Intelligent Self-Describing Power Grids

Andrea Schröder[1] and Inaki Laresgoiti[2]
[1]FGH e.V.,
[2]TECNALIA
[1]Germany
[2]Spain

1. Introduction

The liberalization of the electrical energy market increases the complexity of power grid operation and pushes companies to build cheaper electrical power generation and distribution systems. Automated network management systems based on massively distributed information help to maintain the expected quality performance, manageability and reliability. In this context the European Technology Platform (ETP) Smart Grids was set up in 2005 to create a joint vision for the European networks of 2020 and beyond. Its vision builds both upon an efficient and intelligent integration of distributed resources with other network components and upon two basic concepts upon which the integration of huge amounts of distributed resources is going to be done: "Virtual Power plants" and "Micro Grids" both combined with an extensive use of Information and Communication Technologies (ICT). The Internet and the Web are identified as promising approaches for the integration of large amounts of distributed resources in a dynamic and efficient way.

The EC funded S-TEN (Intelligent Self-describing Technical and Environmental Networks) [1] project running from April 2006 to September 2008 makes a substantial step to let this vision come true by providing decision support in a complex and potentially continuously changing networks based upon the use of semantic web technology [2]. E.g. ontologies are used to represent knowledge associated with the devices, ranging from simple sensors to complex plants such as combined heat and power plants or wind power plants, so that they can register themselves at a registry exposing their capabilities and interfaces to applications searching for a matching service. Besides, semantics describing the equipment design information and measurement data can be used together with the respective data by decision support systems to detect any malfunctioning of the equipment and to trigger corresponding alarms or any other kind of action, such as giving best practice support to the operator.

The following chapters present the work being done to develop and use technologies designed to provide functionalities required by the grid of the future. A clear focus is on the defined ontologies as they form the backbone of intelligent and autonomously acting applications.

2. S-TEN system

In future intelligent grids with lots of distributed resources, such as sensors, photovoltaic plants (PV), wind power plants, combined heat and power plants (CHP) and electrical vehicles, each node has its own intelligence, is able to register in the network autonomously and publishes information about its position, services and data.

This is done by giving devices that may be sensors or plants a semantically interpretable self-description. This self-description is then uploaded into the S-TEN registry. The S-TEN registry is also based on semantically annotated Web Services complemented by a semantically annotated registry schema describing the semantics of data stored in the registry. Authorized control systems can access and query for a particular service (Figure 1).

The registry response contains relevant information to access the Web services offered by the sensor/plant, e.g. monitor current data, browse data for a specific instance in time or a period, control capabilities, subscribe to event notification such as alarms and warnings etc.. In addition, semantic annotation of technical documentation and definition and application of rules support best-practice advice to decision-makers enabling them to react more precisely and faster to unusual situations.

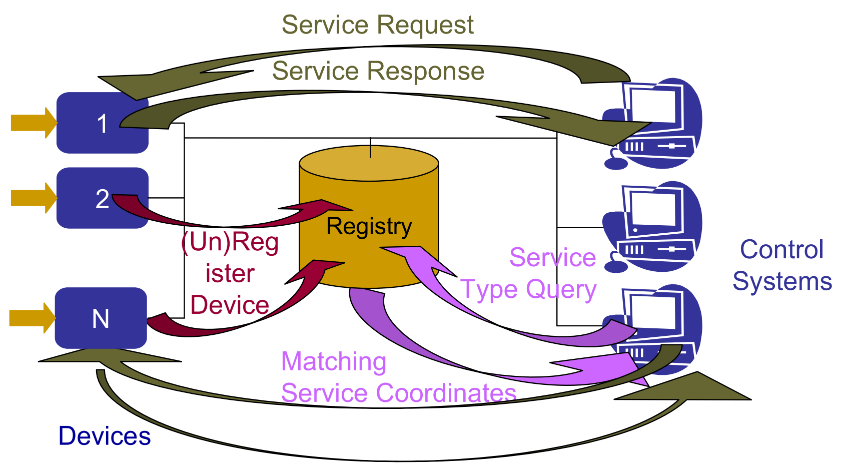

Fig. 1. Overview of the S-TEN system.

3. S-TEN technology

The S-TEN system is based on the S-TEN technology consisting of the following main components:

- **Self-description and registry.** S-TEN systems publish their **self-description** at the S-TEN registry exposing their capabilities and interfaces to applications searching for a matching service. An application first accesses the registry in order to get the endpoints and relevant information of the registered resources.
- **Ontologies. S-TEN top ontologies** and **demonstrator specific ontologies** referencing concepts of the S-TEN top ontologies have been defined.
- **Semantically annotated web services.** The semantics used refer to concepts of the S-TEN ontology for self-describing networks and the S-TEN ontologies for physical quantity space and scale. The web service based implementation of S-TEN services is based on Axis2 [3].

The following section describes the above listed main components of the S-TEN technology in more detail.

3.1 Self-description and registry

For physical devices it is necessary to specify the way in which they announce their presence in a network, so that they can be found when it is necessary to carry out a control or monitoring activity. In the following example, an implementation of the register function is shown: The Register message is used to register an XML-File with the registry. The request to activate this function comprises a HTTP header, a SOAP header with authentication information according to WS-security and an XML-File in the SOAP body which complies with the semantically annotated S-TEN registry schema:

```
POST /STENRegistry/services/Registry HTTP/1.1
Content-Type: text/xml; charset=UTF-8
SOAPAction: "urn:register"
User-Agent: Axis2
....
<?xml version='1.0' encoding='UTF-8'?>
  <soapenv:Envelope>
    <soapenv:Header>
      <wsse:Security>
   <wsse:UsernameToken>
          <wsse:Username>XXX</wsse:Username>
          <wsse:Password      Type="...#PasswordDigest">      1mFoYgXbWQ72/6bvnWNEnxtatmU=
</wsse:Password>
        </wsse:UsernameToken>
      </wsse:Security>
      ...
    </soapenv:Header>
    <soapenv:Body>
      <ns1:register>
        <xmlfile> the xml file </xmlfile>
      </ns1:register>
    </soapenv:Body>
  </soapenv:Envelope>
```

In reaction to the register request, the registry sends a HTTP response back to the client:

```
HTTP/1.1 200 OK
Content-Type: text/xml;charset=UTF-8
....
<?xml version='1.0' encoding='UTF-8'?>
  <soapenv:Envelope>
    <soapenv:Header> ... </soapenv:Header>
    <soapenv:Body>
      <ns:registerResponse>
        <ns:return>
          <result></result>
        </ns:return>
      </ns:registerResponse>
    </soapenv:Body>
  </soapenv:Envelope>
```

If the registration request was successful, a HTTP response with an empty result is sent back to the client. Otherwise an error message will be returned in the SOAP body of the HTTP response. To enable the invoking client to semantically interpret the register service offered by the registry server, the WSDL [4] specification of the registry service has to be semantically annotated according to SAWSDL [5]. As at this time Axis2 did not fully support WSDL 2.0 it was necessary to base the work on WSDL 1.1, and annotate the respective S-TEN registry WSDL specification accordingly. The data-type associated with the Register Message looks in its semantically annotated form the following way:

```
<xs:element name="register" sawsdl:modelReference="http://www.s-ten.eu/sten-web-
services#AddRequestMessage">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="xmlfile" nillable="true"
             sawsdl:modelReference="
             http://www.s-ten.eu/sten-web-services#XMLObject
             http://www.w3.org/2000/01/rdf-schema#isDefinedBy
             http://www.s-ten.eu/sten-web-services#RegistrySchema"/>
         </xs:sequence>
     </xs:complexType>
</xs:element>
```

