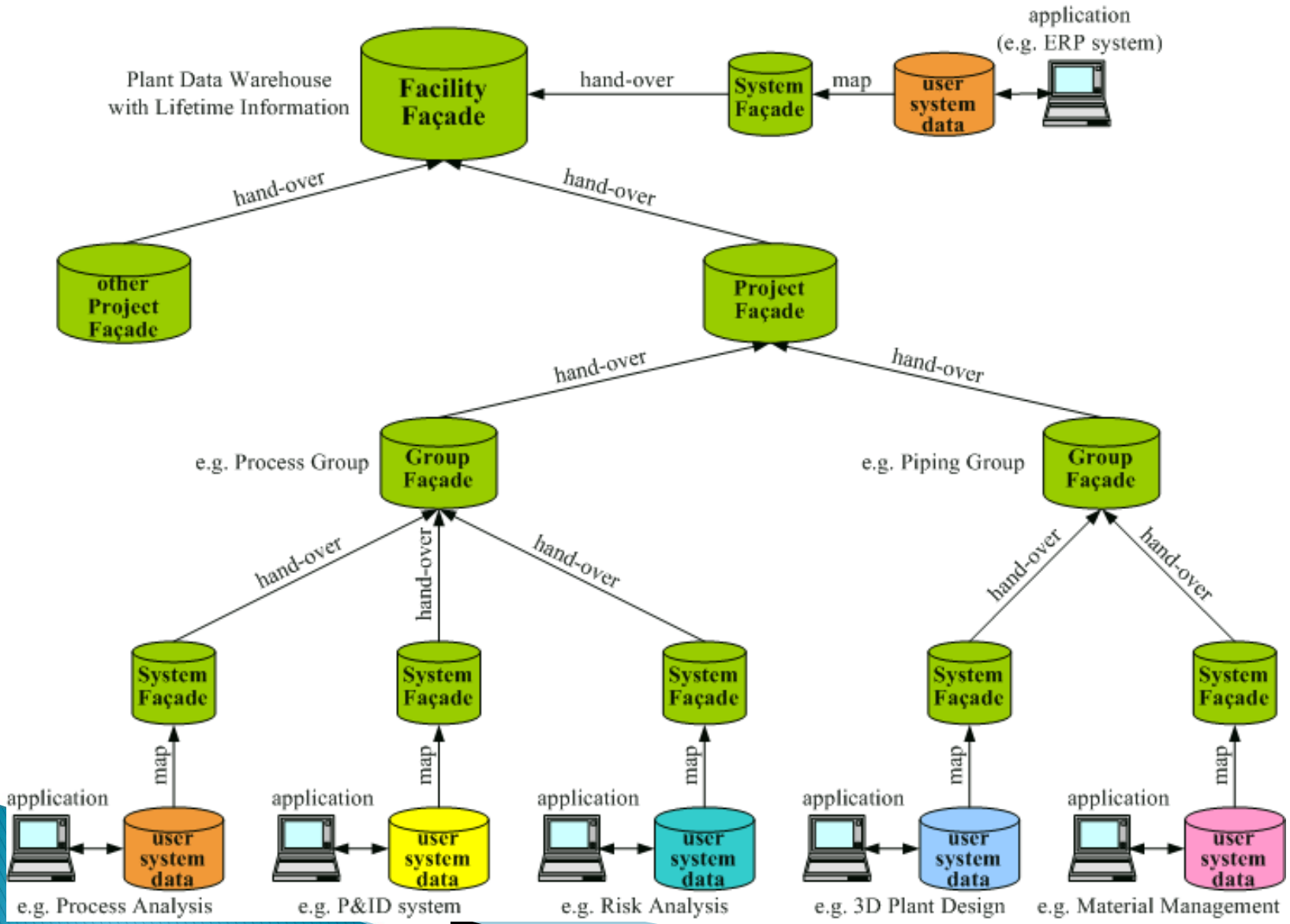
For each template class a Template Specification is made. These can be found at:
http://www.infowebml.ws/TemplateSpecs/index.htm

A template specification has following chapters:

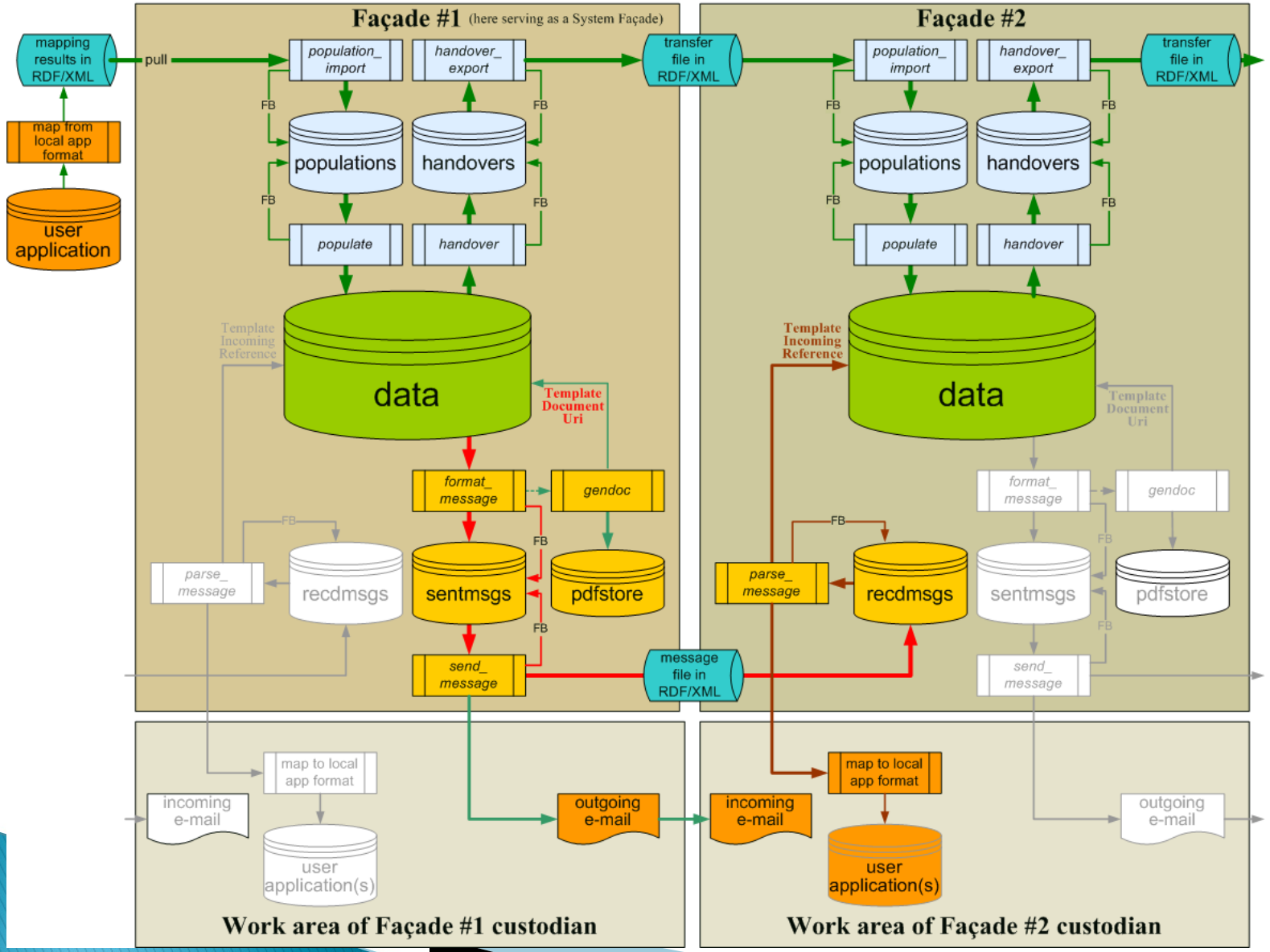
- Template Name
- Intent
- Description
- Lifted and lowered graph
- Lifted template elements
- Signature of lowered template
- OWL code of lowered template
- Specification in First-Order Logic
- Sample of lowered template instance

Let’s have a break
in order to digest this material

ISO 15926-9 Façades [1]



ISO 15926-9 Façades [2]



ISO 15926-9 Façades [3]

Data in the various computer systems are being mapped to template instances in the **RDF/XML** format.

This RDF/XML file is made input to a “triple store” that is called a “Façade”. The reason for the name “façade” is that to the outside world all systems can be approached in the same manner. A façade is a kind of Extranet.

The RDF/XML file is transformed into triples. On the next slide a small example is given.

ISO 15926-9 Triples

Here are the triples of an instance of PropertyWithValueOfTemporalPart

http://www.infowebml.ws/owl/xyzhr#T-340982	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2006/12/owl2-xml#Thing
http://www.infowebml.ws/owl/xyzhr#T-340982	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.infowebml.ws/owl/tpl#PropertyWithValueOfTemporalPart
http://www.infowebml.ws/owl/xyzhr#T-340982	http://www.infowebml.ws/owl/meta#annUniqueName	"A temporal part of MPO349818 has a temperature of 67.7 Celsius since 2011-08-14T17:46:00Z"^^http://www.w3.org/2001/XMLSchema#string
http://www.infowebml.ws/owl/xyzhr#T-340982	http://www.infowebml.ws/owl/meta#annRule	"#com1_4598292121"^^http://www.w3.org/2001/XMLSchema#string
http://www.infowebml.ws/owl/xyzhr#T-340982	http://www.infowebml.ws/owl/meta#annAccessCode	"#com1_273872"^^http://www.w3.org/2001/XMLSchema#string
http://www.infowebml.ws/owl/xyzhr#T-340982	http://www.infowebml.ws/owl/tpl#hasTemporalWhole	http://www.infowebml.ws/owl/xyzhr#MPO349818
http://www.infowebml.ws/owl/xyzhr#T-340982	http://www.infowebml.ws/owl/tpl#hasPropertyPossessor	http://www.infowebml.ws/owl/xyzhr#MPO349818_2011-08-14T17-46-00Z
http://www.infowebml.ws/owl/xyzhr#T-340982	http://www.infowebml.ws/owl/tpl#hasPropertyType	http://www.infowebml.ws/owl/rdl#R41192093771
http://www.infowebml.ws/owl/xyzhr#T-340982	http://www.infowebml.ws/owl/tpl#valPropertyValue	"67.7"^^http://www.w3.org/2001/XMLSchema#float
http://www.infowebml.ws/owl/xyzhr#T-340982	http://www.infowebml.ws/owl/tpl#hasPropertyScale	http://www.infowebml.ws/owl/rdl#R74877992703
http://www.infowebml.ws/owl/xyzhr#T-340982	http://www.infowebml.ws/owl/tpl#valStartTime	"2011-08-14T17:46:00Z"^^http://www.w3.org/2001/XMLSchema#dateTime

