Tekes Project

SmartResource

Deliverable D1.1

Resource State/Condition Description Framework (RS/CDF)

Industrial Ontologies group
Agora Center, University of Jyväskylä

June-August, 2004
Jyväskylä, Finland

SmartResource Project

Technical report

“*Deliverable 1.1, Resource State/Condition Description Framework*”

1. Introduction

Original idea of Semantic Web as next-generation of the Web assumes that besides existing content there is a conceptual layer of machine-understandable metadata, which makes the content available for processing by intelligent software, allows automatic resource integration and provides interoperability between heterogeneous systems. To be understood by software application, semantics must be presented explicitly in some form, which would allow intelligent information processing substituting human. Such semantic description is a metadata (data about data), attached to a resource. Addressing these problems W3C consortium has started Semantic Web Activity, which resulted in development of *Resource Description Framework* (RDF) as a basic model for semantic descriptions.

The RDF [Klyne & Caroll, 2004, Manola & Miller, 2004] is a system for making statements about Web resources. Each RDF statement is a triple (subject-predicate-object) and may be interpreted as a subject that has a predicate with value object. Also RDF provides a well-defined framework to make statements about statements, which is called reification. According to traditional view a Web resource contains some data (either structured in some or other way or unstructured) about certain domain entity(s). Thus an RDF description is in a way a metadata.

Originally the dynamics of Web resources was not seriously considered as a subject for RDF community consideration. Everything was simple. A human creates Web resource and responsible for its annotation. Resource data is stable for some period of time. When essential changes in data are necessary then the human renews the resource. If changes are such that they effect also the metadata then the human will update also appropriate RDF descriptions. Automated metadata updates are becoming very complicated when we are dealing with dynamic Web resources. In this case resource content is generated automatically according to continuously changing content of some databases and there would be really a challenge to accordingly update the metadata, which summarizes every new state of the resource [Benjamins at al., 1999]. However still these dynamic resources are traditionally informational in a sense that the informational content of original databases is collected by humans and from time to time maintained by humans, which can lead to optimistic assumption that RDF (of course with appropriate ontological support) has enough means already to describe such “discrete” dynamics.

More complicated case is when we are dealing with industrial resources (machines, processes, personnel, etc.), which dynamics is often naturally continuous even if monitored during discrete time intervals. The challenge is to consider such resources as Web resources and thus as subject for semantic annotation as it was discussed in [Kaykova et. al., 2004]. This approach is the part of going-on SmartResource project “Proactive Self-Maintained Resources in Semantic Web” [SmartResource, 2004] leaded by Industrial Ontologies Group [IOG, 2004]. Industrial resources can be linked to the Web by special adapters and sensors. Sensors keep track of data about internal states of the resource and adapters help to make this data readable for external applications. It is obvious that RDF is not very suitable for making statements about continuously changing resources. Another problem is reification, which deals with dynamically changed statements, e.g. descriptions for certain processes or trends taking into account that such descriptions are often context-dependent. In [Kaykova et. al., 2004] it is supposed that semantic descriptions for the industrial resources state and conditions should be generated automatically by appropriate Web-services to be further used when appropriate for predictive maintenance of the original industrial resources.

The goal of this paper is to present *Resource State/Condition Description Framework (RSCDF)*, as an extension to RDF, which introduces upper-ontology for describing maintenance-oriented characteristics of resources: states and correspondent conditions, dynamics of state changes that happen, target condition of the resources and historical data about previous states. Resources (e.g. devices) are assumed to have their own state presented as RSCDF descriptions. These descriptions are used by external applications (e.g. remote diagnostics) that support RSCDF and are able to process data presented in such format. Introduction of RSCDF allows solving problems of interoperability and resource heterogeneity. Enabling RSCDF can be considered as an important stage of the SmartResource project activities [SmartResource, 2004].

Very basic view to the role of RSCDF for the SmartResource concept is shown in Figure 1.

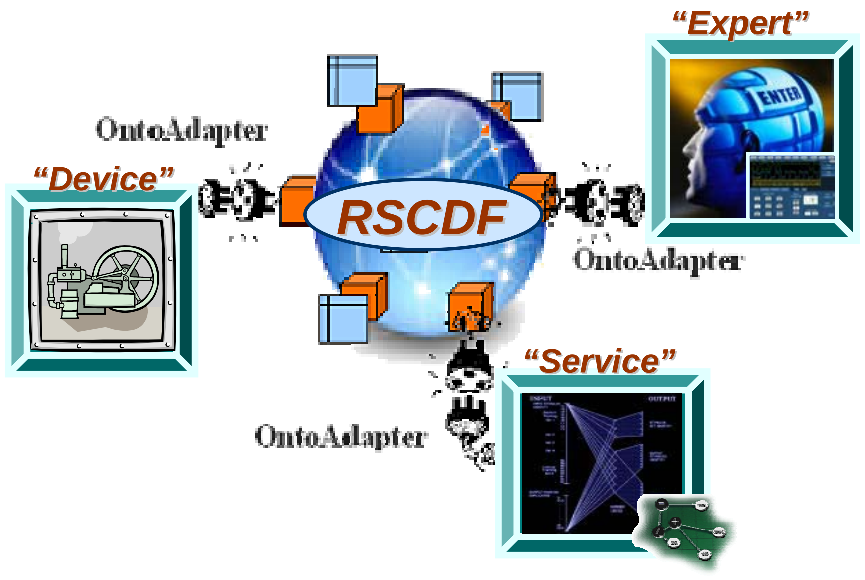


Figure 1 – SmartResource: collaborative environment for field devices, Web-services and human experts with RSCDF-based metadata exchange [SmartResource, 2004]

Data (e.g. diagnostics query) from some field "*Device*" is automatically being translated to RSCDF due to adapter, which is linked to the appropriate ontology. RSCDF will contain ontologically standardized description of the device state (temporal track of the values of parameters taken by sensors) in a form suitable for machine processing and free from the specifics (nature, producer, etc.) of the device itself. After the stage of a suitable "*Expert*" discovery in P2P network of various Web resources the RSCDF query can be shown to the expert in diagnostics (specific adapter will visualize RSCDF to the human) who can return diagnosis (translated automatically back to RSCDF) to the device. Thus device (actually some agent on behalf of the device) will collect some history of device states which is labeled with expert diagnoses. Such history sooner or later will be enough as training set for some machine learning algorithm, which can be represented as Web “*Service*”. Thus labeled device state history in RSCDF will be sent to an external machine-learning service; service will build a model (e.g. neuro-fuzzy network); service is ready for remote diagnostics of the device instead of querying expensive experts for the diagnostics. So RSCDF is required to be a semantically rich RDF-based language for enabling interoperability within global systems for automated online condition monitoring and remote diagnostics of heterogeneous field devices.

2. RSCDF extensions

Resource State/Condition Description Framework (RSCDF) extends RDF to make it expressive enough for the needs of automated resource monitoring and maintenance. Of course we are assuming that there are some Web-accessible industrial resources, which will need monitoring and maintenance. Consider two major extensions: temporal and contextual, which are described in more details below.

2.1 Temporal RSCDF extension

Resource maintenance: a need in temporal metadata

RSCDF ontology, in order to be a data format that supports advanced resource maintenance techniques, must contain temporal concepts as a basis for temporal metadata. Efficient resource maintenance must be based not only on analysis of static resource states, but take into account temporal relations between those states, too. This enables us to have a broader view on the situation: sometimes just considering temporal evolution of relations between different resource states/symptoms can give us a hint as to precise diagnostics. Especially important in this approach is to have ready temporal scenarios of possible critical situations that can occur. Any particular situation can be compared with existing scenarios and possibly classified as belonging to one of them [Ryabov & Terziyan, 2003]. Thus, RSCDF must contain a temporal extension to enable intelligent decision making systems based on temporal diagnostics, i.e. diagnostics based on temporal data. Generally, temporal diagnostics is one important area of application of temporal representation and reasoning formalisms. It includes medical and industrial diagnostics, diagnostics in field device management, etc. [Terziyan & Ryabov, 2003].

Analyzing recent research efforts related to automated monitoring and maintenance of organizations based on multi-agent systems [Jonker et. al., 2001], it becomes clear that automated maintenance of the abstract resource will need means and data models for accumulation of its state history – a sequence of resource states over a time. Thus, it is more correct to describe an object by its history, than by its current state.

To summarize, the three main possibilities that the temporal RSCDF extension must have are: *temporal marker*, *temporal relation* and *temporal trace*. The first assumes assigning to any resource state a temporal stamp; the second assumes defining a temporal relation between two resource conditions. The last one, temporal trace will be used as a container for sequences of resource states – the dynamics of their change through time.

Standardization efforts for temporal metadata

As most comprehensive standardization activity related to temporal metadata, DAML-Time effort [DAML-Time] can be mentioned. This is a collaborative project, which is led by Jerry Hobbs and is a part of the DARPA Agent Markup Language (DAML – [DAML]) project. The DAML-Time project “aims to develop a representative ontology of time that expresses temporal concepts and properties common to any formalization of time” [DAML-Time].

The DAML-Time effort originated from the significant needs of the Semantic Web community in the standardized temporal metadata, ontologies [Hobbs, 2002].

In the development of the RSCDF temporal extension we will orient ourselves at the results of the DAML-Time effort and most obviously the *temporal marker* will be represented by *TemporalThing* - a top-level concept in the DAML-Time ontology (see description of the latter and links to the documentation in [Pan & Hobbs, 2004]). This adoption is made by reason of RSCDF compatibility with widely-adopted standards and by reason of reuse of world-wide expertise in temporal metadata. As yet, the DAML-Time ontology has been developed to the quite vast extent, which makes it ready-to-use for real-life implementations (see [DAML-Time ontology]).

Among other comparably widely-adopted standards that contain aspects of temporal metadata is Dublin Core Metadata Initiative [DublinCore]. However, the temporal metadata elements that it defines (see DC:date, DC:created, DC:dateAccepted, DC:dateCopyrighted, DC:dateSubmitted, DC:Period, DC:Coverage elements in [DCMI Metadata Terms]) are too poor to be used in RSCDF. It is obvious, because the Dublin Core ontology is dedicated to other purposes than RSCDF, namely, for description of digital documents. Hence, Dublin Core cannot be used as a base for temporal extension of RSCDF.