The referenced registry schema is then again semantically annotated which enables the client using the GetSchema message to arrive at a complete, semantically correct interpretation of the expected registry content.

```
<xsd:schema                                    xmlns:Ontology0="http://www.s-ten.eu/sten#"
xmlns:sawsdl="http://www.w3.org/ns/sawsdl"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"               elementFormDefault="qualified"
attributeFormDefault="unqualified">
 <xsd:complexType  name="SystemType"  sawsdl:modelReference="http://www.s-ten.eu/sten-web-
services#S-TENSystem">
  <xsd:sequence>
   <xsd:element name="SystemID" type="xsd:anyURI"
   sawsdl:modelReference="http://www.s-ten.eu/sten-web-services#ReferenceDesignation"/>
   <xsd:element name="Name" type="xsd:string"
   sawsdl:modelReference="http://www.s-ten.eu/sten-web-services#Designation"/>
   <xsd:element name="S-TENServiceDescriptionEndpoint" type="xsd:anyURI"
   sawsdl:modelReference="http://www.s-ten.eu/sten-web-services#WSDLDescription"/>
   <xsd:element name="TextualDescription" type="xsd:string" minOccurs="0"
   sawsdl:modelReference="http://www.s-ten.eu/sten-web-services#Description"/>
   <xsd:element name="SymbolicLocation" type="xsd:string" minOccurs="0"
   sawsdl:modelReference="http://www.s-ten.eu/sten-web-services#LocationDesignation"/>
   <xsd:element name="S-TENDescriptionEndpoint" type="xsd:anyURI" minOccurs="0"
   sawsdl:modelReference="http://www.s-ten.eu/sten-web-services#OWLDescription"/>
   <xsd:element name="GISLocation" minOccurs="0"
   sawsdl:modelReference="http://www.s-ten.eu/sten-web-services#PointInSpace">
    <xsd:complexType>
     <xsd:sequence>
      <xsd:element name="Longitude" type="xsd:decimal"
      sawsdl:modelReference="http://www.s-ten.eu/sten-web-services#Longitude"/>
      <xsd:element name="Latitude" type="xsd:decimal"
      sawsdl:modelReference="http://www.s-ten.eu/sten-web-services#Latitude"/>
      <xsd:element name="HeightOverSeaLevel" type="xsd:decimal"
      sawsdl:modelReference="http://www.s-ten.eu/sten-web-services#HeightOverSeaLevel"/>
     </xsd:sequence>
    </xsd:complexType>
   </xsd:element>
   <xsd:element name="ObservedPhysicalQuantitySpace" type="xsd:string" minOccurs="0"
   maxOccurs="unbounded"                    sawsdl:modelReference="http://www.s-ten.eu/sten-web-
services#PhysicalQuantitySpace"/>
   <xsd:element name="MonitoredDevices" type="xsd:string" minOccurs="0" maxOccurs="unbounded"
   sawsdl:modelReference="http://www.labein.es/Microgrid.xml#Device"/>
   <xsd:element name="OfferedServices" type="xsd:string" minOccurs="0" maxOccurs="unbounded"
   sawsdl:modelReference="http://www.labein.es/Microgrid.xml#Service"/>
   <xsd:element name="InventaryNumber" type="xsd:string" minOccurs="0"
   sawsdl:modelReference="http://www.s-ten.eu/sten-web-services#InventaryNumber"/>
   <xsd:element name="Producer" type="xsd:string" minOccurs="0"
   sawsdl:modelReference="http://www.s-ten.eu/sten-web-services#Manufacturer"/>
   <xsd:element name="Owner" type="xsd:string" minOccurs="0"
   sawsdl:modelReference="http://www.s-ten.eu/sten-web-services#Owner"/>
   <xsd:element name="Possessor" type="xsd:string" minOccurs="0"
   sawsdl:modelReference="http://www.s-ten.eu/sten-web-services#Possessor"/>
   <xsd:element name="Operator" type="xsd:string" minOccurs="0"
   sawsdl:modelReference="http://www.s-ten.eu/sten-web-services#Operator"/>
   <xsd:element name="LegacySystemSpecificInformation" minOccurs="0" maxOccurs="unbounded"
   sawsdl:modelReference="http://www.s-ten.eu/sten-web-services#LegacySystemInformation">
    <xsd:complexType>
     <xsd:sequence>
      <xsd:element name="URN" type="xsd:anyURI"/>
      <xsd:element name="Name" type="xsd:string" minOccurs="0"/>
      <xsd:element name="TextualDescription" type="xsd:string" minOccurs="0"/>
      <xsd:element name="Value" type="xsd:string"/>
     </xsd:sequence>
    </xsd:complexType>
   </xsd:element>
  </xsd:sequence>
 </xsd:complexType>
 <xsd:element name="System" type="SystemType"/>
</xsd:schema>
```

3.2 Ontologies

Ontologies are an important part of the Semantic Web and they are thought to be one of the most important advances in software development. They define the terms used to describe and represent an area of knowledge and have the advantage of being a way for a much more formal and less ambiguous definition of the meaning of the concepts.

They are composed of a formal description of concepts (classes) in a domain of discourse, properties of each concept describing various features of the concept, and restrictions on properties. If we add a set of individual instances of classes to the ontology we obtain a knowledge base.

There are different languages that can be used to express ontologies such as XML, RDF(S), SHOE, OWL, etc. In the S-TEN project the Web Ontology Language (OWL) [6] is employed for the definition of the case study because it provides the most expressive language for ontology description.

OWL was the W3C recommended standard for ontologies when the S-TEN project started. It is an extension of RDF(S) [7] that provides additional vocabulary in order to represent more characteristics between concepts. It distinguishes three types of entities: individuals, classes and properties:

- Individuals (or instances) represent objects in the domain of interest (e.g. Transformer_1)
- Classes are groups of individuals (e.g. Equipment, Transformer). Classes can be organised into a superclass-subclass hierarchy, which is also known as taxonomy.
- Properties are binary relations between individuals. There are two main types of properties: object properties that link two individuals and datatype properties that link an individual to an XML Schema Datatype value or an rdf literal (e.g. hasActivePower, name).

There are three different OWL sublanguages:

- OWL-Lite: It is the syntactically simplest sublanguage. It is intended to be used in situations where only a simple class hierarchy and simple constraints are needed.
- OWL-DL: It is much more expressive than OWL-Lite and it is based on Description Logics. It can be used when a maximum expressiveness is required but the computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite time) must be ensured.
- OWL-Full: It provides the maximum expressiveness and the syntactic freedom of RDF with no computational guaranties.

For S-TEN OWL-DL was chosen because the existent reasoners were based on OWL-DL and although they would work also in OWL-Full they could enter into endless loops.

The S-TEN ontology [8] describes all S-TEN system components in a computer understandable way and provides the semantics necessary for exchanging messages between the different components of the S-TEN system including those coming from external users and the ones used internally by the system. Key features of the S-TEN ontology are as follows:

- it specifies an ontology for describing a physical network;
- it can be used by intelligent devices to publish information about their existence and state;
- it encompasses measurements made by sensors within a network and the storage and processing of measurements and
- it enables the use of inferencing to support activities carried out upon a network, including operations and maintenance.

A top level ontology for sensor related information and Web Services has been developed taking into account existing standards such as ISO 15926, IEEE SUMO, IEC 61850 and ISO 10303 (Figure 3). Based on the top S-TEN ontologies, application specific ontologies referencing concepts of the S-TEN top level ontology have been defined for the prototype applications.

The advantage of having an ontology is that automated reasoning can be applied to the data of the network components. Within S-TEN, rules supporting system operations are developed and applied to information available in the Web, e.g. measurements. Rules trigger corresponding alarms and generate Best Practice advice for the system operator based on the currently valid state of the network. Different organisations usually have different views on the same data. Therefore each organisation can have their own rule-bases to generate appropriate Best Practice Advice according to the nature of the system status enabling decision-makers to react more precise and faster to unusual situations and unbalanced systems.