ISO 15926-9 Security [1]

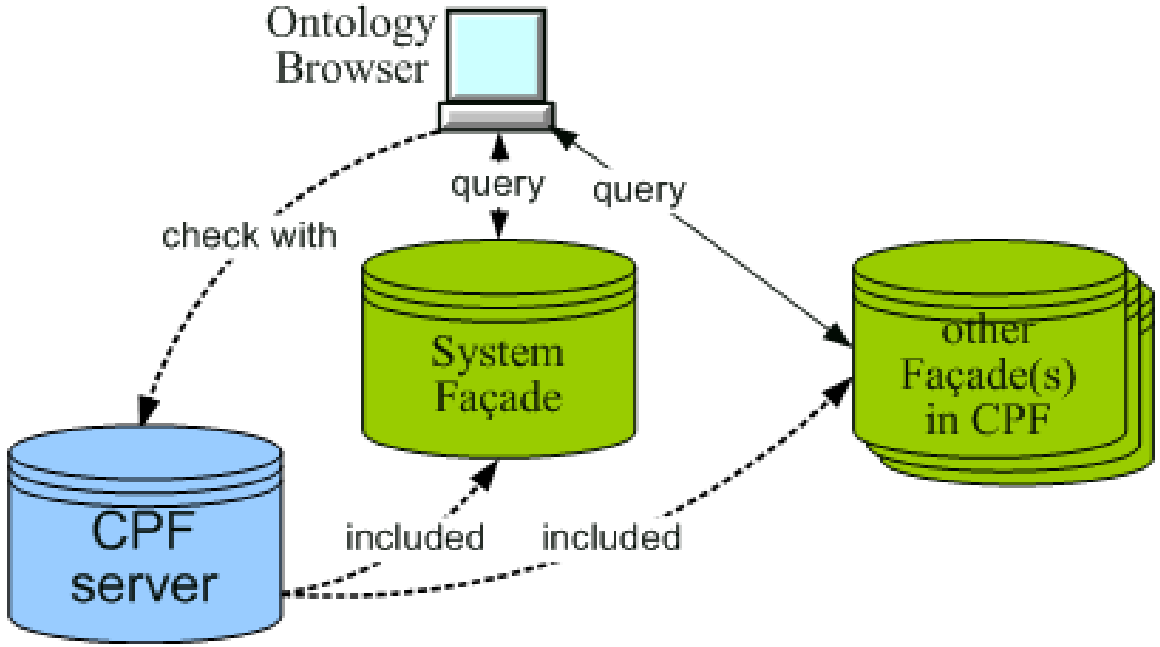
The Internet is a dangerous place, and therefore security measures must be stringent. This is still under development.

In these measures the **CPF** (Confederation of Participating Façades) will play an important role.

A CPF is a register of Façade **URIs** with information about access rights of any one Façade towards any of the other Façades in that CPF.

ISO 15926-9 Security [2]

When launching a **SPARQL** query it is checked to which façades the querier has access.



Let’s have a break
in order to digest this material

Modeling issues

Important concepts of ISO 15926

Temporal parts [1]

One of the key concepts of ISO 15926 is that the validity of information can be placed in time.

To that end we have the concept of "**temporal part**" of a PossibleIndividual.

The spatial extent of the temporal part is that of the temporal whole for the period of the existence of the temporal part.

Temporal parts [2]

Let us take the example of John Doe.

He was born on April 12th 1984. That was when this WholeLifeIndividual started to exist.

He has as temporal parts:

- His school years
- His membership of the tennis club
- His employment with the XYZ Corp.

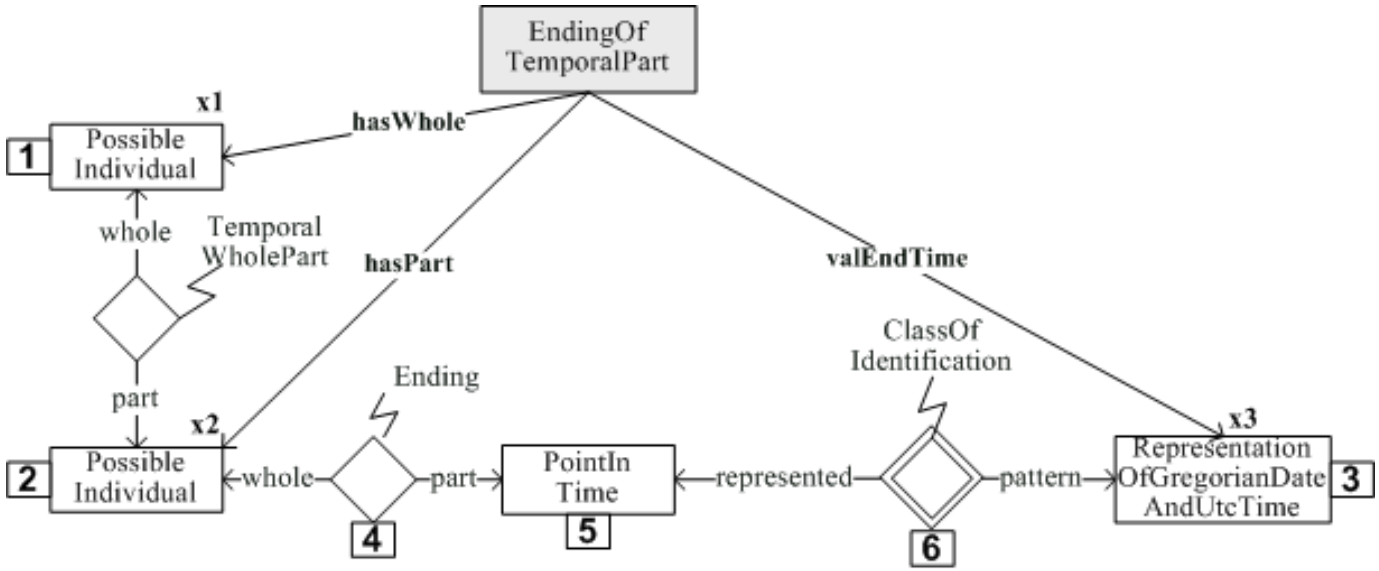
For the latter of these temporal parts one or more temporal parts may apply, like for the information what his 2012 salary is, his rank and function, his performance, etc.

Temporal parts [3]

A temporal part inherits the information (except the metadata) from its temporal whole.

Use **one** temporal part for **one** information chunk (template). When that information no longer is valid, then end that temporal part with a template EndingOfTemporalPart:

Temporal parts [4]



Role No.	Role Name	Role Object Type
1	hasTemporalWhole	http://dm.rdlfacade.org/data#PossibleIndividual
2	hasTemporalPart	http://dm.rdlfacade.org/data#PossibleIndividual
3	valEndTime	http://dm.rdlfacade.org/ data#RepresentationOfGregorianDateAndUtcTime

Ending a TemporalPart [1]

Let us return to our template example:

```
<owl:Thing rdf:ID="T-54598549">
   <rdf:type rdf:resource="http://p7tpl.rdlfacade.org/data#AssemblyOfAnIndividual"/>
   <p7tpl:hasTemporalWholeOfWhole rdf:resource="http://www.xyz-corp.com/plant21#MPO-498439"/>
   <p7tpl:hasWhole rdf:resource="http://www.xyz-corp.com/plant21#MPO-498439_2012-02-14T14-37-28Z"/>
   <p7tpl:hasAssemblyType rdf:resource="http://rdl.rdlfacade.org/data#R560949833"/>
   <p7tpl:hasTemporalWholeOfPart rdf:resource="http://www.xyz-corp.com/plant21#MPO-430053"/>
   <p7tpl:hasPart rdf:resource="http://www.xyz-corp.com/plant21#MPO-430053_2012-02-14T14-37-28Z"/>
   <p7tpl:valStartTime rdf:datatype="&xsd;dateTime">2012-02-14T14:37:28Z</p7tpl:valStartTime>
</owl:Thing>
```

About a five-stage centrifugal pump and its impellers (here one such impeller is shown).

The question is: how do we find that impeller in one Terabyte of triples?

