ScadaOnWeb – Web Based Supervisory Control and Data Acquisition

Thomas Dreyer[1], David Leal[2], Andrea Schröder[3], and Michael Schwan[3]

[1] RACOS Technische Informationssysteme
Max-Stromeyer-Str. 172, 78467 Konstanz, Germany
tdreyer@racos.com
[2] CAESAR Systems Limited
29 Somertrees Avenue, SE12 0BS, London, United Kingdom
david.leal@caesarsystems.co.uk
[3] Forschungsgemeinschaft für Elektrische Anlagen und Stromwirtschaft (FGH)
Postfach 81 01 69, 68201 Mannheim, Germany
{schroeder.andrea,schwan}@fgh-ma.de

Abstract. W3C envisages the future of the web as a "Semantic Web" – an extended web of machine-understandable information and automated services going far beyond current capabilities. The EC funded project ScadaOnWeb contributes to the Semantic Web by defining a new standard giving semantics to structured numeric data. An ontology for SCADA (supervisory control and data acquisition) applications is made available, enhancing semantics for the engineering domain. This ontology is defined in RDF, RDF schema and OWL. It comprises a basic ontology providing a general model for SCADA data, an ontology for physical quantities, units and data quality, and specific ontologies for different SCADA applications. Large amounts of structured numeric data inherent to most SCADA applications can either be stored within the XML file or within an external binary file referenced from within the XML file. The semantics wrapped around the structured numeric data enables the understanding of the numbers.

1 Introduction

Typically, SCADA Systems are used in industry to monitor and control plant status and provide data logging facilities. SCADA systems are highly configurable, and usually interface to the plant via PLCs (programmable logic controller). But process monitoring and control is a key technology in many areas of life. The process can be an environmental process, such as flows in a hydraulic basin or the dispersion of pollutants, an industrial production process, a transmission and distribution process carried out by a utility, or a commercial process such as the operation of the energy market. Today, in each of these areas, one-off systems are created using proprietary formats and systems with their inherent problems and disadvantages, e.g. lacking flexibility and high costs.

The ScadaOnWeb technology, developed within the EC funded project ScadaOnWeb [1] provides a standard platform that addresses this wider understanding of process monitoring and control applications that are distributed over

D. Fensel et al. (Eds.): ISWC 2003, LNCS 2870, pp. 788–801, 2003.

the web and make proprietary systems and formats superfluous. It extends current technologies by defining a standard for meta-data that gives semantics to engineering data and references standards for units of measure and ontologies for properties in different engineering domains. Standard transaction templates are developed to support process monitoring and control over the web and means for access control appropriate to process monitoring and control applications are provided [2].

In order to validate the ScadaOnWeb technology, the following prototype applications from different areas of process monitoring and control are developed:

- Flood warning system based upon remote sensors,
- Balance group energy management,
- Flexible metering of domestic and small industrial consumers,
- Condition based maintenance of remote equipment,
- Control of distributed wind and hydro electricity generation.

These are examples of a wide range of possible future applications in the area of engineering.

2 SCADA Web Data Type

Process monitoring and control can involve large amounts of numeric data with a regular structure. In order to process this type of data efficiently, it may be to the best advantage to maintain the structure as well as to hold the numbers in binary format, so that storage is minimized and computation is efficient. But information exchange also requires precision about the semantics of the data.

E.g. in a three-dimensional data-field data of many sensors can be stored for various time-steps. The binary file holds the measured value for time-step 10.000 and for sensor 1.000 in position (10.000, 1.000) of an array. It is necessary to know the identity of sensor 1.000 and where it is placed in the system, that time-step 10.000 was at 10:31:15 on 2001-12-14, and that the measured value is a flow expressed in $m^3 sec^{-1}$.

Using XML (Extensible Markup Language) [3] in conjunction with RDF (Resource Description Framework) [4] and RDFS provides the semantics, but a traditional XML approach may turn the data into millions of separate XML elements. Therefore, an approach was chosen which enables the information to be recorded and, where suitable, to be exchanged as a pair of files: An XML file which describes the semantics and a HDF5 (Hierarchical Data Format) [5] file with structured numeric data. HDF5 is a compact machine-independent binary file format developed by NCSA (National Center for Supercomputing Applications) and is used by NASA (North American Space Agency) and ESA (European Space Agency) in different applications.