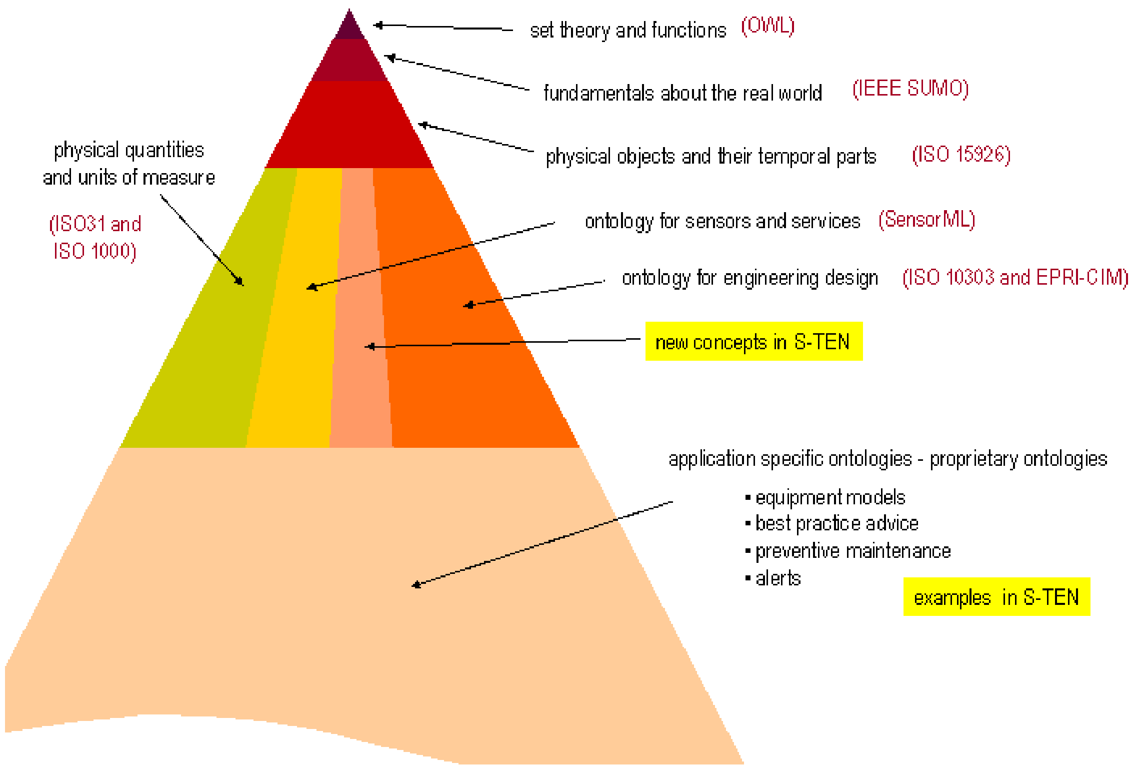
set theory and functions (OWL)
fundamentals about the real world (IEEE SUMO)
physical objects and their temporal parts (ISO 15926)
physical quantities and units of measure
(ISO31 and ISO 1000)
ontology for sensors and services (SensorML)
ontology for engineering design (ISO 10303 and EPRI-CIM)
new concepts in S-TEN
application specific ontologies - proprietary ontologies
• equipment models
• best practice advice
• preventive maintenance
• alerts
examples in S-TEN

Fig. 2. S-TEN ontology.

The scope of the S-TEN ontology includes:

- **physical objects (physical thing)**

  A network is made up of physical objects. Information about these objects includes:

  - what it is;
  - what it is connected to;
  - where it is (e.g. input from GPS);
  - what it does;
  - what it measures (if it is a measurement device);
  - what the quality of the measurement is (if it is a measurement device).
- **activities**

  An activity can be:

  - a manufacturing or processing activity performed by a physical object within a network as part of its normal operation;

- a measurement activity performed by a measurement device;
- a control, maintenance or measurement activity performed by a person

The information about any activity, can include:

- what it is;
- the performer, which can be a person or device; and
- other physical objects which participate in the activity, particularly the inputs and outputs.

An input or output can be an information object.

- **information**

  It is necessary to record information about both information objects and their information contents.

- **Physical quantities, property and scale**

For physical things the definition of the ontology will be described in detail. The ontologies for the other categories can be found at http://www.s-ten.eu/deliverables/D2.1/.

Physical things

Types of physical thing

The purpose of the structure at the top of the ontology is to specify which relationships are valid for which types of physical thing. These relationships are inherited down the ontology. The principal relationships are discussed in the section Principal relationships. The top of the ontology for S-TEN, is a variant of that in SUMO and ISO 15926-2 as follows:

2. A sten:physical_thing is one of:
   - sten:classical_object which consists of very many interacting particles for a period;
   - sten:classical_object_at_instant which is a sten:classical_object at an instant; or
   - non-classical thing which is everything else, including small numbers of particles isolated from the rest of the universe, and packets of energy.

   For a sten:classical_object, time, position and causality are all valid concepts. Non-classical things are not within the scope of S-TEN.

3. A sten:classical_object can be:
   - sten:persistent_object which is created by a sten:activity and which continues to exist until destroyed by another sten:activity; and
   - sten:ephemeral_object which is a part of a sten:persistent_object which existed before or continued to exist after.

   The different between sten:persistent_object and sten:ephemeral_object is contentious amongst upper ontologists. An sten:ephemeral_object is something which has been defined by a person for a purpose. The beginning and end of an sten:ephemeral_object is not necessarily obvious to another person. Unfortunately, the same can be sometimes be said for a sten:persistent_object.

4. There are many types of sten:persistent_object. These include:
   - sten:facility, which is an object defined by the role it performs, and which is of interest for operations;
   - sten:physical_asset, which is an object defined by its material, and which is of interest for materials management;
   - computer file, which is an information bearing object defined by a computer system.

   EXAMPLE: Consider a pump in a process plant. The switch in the control room which operates this pump is labeled "P_101". On 2007-03-19, the pump with serial number "98-1234" is installed as "P_101". A maintenance activity is carried out overnight, so that on 2007-03-20 the pump with serial number "03-5678" is installed.

   The object "P_101" is a facility, and is defined by the role it performs. The object "98-1234" is an asset and is defined *more or less* by the matter it consists of.

   The qualification *more or less* is important, because a major refurbishment may change many of the parts of asset "98-1234". In this case, whether or not a new asset is created is an administrative choice.

   The justification for regarding a computer file as something physical is discussed in Section Information.

5. There are many types of sten:ephemeral_object. These include:
   - sten:period_of_life, in which a sten:persistent_object operates during a sten:period;
   - sten:activity, which is one or more instances of sten:period_of_life, and which is defined in order to identify a purpose, cause or result.

   NOTE 5: There are particular relationships between an sten:activity and its parts, such as sten:has_input, sten:has_output, sten:creates, sten:destroys, and sten:performed_by.

Top classes in the S-TEN, SUMO and ISO 15926-2 ontologies are shown in Figure 3.

Principal relationships

Relationships which are valid for the different types of physical thing are as follows:

relationship	**rdfs:Property**	**rdfs:domain**	**rdfs:range**
whole-part	sten:part_of	sten:physical_thing	sten:physical_thing
	sten:has_part	sten:physical_thing	sten:physical_thing
temporal whole-part	sten:temporal_part_of	sten:physical_thing	sten:classical_object
	sten:has_temporal_part	sten:classical_object	sten:physical_thing
assembly	sten:component_of	sten:classical_object	sten:classical_object
	sten:has_component	sten:classical_object	sten:classical_object
at time	sten:at_time	sten:classical_object_at_instant	sten:instant
during	sten:during	physical_thing	sten:part_of_time

The difference between the relationships sten:part_of, sten:temporal_part_of and sten:component_of is illustrated by Figure 4.