Ending a TemporalPart [2]

In structured English that goes as follows:

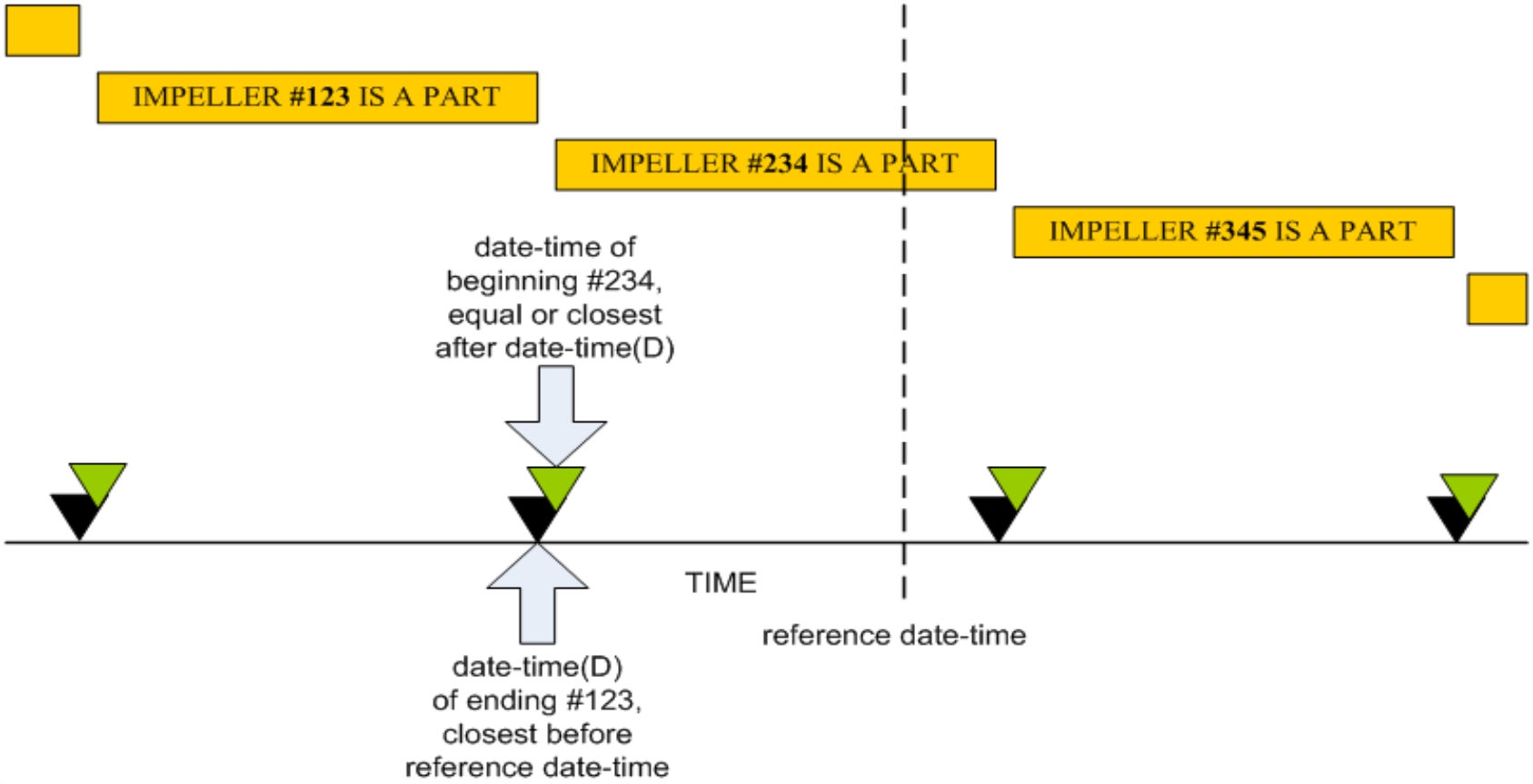
- Enter a reference date-time at which the information must be valid
- Search all templates that involve MPO-498439 and store that set(A)
- From set(A) search all instances of the template class AssemblyOfAnIndividual and store that set(B)
- From set(B) search all objects of the hasPart property and store that set(C); (this set contains all impellers that were part of the 'whole' MPO_498439 at any time)

THEN:

- Using set(C) search for the instance of EndingOfTemporalPart from set(A) that has a date-time that is closest before the given reference date-time, and note that date-time(D)
- Search from set(B) the template that has a valStartTime value that equals or is closest after date-time(D)
- The object of the hasPart property is the one you were looking for.

Ending a TemporalPart [3]

We can show this graphically:



Ending a TemporalPart [4]

The **SPARQL** language cannot do this kind of logic alone.

It can do the searching, but the algorithms for the dates, as shown on the previous slide must be programmed in a procedural language such as **Java**, **PHP**, etc

Let’s have a break
in order to digest this material

Functional vs Materialized PhysicalObject [1]

Anything you create when designing and engineering a plant, like:

- Diagrams (PFD, P&ID, Loop Diagrams, etc)
- Process Data Sheets
- Specifications (Equipment Data Sheets)

is about **classes**, and hence you should use templates about classes.

Once the ordered goods have been delivered, the world of **physical objects** starts.

Functional vs Materialized PhysicalObject [2]

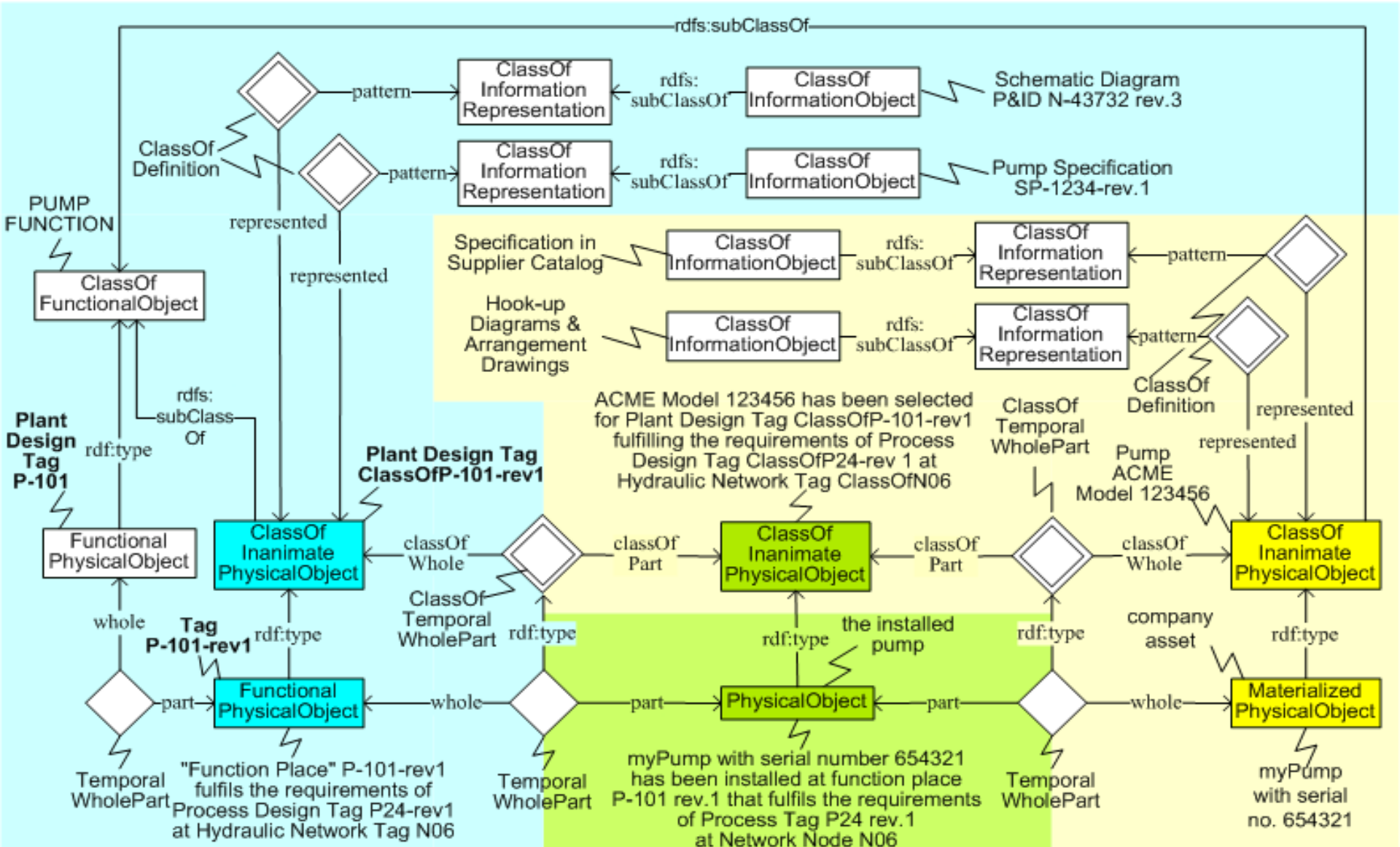
You can order more than one physical object against a specification.

You can build more than one plant against the P&IDs (e.g. P&IDs of a process licensor) and 3D models.

Pump P-101, as specified on a specification, is a **FunctionalPhysicalObject** that is classified with the Class that is defined by that specification.

Pump P-101 only exists in the context of its function within the design of the plant, as shown on the P&ID and 3D model.

Functional vs Materialized PhysicalObject [3]



Functional vs Materialized PhysicalObject [4]

The **MaterializedPhysicalObject** is the one coming from the manufacturer.

The green **ClassOfInanimatePhysicalObject** in the middle inherits the information of the designed class P-101 and the supplier class Model 123456, **provided it fulfils the requirements of P-101!!!** .