This approach provides maximum flexibility: On server side, a ScadaOnWeb application with a high amount of recorded data may store it in a HDF5 file and keep the description of its semantic in a separate XML file or data-base referencing its respective HDF5 file (see Fig. 3). A similar application with a smaller amount of data recorded may choose to keep data and semantics in one and the same XML file. The same options hold if the server is queried by a client application: Depending on

average amounts of data resulting from typical queries, the data may be exchanged in a single file or as a pair of files separating data and respective semantics. Fig. 1 symbolizes the case where large amounts of data are kept separate from its semantics in a structured HDF5 data block.

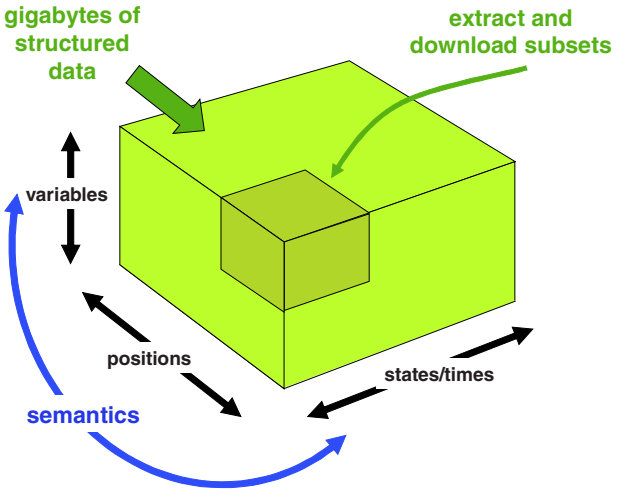

Fig. 1. SCADA web data type

The serialization in Fig. 2 is an example of a sensor 'sensor_xyz' making 96 measurements during a day. Each measurement is of the average electric power during the preceding 15 minutes. The distribution 'sensor_xyz_on_2002-05-02' is defined by the following equality:

$$\text{Distribution} = \text{Parameterization} \cdot \text{Table} \cdot (\text{Scale})^{-1} \quad (1)$$