One more effort related to the temporal metadata standardization and which is worth considering is TimeML – Markup Language for Temporal and Event Expressions [TimeML]. This project is funded by the ARDA organization and lead by James Pustejovsky. TimeML aims at developing “a robust specification language for events and temporal expressions in natural language” [TimeML]. Recently (April 2004) the project team issued a series of documentation supporting this standardization effort, particularly TimeML Specification (v1.1): see [TimeML docs] for details. Obviously, the initial orientation of TimeML to the natural language makes it inappropriate to utilization for the temporal RSCDF extension. The joint publication of the leaders – Jerry Hobbs and James Pustevjovsky - of the two mentioned projects (DAML-Time and TimeML respectively) (see [Hobbs & Pustejovsky, 2003]) makes an attempt of defining mappings between those two standards. The results of this cooperative analysis are still obscure and increase our initial opinion about TimeML as a standard not very applicable for the needs of the temporal RSDCF extension. Moreover, the current TimeML specification is fully based on XML standard [TimeML docs] and the absence of any representation that conforms to the Semantic Web based standards makes TimeML unready for its immediate practical utilization. In principle, most tasks (see in [Hobbs & Pustejovsky, 2003]), which TimeML basically addresses, are quite appropriate for the temporal RSCDF extension.

Implementation of the RSCDF temporal extension

In this subsection three requirements that were imposed on the temporal extension are fulfilled. They were namely *temporal marker*, *temporal relation* and *temporal trace*.

Temporal marker or temporal stamp assumes adding temporal features to resource properties (SmartResource, container of statements, resource as a result of a transformation, etc.) with help of instances of the *TemporalThing* class from the DAML-Time ontology, which relates to a resource through the *rscdf:time* property (see Figure 2).

In order to reuse the recent results in the domain of temporal metadata, the DAML-Time OWL ontology will be adopted as it was decided earlier. For the first version of RSCDF, we will orient at the sub-ontology of time, which was developed by Feng Pan and Jerry Hobbs [DAML-Time ontology]. It is also expressed in OWL and is much simpler than the full ontology and “provides all of what most users would need, i.e., a vocabulary for expressing facts about topological relations among instants, intervals, and events, together with information about durations and about dates and times” [DAML-Time].

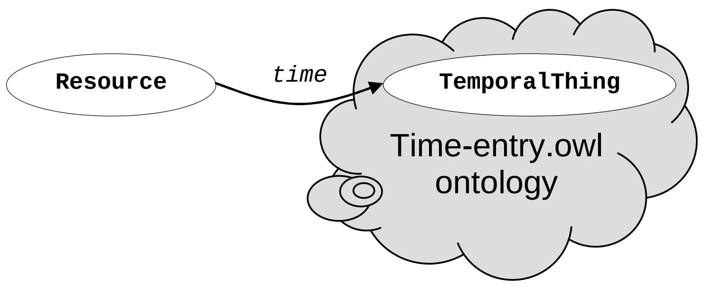


Figure 2 – Adoption of the *time-entry* sub-ontology

In fact, in RSCDF a specific class, which represents the temporal marker, might be declared. It might be, for example, *rscdf:TempMark*. However, the necessity in this potentially redundant concept in the RSCDF ontology is a disputable point. *TempMark* would exist only to carry a specific semantic sense, defining the purposes of the *TemporalThing* utilization more exactly.

As for the last requirement, namely a *temporal trace*, this kind of RDF extension is realized through a specific *rscdf:SR_Container*, which contains a set of triplets (SmartResource-Its_property-Property_value) ordered by a time context (see Figure 3). It is a more universal approach not to distinguish a separate class for the resource history, but provide a possibility of structuring all statements about a resource into custom containers. For this purpose the class *SR_Container* has been introduced in the RSCDF-schema. Similarly, RSCDF-schema does not introduce a separate class for a resource state as a set of triplets that represent values of all resource properties in a given time period. This specific container can be generated as an instance of the mentioned *rscdf:SR_Container*.

In RSCDF-schema *rscdf:SR_Container* is defined as a descendant of *rdf:Container*. As it is well known, *Sequence* assumes ordering its members; hence, it must be used, if the instance container will include a set of statements ordered according to a certain feature, for example in a time order. However, if instances of this feature could form intersections, in this case the container would have to be declared as an instance of *Bag*. Consider an example with a temporal feature as the ordering base: one could describe a value of resource property as a periodically occurring one. The latter can be, for instance, a systematic repair or inspecting of *SmartResource*, known in advance for a certain period in future. For a detailed description of the *rscdf:SR_Container*.

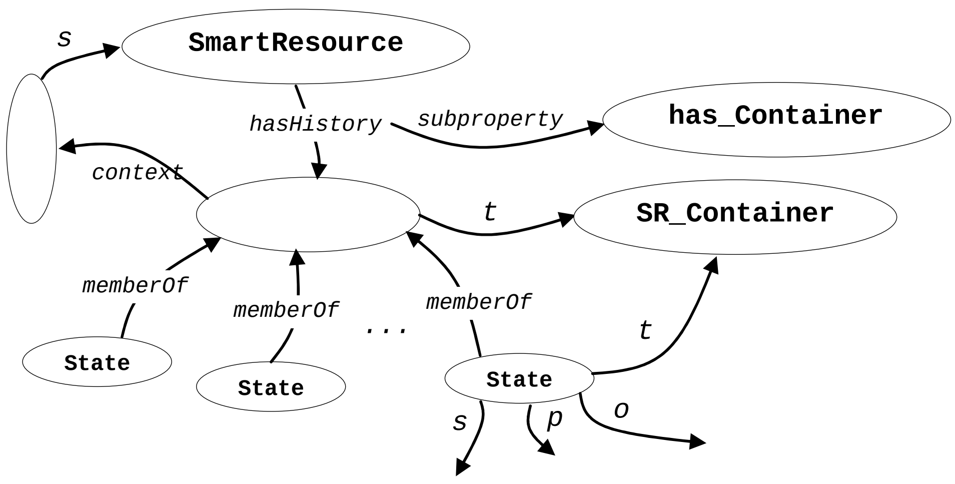


Figure 3 – Temporal trace realization for SmartResource

Now, having all important temporal features realized in RSCDF schema, it is clear, how to query vital data about a certain SmartResource. Most frequently performed requests for resource state can acquire values of a certain number of resource properties (spectrum) for a given point of time (a) and values of a given resource property (spectral constituent) through the time (b). To examine those cases, it is necessary to look a little deeper into the RSCDF schema. For this purpose it is necessary to know, that every resource *State* entry in the history has associated set of parameter value containers, which in their turn have specific related parameter type, denoted by the *ParamType* class.

Now, the case *a* can be performed via the following RQL-query [RQL], if to assume that a resource history will be stored on a Sesame RQL Server [openRDF] (see Figure 4).

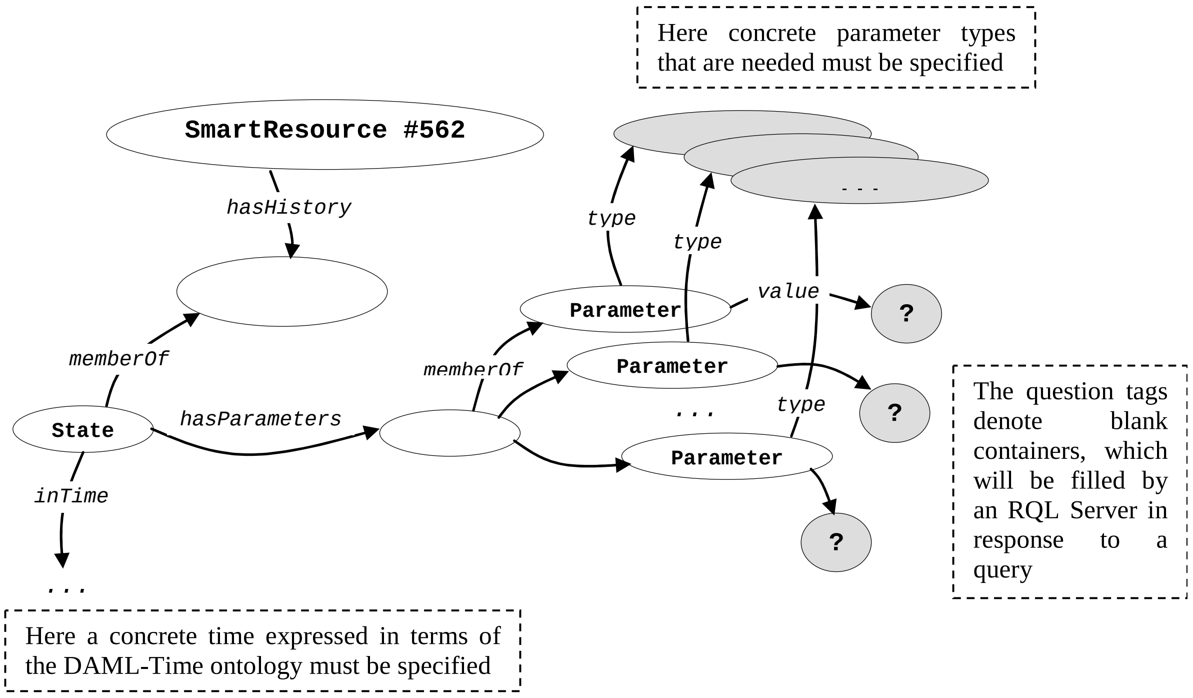

Here concrete parameter types that are needed must be specified
SmartResource #562
hasHistory
type
type
. . .
value
Parameter
?
memberOf
memberOf
Parameter
The question tags denote blank containers, which will be filled by an RQL Server in response to a query
State
hasParameters
. . .
type
?
Parameter
inTime
?
. . .
Here a concrete time expressed in terms of the DAML-Time ontology must be specified

Figure 4 – Querying resource state history, case (a)

In the Figure 4 the necessary RQL-query is represented in a graphical form. It shows the structure of a graph that will be constructed in response to a query for values of a certain number of resource properties (spectrum) for a given point of time. The necessary parameter types specified by a user (in our case it will be a SmartResource) will be put in the containers filled by grey colour in the figure. Also, a user will have to specify a concrete time entity in the query, which must be expressed following the DAML-Time ontology. Sesame RQL Server, which will store a SmartResource history, will reply with corresponding parameter values enclosed in the blank containers instead of question marks.

The case *b* can be performed via the RQL-query, depicted in Figure 5, which is quite similar to the query used for the *a* case.

The requestor in order to fetch values of a given resource property (spectral constituent) through the time, must specify the type of that property/parameter. Optionally, the requesting side can specify a blank container to be filled by time entities, which correspond to the retrieved parameter values. Finally, the server reply will have a form of pairs: *time-entity – parameter-value*.