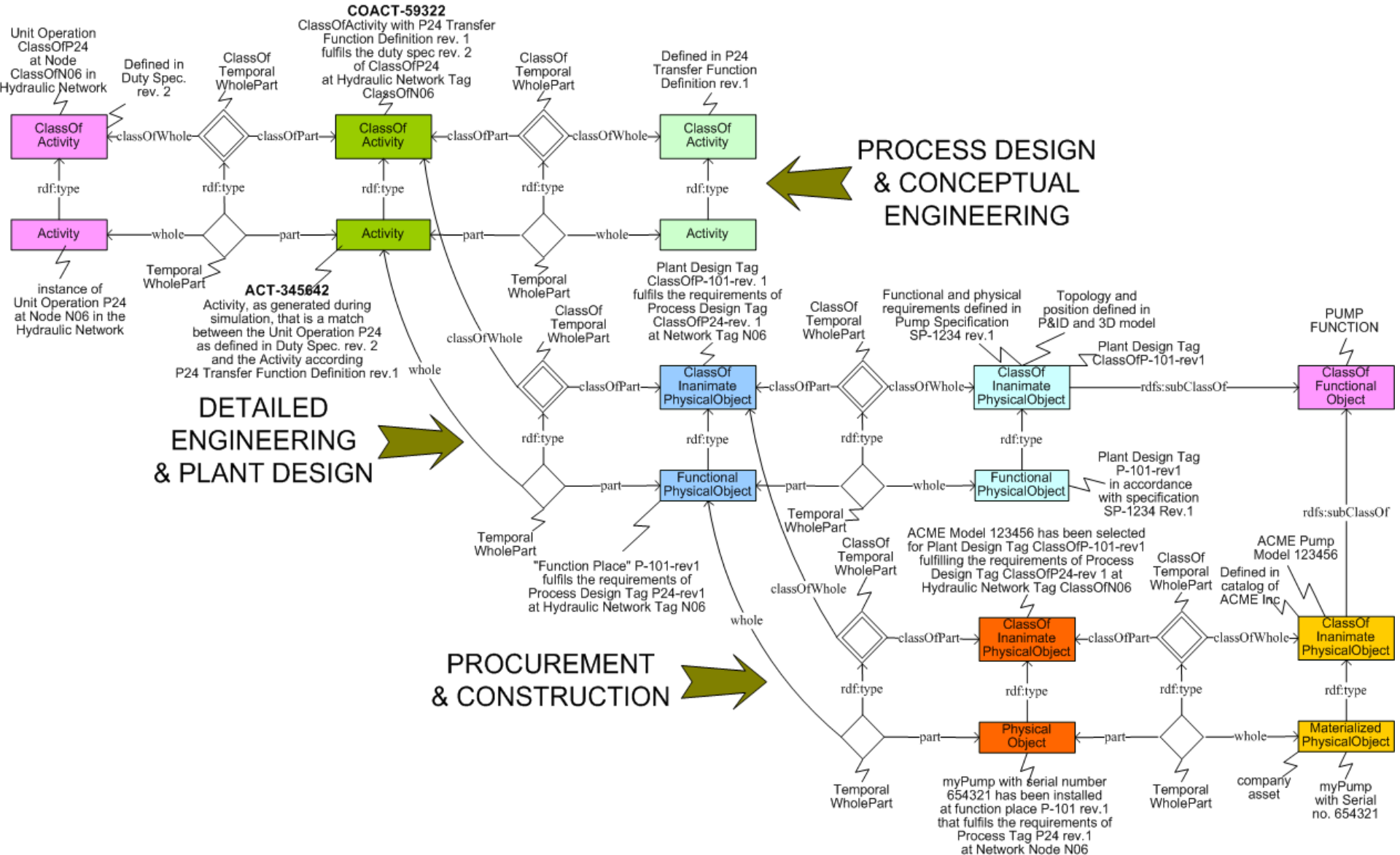
The **installed PhysicalObject** is a temporal part of the "function place" P-101 and of the supplied pump with serial number 654321, and inherits all properties of both.

Functional vs Materialized PhysicalObject [5]

Since the TemporalWholePart relationship is transitive, the installed PhysicalObject inherits all information about its both temporal wholes.

Study is made to include the FEED (Front End Engineering & Design) information. Visit: http://www.15926.info/cradle-to-grave/index.htm

Functional vs Materialized PhysicalObject [6]



ISO 15926-7 & -8

Some highlights

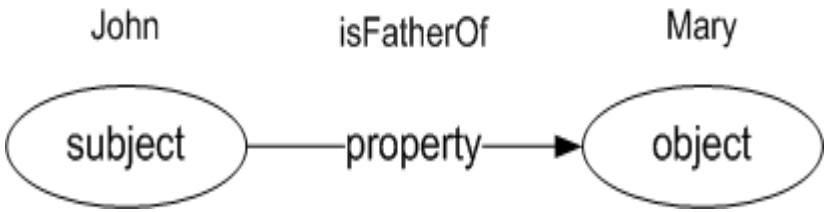
Incompatibilities & Solutions

ISO 15926-2 and **OWL** are, in most aspects, compatible. But there are a few problems.

In the following slides the most important of these problems, and their solutions, will be discussed.

(ClassOf)Relationship in OWL [1]

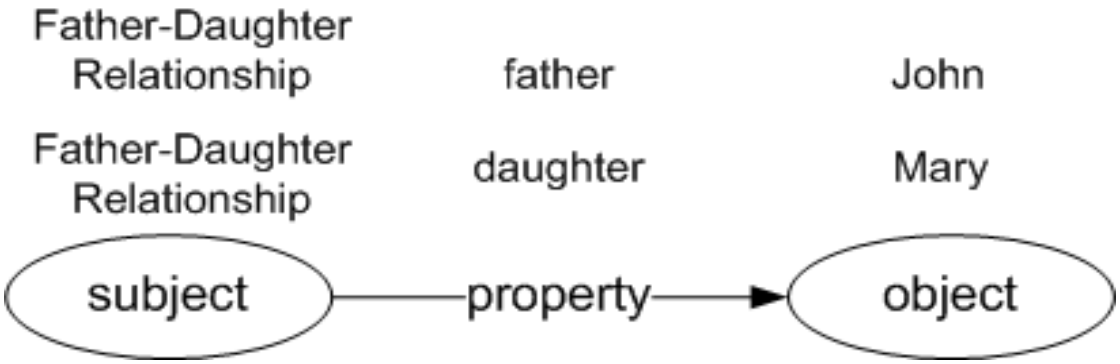
Relationships and **ClassOfRelationships** present a problem, at least in the perception of some logicians. They want to use standard **OWL tools** for reasoning, and these expect an **rdf:Property**:



instead of what ISO 15926 offers:

(ClassOf)Relationship in OWL [2]

That “Father-Daughter Relationship” could be written as two subject-property-object instances:



but, although syntactically OWL-compliant, that is unacceptable for the logicians.

(ClassOf)Relationship in OWL [3]

The solution is found in the "**Proto Templates**".

These provide a layer of abstraction immediately above the relational entity types of **ISO 15926-2** by hiding the reified relationships of ISO 15926-2.

Reification is a modeling style in which a relationship is expressed as an object class.

In terms of First-Order Logic an ISO 15926-2 Relationship is expressed in two axioms:

(ClassOf)Relationship in OWL [4]

RTriple(z,x,y) <-> **R(z) ∧ hasR1(z,x) ∧ hasR2(z,y)**

RTemplate(x,y) <-> ∃z(RTriple(z,x,y))

where:

R = the name of a Part 2 (ClassOf)Relationship

hasR1 = replacement for role r1 in that Part 2 (ClassOf)Relationship

∃z(RTriple(z,x,y)) = there exists at least one z such that RTriple(z,x,y) is true

RTemplate has the **rdf:Property** format, and is derived from the Part 2 (ClassOf)Relationship.

Only in cases where the (ClassOf)Relationship is the rdf:object in another (ClassOf)Relationship, the RTriple is used.

(ClassOf)Relationship in OWL [5]

An example:

ApprovalTriple(x,y,z) <-> **Approval(x) ∧ hasApproved(x,y) ∧ hasApprover(x,z)**

ApprovalTemplate(y,z) <-> ∃u(ApprovalTriple(u,y,z))

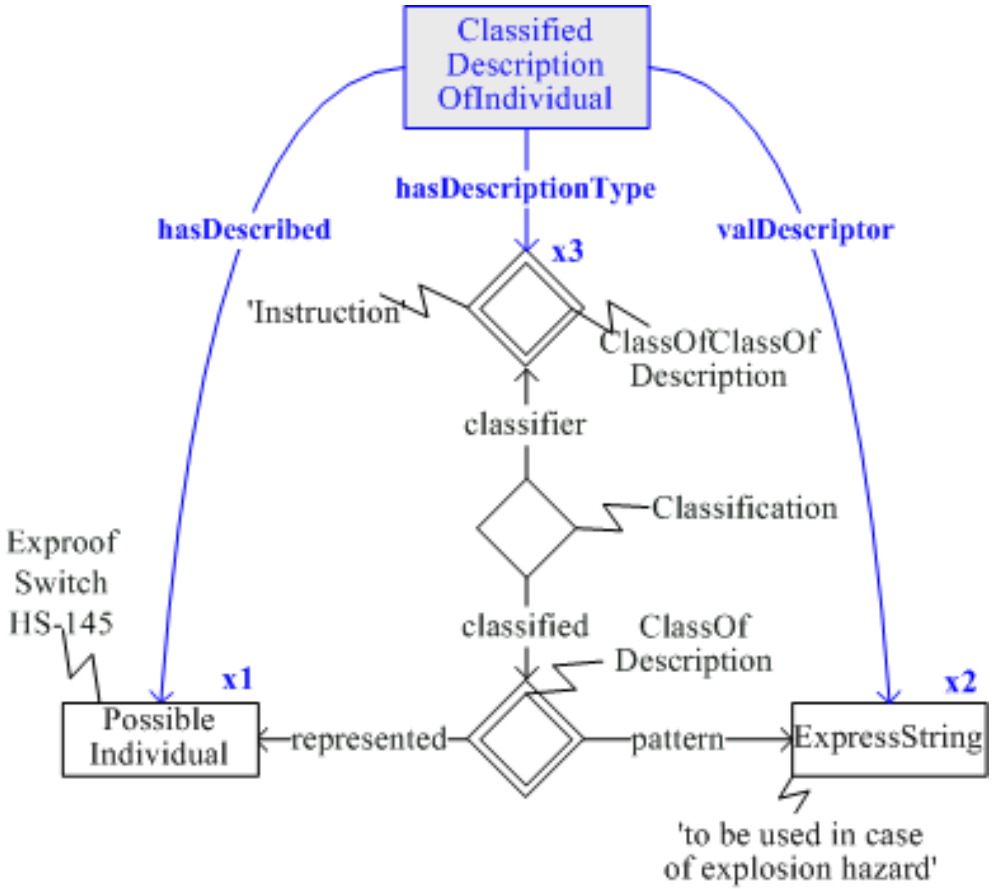
where **Approval(x) ∧ hasApproved(x,y) ∧ hasApprover(x,z)** :

- represents the Part 2 relationship Approval, with the attributes ‘approved’ and ‘approver’
- is defined in what is called "ISO 15926-2 in first-order logic“ (Part 7, Annex B).

NOTE – Because word processors have problems with logical symbols like ∃ and ∧ these symbols are presented as *exists* and *&* in the FOL listings in the template specifications.