Each of "Distribution", "Parameterization", "Table" and "Scale" is a property. Hence the "Distribution" is described by a composition of the other three, as shown in the following serialization.

```
<?xml version="1.0"?>
<rdf:RDF
  <!-- Definition of namespaces and assignment of XML-Schema-->
      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:po="http://www.scadaonweb.com/publications/ontologies/physicalObject.owl#"
      xmlns:ppqs="http://www.scadaonweb.com/publications/ontologies/physicalPropertyQuantityAndS
  cale.owl#"
      xmlns:pqs="http://www.scadaonweb.com/publications/ontologies/physicalQuantitySpaces.owl#"
      xmlns:snt="http://www.scadaonweb.com/publciations/ontologies/structureNumberAndText.owl#"
      xmlns:time="http://www.scadaonweb.com/publications/ontologies/time.owl#"
      xmlns:pods="http://www.scadaonweb.com/publications/ontologies/physicalObjectDecomposition
  AndSampling.owl#"
      xmlns:dist="http://www.scadaonweb.com/publications/ontologies/distributionAndDescription.owl
  #">
```

```
<po:Product rdf:ID="sensor_xyz"> </po:Product>

<po:ProductLifeSegment rdf:ID="sensor_xyz_on_2002-05-02">
 <po:lifeSegmentOfProduct rdf:resource="#sensor_xyz"/>
 <time:timeIntervalOfProductLifeSegment>
  <time:TimeInterval>
   <ppqs:closedLowerBoundOf1dPhysicalQuantitySpace>
    <time:Time>
     <time:utc>
      <time:UtcRepresentation>
       <snt:hasTerms rdf:parseType="Collection">
        <snt:Integer><snt:decimal>2002</snt:decimal></snt:Integer>
        <snt:Integer><snt:decimal>5</snt:decimal></snt:Integer>
        <snt:Integer><snt:decimal>2</snt:decimal></snt:Integer>
        <snt:Integer><snt:decimal>0</snt:decimal></snt:Integer>
        <snt:Integer><snt:decimal>0</snt:decimal></snt:Integer>
        <snt:Real><snt:decimal>0.0</snt:decimal></snt:Real>
        <snt:Boolean rdf:about="&snt;False"/>
       </snt:hasTerms>
      </time:UtcRepresentation>
     </time:utc>
    </time:Time>
   </ppqs:closedLowerBoundOf1dPhysicalQuantitySpace>
   <time:durationOfTimeInterval>
    <pqs:TimeDuration>
     <time:hour>
      <snt:Real>
       <snt:decimal>24.0</snt:decimal>
      </snt:Real>
     </time:hour>
    </pqs:TimeDuration>
   </time:durationOfTimeInterval>
  </time:TimeInterval>
 </time:timeIntervalOfProductLifeSegment>
</po:ProductLifeSegment>

<po:ProductLifeSegmentSet rdf:ID=" sensor_xyz_on_2002-05-02_at_15_minutes">
 <rdf:type rdf_resource="&pods;UniformlySizedProductLifeSegmentSet"/>
 <pods:lifeSegmentTemporalDecompositionOfProductLifeSegment
   rdf:resource="#sensor_xyz_on_2002-05-02"/>
 <pods:durationOfUniformlySizedSegments>
  <pqs:TimeDuration>
   <time:minute>
    <snt:Real><snt:decimal>15.0</snt:decimal></snt:Real>
   </time:minute>
  </pqs:TimeDuration>
 </pods:durationOfUniformlySizedSegments>
</po:ProductLifeSegmentSet>

<dist:PhysicalDistribution>
 <rdfs:domain rdf:resource="sensor_xyz_on_2002-05-02_at_15_minutes"/>
 <rdfs:range rdf:resource="&pqs;Power"/>
 <rdfs:subPropertyOf rdf:resource="&myPp;timeAveragePower"/>
 <dist:factorsInto>
  <dist:PhysicalDistributionDescriptionVector>
   <snt:hasTerms rdf:parseType="Collection">
    </dist:PhysicalParameterization>
     <rdf:type
       rdf:resource="&dist;DiscreteForwardTimeParameterisation"/>
```

```
    </dist:PhysicalParameterization>
    <snt:SymbolTable>
     <rdfs:domain>
      <snt:FiniteIntegerInterval>
       <snt:lowerBoundOfIntegerInterval>
        <snt:Integer><snt:decimal>1</snt:decimal></snt:Integer>
       </snt:lowerBoundOfIntegerInterval>
       <snt:upperBoundOfIntegerInterval>
        <snt:Integer><snt:decimal>96</snt:decimal></snt:Integer>
       </snt:upperBoundOfIntegerInterval>
      </snt:FiniteIntegerInterval>
     </rdfs:domain>
     </rdfs:range rdf:resource="&snt;Real"/>
     <snt:hasLinearizedSymbolTableTerms>
      <snt:SymbolVector>
       <snt:hasTerms rdf:parseType="Collection"/>
        <snt:Real><snt:decimal>2.4</snt:decimal></snt:Real>
        <snt:Real><snt:decimal>2.3</snt:decimal></snt:Real>
          ...
       </snt:hasTerms>
      </snt:SymbolVector>
     </snt:hasLinearizedSymbolTableTerms>
    </snt:SymbolTable>
    <dist:InverseCompoundScale>
     <owl:inverseOf>
      <ppqs:CompoundScale>
       <snt:compoundPropertyCompoundDomainComponents>
        <snt:PropertyVector>
         <snt:hasTerms rdf:parseType="Collection">
          <ppqs:Scale rdf:about="&myScales;kilowatt"/>
         </snt:hasTerms>
        </snt:PropertyVector>
       </snt:compoundPropertyCompoundDomainComponents>
      </ppqs:CompoundScale>
     </owl:InverseOf>
    </dist:InverseCompoundScale>
   </snt:hasTerms>
  </dist:PhysicalDistributionDescriptionVector>
 </dist:factorsInto>
</dist:PhysicalDistribution>

<rdf:RDF>
```

Fig. 2. XML-serialization for a temperature distribution

The prefixes of the tag names refer to namespaces in which the ontology is defined. For example the prefix “dist” refers to a file called http://www.scadaonweb.com/ publications/ontologies/distributionAndDescription.owl containing a part of the ontology that is relevant for distributions.