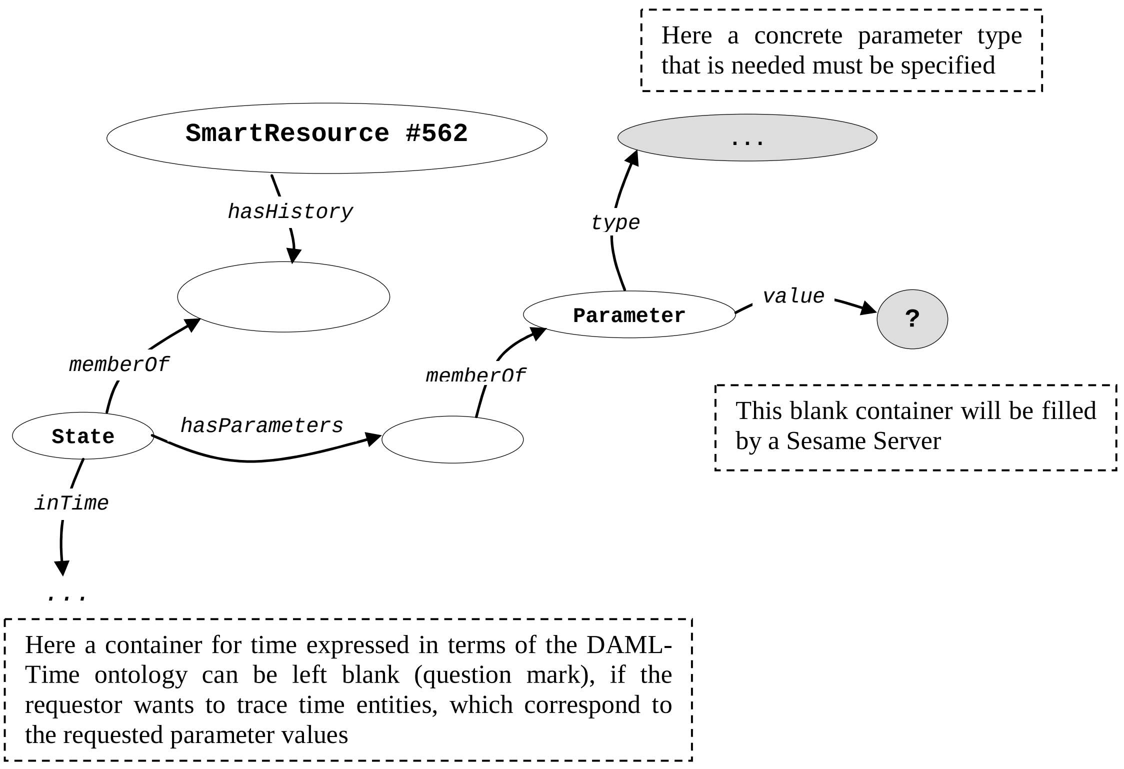

Here a concrete parameter type that is needed must be specified
SmartResource #562
...
hasHistory
type
value
Parameter
?
memberOf
memberOf
State
hasParameters
This blank container will be filled by a Sesame Server
inTime
...
Here a container for time expressed in terms of the DAML-Time ontology can be left blank (question mark), if the requestor wants to trace time entities, which correspond to the requested parameter values

Figure 5 – Querying resource state history, case (b)

Further development of the RSCDF temporal extension

In next versions of the RSCDF temporal extension, Temporal Validity Intervals would be very useful, too. The latter are intended for expressing a time-to-live of a resource property and might be declared using *rdf:Statement* construction. This extension makes possible, for instance, temporal diagnoses (which expire after certain period of time or occur periodically) and opinions. In this case, a necessity in the specific maintenance agent, which changes the RDF-schema according to the time-to-live properties, must be analyzed carefully.

Time aggregation, which relates to a frequency of parameter value acquisition and precision in time, has to be realized within the RSCDF temporal extension, too. This aspect must be supported by Time Aggregation Ontology. Diagnostic Services must have a possibility to vary a time precision in a request for parameters values. Aggregation concepts can be used for realization of the archiving function for resource state that will be accumulated in the resource history ("Resource Lifeblog"). The archiving assumes generalization of the sequence of resource states by one term for the purpose of storage space saving and increasing access time to the resource properties. Some services for diagnostics may need just generalized description of the resource state during the given period of time instead of the detailed log of its precise states.

Current version of RSCDF still supports basic time aggregation. Let us consider an example of how to perform such aggregation using current possibilities of RSCDF. In this example we will focus just on one parameter that is included in the whole resource state among many other parameters-constituents. Assume that the parameter values change through the time as it is shown in Figure 6 below.

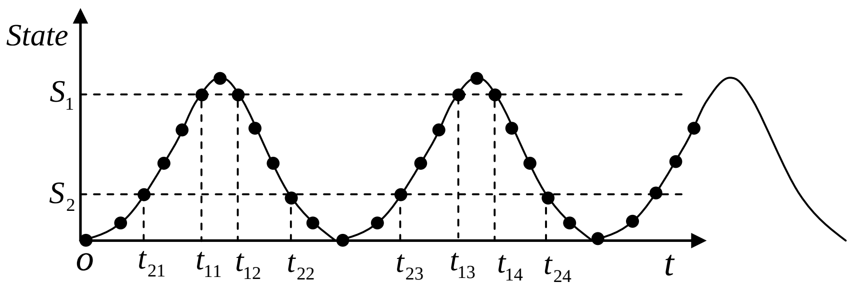


Figure 6 – Simple time aggregation in RSCDF

In Figure 6 the vertical axis denotes the resource states that occur throuh the time (the horizontal axis is dedicated to the time). In the Figure it is shown that the dynamics of the resource state change has a periodic character. Such cases are the most typical to be used for a simple time aggregation in RSCDF. In Figure 6, there are two instances of the resource state (parameter value) occurance: S_1 and S_2. One can see that, for example, state S_1 occures in the time moments $t_{11}, t_{12}, t_{13}, t_{14} ... t_{1n}$, which have a certain period. Hence, if the resource history contained a set of pairs $S_x; t_{xy}$, a long sequence of such pairs can be substituted by its aggregation in form of only one pair $S_x; t_{xagg}$. If to take the S_1 state as an example, its corresponding history sequence can be aggregated into the pair $S_1; t_{1agg}$, where S_1 instance remains the same, but t_{1agg} is a unit of two instant time entities and a time period that corresponds to them. The t_{1agg} concept must be expressed in the terms of the DAML-Time ontology.

The described simple time aggregation can be realized using the available means of the DAML-Time ontology. However, to support advanced time aggregation, like e.g. substitution of the state history sequence by its analytical representation (mathematical formula, etc.), specific concepts in RSCDF must be developed for this purpose.

2.2 Contextual RSCDF extension

[Sayers & Wilkinson, 2003] have discovered that with the original RDF it is quite complicated to represent and query collections of statements in context. They introduced an original higher-level object, the *Snippet*, to hold a fragment of RDF that is about a single subject and made within a particular context. Each snippet may be represented as a bag of reified statements where all the statements are about a single subject and all are made within a particular context. Approach allows also making statements about snippets and snippets about snippets when necessary.

In earlier work [Terziyan & Puuronen, 1997] a Semantic Metanetwork was presented as modelling tool for knowledge, which is valid in certain context and allowing the context itself to be valid within some metacontext, etc. The context itself was considered in a broad meaning (source of information, time, space, etc.). The model includes powerful tools for reasoning with contexts, e.g. interpretation, decontextualization, lifting, discovering context, etc. Most of these properties, which are based on multilevel context, are quite complicated to realize within RDF.

Contextual RSCDF extension implies a possibility to assign additional properties to a series of resource properties (RDF subgraphs) – see Figure 7. This contextual extension can make possible assigning statements to a piece of a resource history (e.g. diagnosis of a doctor). Of course, this extension must provide a possibility of indicating a source of the statement (e.g. Web Service X135-F or Expert An-17).

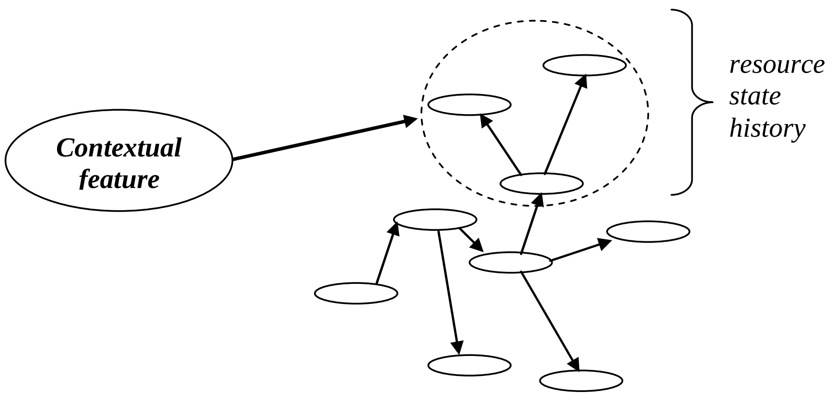


Figure 7 – Contextual RSCDF extension

3. Support for data types in RSCDF

RSCDF is a framework for adding metadata to the actual data. However, what is considered as data? Well, it can by standard data primitives, like Real, Integer, String. They will be enclosed into a necessary RDF-markup. Additionally, RSCDF in order to become universal must be provided with metadata concepts that will allow supporting compound data types, like XML, database formats, pieces of data formatted according a certain protocol. MathML can be used for aggregation of data about the dynamics of parameter values change. Thus, RDF must include upper-classes to denote different existing data formats.

While transforming legacy data format into the RSCDF-based, different levels of a structural detailed elaboration can be used. The initial data format can be totally transformed into the RSCDF-based format, broken down into data primitives. At the other extreme, initial data format can be wrapped by the RSCDF as it is with an indication of its format class. The RSCDF must provide this flexibility to make possible accelerated data exchange between active resources that are based on the same data format. The acceleration of the exchange is achieved through avoiding unnecessary data format transformations inside resource adapters. We also must take into account situations, when a partial interoperability between a pair of resources occurs on a data format level. When, for instance, both resources utilize Relational Databases as a base for data format, but use different structures of the databases.

It is also reasonable to provide for specifying an URI of the actual physical data storage, where the data reside, instead of inserting long data sequences into RDF-files. Is the URI specification developed enough to make such references to all existing data types (consider, for instance, a database, Flat Files)?