Example [1]

Take this simple template:



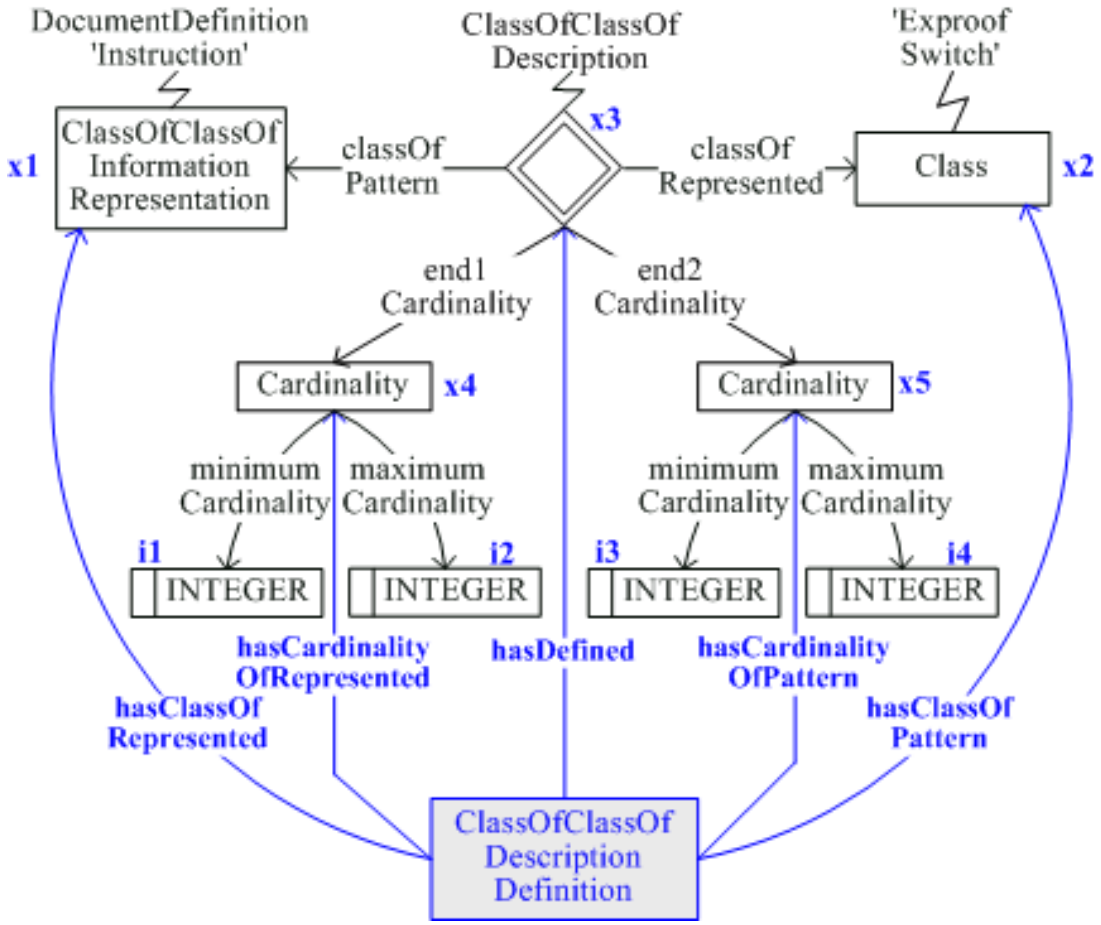
Example [2]

The **FOL listing** for that is:

```
ClassifiedDescriptionOfIndividual(x1, x2, x3) <->
PossibleIndividual(x1) &
ExpressString(x2) &
ClassOfClassOfDescription(x3) &
exists u(ClassOfDescriptionTriple(u, x2, x1) &
        ClassificationTemplate(u, x3)) .
```

The ClassOfClassOfDescription(x3) actually is a Triple that is defined in the RDL as “Instruction”:

Example [3]



resulting in the following FOL listing:

Example [4]

ClassOfClassOfDescriptionDefinition(x1, x2, x3, x4, x5, i1, i2, i3, i4) <->
Class(x1) &
ClassOfClassOfInformationRepresentation(x2) &
ClassOfClassOfDescriptionTriple(x3) &
hasEnd1Cardinality(x3, x4) &
hasEnd2Cardinality(x3, x5) &
exists u(Cardinality(x4) &
    INTEGER(i1) &
    INTEGER(i2) &
    hasMinimumCardinality(x4; i1) &
    hasMaximumCardinality(x4; i2)) &
exists v(Cardinality(x5) &
    INTEGER(i3) &
    INTEGER(i4) &
    hasMinimumCardinality(x5; i3) &
    hasMaximumCardinality(x5; i4)) .

Example [5]

Combining the two by '**expansion**' gives:

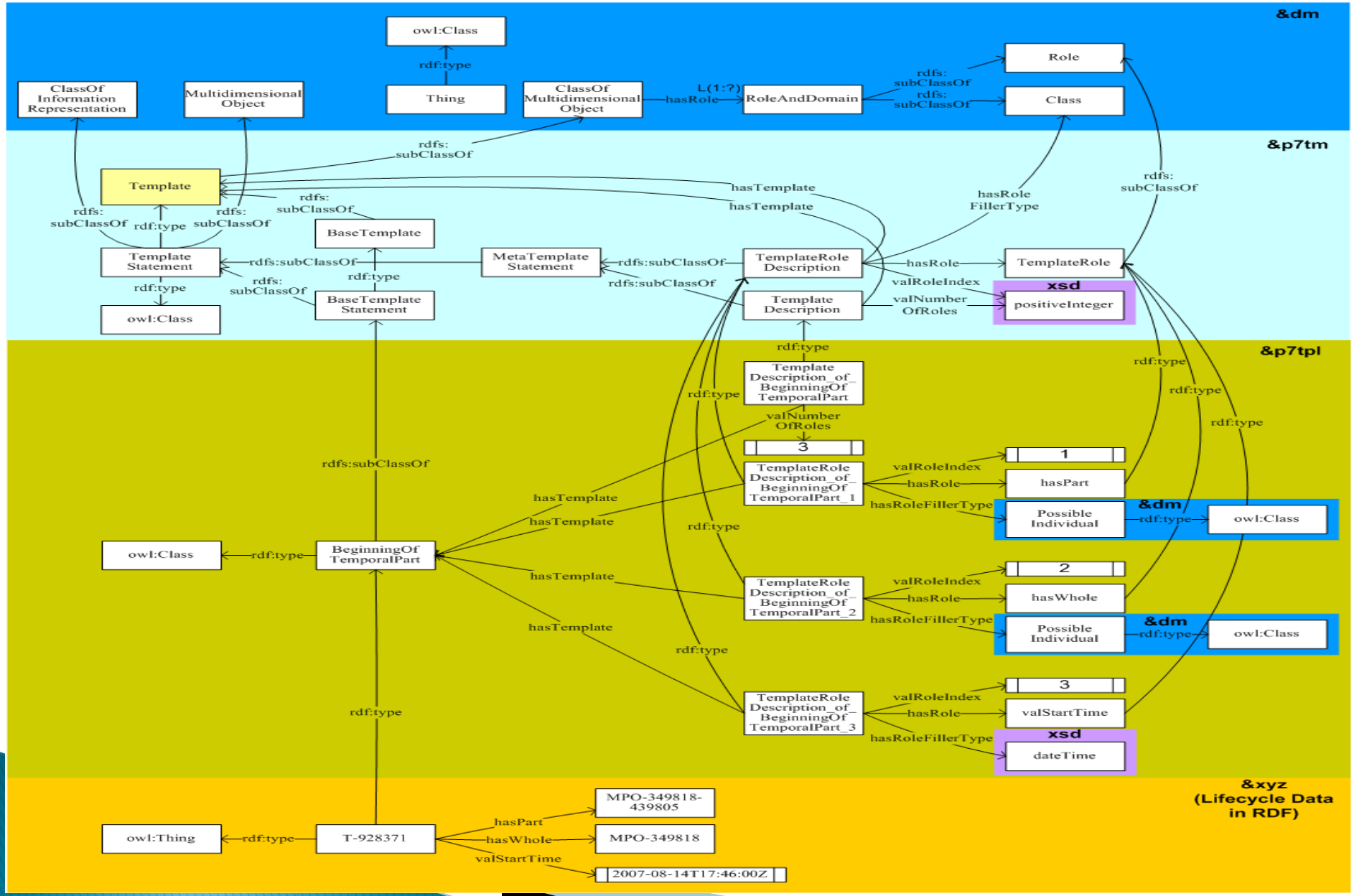
```
ClassifiedDescriptionOfIndividual(x1, x2, x3) <->
PossibleIndividual(x1) &
ExpressString(x2) &
exists u(ClassOfDescriptionTriple(u, x2, x1) &
        ClassificationTemplate(u, x3)) &
exists x3(ClassOfClassOfDescriptionTriple(x3, x6, x7) &
        Class(x6) &
        ClassOfClassOfInformationRepresentation(x7) &
        hasEnd1Cardinality(x3, x4) &
        hasEnd2Cardinality(x4, x5)) &
exists x4(Cardinality(x4) &
        INTEGER(i1) &
        INTEGER(i2) &
        hasMinimumCardinality(x4; i1) &
        hasMaximumCardinality(x4; i2)) &
exists x5(Cardinality(x5) &
        INTEGER(i3) &
        INTEGER(i4) &
        hasMinimumCardinality(x5; i3) &
        hasMaximumCardinality(x5; i4))) .
```

Example [6]