The table is defined explicitly in the XML document starting with the tag <snt:SymbolTable>. Alternatively if the table were much larger, a reference could be made to a table that is defined in a binary file, as shown in Fig. 3.

```
<snt:SymbolTable
  rdf:resource="http://www.company.eu/myFile.hdf5#myDataSet"/>
```

Fig. 3. Reference of data kept within an HDF5 file

In following chapters a basic understanding of the ontology referenced in the serialization of Fig. 2 is given. The main focus is the ontology of physical quantities, units and data quality.

3 Ontology

The ontology [6] underlying the example given in Fig. 2 serves as reference data dictionary for the SCADA web data type and is based on existing standards such as ISO 10303 (Industrial automation systems and integration - Product data representation and exchange) and ISO 15926 (Lifecycle data for process plant, including oil and gas production). It is defined in XML and is based on RDF, OWL (Web Ontology Language) [7] and MathML [8].

XML is unequalled as an exchange format on the Web but by itself, it doesn't provide what is needed for building the semantic web. Compared to RDF one of XML's drawbacks is that there are a lot of possible serializations for a statement. E.g. the statement "Otto Maier is the author of the web-site http://www.ottomaier.com" can be serialized as:

1. <Author>
   <uri> http://www.ottomaier.com</uri>
   <name>Otto Maier</name>
   </Author>

2. <Document uri="http://www.ottomaier.com"
   <Author>Otto Maier</Author>
   </Document>

3. <Document uri=http://www.ottomaier.com Author="Otto Maier"/>

For a human being all these serializations mean the same but an application interprets them differently because a parser generates different XML-trees. RDF in contrast has exactly one serialization matching a statement. Therefore RDF in combination with OWL that is layered on top of RDF is used to define an ontology for web-based SCADA applications.

The basic ontology can be referenced by specific ontologies, e.g. an ontology for an application in the field of condition based maintenance. This is shown in Fig. 4, Fig. 5, Fig. 6 and Fig. 7.

The basic ontology defines three principal types of physical objects: Product, Product life segment and Product at instant.

Product describes the complete lifetime of a physical object, e.g. the generator with the serial number 98/1234 is a product. Hence the class "Generator" is a specialization or subclass of "Product" (Fig. 4).

```
<rdfs:Class rdf:ID="Generator">
  <rdfs:subClassOf rdf:resource="&po;Product"/>
  <rdfs:comment>
     A product for producing electrical energy.
  </rdfs:comment>
</rdfs:Class>
```

Fig. 4. Definition of the class "Generator" as a subclass of "Product"

Product life segment describes a segment and an activity, respectively, in the life of a product. For example, the trip of the generator between 2002-02-28 T10:00:00 and 2002-02-28 T10:00:01 is a product life segment. Hence the class “GeneratorTrip” is a specialization or subclass of “ProductLifeSegment” (Fig. 5).

```
<rdfs:Class rdf:ID="GeneratorTrip">
  <rdfs:subClassOf rdf:resource="&po;ProductLifeSegment"/>
  <rdfs:comment>
    An activity that is a transition from a generating state to a non-generating state.
  </rdfs:comment>
</rdfs:Class>
```

Fig. 5. Definition of the class “GeneratorTrip” as a subclass of “ProductLifeSegment”

Product at instant describes an instant and state, respectively, in the lifetime of a product. The generation state of the generator at 2002-02-28 T10:00:00 is a product at instant. Therefore the class 'Generating' is a specialization or subclass of “ProductAtInstant” (Fig. 6).

```
<rdfs:Class rdf:ID="Generating">
  <rdfs:subClassOf rdf:resource="&po;ProductAtInstant"/>
  <rdfs:comment>A state that causes a flow of energy into an electrical network.</rdfs:comment>
</rdfs:Class>
```

Fig. 6. Definition of the class “Generating” as subclass of “Product at instant”

All classes are a specialization of these principal concepts and can be defined as their subclasses. According to these three principal physical objects, the following principal properties are defined: lifeSegmentOfProduct (assignment of a product life segment to a product), instantOfProduct (assignment of a state to a product), instanceOfLifeSegment (assignment of a state to a product life segment).