This aspect of the RSCDF opens a wide area for different Web Services dedicated for data transformation between different formats, different structural detailed elaboration levels. This aspect is also related to the challenge of *Service Orchestration*, which assumes composing sequences of Web Services, which have data in X format on its input and has data in Y format on its output.

Hence, corresponding ontology must include information about aggregation, normalization of values and method/service/type of a processing that were applied for the initial data. Most likely, these concepts will be included into Adaptation Framework, particularly on its Semantic Level (see Adaptation Layers). They might also enter into Behavioural Framework in form of instructions concerning data processing and transformation. In this context, a challenge of responsibilities distribution comes up: who will be responsible for a management of the data transformation process (search and orchestration of the necessary services): a Device or a Web Service?

The possible view of a part of ontology that reflects data format aspects is shown in Figure 8.

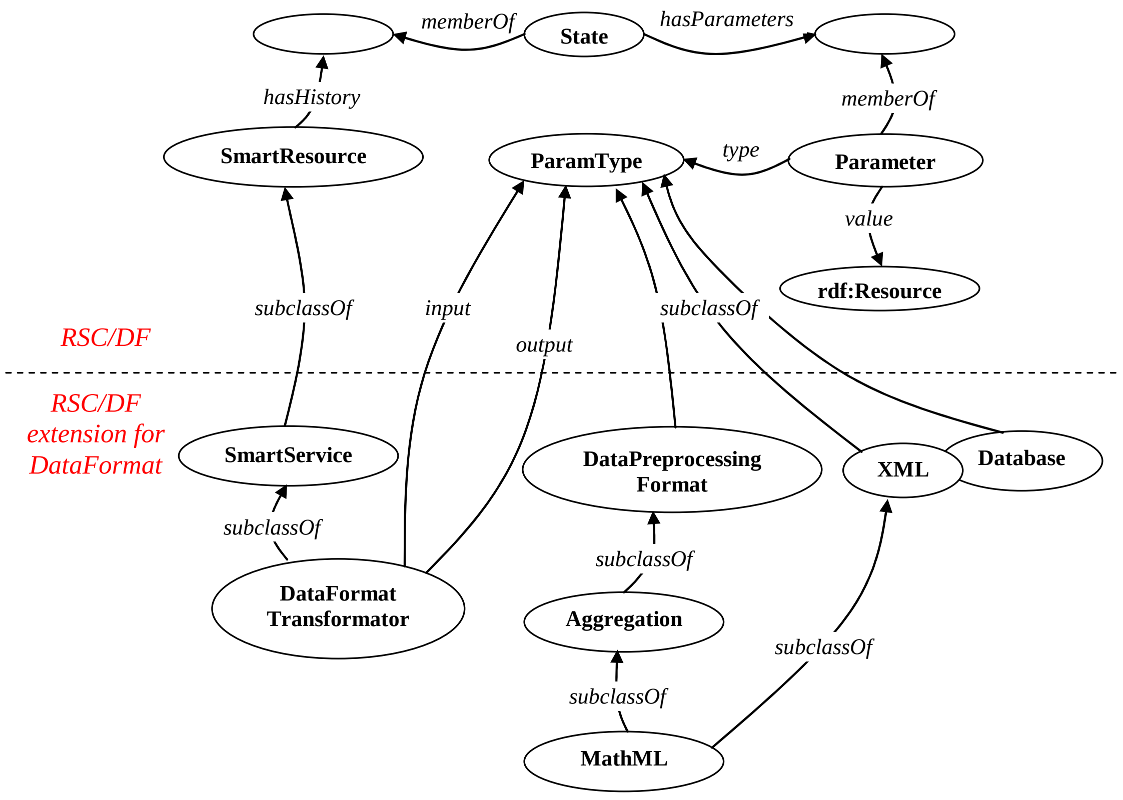


Figure 8 – RSCDF extension for data formats support

The core RSCDF triplets and the ones that belong to the RSCDF extension for DataFormat are separated by a dotted line. One can see that in the extension the SmartResource has a preliminary subclass SmartService, which corresponds to the abstract WebService component in the conceptual *Device-Expert-WebService* interaction triangle. SmartService, in its turn, has a subclass – DataFormatTransformator, which denotes a specific WebService, which performs transformation of one data format to another. Therefore, DataFormatTransformator has two properties: *input* and *output*, which relate it to the ParamType concept.

On the left of DataFormatTransformator, a preliminary ontology branch for different data format types is located. DataPreprocessingFormat serves as an upper class for all intermediate data formats. One can see an example extension of this upper class by the Aggregation class, which corresponds to aggregative data formats, discussed above. Aggregation has a child – MathML, which is a specific data format for aggregated real-time data representation.

It is assumed that descendants of DataFormatTranslator – concrete WebServices transformating data formats – will have their *input* and *output* properties, referred to more specific ParamTypes - its descendants from the mentioned ontology branch.

In the considered RSC/DF extension there are few subclasses of ParamType, which denote general data formats, like XML and Database data format. They are not intermediate data formats on their own; they are rather basic data formats. This part of ontology will be also branchy. The relations between this part of ontology and descendants of DataProcessingFormat are still obscure. As a preliminary, those relations are expected to be *subclassOf* (see this relation between MathML and XML).

4. SSDDM-structure for the RSCDF schema

SSDDM-model (State-Symptom-Diagnosis-Decision-Maintenance) must contain all necessary concepts, their properties and relations between the concepts to support the overall lifecycle of the Maintained Resource. The lifecycle includes the following stages:

1. Sensing resource state (a set of parameter values across a timeline) or reading it from the resource state history (resource state lifeblog). Essential is that the resource state can comprise as its part opinions of the other resources about its state. The resource can grade these opinions by the parameter of their quality.
2. Making preliminary state processing (alarm systems), generating symptoms (preliminary assumption about a diagnosis). Alarm system is a Web Service, which implements an algorithm that makes a decision about a symptom based on the analysis of the resource state.

Below there are different cases of a location of Alarm Service in the general architecture of the maintenance system. Figure 9 depicts a case, when Alarm Service has been configured for a certain field device, for its specific format of state data. In this case the adapter is not involved into the data exchange process between a device and Alarm Service.

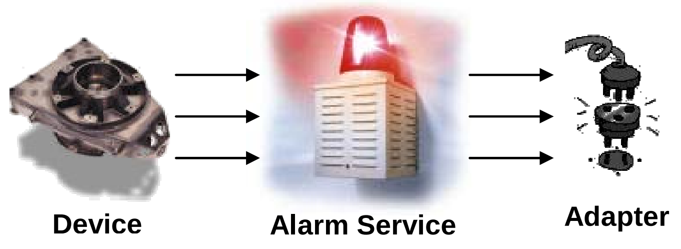


Figure 9 – Case *Device-Alarm-Adapter*

Figure 10 describes other case, when Alarm Service is external ("alien") with regard to Device. For this case, the state of the Device described in the internal standard, when transferred to the "alien" Alarm Service, come through Adapter for transformation into RSCDF standard.

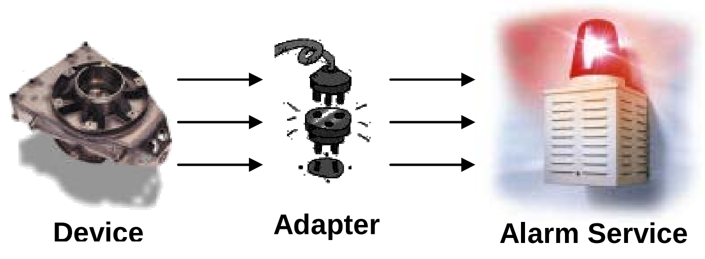


Figure 10 – Case *Device-Adapter-Alarm*

3. Making detailed state processing (diagnostic services, previously generated symptoms can be taken into account) and making final decisions about diagnosis. Diagnosis can be complex (different diagnoses from different sources and distributed in time). Resource might store its "out-patient card", which will contain a history of diseases/diagnoses (resource diseases lifeblog). On this stage a notion of a "Doctor" can be distinguished as someone who sensing a resource state (mark) makes an advice about further actions in order to change the state (mark) to normal.
4. Performing necessary maintenance actions for a given resource based on the previously stated diagnoses. Thus, there must be a service that will generate a *recommendation* (a set of behavioral instructions) for a corresponding resource according to the assigned diagnosis. Maintenance

actions must be put down to the resource history/lifeblog. System must "know" what has been made with it, a cause of changes happened with its parameters.

5. RSCDF-Schema

All concepts declared in the RSCDF-schema will have a namespace rscdfs. The RDF/XML serialization for the whole RSCDF-schema is contained in Appendix A.

5.1 Classes

This section provides a description of all classes declared in the RSCDF-schema. Their hierarchy is depicted in Figure 11.