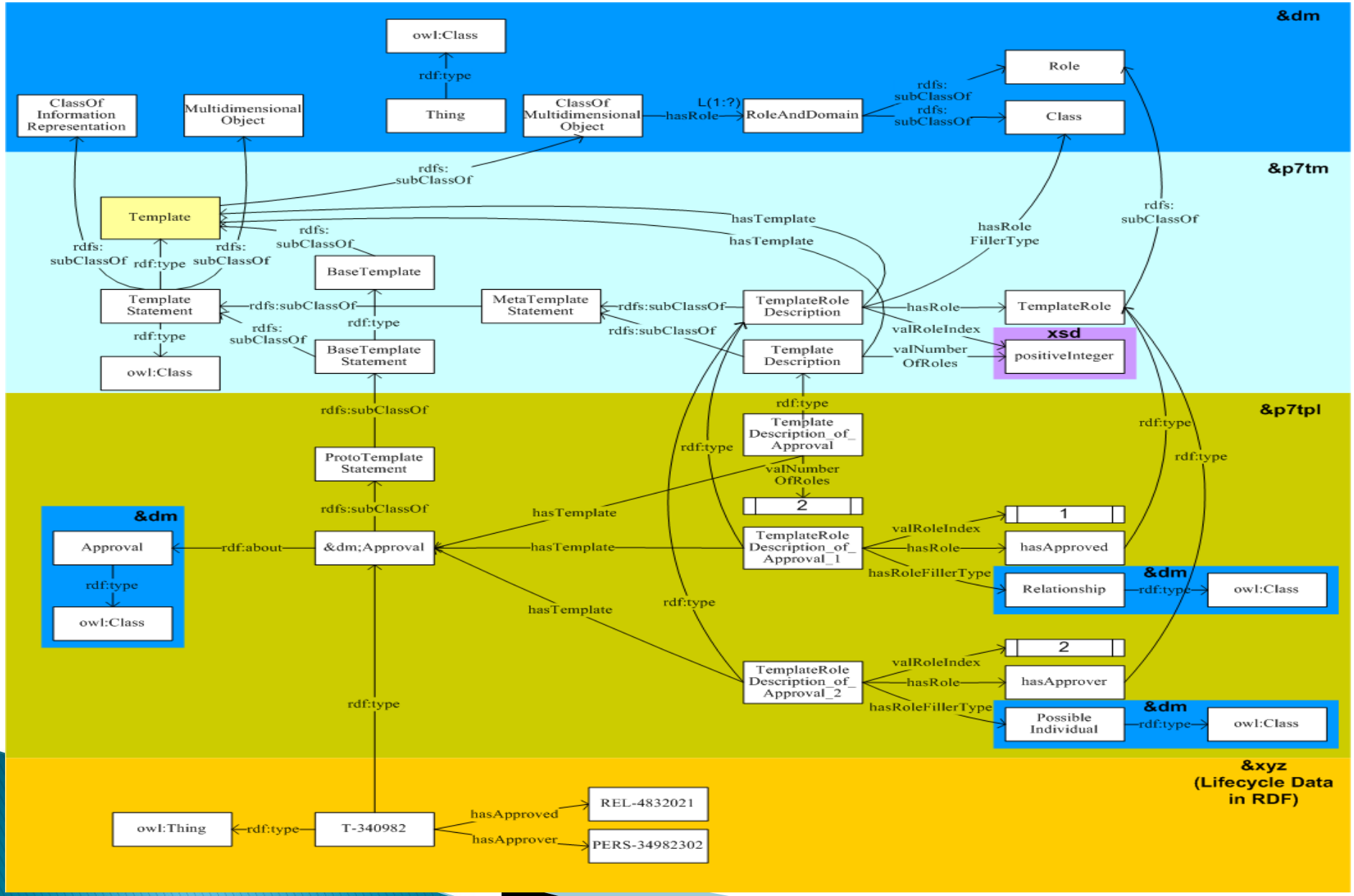
Finally the following can be found by means of **reasoning**:

- The PossibleIndividual(x1) shall be a member of the Class(x6)
- The ExpressString(x2) is a member of the ClassOfClassOfInformationRepresentation(x7)

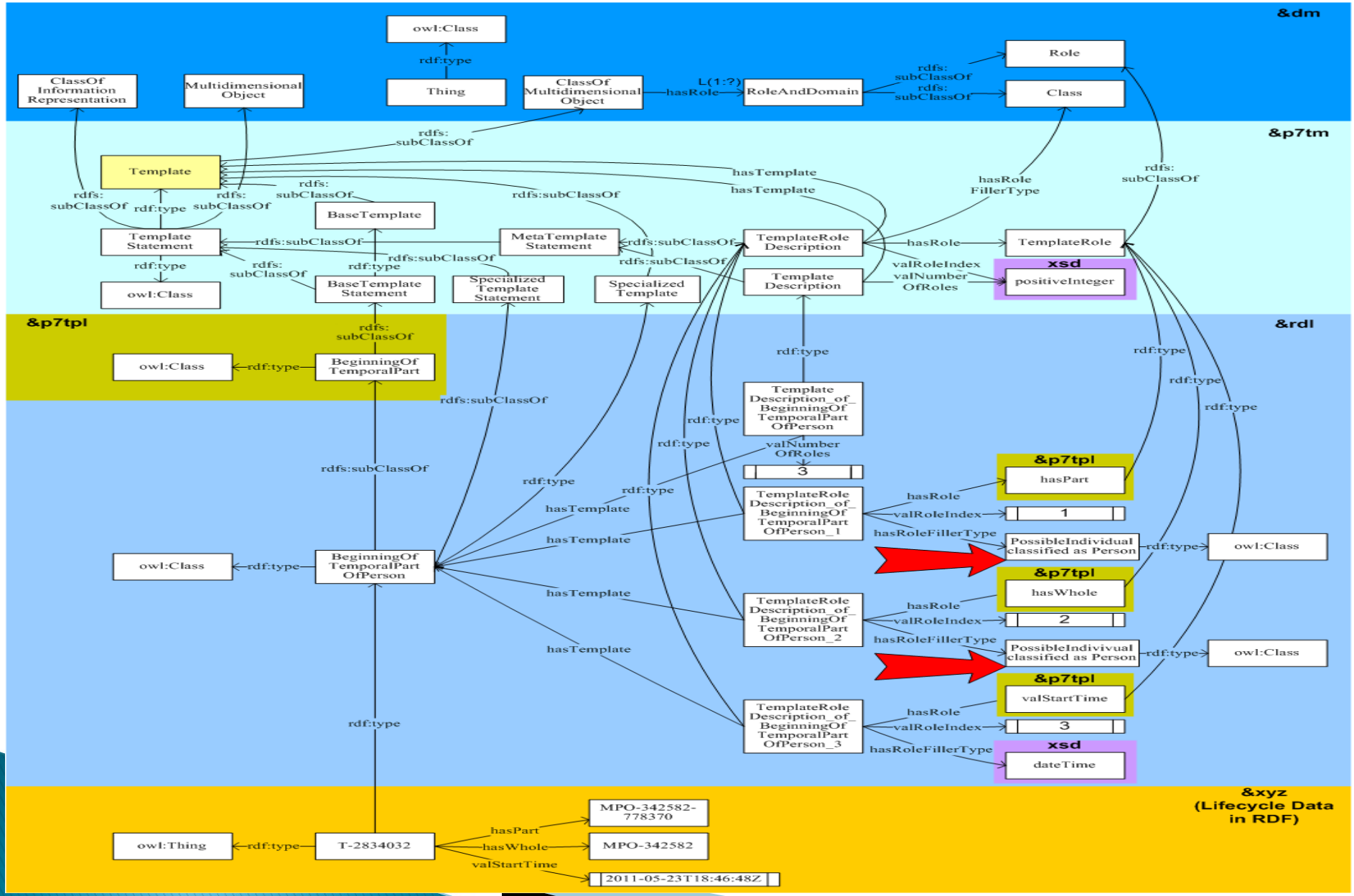
Base Templates



Proto Templates



Specialized Templates



Template definition in OWL [1]

The definition in OWL is, per template property:

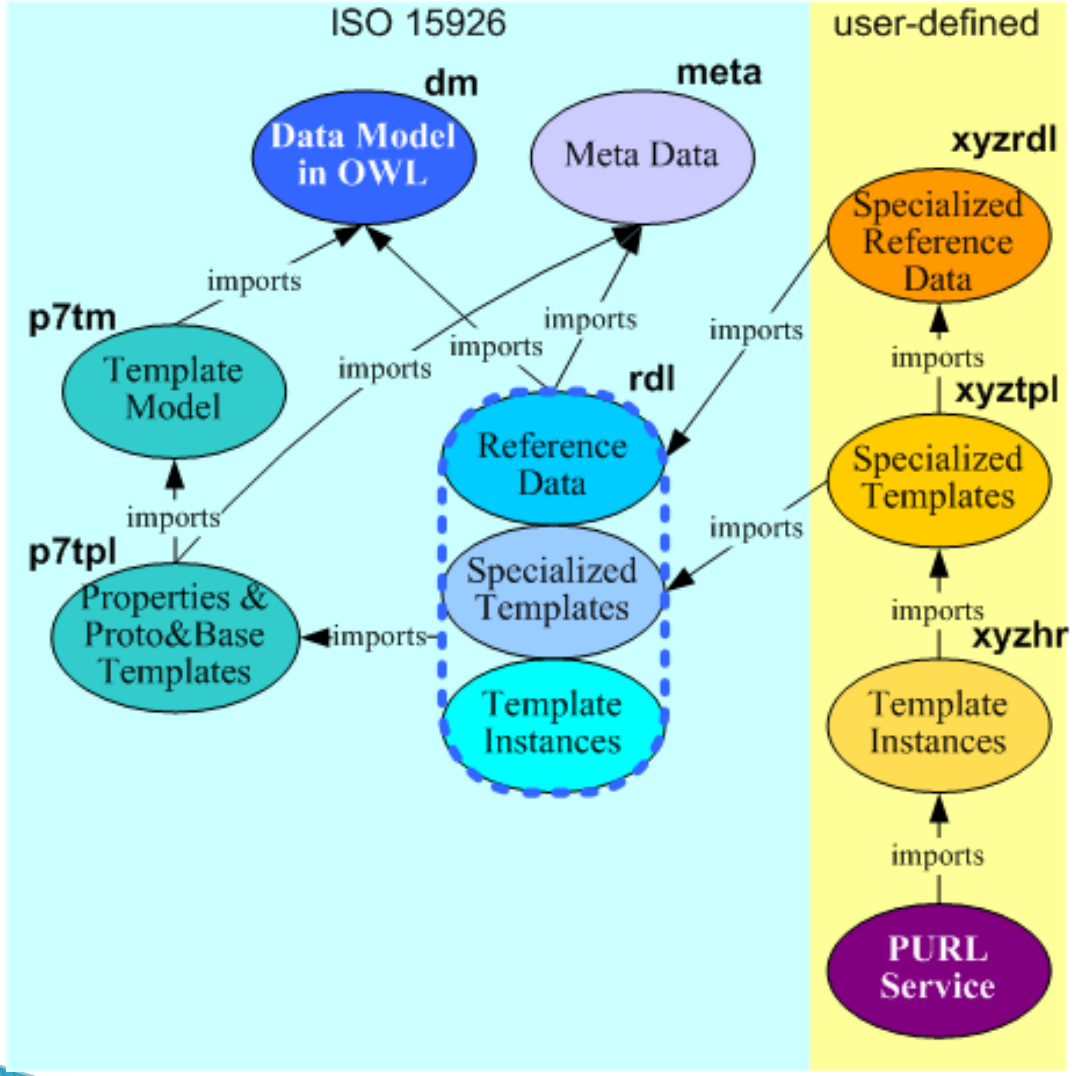
```
<owl:Class rdf:ID="BeginningOfTemporalPart">                    )
    <rdfs:label>BeginningOfTemporalPart</rdfs:label>            ) heading
    <rdfs:subClassOf rdf:resource="#BaseTemplateStatement"/>    )
    <rdfs:subClassOf>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hasPart"/>
            <owl:allValuesFrom rdf:resource="&dm;PossibleIndividual"/>
          </owl:Restriction>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hasPart"/>
            <owl:onClass rdf:resource="&dm;PossibleIndividual"/>
            <owl:qualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">
              1
            </owl:qualifiedCardinality>
          </owl:Restriction>
        </owl:intersectionOf>
      </owl:Class>
    </rdfs:subClassOf>
  etc
```