The example given in Fig. 4 is extended by defining that an activity of type 'GeneratorTrip' is only performed by a product of type “Generator”. This is accomplished by formulating a restriction on the property “lifeSegmentOfProduct” (Fig. 7).

```
<rdfs:Class rdf:ID="GeneratorTrip">
  <rdfs:subClassOf rdf:resource="&po;ProductLifeSegment"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&po;#lifeSegmentOfProduct"/>
      <owl:toClass rdf:resource="#Generator"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:comment>
    An activity that is a transition from a generating state to a non-generating state.
  </rdfs:comment>
</rdfs:Class>
```

Fig. 7. Definition of a property restriction for the class “GeneratorTrip”

Except for the above described basic ontology for SCADA data and application specific SCADA ontologies, an ontology for physical quantity spaces and units is provided and can be referenced from XML files containing the data, e.g. ”Power” and

"kilowatt" as shown in Fig. 2. The definitions necessary for this ontology are described in the following paragraphs.

4 Physical Quantities and Properties

The value (e.g. 10 kg) of a physical quantity (e.g. mass) is a property of a physical object (e.g. transformer), that can be observed or measured. Additionally to the value (e.g. 10) the corresponding unit has to be given (e.g. kilogram). The RDF graph (Fig. 8) shows the definition of a physical quantity value and corresponds to the RDF serialization given in Fig. 9. An approach derived from MathML (rather than the approach used in XML Schema) has been used to specify the numeric value, because many physical properties are identified by complicated numeric values such as vectors or matrices.

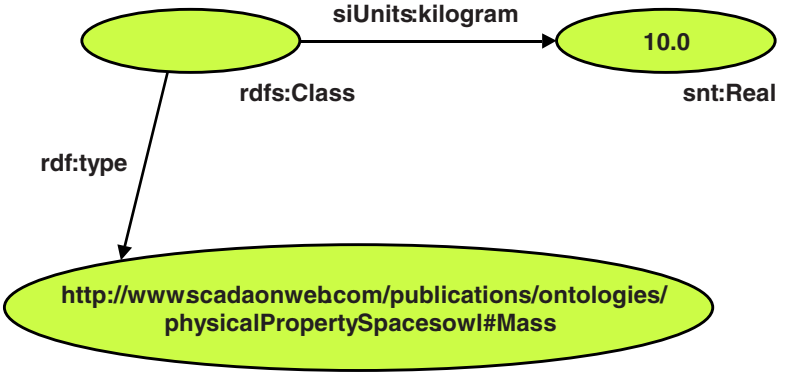

Fig. 8. Definition of a physical quantity value (graph)

```
<rdf:RDF
 ... (Definition of namespaces)>
  <rdfs:Class>
    <rdf:type rdf:resource="&pqs;Mass"/>
    <siUnits:kilogram>
      <snt:Real>
        <snt:decimal>10.0</snt:decimal>
      </snt:Real>
    </siUnits:kilogram>
  </rdfs:Class>
</rdf:RDF>
```

Fig. 9. Definition of a physical quantity value (serialization)

A scale for a physical quantity space is a property of the physical quantity space. The resulting RDF-graph shown in Fig. 10 corresponds to the RDF serialization given in Fig. 11.

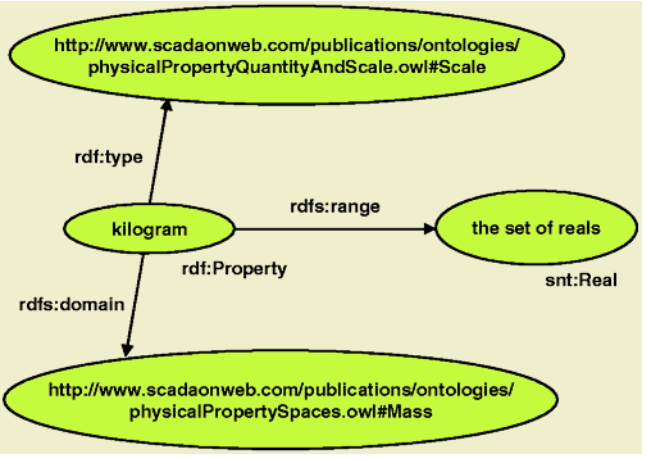

Fig. 10. Definition of a scale for a physical quantity space (graph)

```
<rdf:RDF
 ... (Definition of namespaces)>
  <rdf:Property rdf:ID="kilogram">
    <rdf:type rdf:resource="&ppqs;Scale"/>
    <rdfs:comment>kilogram scale defined by ISO 1000:1992</rdfs:comment>
    <rdfs:domain rdf:resource="&pqs;#Mass"/>
    <rdfs:range rdf:resource="&snt;Real"/>
  </rdf:Property>
</rdf:RDF>
```