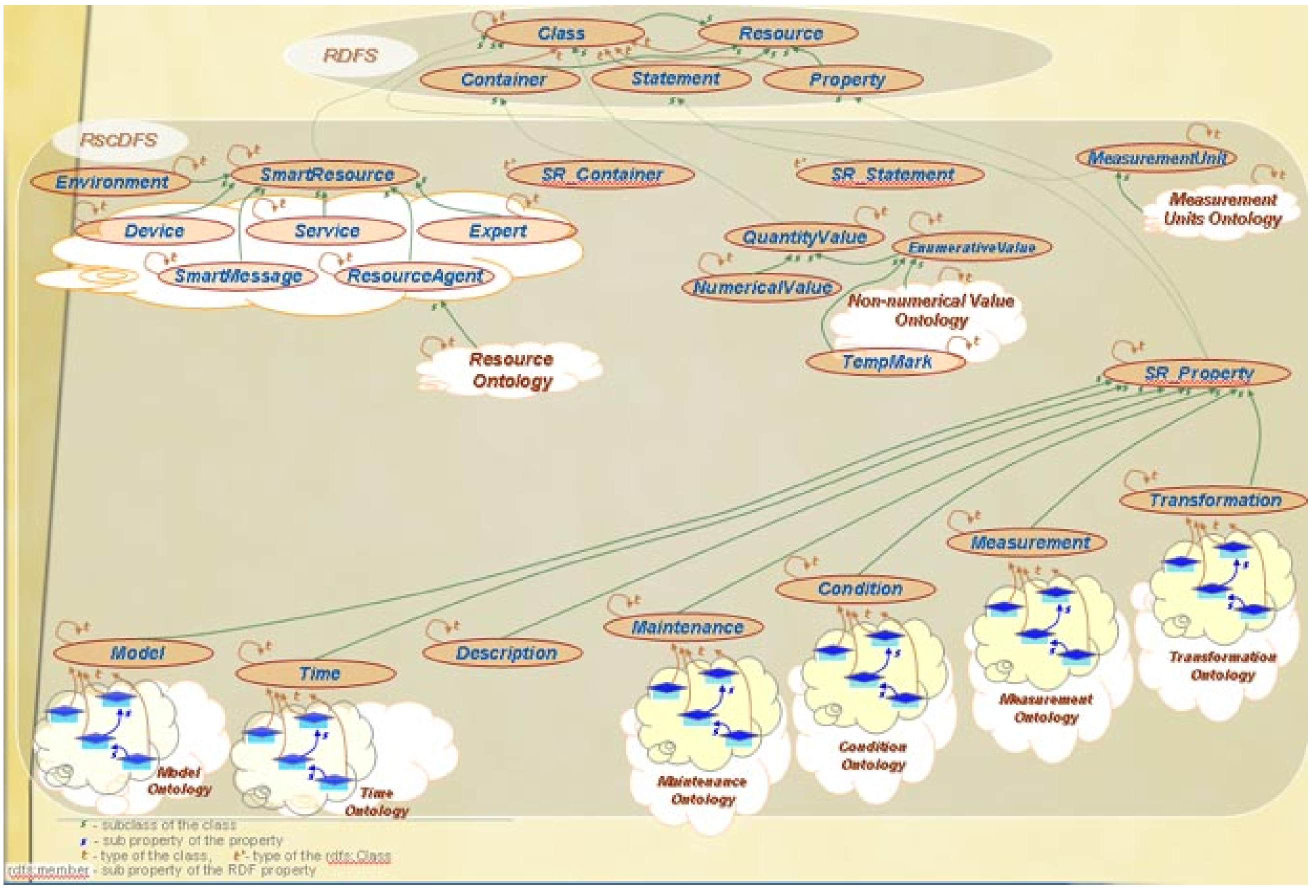


Figure 11 - The hierarchy of classes in RSCDF-schema

As you can see from the Figure, the hierarchy of classes has a specifics: all classes except *rscdfs:SR_Container*, *rscdfs:SR_Statement* and *rscdfs:SR_Property* are subclasses of *rdfs:Class* and are instances of themselves, like *rdfs:Class*.

Family of upper classes for all Smart Resources

This subsection contains descriptions of the upper classes for annotation of all types of Smart Resources that will be integrated in the target system. These classes are *SmartResource*, *Device*, *Service*, *Expert*, *SmartMessage*, *ResourceAgent* and *Environment*. In the Figure 11, this family is included in a white cloud and another cloud – Resource Ontology – represents all descendants of the subject family (see *`subClassOf`* relation from Resource Ontology).

rscdfs:SmartResource

`rscdfs:SmartResource` is an upper class for all concepts that will correspond to SmartResources, which will be integrated into the target system of self-maintained resources.

This class is a subclass of *`rdfs:Class`* and an instance of itself.

rscdfs:Device

`rscdfs:Device` is a direct subclass of *`rscdfs:SmartResource`* and represents an upper ontological concept/class, used for annotation of all devices being integrated into the target system.

This class is an instance of itself.

rscdfs:Service

`rscdfs:Service` is a direct subclass of *`rscdfs:SmartResource`* and represents an upper ontological concept/class, used for annotation of all services being integrated into the target system.

This class is an instance of itself.

rscdfs:Expert

`rscdfs:Expert` is a direct subclass of *`rscdfs:SmartResource`* and represents an upper ontological concept/class, used for annotation of all experts being integrated into the target system.

This class is an instance of itself.

rscdfs:SmartMessage

`rscdfs:SmartMessage` is a direct subclass of *`rscdfs:SmartResource`* and represents an upper ontological concept/class, used for annotation of all kinds of Smart Messages used in the target system. Actually, this class is reserved for further versions of the RSCDF-schema, the notion of Smart Message is described in further versions of Resource State/Condition Description Framework.

This class is an instance of itself.

rscdfs:ResourceAgent

`rscdfs:ResourceAgent` is a direct subclass of *`rscdfs:SmartResource`* and represents an upper ontological concept/class, used for annotation of all kinds of Resource Agents used in the target system. Actually, this class is reserved for further versions of the RSCDF-schema, the Resource Agent corresponds to Maintenance Agent, which will be designed in 2005 as the next step of the target system development. Thus, see details related to *`rscdfs:ResourceAgent`* class in further versions of Resource State/Condition Description Framework.

This class is an instance of itself.

rscdfs:Environment

`rscdfs:Environment` is a direct subclass of *`rscdfs:SmartResource`* and represents an upper ontological concept/class, used for annotation of all kinds of Environments used in the target system. This class is reserved for further versions of the RSCDF-schema, and it corresponds to a runtime environment, where all components of a Smart Resource execute. Details related to *`rscdfs:Environment`* class, see in further versions of Resource State/Condition Description Framework.

`rscdfs:Environment` is an instance of itself.

Family of upper classes for all Smart Resource properties

The family of all Smart Resource properties originate from the upper class – *`rscdfs:SR_Property`*, which is a subclass of *`rdf:Property`*. The more concrete upper classes for properties, which refine/extend *`rscdfs:SR_Property`*, are the following (they are the siblings of *`SR_Property`*): *`Transformation`*, *`Measurement`*, *`Condition`*, *`Maintenance`*, *`Description`*, *`Time`* and *`Model`*. All further declarations of more concrete Smart Resource properties (e.g. *`temperature`* or *`volume`*) will be made as *instances* of these upper classes. While defining the hierarchy of such properties a *`rdfs:subPropertyOf`* relation can be used for this. This aspect is shown in Figure 1 in a form of schematic hierarchies of properties in clouds, located near the corresponding upper class (e.g. Measurement Ontology near *`Measurement`* upper class, etc.).

rscdfs:Measurement

The upper class for Smart Resource properties, which are a part of its state and values of which change dynamically. It can be, for instance, a temperature of a kettle and a volume of water in it. *`rscdfs:Measurement`* is a subclass of *`rscdfs:SR_Property`* and an instance of itself.

rscdfs:Condition

The upper class for Smart Resource properties, which refer to its condition - a statement/label of one Smart Resource about the state of other Smart Resource derived based on analysis of observed facts about that Smart Resource. *Fact* in this context means any statement expressed in format of the RSCDF-schema. *`rscdfs:Condition`* is a subclass of *`rscdfs:SR_Property`* and an instance of itself.

rscdfs:Transformation

The upper class for all types of different transformations, which can be performed over a resource. These properties relate one resource (*initial*) to another one (*resulting*), which was got in consequence of applying the subject transformation. *`rscdfs:Transformation`* is a subclass of *`rscdfs:SR_Property`* and an instance of itself.

rscdfs:Maitenance

The upper class for all types of different maintenance procedures, which can be performed over a Smart Resource with goal of returning its state to normal. This property can be used for logging a history of all maintenance activities performed over a Smart Resource. This history can be taken into account by decision-making tools as a context for diagnostics or recommendation. *`rscdfs:Maintenance`* is a subclass of *`rscdfs:SR_Property`* and an instance of itself.

rscdfs:Description

The upper class for all types of different Smart Resource properties, which serve as descriptions of its static features unlike *`rscdfs:Measurement`* that describes dynamic properties. Static properties of a Smart Resource can be its voltage (for Smart Resources, based on electricity); structural characteristics (e.g., number of motors). *`rscdfs:Description`* covers a large variety of Smart Resource descriptive properties, even those, which relate a Resource to a container (see *`rscdfs:has_Container`*). *`rscdfs:Description`* is a subclass of *`rscdfs:SR_Property`* and an instance of itself.

rscdfs:Time

The upper class for all types of different temporal properties of Smart Resource. One example of such temporal property can be a time of a measurement performed for a dynamic property of smart Resource. *`rscdfs:Time`* is a subclass of *`rscdfs:SR_Property`* and an instance of itself.

rscdfs:Model

The upper class for all types of different properties of a Smart Resource, which relate to its internal underlying model. In case of a Service as a Smart Resource, the model can be, for example, a neural network. *`rscdfs:Model`* is a subclass of *`rscdfs:SR_Property`* and an instance of itself.

Upper classes for values of Smart Resource properties

The core element of resource descriptions made using RDF is the RDF-statement, which consists of subject, predicate and object. Descriptions about Smart Resources will also be created in a form of RDF triplets. So far, we have described subjects (a family of Smart Resource classes) of these triplets and their predicates (a family of Smart Resource properties). This subsection describes the third part of the statements about Smart Resources – objects.

rscdfs:QuantityValue

The upper class for all types of values of Smart Resource properties (descendants of *`rscdfs:SR_Property`*). *`rscdfs:QuantityValue`* is a subclass of *`rdfs:Class`* and an instance of itself.

rscdfs:NumericalValue

The direct descendant of *`rscdfs:QuantityValue`* and an upper class for all types of numerical values of Smart Resource properties. An example of numerical value is a temperature: its values are numbers. This class is compound: it has two structural parts – *`rdfs:Literal`*, which is a metadata container for the numerical value itself, and *`rscdfs:MeasurementUnit`*, which defines the semantics of the unit of that numerical value. *`rscdfs:NumericalValue`* will be used as a class of values of many Smart Resource properties, especially of those, which originate from Measurement Ontology (see Figure 11). *`rscdfs:QuantityValue`* is a subclass of *`rdfs:Class`* and an instance of itself.

rscdfs:EnumerativeValue

This class is an upper class for all types of non-numerical values of Smart Resource properties, all possible values of which can be enumerated as instances. *`rscdfs:EnumerativeValue`* is the upper class in Non-numerical Value Ontology (see Figure 11) and a direct descendant of *`rscdfs:QuantityValue.`* An example of enumerative property value can be a state of a trigger: it can be either switched on (this state will have a corresponding instance in the ontology – instance of a subclass of *`rscdfs:EnumerativeValue`*) or switched off (also will have a corresponding instance). This class is also an instance of itself.

rscdfs:TempMark

This class is an upper class for all classes, which will denote values of temporal properties of a Smart Resource. The characteristics of this class are still unclear: it might be a container of Temporal Entities from DAML-Time ontology or even be substituted by a Temporal Thing class from the same DAML-Time.