Template definition in OWL [2]

In case of specialization, the applicable property object is specialized:

```
<rdfs:subClassOf>
        <owl:Class>
           <owl:intersectionOf rdf:parseType="Collection">
              <owl:Restriction>
                 <owl:onProperty rdf:resource="#hasPart"/>
                 <owl:allValuesFrom rdf:resource="&rdl;PhysicalObject"/>
              </owl:Restriction>
              etc
```

ISO 15926 configuration



OWL Listings [1]

In the &dm ontology:

```
<owl:Class rdf:ID="Thing"/>
<owl:Class rdf:ID="PossibleIndividual">
      <rdfs:subClassOf rdf:resource="#Thing"/>
</owl:Class>
<owl:Class rdf:ID="PhysicalObject">
      <rdfs:subClassOf rdf:resource="#PossibleIndividual "/>
</owl:Class>
<owl:Class rdf:ID="AbstractObject">
      <rdfs:subClassOf rdf:resource="#Thing"/>
</owl:Class>
<owl:Class rdf:ID="Class">
      <rdfs:subClassOf rdf:resource="#AbstractObject"/>
</owl:Class>
<owl:Class rdf:ID="ClassOfIndividual">
      <rdfs:subClassOf rdf:resource="#Class"/>
</owl:Class>
<owl:Class rdf:ID="ClassOfArrangedIndividual">
      <rdfs:subClassOf rdf:resource="#ClassOfIndividual"/>
</owl:Class>
<owl:Class rdf:ID="ClassOfInanimatePhysicalObject"/>
      <rdfs:subClassOf rdf:resource="#ClassOfArrangedIndividual"/>
</owl:Class>
```

OWL Listings [2]

In the RDL:

```
<owl:Class rdf:ID="R35802804974"/> <!-- Inanimate Physical Object -->
   <rdf:type rdf:resource="&dm;ClassOfInanimatePhysicalObject"/>
</owl:Class>

<owl:Class rdf:ID="R20735180747"/> <-- Pump -->
   <rdf:type rdf:resource="&dm;ClassOfInanimatePhysicalObject"/>
   <rdfs:subClassOf rdf:resource="#R35802804974”/>
</owl:Class>

<owl:Class rdf:ID="R84144178538"/> <!-- Centrifugal Pump -->
   <rdf:type rdf:resource="&dm;ClassOfInanimatePhysicalObject"/>
   <rdfs:subClassOf rdf:resource="#R20735180747"/> <!-- Pump -->
</owl:Class>
```

OWL Listings [3]

In a PURL service:

```
<owl:Thing rdf:ID="PHO-5398439"/>
   <rdf:type rdf:resource="&dm;PhysicalObject"/> <!-- means: is an instance of -->
   <rdf:type rdf:resource="&dm;WholeLifeIndividual"/>
   <rdf:type rdf:resource="&dm;ArrangedIndividual"/>
   <rdf:type rdf:resource="&dm;ActualIndividual"/>
   <rdf:type rdf:resource=" &rdl;R35802804974"/>  <!-- means: is a member of Inanimate Physical
                                                        Object, the top of that class hierarchy →
</owl:Thing>
```

In Project data:

```
<owl:Thing rdf:ID="T-340981">
   <rdf:type rdf:resource="&p7tpl;BeginningOfIndividual"/>
   <p7tpl:hasIndividual rdf:resource="&purl;PHO-5398439"/>
   <p7tpl:valStartTime rdf:datatype="&xsd;dateTime">2011-08-14T17:46:00Z</p7tpl:valStartTime>
</owl:Thing>
<owl:Thing rdf:ID="T-340982">
   <rdf:type rdf:resource="&p7tpl;ClassificationOfTemporalPart"/>
   <p7tpl:hasTemporalWhole rdf:resource="&purl;PHO-5398439"/>
   <p7tpl:hasClassified rdf:resource="&purl;PHO-5398439_2011-08-14T17-46-00Z"/>
   <p7tpl:hasClassifier rdf:resource=" &rdl;R84144178538"/> <!-- CentrifugalPump -->
   <p7tpl:valStartTime rdf:datatype="&xsd;dateTime">2011-08-14T17:46:00Z</p7tpl:valStartTime>
</owl:Thing>
```

OWL Listings [4]

The ideas about **PURL** (Persistent URL) are still under debate.

Part 8 states that IDs shall not change ever, so also not at handover. The plant owner/operator should organize his IDs in a PURL mode, valid for the entire corporation. It prevents getting into trouble in cases where an asset is moved from one of his plants to another.

Interestingly enough this rules out the manufacturer, who is the creator of the asset. In the above scheme he cannot assign IDs in his PURL service to his creations.

We could give the manufacturer the IDs that he shall use when giving information about the goods that he will deliver (e.g. assigned serial number, heat treating, testing).

What about documents? [1]

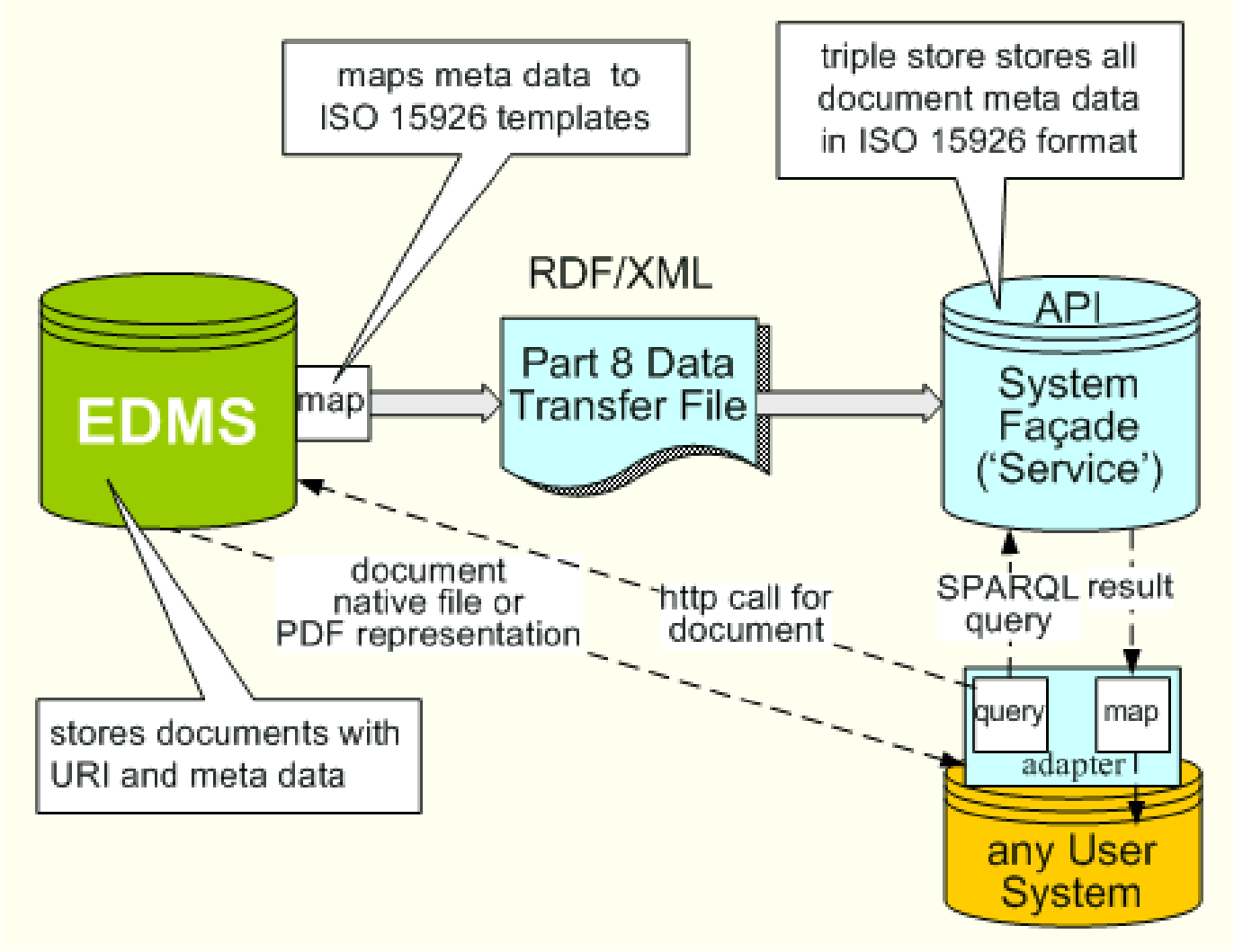
Most information is exchanged right now in the form of **documents**.

ISO 15926 will not change that, but will add:

- lifecycle information of the facility (e.g. :
  - maintenance activities and related costs)
  - measured values of pressure, flow rate, temperature, etc
- a standard way to get access to documents.

We will explore the latter somewhat.

What about documents? [2]



Functional Diagram for the integration of an EDMS

Documents [1]

Finally I present a graph about revising a document.