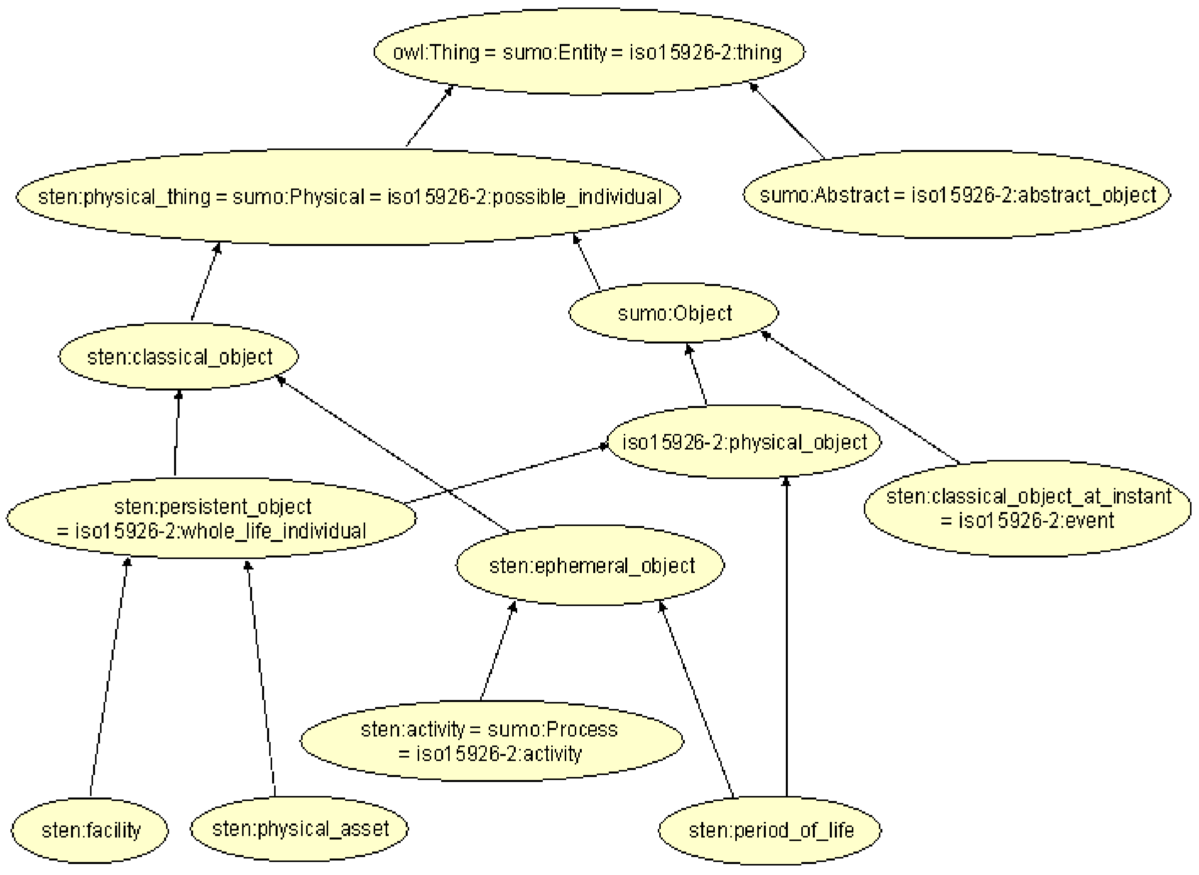
owl:Thing = sumo:Entity = iso15926-2:thing
sten:physical_thing = sumo:Physical = iso15926-2:possible_individual
sumo:Abstract = iso15926-2:abstract_object
sumo:Object
sten:classical_object
iso15926-2:physical_object
sten:persistent_object = iso15926-2:whole_life_individual
sten:classical_object_at_instant = iso15926-2:event
sten:ephemeral_object
sten:activity = sumo:Process = iso15926-2:activity
sten:facility
sten:physical_asset
sten:period_of_life

Fig. 3. Top classes in the S-TEN ontology.

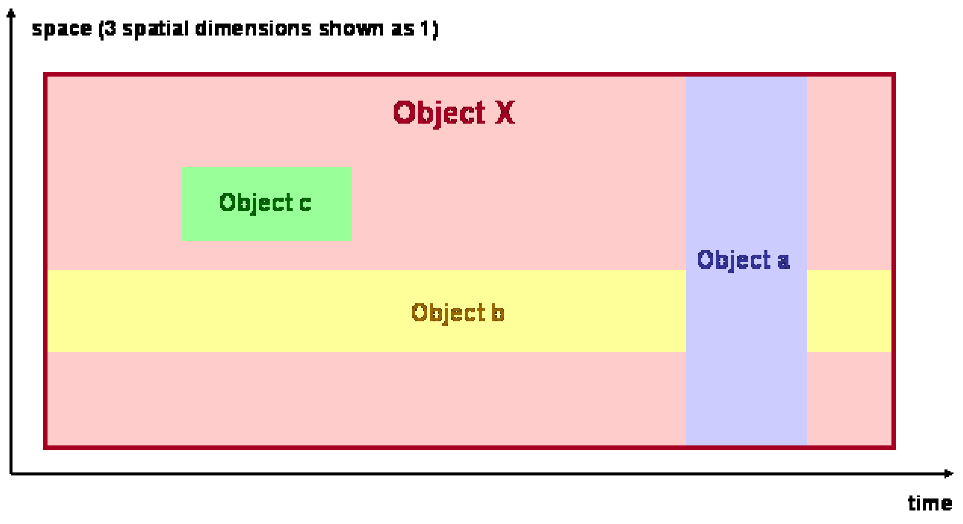

Fig. 4. Temporal part and component.

In this example:

- "object a" has a sten:temporal_part_of relationship with "object X";
- "object b" has a sten:component_of relationship with "object X";
- "object c" has a sten:part_of (but not sten:temporal_part_of or sten:component_of) relationship with "object X";

Activity and role

An sten:activity (= sumo:Process) is defined in order to identify a purpose, cause or result.

S-TEN defines the sten:has_role_player relationship between an sten:activity and a sten:persistent_object. This is not a relationship with a temporal part because an sten:activity is specifically defined to express a relationship between instances of sten:persistent_object.

EXAMPLE 1: Consider the manufacturing sten:activity "F. Bloggs production on 2007-04-23". This sten:activity creates the sten:physical_asset with serial number "07/123456". The sten:activity does not create a temporal part of "07/123456", but instead the sten:persistent_object which continues to exist after the sten:activity has finished.

There are many possible roles in an activity, which are specific to the type of activity.

EXAMPLE 2 A meeting activity has the roles:

- chair person;
- minute taker;
- teller (who counts votes); and
- attender.

For S-TEN, the roles "input" and "output" have been introduced. The part of speech has also been changed so that the relationships are not identified by nouns and cannot be easily mistaken for objects. This gives the following generic roles:

role	**definition**
sten:performed_by	identical to sumo:agent.
sten:has_input	identical to sumo:resource.
sten:has_output	identical to sumo:result.
sten:destroys	a sten:has_input, where the input does not exist after the activity.
sten:creates	a sten:has_output, where the output did not exist before the activity.

A temporal part can be related to an sten:activity by a sten:has_part relationship.

An sten:activity can have a relationship with an object which is neither sten:has_role_player or sten:has_part. Examples of this are:

- sten:causes where one sten:activity causes another to happen;
- sten:has_input_record and sten:has_output_record where a sten:information_carrier is input or output.
- sten:investigates where information is sought about an object which may not play a role in the sten:activity.

Relationships with parts of life

The relationship sten:component_of is a sten:simultaneous_mapping which applies to each temporal part of the part and of the whole.

EXAMPLE 1: The statement "*The gearbox with serial number 98/1234 has a **sten:component_of** relationship with the car with chassis number 99/4567.*" is untrue because the gearbox existed before the car, and may continue to exists after the car if the car is disassembled. If the car has the same gearbox throughout its life, then the correct statement is:

“A temporal part of the gearbox with serial number 98/1234 has a ***sten:component_of*** relationship with the car with chassis number 99/4567.“ There is no need to be specific about the start and end times of the temporal part of :98/1234, because this can be deduced from the start and end times of :99/4567.

Ontology for physical thing - taxonomy

The following section is an excerpt of the ontology for physical thing and additionally lists some explanations and examples. The ontology is defined in OWL and RDF.

physical_thing

An object is an sten:physical_thing if and only if it exists in space and time.

```
<owl:Class rdf:about="&sten;physical_thing">
 <owl:sameAs rdf:resource="&sumo;Physical"/>
 <owl:sameAs rdf:resource="&iso15926-2;possible_individual"/>
 <meta:defined_by rdf:resource="&D2.1;/physical_things.htm#physical_thing"/>
</owl:Class>
```

classical_object

An object is an sten:classical_object if and only if:

- it is an sten:physical_thing;
- it is a large number of interacting particles which exists for finite periods of time.