`rscdfs:TempMark` is temporarily declared as a subclass of *`rscdfs:EnumerativeValue`*, however its details will be defined more exactly in further versions of the RSCDF-schema. *`rscdfs:TempMark`* is comprised into the Non-numerical Value Ontology and is an instance of itself.

rscdfs:MeasurementUnit

`rscdfs:MeasurementUnit` is an upper class for all types of units that will correspond to numerical values of Smart Resource properties. *`rscdfs:MeasurementUnit`* is in fact a structural part of the *`rscdfs:NumericalValue`* class and relates to it via a property *`rscdfs:unit`*.

`rscdfs:MeasurementUnit` is an upper class in Measurement Units Ontology (see Figure 11) and concrete unit enumerations will be instances of its subclasses. For example, units, which will accompany temperature values, will be enumerated as instances of a TemperatureUnit class – a descendant of the `rscdfs:MeasurementUnit`, like Fahrenheit, Celsius, Kelvin, etc.

`rscdfs:MeasurementUnit` is defined as a subclass of `rdfs:Class` and an instance of itself.

Upper classes for describing a context of statements about Smart Resources

RSCDF provides a means for annotating/describing context statements, which correspond to other statements about a Smart Resource. The latter can be statements about a fact of a performed measurement of a Smart Resource property value or a fact of a condition assigned by one Smart Resource to other Smart Resource. These means include `rscdfs:SR_Statement` and `rscdfs:SR_Container` classes.

rscdfs:SR_Statement

This is an upper class for all reifications of statements about a Smart Resource. This reification is defined in order to be able to express additional facts related to the main statements about Smart Resource. Very important addition is a possibility of expressing a context of a statement about a Smart Resource. `rscdfs:SR_Statement` is a subclass of `rdfs:Statement` and an instance of `rdfs:Class`.

rscdfs:SR_Container

This is an upper class for all containers that include `SR_Statement`s. The instances of this container can represent a context for a statement as a set of other statements. Alternatively, another instance can contain statements about a certain Smart Resource, selected according to special conditions. Thus, `rscdfs:SR_Container` also exists with a purpose of structuring the initial plain set of statements about a Smart Resource. `rscdfs:SR_Container` is a subclass of `rdfs:Container` and an instance of `rdfs:Class`.

5.1 Properties

This section provides a description of all properties declared in the RSCDF-schema. Their hierarchy is depicted in Figure 12.

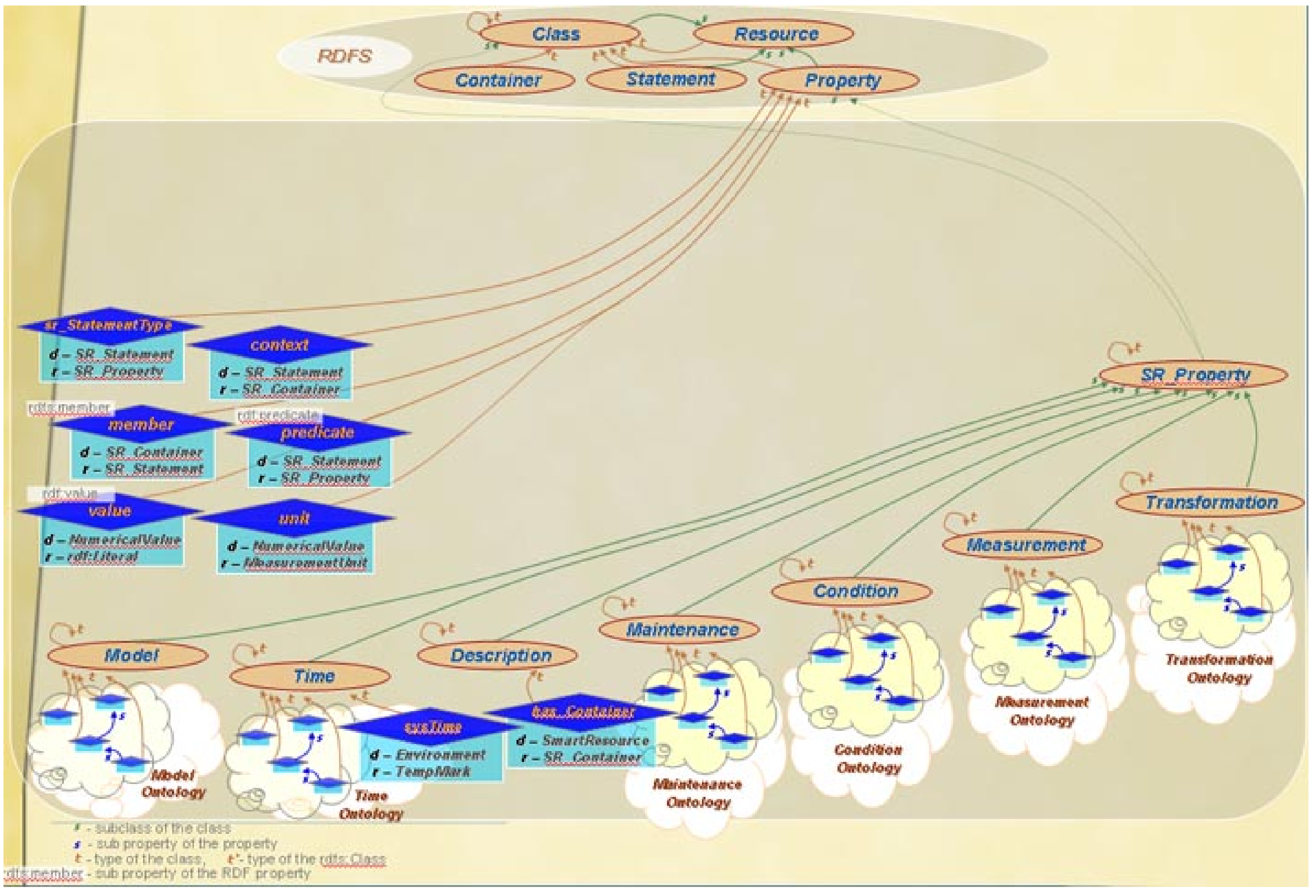


Figure 12 – The hierarchy of properties in RSCDF-schema

Family of Smart Resource properties

The family of Smart Resource properties is represented by RSCDFS concepts, which are declared as instances of the upper classes from the family of upper classes for all Smart Resource properties (see Section 6.1). At the moment, the majority of these properties aren't declared explicitly – they are showed on the RSCDF-schema in the generalized form of reserved ontology branches (see cloud-like regions).Those reserved ontology branches are (see Figure 12): Transformation Ontology, Measurement Ontology, Condition Ontology, Maintenance Ontology, Time Ontology and Model Ontology.

This Section describes in details just concrete properties, defined in the RSCDF-schema. See more information about the reserved ontology branches in Section 6.1.

rscdfs:has_Container

This property allows Smart Resources to have any associated containers of `SR_Statement`s, which can bear any semantic meaning. It can be a certain period of Smart Resource's history in a form of a set of statements about that resource, or other set of such statements, selected according to a specific requirements.

`rscdfs:has_Container` is an instance of `rscdfs:Description` class. Its *domain* is `rscdfs:SmartResource` and *range* is `rscdfs:SR_Container`.

rscdfs:sysTime

This property is defined for specifying an absolute time in the system – instance of `rscdfs:Environment`, where Smart Resource and other components exist. All other temporal markers, which accompany statements about smart Resources, will refer to the temporal instances, which form the original system timeline.

rscdfs:sysTime is an instance of *rscdfs:Time* class. Its *domain* is *rscdfs:Environment* and *range* is *rscdfs:TempMark*.

rscdfs:value

This property is defined for specifying an actual value of the Smart Resource numerical property in a form of string. The *domain* of this property is *rdfs:Literal*, hence the values of this property can represent integer, float, etc. values with additional specification of their data type using XML schema. The *range* of *rscdfs:value* refers to *rscdfs:NumericalValue* – a compound concept, declared in order to be able to specify additional properties for numerical values of Smart Resource properties, like units (see *rscdfs:unit*).

rscdfs:value is a direct instance of *rdf:Property* class and sub-property of the original *rdf:value* property.

rscdfs:unit

This property is defined for specifying units that correspond to actual values of Smart Resource numerical properties (see *rscdfs:value*). An example of the unit can be a Celsius for property *temperature*, all of them are defined in the Measurement Units Ontology and extend the upper class *rscdfs:MeasurementUnit* (*range* of the *rscdfs:unit)*. The domain of *rscdfs:unit* is the compound class *rscdfs:NumericalValue*.

rscdfs:unit is a direct instance of *rdf:Property* class.

Upper properties for describing a context of statements about Smart Resources

rscdfs:predicate

This property overrides original *rdf:predicate* constricting its *domain* to *rscdfs:SR_Statement* and *range* to *rscdfs:SR_Property*. *rscdfs:predicate* is a sub-property of *rdf:predicate* and instance of *rdf:Property*.

rscdfs:sr_StatementType

This property allows specifying a type of the *SR_Statement* by appropriate class-descendant of *rscdfs:SR_Property*. The *domain* of this property is *rscdfs:SR_Statement* and *range* is *rscdfs:SR_Property*. *rscdfs:sr_StatementType* is an instance of *rdf:Property*.

rscdfs:context

This property relates a statement about Smart Resource to the context, in which it was asserted. The context is a container (set) of other SR_Statements, which are statements about separate contextual aspect. *rscdfs:context* is an instance of *rdf:Property* and has *rscdfs:SR_Statement* class as *domain*, *rscdfs:SR_Container* class – as *range*.

rscdfs:member

This property overrides original *rdfs:member* constricting its *domain* to *rscdfs:Container* and *range* to *rscdfs:SR_Statement*. This property serves as a relation of membership of SR_Statement to the particular SR_Container. *rscdfs:member* is a sub-property of *rdf:member* and instance of *rdf:Property*.

6. Examples

Sample instances annotated based on the RSCDF-Schema see in Appendix B and their RDF/XML serialization in the file *resourceDescription.doc*, which accompanies this technical report.