The model of a document has been, and still is, considered to be difficult to understand and counter-intuitive.

But once you know the basic principles, things fall in place.

Documents [2]

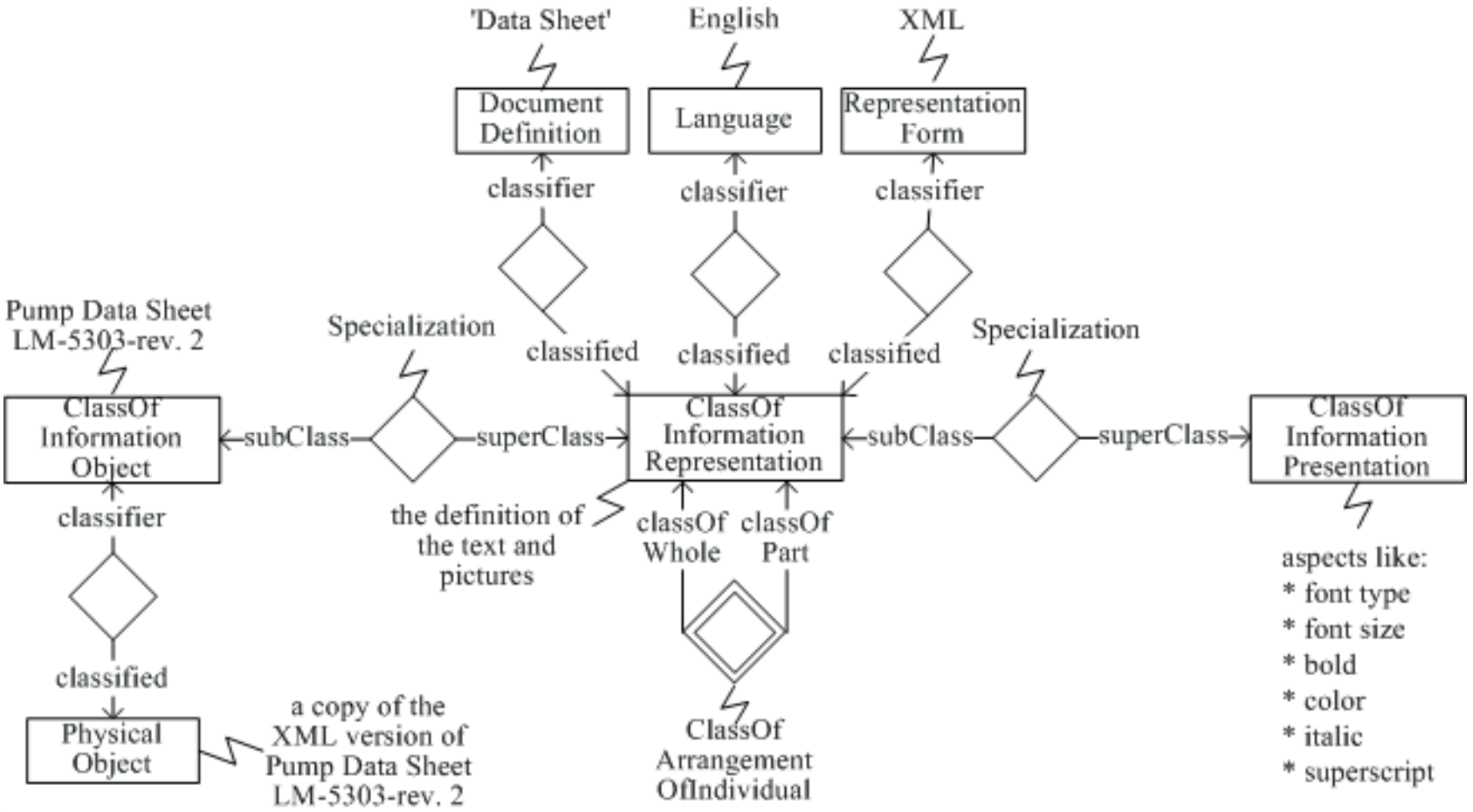
Important to know is that a string, number, binary, etc is a Class (in the ISO 15926 sense): ClassOfInformationRepresentation or a subtype thereof.

The physical text, number, picture, etc is a physical object that is a member of such a Class.

A paper document is a member of a ClassOfInformation**Object**.
This class is a subclass of the applicable ClassOfInformation**Representation**

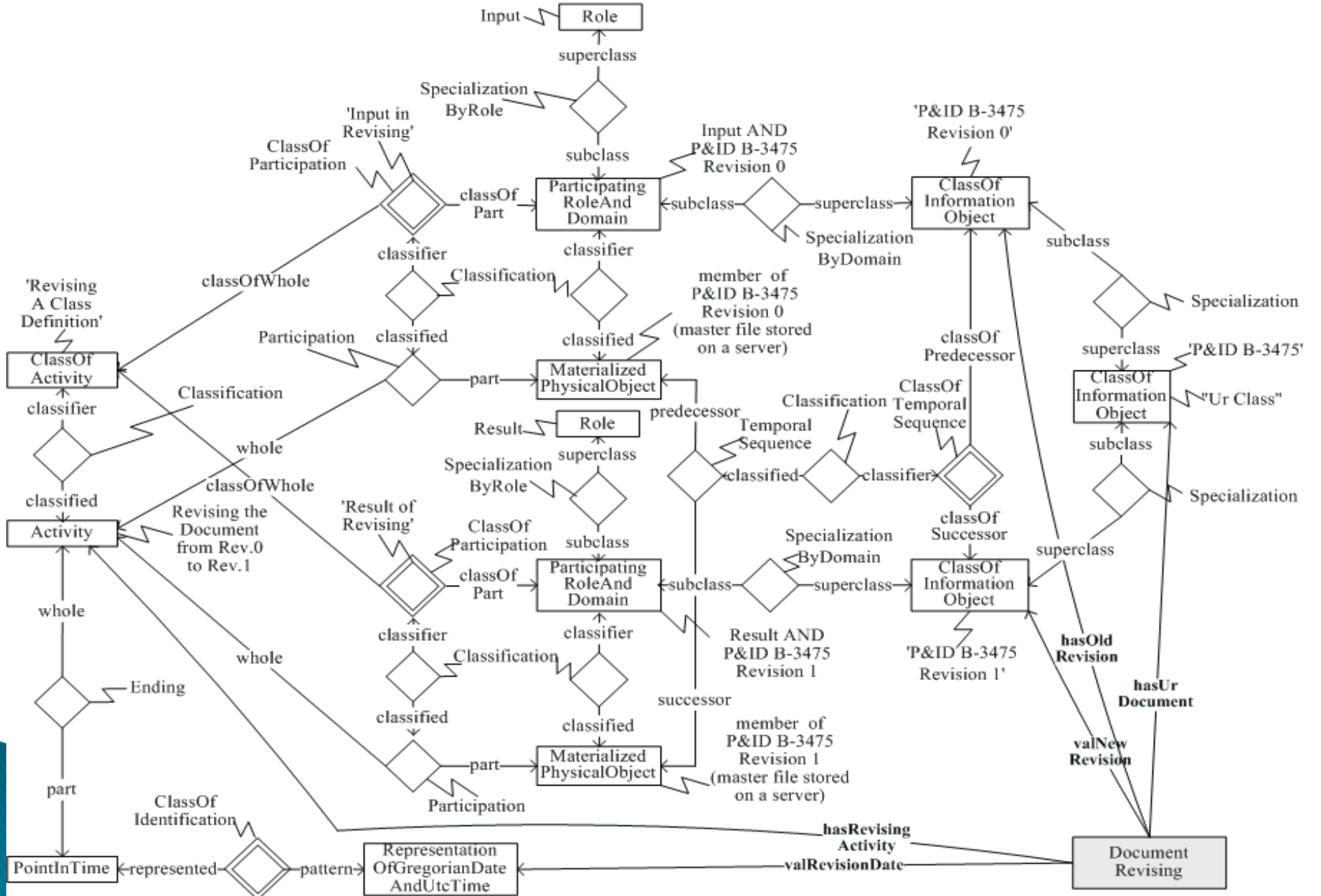
Documents [3]

So the basic model is:



Now you can understand a bigger model:

Documents [4]


Input
Role
superclass
Specialization ByRole
subclass
'Input in Revising'
ClassOf Participation
classOf Part
Participating RoleAnd Domain
Input AND P&ID B-3475 Revision 0
subclass
superclass
Specialization ByDomain
'P&ID B-3475 Revision 0'
ClassOf Information Object
classifier
classifier
classOfWhole
Classification
member of P&ID B-3475 Revision 0 (master file stored on a server)
'Revising A Class Definition'
ClassOf Activity
Participation
classified
classified
part
Materialized PhysicalObject
predecessor
Classification
ClassOf Temporal Sequence
classOf Predecessor
subclass
Specialization
superclass
'P&ID B-3475'
ClassOf Information Object
"Ur Class"
Result
Role
Temporal Sequence
classified
classifier
subclass
Specialization
classified
Activity
whole
classOfWhole
Revising the Document from Rev.0 to Rev.1
'Result of Revising'
ClassOf Participation
subclass
Specialization ByRole
superclass
classOf Successor
Specialization ByDomain
superclass
Participating RoleAnd Domain
subclass
superclass
ClassOf Information Object
classOf Part
classifier
classifier
Classification
Result AND P&ID B-3475 Revision 1
'P&ID B-3475 Revision 1'
hasOld Revision
hasUr Document
whole
whole
Ending
successor
classified
classified
member of P&ID B-3475 Revision 1 (master file stored on a server)
part
part
Materialized PhysicalObject
Participation
valNew Revision
ClassOf Identification
hasRevising Activity
valRevisionDate
PointInTime
represented
pattern
Representation OfGregorianDate AndUtcTime
Document Revising

Further information

Further information can be obtained from:

www.15926.org (modeling)

www.15926.info (topics)

www.infowebml.ws (overall concept)

www.iringug.org (implementation)

munduhwan@gmail.com (1st contact in Korea)

onno.paap@gmail.com (implementation)

hans.teijgeler@quicknet.nl (modeling)

++++++++++++++++++

Thank you!