A sten:classical_object can have more than one creation and destruction activity because can cease to exist for periods of time.

```
<owl:Class rdf:about="&sten;classical_object">
 <rdfs:subClassOf rdf:resource="&sten;physical_thing"/>
 <meta:defined_by rdf:resource="&D2.1;/physical_things.htm#classical_object"/>
</owl:Class>
```

classical_object_at_instant

An object is an sten:classical_object_at_instant if and only if:

- it is an sten:physical_thing;
- it is a sten:classical_object at an instant.

```
<owl:Class rdf:about="&sten;classical_object_at_instant">
 <rdfs:subClassOf rdf:resource="&sten;physical_thing"/>
 <rdfs:subClassOf rdf:resource="&sumo;Object"/>
 <owl:sameAs rdf:resource="&iso15926-2;event"/>
 <meta:defined_by rdf:resource="&D2.1;/physical_things.htm#classical_object_at_instant"/>
</owl:Class>
```

persistent_object

An object is an sten:persistent_object if and only if:

- it is an sten:classical_object;
- the periods of time during which it exists results from creation and destruction activities.

A single sten:persistent_object can have more than one creation and destruction activity because can cease to exist for periods of time.

```
<owl:Class rdf:about="&sten;persistent_object">
 <rdfs:subClassOf rdf:resource="&sten;classical_object"/>
 <owl:sameAs rdf:resource="&iso15926-2;whole_life_individual"/>
 <rdfs:subClassOf rdf:resource="&iso15926-2;physical_object"/>
 <meta:defined_by rdf:resource="&D2.1;/physical_things.htm#persistent_object"/>
</owl:Class>
```

ephemeral_object

An object is an sten:ephemeral_object if and only if:

- it is an sten:classical_object;
- its spatial and temporal boundaries are a choice to support the recording of information.

A boundary of an sten:ephemeral_object in time or space is not necessarily obvious to an observer.

```
<owl:Class rdf:about="&sten;ephemeral_object">
 <rdfs:subClassOf rdf:resource="&sten;classical_object"/>
 <meta:defined_by rdf:resource="&D2.1;/physical_things.htm#ephemeral_object"/>
</owl:Class>
```

facility

An object is a sten:facility if and only if:

- it is a sten:persistent_object;
- it is created by commissioning activities and is destroyed by decomissioning activities.

A sten:facility is operated. The creation and destruction activities are recognised as such by the operator. A single sten:facility can have more than one creation activity because it can cease to exist for periods of time.

A sten:facility is distinguished from a sten:physical_asset by the nature of the creation and destruction activities. A sten:facility is created when physical assets are combined into a whole that can be operated.

The quantity of matter that is a facility can change during the life of the facility, by the removal of some matter and the addition of other matter. It often changes completely, so that all the matter previously present is removed and replaced by different matter.

EXAMPLE: The feedwater pump for boiler "B_101" is a facility with tag "P_101". Operating the boiler involves operating pump "P_101". The quantity of matter that is pump "P_101" changes a little whenever pump "P_101" is maintained, perhaps by replacing a seal or a bearing.

If the pump with serial number 98-1234 is installed as "P_101" until 2006-10-16, and is then replace by the pump with serial number 06-4567, then the quantity of matter that is pump "P_101" changes completely on 2006-10-16.

```
<owl:Class rdf:about="&sten;facility">
 <rdfs:subClassOf rdf:resource="&sten;persistent_object"/>
 <meta:defined_by rdf:resource="&D2.1;/physical_things.htm#facility"/>
</owl:Class>
```

physical_asset

An object is a sten:physical_asset if and only if:

- it is a sten:persistent_object;
- it is created by an assembly or fabrication activity and is ended by a disassembly or recycling activity.

A sten:physical_asset is owned and maintained. The creation and destruction activities are recognised by the owner or maintainer.

The quantity of matter that is a physical_asset can change during the life of the physical_asset, by the removal of some matter and the addition of other matter. There is only rarely a complete change, in which that all the matter previously present is removed and replaced by different matter.

EXAMPLE: The pump with serial number "98-1234" is a physical_asset. This pump is recorded in the inventory of equipment owned by J. Bloggs and Co..

If the pump with serial number 98-1234 is maintained by replacing the bearings and seals, then the quantity of matter changes, but the physical_asset continues to exist. Replacing the body of the pump would probably be regarded as a manufacturing a new physical_asset, rather than a maintenance activity carried out on a existing physical_asset.

```
<owl:Class rdf:about="&sten;physical_asset">
 <rdfs:subClassOf rdf:resource="&sten;persistent_object"/>
 <meta:defined_by rdf:resource="&D2.1;/physical_things.htm#physical_asset"/>
</owl:Class>
```

period_of_life

An object is a sten:period_of_life if and only if:

- it is an sten:ephemeral_object;
- the sten:period during which it exists is a choice to support the recording of information.

```
<owl:Class rdf:about="&sten;period_of_life">
 <rdfs:subClassOf rdf:resource="&sten;ephemeral_object"/>
 <rdfs:subClassOf rdf:resource="&iso15926-2;physical_object"/>
 <meta:defined_by rdf:resource="&D2.1;/physical_things.htm#period_of_life"/>
</owl:Class>
```

activity

An object is an sten:activity if and only if:

- it is an sten:ephemeral_object;
- it is defined in order to identify a purpose, cause or result.

The ISO 15926 term "activity" is prefered to the SUMO term "process", because it is more general and does not imply that there is an intention or that there is a well defined outcome.

```
<owl:Class rdf:about="&sten;activity">
 <rdfs:subClassOf rdf:resource="&sten;ephemeral_object"/>
 <owl:sameAs rdf:resource="&sumo;Process"/>
 <owl:sameAs rdf:resource="&iso15926-2;activity"/>
 <meta:defined_by rdf:resource="&D2.1;/physical_things.htm#activity"/>
</owl:Class>
```