References

[Benjamins et al., 2002]	R. Benjamins, J. Contreras, O. Corcho and A. Gómez-Pérez, The Six Challenges for the Semantic Web, KR-2002 Workshop on Semantic Web, Toulouse, France, 2002.
[DAML-Time]	Official Homepage of the DAML-Time effort, http://www.cs.rochester.edu/~ferguson/daml/.
[DAML]	Official homepage of the DAML program, http://www.daml.org/.
[Hobbs, 2002]	J. Hobbs. Towards an ontology of time for the semantic web. In Proc. of Workshop on Annotation Standards for Temporal Information in Natural Language, LREC2002, Las Palmas, Spain, May 2002.
[Hobbs & Pustejovsky, 2003]	Jerry R. Hobbs and James Pustejovsky. Annotating and reasoning about time and events. In Patrick Doherty, John McCarthy, and Mary-Anne Williams, editors, *Working Papers of the 2003 AAAI Spring Symposium on Logical Formalization of Commonsense Reasoning*, pages 74-82. AAAI Press, Menlo Park, California, 2003.
[DAML-Time ontology]	DAML-Time ontology serialized into OWL/XML: http://www.isi.edu/~pan/damltime/time.owl, http://www.isi.edu/~pan/damltime/time-entry.owl.
[DCMI Metadata Terms]	"DCMI Metadata Terms" - DCMI Recommendation, 14 June 2004, http://dublincore.org/documents/dcmi-terms/.
[DublinCore]	Official website of the Dublin Core Metadata Initiative, http://dublincore.org/.
[IOG, 2004]	Official Web-Site of Industrial Ontologies Group, http://www.cs.jyu.fi/ai/OntoGroup.
[Jonker et al., 2001]	C. M. Jonker, I. A. Letia, J. Treur, Diagnosis of the Dynamics within an Organization by Trace Checking of Behavioural Requirements. AOSE 2001, pp. 17-32.
[Kaykova et al., 2004]	Kaikova H., Khriyenko O., Kononenko O., Terziyan V., Zharko A., Proactive Self-Maintained Resources in Semantic Web, *Eastern-European Journal of Enterprise Technologies*, Vol. 2, No. 1, 2004, ISSN: 1729-3774, Kharkov, Ukraine, pp. 37-49.
[Klyne & Caroll, 2004]	G. Klyne and J. Caroll, Resource Description Framework (RDF) Concepts and Abstract Syntax, W3C Recommendation, http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/, February 2004.
[Manola & Miller, 2004]	F. Manola and E. Miller, RDF Primer, W3C Recommendation, http://www.w3.org/TR/2004/REC-rdf-primer-20040210/, February 2004.

[openRDF]	Official website of openRDF.org - a community that is the center for all Sesame-related development, http://www.openrdf.org/ about.jsp.
[Pan & Hobbs, 2004]	F. Pan and J. Hobbs. Time in OWL-S. In Proc. of First International Semantic Web Services Symposium, 2004 AAAI Spring Symposium series, Stanford University, Palo Alto, California, March 22-24, 2004.
[RQL]	RQL Tutorial, http://www.openrdf.org/doc/rql-tutorial.html.
[Ryabov & Terziyan, 2003]	Ryabov V., Terziyan V., Industrial Diagnostics Using Algebra of Uncertain Temporal Relations, In: *Proceedings of the 21-st IASTED International Multi-Conference on Applied Informatics (AI-2003)*, February 10-13, 2003, Innsbruck, Austria, ACTA Press, ISBN 0-88986-341-5, ISSN 1027-2666, pp.351-356.
[Sayers & Wilkinson, 2003]	C. Sayers and K. Wilkinson, A pragmatic Approach to Storing and Distributing RDF in Context using Snippets, Technical Report HPL-2003-231, Hewlett Packard Laboratories, Palo Alto, California, 2003.
[SmartResource, 2004]	Proactive Self-Maintained Resources in Semantic Web, Presentation of SmartResource Tekes Project, http://www.cs.jyu.fi/ai/ OntoGroup/SmartResource.ppt.
[Terziyan & Ryabov, 2003]	Terziyan V., Ryabov V., Abstract Diagnostics Based on Uncertain Temporal Scenarios, In: M. Mohammadian (ed.), *Proceedings of the International Conference on Computational Intelligence for Modelling Control and Automation (CIMCA-2003)*, February 12-14, 2003, ISBN 1740880684, Vienna, Austria, pp. 327-337.
[TimeML]	Official website of the TimeML project, http://www.cs.brandeis.edu/ ~jamesp/arda/time/.
[TimeML docs]	TimeML documentation, http://www.cs.brandeis.edu/~jamesp/arda/ time/timemldocs.html.

Appendix A: RscDF Schema as RDF/XML

```
<?xml version='1.0' ?>
<!DOCTYPE   rdf:RDF [
                <!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
                <!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-schema#'>
                <!ENTITY rscdfs 'http://www.cc.jyu.fi/~olkhriye/rscdfs/0.2/rscdfs#'>]>
<rdf:RDF
           xmlns:rdf="&rdf;"
           xmlns:rdfs="&rdfs;"
           xmlns:rscdfs="&rscdfs;">


<rscdfs:SmartResource rdf:about="&rscdfs;SmartResource"
           rdfs:comment="Type of itself"
           rdfs:label="SmartResource">
          <rdfs:subClassOf rdf:resource="&rdfs;Class"/>
          <rdfs:subClassOf rdf:resource="&rdfs;Resource"/>
</rscdfs:SmartResource>

<rscdfs:Device rdf:about="&rscdfs;Device"
           rdfs:comment="Type of itself"
           rdfs:label="Device">
          <rdfs:subClassOf rdf:resource="&rscdfs;SmartResource"/>
</rscdfs:Device>

<rscdfs:Service rdf:about="&rscdfs;Service"
           rdfs:comment="Type of itself"
           rdfs:label="Service">
          <rdfs:subClassOf rdf:resource="&rscdfs;SmartResource"/>
</rscdfs:Service>

<rscdfs:Expert rdf:about="&rscdfs;Expert"
           rdfs:comment="Type of itself"
           rdfs:label="Expert">
          <rdfs:subClassOf rdf:resource="&rscdfs;SmartResource"/>
</rscdfs:Expert>

<rscdfs:SmartMessage rdf:about="&rscdfs;SmartMessage"
           rdfs:comment="Type of itself"
           rdfs:label="SmartMessage">
          <rdfs:subClassOf rdf:resource="&rscdfs;SmartResource"/>
</rscdfs:SmartMessage>

<rscdfs:ResourceAgent rdf:about="&rscdfs;ResourceAgent"
           rdfs:comment="Type of itself"
           rdfs:label="ResourceAgent">
          <rdfs:subClassOf rdf:resource="&rscdfs;SmartResource"/>
</rscdfs:ResourceAgent>
```

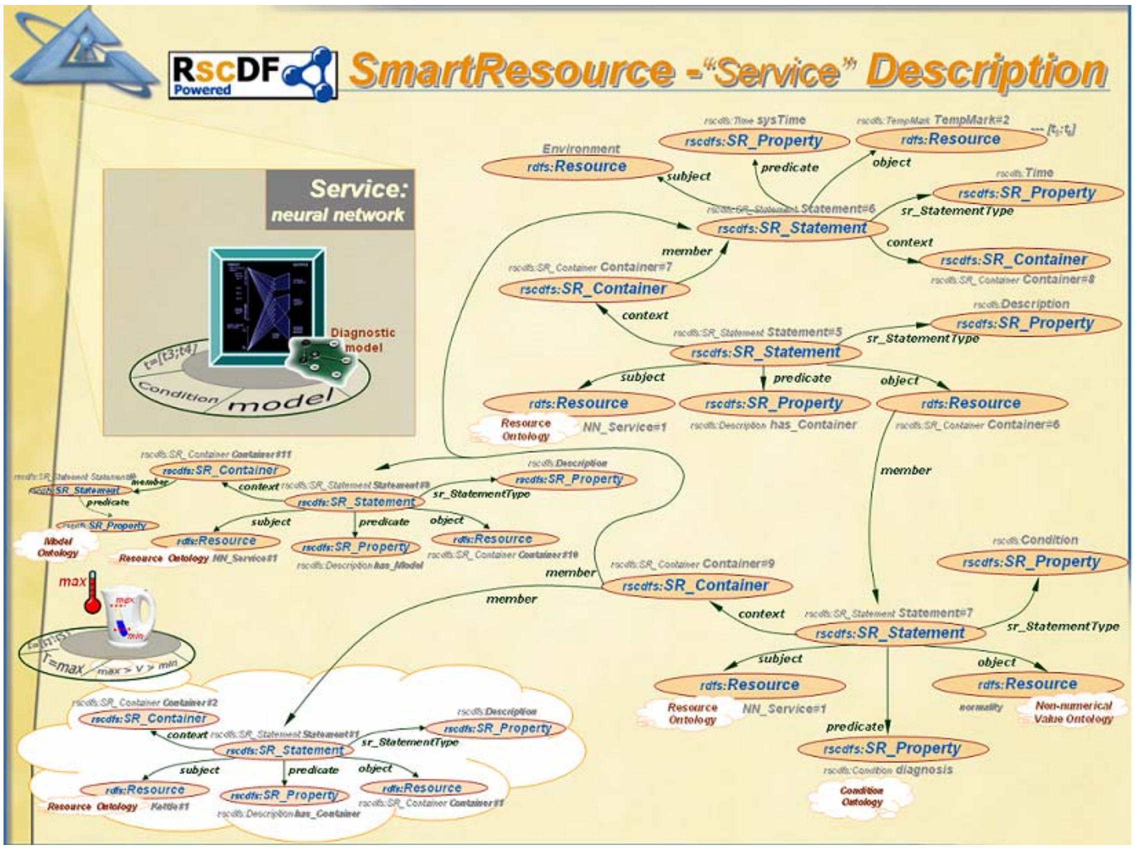
```
<rscdfs:Environment rdf:about="&rscdfs;Environment"
          rdfs:comment="Type of itself"
          rdfs:label="Environment">
         <rdfs:subClassOf rdf:resource="&rscdfs;SmartResource"/>
</rscdfs:Environment>

<rdfs:Class rdf:about="&rscdfs;SR_Container"
          rdfs:comment=""
          rdfs:label="SR_Container">
         <rdfs:subClassOf rdf:resource="&rdfs;Container"/>
</rdfs:Class>

<rdfs:Class rdf:about="&rscdfs;SR_Statement"
          rdfs:comment=""
          rdfs:label="SR_Statement">
         <rdfs:subClassOf rdf:resource="&rdf;Statement"/>
</rdfs:Class>

<rscdfs:QuantityValue rdf:about="&rscdfs;QuantityValue"
          rdfs:comment="Type of itself"
          rdfs:label="QuantityValue">
         <rdfs:subClassOf rdf:resource="&rdfs;Class"/>
         <rdfs:subClassOf rdf:resource="&rdfs;Resource"/>
</rscdfs:QuantityValue>

<rscdfs:EnumerativeValue rdf:about="&rscdfs;EnumerativeValue"
          rdfs:comment="Type of itself"
          rdfs:label="EnumerativeValue">
         <rdfs:subClassOf rdf:resource="&rscdfs;QuantityValue"/>
</rscdfs:EnumerativeValue>

<rscdfs:TempMark rdf:about="&rscdfs;TempMark"
          rdfs:comment="Type of itself"
          rdfs:label="TempMark">
         <rdfs:subClassOf rdf:resource="&rscdfs;EnumerativeValue"/>
</rscdfs:TempMark>

<rscdfs:NumericalValue rdf:about="&rscdfs;NumericalValue"
          rdfs:comment="Type of itself"
          rdfs:label="NumericalValue">
         <rdfs:subClassOf rdf:resource="&rscdfs;QuantityValue"/>
</rscdfs:NumericalValue>

<rscdfs:MeasurementUnit rdf:about="&rscdfs;MeasurementUnit"
          rdfs:comment="Type of itself"
          rdfs:label="MeasurementUnit">
         <rdfs:subClassOf rdf:resource="&rdfs;Class"/>
         <rdfs:subClassOf rdf:resource="&rdfs;Resource"/>
</rscdfs:MeasurementUnit>
```