Fig. 11. Definition of a scale for a physical quantity space (serialization)

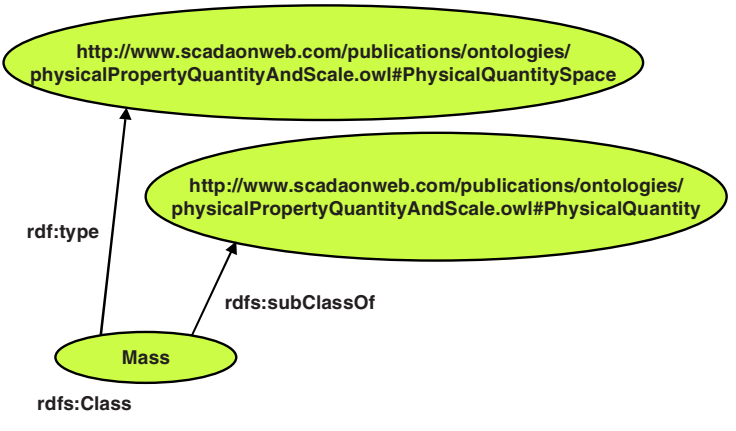

Fig. 12. Definition of a physical quantity space (graph)

A physical quantity space is a class containing all physical quantity values. E.g. “mass” is a physical quantity space and “10 kg” belongs to this class. The RDF graph in Fig. 12 defines a physical quantity space and corresponds to the RDF-serialization in Fig. 13.

```
<rdf:RDF
  ... (Definition of namespaces)>
  <rdfs:Class rdf:ID="Mass">
    <rdf:type rdf:resource="&ppqs;PhysicalQuantitySpace"/>
    <rdfs:subClassOf rdf:resource="&ppqs;PhysicalQuantity"/>
    <rdfs:comment>mass</rdfs:comment>
  </rdfs:Class>
</rdf:RDF>
```

Fig. 13. Definition of a physical quantity space (serialization)

Under the general term "possession of a physical property" the relationship between a physical object and a physical quantity is considered. A physical object is member of the class physical quantity space. E.g. the physical object "transformer at 10:30 h on 15.03.2002" is member of the class "10 kg". Fig. 14 shows an RDF-graph for a simple relationship for the physical property "massWhenEmpty". The corresponding RDF-serialization is given in Fig. 15.

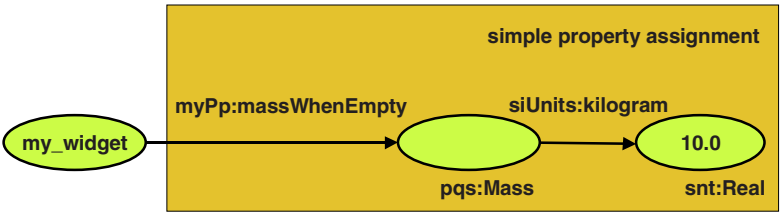

Fig. 14. Definition of a physical property (graph)

```
<rdf:RDF
 ... (Definition of namespaces)>
  <myPo:Widget rdf:ID="my_widget">
    <myPp:massWhenEmpty>
      <pqs:Mass>
        <siUnits:kilogram>
          <snt:Real>
            <snt:decimal>10.0</snt:decimal>
          <snt:Real>
        </siUnits:kilogram>
      </pqs:Mass>
    </myPp:massWhenEmpty>
  </myPo:Widget>
</rdf:RDF>
```

Fig. 15. Definition of a physical property (serialization)

5 Derived Physical Quantities, Scales, and Properties

Additionally to fundamental physical quantity spaces (e.g. mass, time, length), base units (e.g. kg, s, m) and simple physical properties (e.g. mass when empty), derived physical quantity spaces (e.g. velocity, acceleration, force) and their corresponding units (e.g. m/s, m/s^2, kg·m/s^2) have been defined within the ScadaOnWeb ontology. Fig. 16 shows an example for a derived physical quantity.

```
<rdf:RDF
 ... (Definition of namespaces)>
  <ppqs:PhysicalQuantitySpace rdf:ID="Force">
    <rdf:type rdf:resource="#DerivedPhysicalQuantitySpace"/>
    <rdfs:label xml:lang="en">force</rdfs:label>
    <pqs:hasPhysicalDimension rdf:parseType="Literal">
      <mathml:apply>
        <mathml:times/>
        <ppqs:PhysicalQuantitySpace rdf:about="#Mass"/>>
        <ppqs:PhysicalQuantitySpace rdf:about="#Acceleration"/>
      </mathml:apply>
    </pqs:hasPhysicalDimension>
  </rdfs:Class>
</rdf:RDF>
```