3.3 Semantically annotated web services

S-TEN data access is based on semantically annotated Web Services or on agent based systems using as a system component semantically annotated Web Services, too. This approach was chosen, as standard Web Services take into account only the functional or syntactical aspects of a service. This means semantic information is either not available or offered only in an informal and human-interpretable way. It's therefore not possible to derive the intended semantics of input, output, and function or the meaning of parameters of a Web service from its WSDL description only. In that way, two Web services may have the same syntactical definition and, yet, implement different functionality. In this situation, Semantic Web services constitute a promising path to automate the tasks of Web service discovery, composition and invocation and to improve the integration of applications within and across enterprise boundaries. At present, there are three main approaches to enable Semantic Web services: SAWSDL [5], OWL-S [9] and WSMO [10], ranging from pragmatic to more comprehending extensions of standard Web services. Within S-TEN, SAWSDL, as the most pragmatic approach was selected as the only one suitable for implementation at this time, because SAWSDL was supported by the many of the widely used tools of WSDL while the other two had a lack of tools. The basic idea behind SAWSDL is to enhance the WSDL specification with the aim to go beyond a mere syntactical description of a Web service by adding semantics to the functions and the data delivered by a service. As it can be seen in the code excerpt below, featuring a BrowseRequest service, SAWSDL model references provide the link to the semantics defined in the S-TEN top-level ontology.

```
<xs:element name="BrowseRequest">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="SessionID" type="xs:string"
    sawsdl:modelReference="http://www.s-ten.eu/sten-web-services#SessionID"/>
   <xs:element name="ObservedBy" type="xs:string" sawsdl:modelReference="http://www.s-ten.eu/sten-
ore#observedBy"/>
   <xs:element name="Observes" type="xs:string" minOccurs="0"
     maxOccurs="unbounded" sawsdl:modelReference="http://www.s-ten.eu/sten-core#observes"/>
   <xs:element name="PhysicalProperty" type="xs:string" minOccurs="0" maxOccurs="unbounded"
       sawsdl:modelReference="http://www.s-ten.eu/sten-core#PhysicalProperty"/>
   <xs:choice>
    <xs:element name="TimeSpan" type="ns0:TimeSlice"/>
     <xs:element name="At" type="xs:long" nillable="true"
         sawsdl:modelReference="http://www.s-ten.eu/sten-core#ClassicalObjectAtInstant
                               http://www.s-ten.eu/sten-core#atTime
                               http://www.s-ten.eu/sten-core#Time
                               http://www.s-ten.eu/sten-core#milliSecondsSince1970-01-01
                               http://www.s-ten.eu/sten-core#Integer"/>
     <xs:element name="Quantity" type="xs:int" nillable="true"
         sawsdl:modelReference="http://www.s-ten.eu/sten-web-services#NumberOfMembers"/>
    </xs:choice>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

Fig. 5. Browse Request Service.

4. S-TEN applications

Prototype applications have been developed within the S-TEN project validating the S-TEN technology and showing the advantages of the use of web-services, agents and ontologies. Among these prototype applications the following two applications demonstrate the monitoring and control functionality of future power systems:

- Control of Distributed Resources in Electrical Power Networks.
- Secondary control of microgrids

For these prototypes application-specific ontologies have been defined referencing concepts of the S-TEN top ontology and taking into account existing standards such as IEC 61850 and ISO 15926. The development of these applications has been supported by a user-group set up within the S-TEN project. This user-group, consisting of supply companies, manufacturers and research organizations, assisted in defining the requirements of the various applications thereby ensuring its practical relevance. The following section describes the two applications mentioned above in more detail.

4.1 Control of distributed resources in electrical power networks

The integration of distributed energy resources will influence the structure of future power systems considerably. Power generation will not take place only in a few central power stations but also in a constantly increasing number of distributed energy generators such as wind power plants, photovoltaic plants and combined heat and power plants located in the medium and low voltage system. The prototype application "Control of Distributed Resources in Electrical Power Networks" will enable the monitoring of distributed generators with the aim to improve the management of the distribution network in terms of remote control capabilities and automated network control. The monitoring includes both the observation of measuring values and events (alarms and warnings) and provides event handling. The application has a public presence on the web where its client software can be downloaded at www.s-ten.eu.

Ontology

Based on the S-TEN top-level ontology an application specific ontology has been defined featuring combined heat and power plants (CHP), photovoltaic (PV), wind power devices and additional network devices such as circuit-breakers, inverters, batteries, etc.. An excerpt of the CHP ontology is given in Figure 6.

The Web Services the network resources offer are semantically annotated that is they link to the application ontology and the S-TEN top-level ontology. The XML document (Fig. 7.) shows an example of a browse request and the browse response written according to the S-TEN ontology.

Also, rules definition uses concepts from the S-TEN top-level ontology and the application specific ontology.

Architecture and functionality

A hierarchical architecture has been chosen for the demonstrator (Figure 6). For each DER an S-TEN server has been established. The hierarchical architecture allows adding S-TEN

```
<?xml version="1.0"?>

<rdf:RDF xml:base="[redacted]" xmlns="[redacted]"
xmlns:chp="[redacted]" xmlns:der="[redacted]"
xmlns:owl="http://www.w3.org/2002/07/owl#" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" xmlns:sten-PhysicalQuantitySpace="http://www.s-ten.eu/sten-
PhysicalQuantitySpace#" xmlns:sten-core="http://www.s-ten.eu/sten-core#"
xmlns:xsd="http://www.w3.org/2001/XMLSchema#">
        <owl:Ontology rdf:about="">
                <owl:versionInfo xml:lang="en">version 0.1</owl:versionInfo>
                <rdfs:comment xml:lang="en">Ontology describing CHP systems</rdfs:comment>
                <owl:imports rdf:resource="http://www.s-ten.eu/sten-core"/>
                <owl:imports rdf:resource="[redacted]"/>
                <owl:imports rdf:resource="http://www.s-ten.eu/sten-PhysicalQuantitySpace"/>
        </owl:Ontology>
        <owl:Class rdf:ID="CHPSystem">
                <rdfs:subClassOf rdf:resource="#DER"/>
                <rdfs:comment xml:lang="en">A Combined heat and power (CHP) system uses the exhaust heat of an
                engine to simultaneously generate both electricity and useful heat.</rdfs:comment>
        </owl:Class>
        <owl:Class rdf:ID="CHPModule">
                <sten-core:partOf rdf:resource="#CHPSystem"/>
                <rdfs:comment xml:lang="en">A CHP module forms part of a CHP system. The CHP system may consist of
                various CHP Modules.</rdfs:comment>
        </owl:Class>
        <!--        CHP specific data      -->
        <owl:ObjectProperty rdf:about="#nominalMaximumActivePower">
                <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
                <rdfs:label xml:lang="en">nominal maximum active power</rdfs:label>
                <rdfs:domain rdf:resource="#CHPModule"/>
                <rdfs:range rdf:resource="http://www.s-ten.eu/sten-PhysicalQuantitySpace#ActivePower"/>
                <rdfs:comment xml:lang="en">nominal maximum active power (W)</rdfs:comment>
        </owl:ObjectProperty>

        <owl:ObjectProperty rdf:about="#nominalMaximumThermalPower">
                <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
                <rdfs:label xml:lang="en">nominal maximum heating power</rdfs:label>
                <rdfs:domain rdf:resource="#CHPModule"/>
                <rdfs:range rdf:resource="http://www.s-ten.eu/sten-PhysicalQuantitySpace#Power"/>
                <rdfs:comment xml:lang="en">nominal maximum thermal power (kW)) </rdfs:comment>
        </owl:ObjectProperty>
        <owl:ObjectProperty rdf:about="#nominalHeatToPowerEfficiency">
                <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
                <rdfs:label xml:lang="en">nominal heat to power efficiency</rdfs:label>
                <rdfs:domain rdf:resource="#CHPModule"/>
                <rdfs:range rdf:resource="http://www.s-ten.eu/sten-core#Real"/>
        </owl:ObjectProperty>
        <owl:ObjectProperty rdf:about="#typeOfHeatingMedium">
                <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
                <rdfs:label xml:lang="en">type of heating medium</rdfs:label>
                <rdfs:domain rdf:resource="#CHPModule"/>
                <rdfs:range rdf:resource="http://www.s-ten.eu/sten-core#Integer"/>
                <rdfs:comment xml:lang="en">0: not applicable; 1: water; 2: steam; 3: air; 99: other</rdfs:comment>
        </owl:ObjectProperty>

        <!--
        Measured values and time dependent values, respectively     -->
        <owl:Class rdf:about="#CHPModuleAtInstant">
                <rdfs:subClassOf rdf:resource="http://www.s-ten.eu/sten-core#ClassicalObjectAtInstant"/>
                <rdfs:comment xml:lang="en">A CHP at an instant in time</rdfs:comment>
        </owl:Class>
        <!-- Measured active power (W) -->
        <owl:ObjectProperty rdf:about="#measuredActivePower">
                <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
                <rdfs:label xml:lang="en">measured active power</rdfs:label>
                <rdfs:domain rdf:resource="#CHPModuleAtInstant"/>
                <rdfs:range rdf:resource="http://www.s-ten.eu/sten-PhysicalQuantitySpace#ActivePower"/>
        </owl:ObjectProperty>
        <!-- Measured thermal power (W) -->
        <owl:ObjectProperty rdf:about="#measuredThermalPower">
                <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
```

Fig. 6. Excerpt of ontology for Combined Heat and Power plants.

```
<?xml version="1.0" encoding="UTF-8"?>
<BrowseRequest>
  <SessionID>012154782</SessionID>
  <ObservedBy>http://www.fgh.de/sten/DER.xml#MDevice_CHP_1</ObservedBy>
  <Observes>http://www.fgh.de/sten/DER.xml#CHP_1</Observes>
   <PhysicalProperty>http://www.fgh.de/sten/CHP.xml#ActivePower</PhysicalProperty>
  <After>1201508970325</ After>
  <Before>1201512570325</Before>
</BrowseRequest>
```

Fig. 7. Browse request example.

```
<?xml version="1.0" encoding="UTF-8"?>
<BrowseResponse>
 <ObservedBy>http://www.fgh.de/sten/DER.xml#MDevice_CHP_1</ObservedBy>
 <Observes>http://www.fgh.de/sten/DER.xml#CHP_1</Observes>
 <PhysicalProperty>http://www.labein.es/sten/CHP.xml#Active_Power</PhysicalProperty>
  <Scale>http://www.fgh.de/sten/SIUnits.xml#Watts</Scale>
  <DataTimeValue>
    <TimeSpec>1201508970325</TimeSpec>
    <DecimalValue>25400</DecimalValue>
  </DataTimeValue>
  <DataTimeValue>
    <TimeSpec>1201508932630325</TimeSpec>
    <DecimalValue>30000</DecimalValue>
  </DataTimeValue>
  <DataTimeValue>
    <TimeSpec>1201508932690325</TimeSpec>
    <DecimalValue>31200</DecimalValue>
  </DataTimeValue>
  <BrowseResponseMessage>
    < SuccessIndicator>Browse succeeded</SuccessIndicator>
  </BrowseResponseMessage>
</BrowseResponse>
```

Fig. 8. Browse response example.

systems (servers) that aggregate other S-TEN systems, where the high level S-TEN system provides low level systems' aggregated information and functionalities.

Currently, an operator can access "level 1" S-TEN systems in order to browse data and events and monitor data of a DER. This implies that he only sees the contribution of one DER upstream the feeder per started client application. A future control system embedded in an upper level could aggregate data of all available S-TEN systems within the control area and be in charge of sending the corresponding operation commands to "level 1" S-TEN systems in order to balance the system.

Three servers have been implemented in the application: one for a wind turbine, a second for a PV plant and a third for a CHP unit (Figure 10).

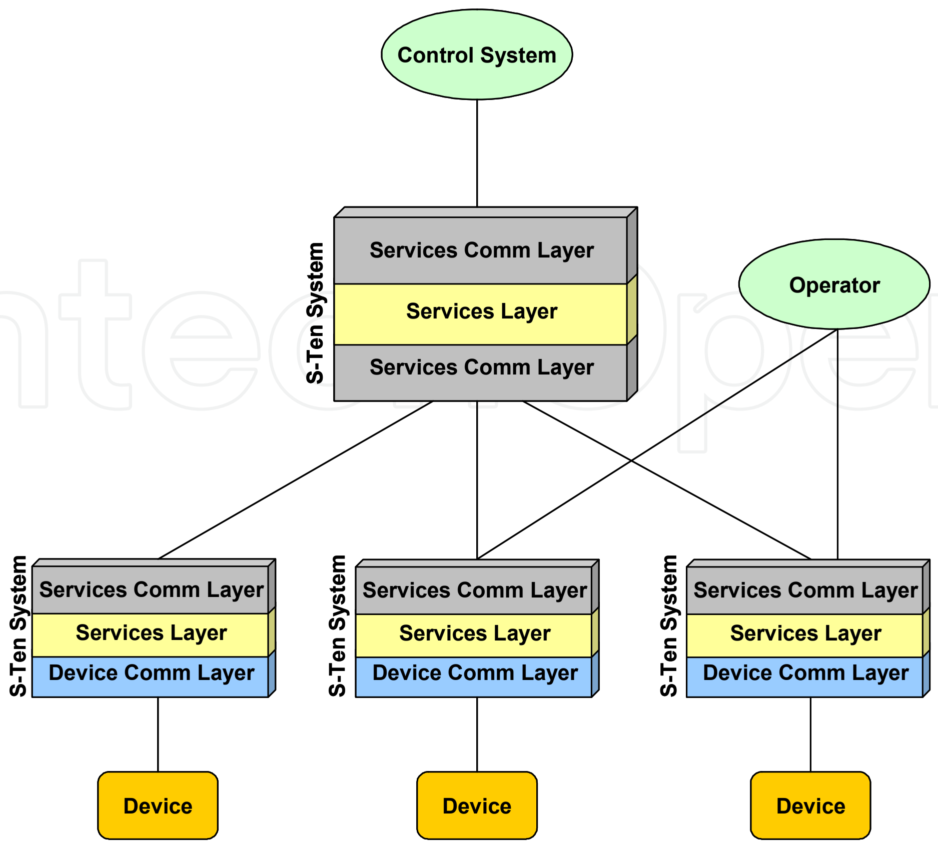

Fig. 9. Hierarchical S-TEN system.

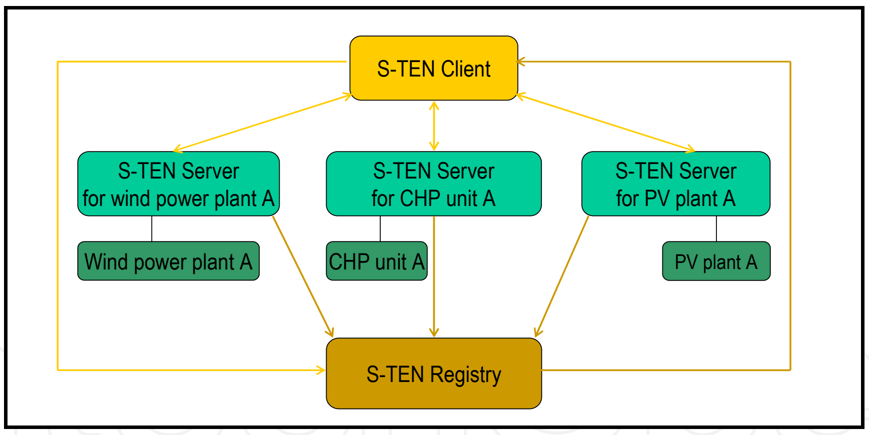
S-TEN Client
S-TEN Server for wind power plant A
S-TEN Server for CHP unit A
S-TEN Server for PV plant A
Wind power plant A
CHP unit A
PV plant A
S-TEN Registry

Fig. 10. Design of the prototype application.

The S-TEN server registers its monitored resource in the S-TEN Registry as soon as it is in operation. The registration file contains both the URL of the resource and the services it offers. Each resource sends its data via a communication interface to the S-TEN server where the data is stored in a database.

The registry can be queried for its registered DERs with the option to ask for all DERs, a special DER type or a DER belonging to a certain system area. As result of this query a list of all registered DERs matching the searching criteria is sent back. The user can now select a DER and access it via its URL known from the self-description file held in the S-TEN registry.

Besides, the web services of the selected DER can be viewed and with the help of the semantic web service validator it can be checked if the offered services match the ones the client understands. If the client and server services match, the user can login the DER of interest.

After successful connection with the DER that is after authorisation and authentication the DER tree is displayed revealing the components of the DER, its measured values along with the measurement devices that measure the values and the warnings and alarms available for the measured value. Within the client application data of the DER can then be monitored and browsed and event subscriptions can be performed leading to a notification when the event occurs. When a web service such as "Browse" is selected, a SOAP query is launched in which the data of interest, e.g. the actual power of wind turbine C belonging to wind power plant A, is held. The SOAP query is sent to the S-TEN server of the requested DER which retrieves the data of interest and sends it back to the client. If a control operation is performed (e.g. drive up a CHP), the way of interaction is the same as described for the monitoring data service only that the client gets a status information of the executed service. The responses and requests can be monitored on request.

Event handling is implemented modeling notifications that are sent when warnings and alarms coming from legacy systems occur. E.g. if a wind turbine is equipped with IEC 61850 IEDs (Intelligent Electronic Device) providing alarms, these events will be made available in the prototype application.

Based on semantically interpretable information rules can be defined enabling automatically derivations and subsequent actions, e.g. switching on or off a disconnector within a ring where a short-circuit occured. Besides, observations and data derived from network resources can be linked together and activated with formal rules in order to provide best practise support for the user.

4.2 Secondary control of microgrids

A microgrid can be defined as an integrated Power Delivery system consisting of interconnected loads and distributed energy resources which as an integrated system can operate in parallel with the grid or in an intentional island mode.

It is important for microgrids that the integration of their resources into the LV (Low Voltage) grids, and their relation with the MV (Medium Voltage) network upstream, will contribute to optimize the general operation of the system. For that purpose a hierarchical system control architecture has been designed, that comprises three critical control levels:

- Local Microsource Controller (MC) and Load Controllers (LC): It responds in milliseconds and uses local information and the requests from the microgrid system Central Controller (MGCC).
- Microgrid system Central Controller: It is the interface with the distribution network and responds to the requests of the Distribution System Operator and optimizes the operation of the equipment that it controls.
- Distribution Management system (DMS): The Distribution Management System is the replica of the Energy Management System (EMS), but for managing distribution networks instead of transmission networks.

In this application a secondary power-frequency control problem is addressed, such that any variation in the power exchange between the Microgrid and the distribution network is detected and the plan to recover from that situation is generated. Once the plan has been generated, the assigned resources are monitored and any deviation from its scheduled behaviour raised and corrected.

Use of the ontology

In order to monitor an industrial process it is necessary to have an operating plan for the physical resources participating in the process, and data from sensors within the system. These are used together to check that everything is taking place according to the plan. Deviations from the plan are detected, and these may require a quick reaction to ensure the safe or economic continuation of the process. The S-TEN project has defined an ontology which covers:

- an observation and measurement activity;
- a time history of a physical property;
- definition of features of a time history.

This ontology is being used to monitor changes of the output active power of generators within a MicroGrid. The monitoring process has two stages, firstly the measurement of values and their recording using RDF, and secondly the monitoring of these values to make higher level statements about the state of the process which are used by the supervisory control to modify the operating plan.

A simple example of monitoring is the detection of changes. The monitoring activity MonitoringActivity_2_OfGenerator_01 is tasked with looking for changes in the output active power of Diesel_01. The following statements are input to the activity and specify what it does.