Fig. 16. Definition of a derived physical quantity space (serialization)

6 Compound Physical Quantities, Scale, and Properties

A compound physical quantity value is a vector of two or more physical quantities (e.g. [2002-05-07 10:30 UTC, 20 degrees Kelvin]). Analogous to that, a compound physical quantity space and scale, respectively, can be defined by a vector of two or more physical quantity spaces (e.g. (instant in time, temperature)) and scales (e.g. [UTC, Kelvin]), respectively.

The identification of a compound physical quantity value is done with respect to a compound scale by a vector of numeric or text values (e.g. [(2002, 5, 7,10, 30), 20]). The serializations given in Fig. 17 and Fig. 18 show the definition of the compound physical quantity "InstantInTimeAndTemperature" and the compound scale "utcAndKelvin".

```
<rdf:RDF
 ... (Definition of namespaces)>
  <ppqs:PhysicalQuantityVectorSpace
      rdf:ID="InstantInTimeAndTemperature">
    <rdfs:label xml:lang="en">instant in time and temperature</rdfs:label>
    <snt:cartesianProductOf>
      <ppqs:PhysicalQuantitySpaceVector>
        <snt:hasTerms rdf:parseType="Collection">
        <ppqs:PhysicalQuantitySpace rdf:about="&time;#Time"/>
        <ppqs:PhysicalQuantitySpace rdf:about="&pqs;#Temperature"/>
        </snt:hasTerms>
      </ppqs:PhysicalQuantitySpaceVector>
    </snt:cartesianProductOf>
  </ppqs:PhysicalQuantityVectorSpace>
</rdf:RDF>
```

Fig. 17. Definition of a compound physical quantity

A compound physical property has a single physical object as its domain, and evaluates to give a compound physical quantity. As shown in Fig. 19 it can be defined by a vector of two or more physical properties (e.g. (time of instant, measured temperature)).

```
<rdf:RDF
 ... (Definition of namespaces)>
 <ppqs:CompoundScale rdf:ID="utcAndKelvin">
  <ppqs:unitSymbol>UTC, K</ppqs:unitSymbol>
  <rdfs:label xml:lang="en">UTC and Kelvin</rdfs:label>
  <rdfs:domain rdf:resource="#InstantInTimeAndTemperature"/>
  <rdfs:range>
   <snt:SymbolVectorSpace>
    <snt:cartesianProductOf>
     <snt:SymbolVector>
      <snt:hasTerms rdf:parseType="Collection">
       <snt:SymbolVector rdf:about="&time;#UtcRepresentation"/>
       <snt:MathsSpace rdf:about="&snt;Real"/>
      </snt:hasTerms>
     </snt:SymbolVector>
    </snt:cartesianProductOf>
   </snt:SymbolVectorSpace>
  </rdfs:range>
  <snt:compoundPropertyCompoundDomainComponents">
   <snt:PropertyVector>
    <snt:hasTerms rdf:parseType="Collection">
     <ppqs:Scale rdf:about="&time;utc"/>
     <ppqs:Scale rdf:about="&siUnits;kelvin"/>
    </snt:hasTerms>
   </snt:PropertyVector>
  </snt:compoundPropertyCompoundDomainComponents>
 </ppqs:CompoundScale>
</rdf:RDF>
```

Fig. 18. Definition of a compound scale

```
<rdf:RDF
 ... (Definition of namespaces)>
 <snt:CompoundPropertySingleDomain rdf:ID="timeOfInstantAndMeasuredTemp">
  <rdfs:label xml:lang="en">time of instant and measured temperature</rdfs:label>
  <rdfs:domain rdf:resource="&po;#ProductAtInstant"/>
  <rdfs:range rdf:resource="InstantInTimeAndTemperature"/>
   <snt:compoundPropertySingleDomainComponents>
    <snt:PropertyVector>
     <snt:hasTerms rdf:parseType="Collection"/>
      <rdf:Property rdf:about="&time;timeOfProductAtInstant"/>
      <rdf:Property rdf:about=&myProperties;measuredTemperature"/>
     </snt:hasTerms>
    </snt:PropertyVector>
   </snt:compoundPropertySingleDomainComponents>
 </snt:CompondPropertySingleDomain>
</rdf:RDF>
```