```
:MonitoringActivity_2_OfGenerator_01
 a                             sten::MonitoringActivity;
 sten:monitors                 :activePowerOfDiesel_01;
 sten:hasMonitoringCriterion :ChangeCriterion1.
```

The time history activePowerOfDiesel_01 is defined as:

```
:activePowerOfDiesel_01
 a                             sten:TimeVariationOfPhysicalProperty;
 rdfs:subPropertyOf            sten:outputActivePower;
 rdfs:domain                   :Diesel_01.
```

A change event is reported (sent as a message or added to the journal formula) if the generator output gets bigger than the 36 kW threshold. Hence the monitoring criterion is defined as follows:

```
:ChangeCriterion1
 a                             sten:ChangeCriterion;
 sten:setlevel                   [scale:kilowatt 36].
```

If the measurement crosses this level between 13:10and 13:20 on 2008-07-22, then the monitoring activity reports the event as follows:

```
[sten:partOfTimeVariation  : activePowerOfDiesel_01;
 rdfs:domain
  [sten:during [t:hasBeginning [t:inXSDDateTime 2008-07-
22T13:10]];
               [t:hasEnd       [t:inXSDDateTime 2008-07-
22T13:20]]]]
  a  :ChangeCriterion1.
```

Description of the control scenario

The technology has been applied for the secondary regulation control of the MicroGrid shown in figure 1. A typical scenario is as follows:

1. A sudden unbalance between generation and load is dealt within the short term by the modification of the exchange of energy between the MicroGrid and the Distribution Network.
2. The controller of the MicroGrid then reacts to the situation and dispatches (instructs) DieselEgine_01 to increase power to 36 kW in order to return the system to the normal balance between generation and load.
3. As soon as the generator is scheduled a "Scheduling Monitoring activity" is started. The main objective of this application is to monitor the ability of DieselEgine_01 to fulfil its assignment. It may not be able to do so, because of a break down of the generator, a loss of efficiency caused by the need to support CHP, or some other reason.
4. If the scheduling application detects that the plan cannot be fulfilled it reports the problem.
5. The plan Plan_Generator_01_1074528964 is defined as the class of time history that:
6. is the outputActivePower of DieselEgine_01;
7. begins at the current time – millisecond count 1074528964;
8. ends when a final output of 36 kW has been reached or when the expected time to achieve it has been exceeded.

The "Scheduling Monitoring activity" assigned to this task is concerned with the time history of active power of DieselEgine_01 beginning at the current time and ending when the new required output level has been reached. The application reports when the requirement has been fulfilled, i.e. when a member of the required class has been found, or when it is clear that the requirement cannot be fulfilled.

The specification of the monitoring activity is as follows:

```
:MonitoringActivity_Generator_01_1074528964
 a                            sten:MonitoringActivity;
 sten:monitors                :activePowerOfDieselEgine_01;
 sten:hasMonitoringCriterion  :Plan_Generator_01_1074528964.
```

The knowledge stored in the ontology greatly facilitates the development of decision support systems and in this case only one simple rule has to be programmed to detect that DieselEgine_01 has achieved its objective. At time T, the time history from the start of the plan to T is assessed as follows:

```
If
activePowerOfDieselEgine01_1074528964 is P1 and
```

```
Current output of DieselEgine_01 is P2 and
Ramping capacity of DieselEgine_01 is R and
Current Time is T and
P2 >= 36 kW and
T < 1074528964 + (36 - P1)/R
Then
DieselEgine_01 has successfully fulfilled his planned schedule.
```

This can be reported by the statement:

```
activePowerOfDieselEgine01_1074528964_T
 a                            :Plan_Generator_01_1074528964.
```

Another rule is used to detect that the DieselEgine_01 can not arrive at its scheduled value and that another generator has to be re-scheduled to cover the difference. In this case, as in the previous rule, design knowledge such as the "ramping" capacity1 of the generator is used to detect the situation. The rule used by the application is as follows:

```
IF
activePowerOfDieselEgine01_1074528964 is P1 and
Current output of DieselEgine_01 is P2 and
Ramping capacity of DieselEgine_01 is R and
Current time is T and
P2 < 36 kW and
T - 1074528964 > (36 kW -P1)/R
Then
Set DieselEgine_01 as unavailable for further scheduling and
schedule anotherGenerator to generate 36 minus P2 kW.
```

5. Outlook

The integration of innovative technologies used by the Semantics Web Community has been proved to be effective for the control of industrial applications and in particular for the control of a MicroGrid. Ontologies derived from standards have been demonstrated to pave the way towards building complex applications by integrating powerful representation mechanisms together with complex decision support systems. The prototype applications have demonstrated that building decision support tools using rule based techniques on top of standard ontologies is a promising way to facilitate the coding of industrial applications and to customize them to the user needs.

Currently the integration of electrical vehicles into the electrical power system is a challenging task that is worked on worldwide. Controllable electrical vehicles could contribute to a reliable and secure power system operation by charging their batteries when the energy demand is low and feeding back energy when the energy demand is high. Applying Web technologies and agent technologies in combination with ontologies and rules to Vehicle To Grid (V2G) applications is a promising approach for realizing the smart grids of the future.

[1] The ramping capacity represents the amount of Power per Time unit that a generator can raise its power.

6. Acknowledgment

The research described in the paper has been carried out within the S-TEN project ('Intelligent Self-describing Technical and Environmental Networks', www.s-ten.eu, FP6-IST-2005-027683) co-funded by the European Community in the Information Society Technologies program.

7. References

[1] S-TEN Project: Intelligent Self-describing Technical and Environmental Networks, Available from: www.s-ten.eu

[2] Berners-Lee, T. Handler, J., Lassila, O. (2001). The Semantic Web, In: *Scientific American Magazine,* Available from: http://www.dblab.ntua.gr/~bikakis/SW.pdf

[3] The Apache Software Foundation: AXIS 2, Available from: http://axis.apache.org/axis2/java/core/

[4] W3C: Web Services Description Language (WSDL) 1.1, Available from: http://www.w3.org/TR/wsdl

[5] W3C: Semantic Annotations for WSDL, Available from: http://www.w3.org/2002/ws/sawsdl/

[6] W3C: OWL Web Ontology Language - W3C Recommendation 10 February 2004, Available from: http://www.w3.org/TR/owl-features/

[7] W3C: RDF Vocabulary Description Language 1.0: RDF Schema W3C Recommendation 10 February 2004, Available from: http://www.w3.org/TR/rdf-schema/

[8] S-TEN consortium(2007): Ontology for self-describing networks (D2.1), Available from: http://www.s-ten.eu/deliverables/D2.1/.

[9} W3C: OWL-S: Semantic Markup for Web Services - W3C Member Submission 22 November 2004, Available from: http://www.w3.org/Submission/OWL-S/

[10] W3C: Web Service Modeling Ontology (WSMO) - W3C Member Submission 3 June 2005, Available from: http://www.w3.org/Submission/WSMO/