Fig. 19. Definition of a compound physical property

7 Data Quality

Except for the measured data value and the time of measurement the quality of the measured data is of relevance and must be considered in an ontology for the engineering domain. The ScadaOnWeb approach for defining quality is based on the

common data attribute type “Quality” of IEC 61850-7-3 (Communication Networks and Systems in Substations, Part 7-3: Basic communication structure for substations and feeder equipment - Common data classes) which contains information on the validity, the detail quality (e.g. overflow, old data, etc.) and the source giving information related to the origin of a value. An XML/RDF file has been created for data quality and is part of the ScadaOnWeb ontology. It can therefore be referenced from within the definition of compound properties (e.g.: [time of instant, measured temperature, quality]).

8 Conclusion

Because of the lack of W3C standards for units, the Web has been little used for engineering data up to now. With the ontology for engineering data defined within the ScadaOnWeb project, containing the definitions for physical quantities and scales described above, an ontology for the engineering domain is finally available. These definitions, comprising a standardized XML Schema for physical quantities, units, data quality and co-ordinate systems, will be input to standardization bodies such as W3C and ISO. They form a basis for the handling of engineering data on the Web which applications in the engineering domain can reference. The ScadaOnWeb platform provides a generic solution to the efficient handling of SCADA data which can be used within applications, and which can replace proprietary systems. Five software applications of the ScadaOnWeb project use this generic platform in order to validate the technology. With the provision of a ScadaOnWeb plug-in it is expected to handle structured numeric data using web browsers in a simple way as it is possible for other standardized formats. Consequently, the use of the internet and intranet for SCADA applications is likely to become widespread.

9 Outlook

Basing itself on XML and Semantic Web technologies, ScadaOnWeb can be rightly said to be the first major project to apply Semantic Web technologies to the domain of engineering data opening thereby the way for a vast range of new non-proprietary applications. Already, the range of application domains covered by the ScadaOnWeb demonstrators is quite impressive:

- meteorology: disaster prediction
- supply of electrical power:
  - planning of power consumption
  - metering electrical power consumption
  - metering and control of small generators
  - maintenance of electrical supply networks
- electrical and process industries: maintenance of electrical machines and devices

But ScadaOnWeb technology is by no means limited to the range of application domains covered by the demonstrators: It has the potential to access and control in a many client, many server environment any device that supplies or receives data for

which a standardized dictionary is available or can be made available. Therefore, future applications of ScadaOnWeb technology may be found in application domains as remote to each other as emission detection, disaster prediction, distributed power generation, supervision of off-shore installations and applications fed by non proprietary sources of stock market data.

Further in the future, a natural extension of the ScadaOnWeb technology will be to enrich it with rule-bases and build web services on top of it. Thus, ScadaOnWeb technology will be made available to and usable by agent software on the web. This work will be an important step towards realizing the vision of "Intelligent Web Services" build upon the "Semantic Web": In this context, rule-bases can play an important role as they can act as a means to draw inferences, to express constraints, to react to events/changes, etc. allowing applications to make deductions from the data supporting thereby the process of decision making.

Acknowledgement. This research project is sponsored by the European Community in the Information Society Technologies (IST) program and BBW (Bundesamt für Bildung und Wissenschaft, Bern, Switzerland).

References

[1] http://www.scadaonweb.com
[2] Bacher, R., Leal, D., Schröder, A.: "ScadaOnWeb – Modelling and Web-Exchange of Process and Engineering Information", International Semantic Web Conference, Cagliari, Italy, (2002)
[3] http://www.w3.org/XML/
[4] http://www.w3.org/RDF/
[5] http://hdf.ncsa.uiuc.edu/HDF5/
[6] Leal, D., Schröder, A.: "RDF reference data library format. Available online at: http://www.scadaonweb.com/publications/NOTE-RDL/NOTE-RDL.html
[7] http://www.w3.org/TR/owl-features/
[8] http://www.w3.org/Math/