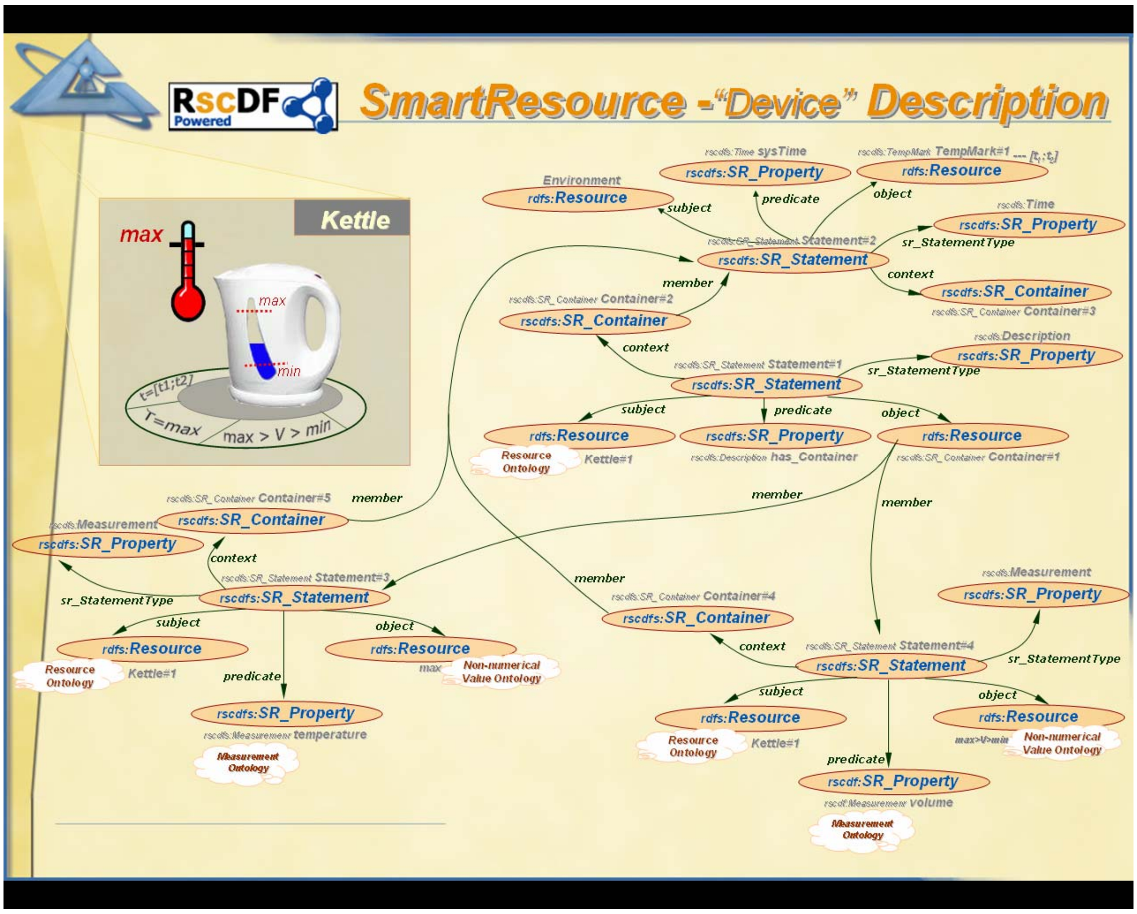
```
<rscdfs:SR_Property rdf:about="&rscdfs;SR_Property"
            rdfs:comment="Type of itself"
            rdfs:label="SR_Property">
           <rdfs:subClassOf rdf:resource="&rdfs;Class"/>
           <rdfs:subClassOf rdf:resource="&rdf;Property"/>
</rscdfs:SR_Property>

<rscdfs:Time rdf:about="&rscdfs;Time"
            rdfs:comment="Type of itself"
            rdfs:label="Time">
           <rdfs:subClassOf rdf:resource="&rscdfs;SR_Property"/>
</rscdfs:Time>

<rscdfs:Description rdf:about="&rscdfs;Description"
            rdfs:comment="Type of itself"
            rdfs:label="Description">
           <rdfs:subClassOf rdf:resource="&rscdfs;SR_Property"/>
</rscdfs:Description>

<rscdfs:Maintenance rdf:about="&rscdfs;Maintenance"
            rdfs:comment="Type of itself"
            rdfs:label="Maintenance">
           <rdfs:subClassOf rdf:resource="&rscdfs;SR_Property"/>
</rscdfs:Maintenance>


<rscdfs:Condition rdf:about="&rscdfs;Condition"
            rdfs:comment="Type of itself"
            rdfs:label="Condition">
           <rdfs:subClassOf rdf:resource="&rscdfs;SR_Property"/>
</rscdfs:Condition>


<rscdfs:Measurement rdf:about="&rscdfs;Measurement"
            rdfs:comment="Type of itself"
            rdfs:label="Measurement">
           <rdfs:subClassOf rdf:resource="&rscdfs;SR_Property"/>
</rscdfs:Measurement>


<rscdfs:Model rdf:about="&rscdfs;Model"
            rdfs:comment="Type of itself"
            rdfs:label="Model">
           <rdfs:subClassOf rdf:resource="&rscdfs;SR_Property"/>
</rscdfs:Model>


<rscdfs:Transformation rdf:about="&rscdfs;Transformation"
            rdfs:comment="Type of itself"
            rdfs:label="Transformation">
           <rdfs:subClassOf rdf:resource="&rscdfs;SR_Property"/>
</rscdfs:Transformation>
```

```
<rscdfs:Description rdf:about="&rscdfs;has_Container"
           rdfs:label="has_Container">
          <rdfs:domain rdf:resource="&rdfs;Resource"/>
          <rdfs:range rdf:resource="&rscdfs;SR_Container"/>
</rscdfs:Description>


<rdf:Property rdf:about="&rscdfs;predicate"
           rdfs:label="predicate">
          <rdfs:subPropertyOf rdf:resource="&rdf;predicate"/>
          <rdfs:domain rdf:resource="&rscdfs;SR_Statement"/>
          <rdfs:range rdf:resource="&rscdfs;SR_Property"/>
</rdf:Property>


<rdf:Property rdf:about="&rscdfs;sr_StatementType"
           rdfs:label="sr_StatementType">
          <rdfs:domain rdf:resource="&rscdfs;SR_Statement"/>
          <rdfs:range rdf:resource="&rscdfs;SR_Property"/>
</rdf:Property>


<rdf:Property rdf:about="&rscdfs;value"
           rdfs:label="value">
          <rdfs:subPropertyOf rdf:resource="&rdf;value"/>
          <rdfs:domain rdf:resource="&rscdfs;NumericalValue"/>
          <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>


<rdf:Property rdf:about="&rscdfs;unit"
           rdfs:label="unit">
          <rdfs:domain rdf:resource="&rscdfs;NumericalValue"/>
          <rdfs:range rdf:resource="&rscdfs;MeasurementUnit"/>
</rdf:Property>


<rdf:Property rdf:about="&rscdfs;member"
           rdfs:label="member">
          <rdfs:subPropertyOf rdf:resource="&rdfs;member"/>
          <rdfs:domain rdf:resource="&rscdfs;SR_Container"/>
          <rdfs:range rdf:resource="&rscdfs;SR_Statement"/>
</rdf:Property>


<rdf:Property rdf:about="&rscdfs;context"
           rdfs:label="context">
          <rdfs:domain rdf:resource="&rscdfs;SR_Statement"/>
          <rdfs:range rdf:resource="&rscdfs;SR_Container"/>
</rdf:Property>
```

```
<rscdfs:Time rdf:about="&rscdfs;sysTime"
            rdfs:label="sysTime">
           <rdfs:domain rdf:resource="&rscdfs;Environment"/>
           <rdfs:range rdf:resource="&rscdfs;TempMark"/>
</rdf:Time>

</rdf:RDF>
```

Appendix B: Example of Instances in RSCDFS


RscDF Powered
SmartResource -"Service" Description
Service: neural network
Diagnostic model
t=[t3;t4]
Condition model
Environment
rdfs:Resource
rscdfs:Time sysTime
rscdfs:SR_Property
rscdfs:TempMark TempMark#2
rdfs:Resource
[t1;t2]
subject
predicate
object
rscdfs:Time
rscdfs:SR_Property
rscdfs:SR_Statement Statement#6
rscdfs:SR_Statement
sr_StatementType
context
member
rscdfs:SR_Container Container#7
rscdfs:SR_Container
rscdfs:SR_Container
rscdfs:SR_Container Container#8
rscdfs:Description
rscdfs:SR_Property
rscdfs:SR_Statement Statement#5
rscdfs:SR_Statement
Resource Ontology
NN_Service#1
rscdfs:Description has_Container
rscdfs:SR_Container Container#6
rscdfs:SR_Container Container#11
rscdfs:SR_Statement Statement#8
rscdfs:Description
rscdfs:Description has_Model
rscdfs:SR_Container Container#10
Model Ontology
rscdfs:Condition
rscdfs:SR_Container Container#9
rscdfs:SR_Statement Statement#7
max
min
T=max
max > V > min
Non-numerical Value Ontology
normality
rscdfs:Condition diagnosis
Condition Ontology
rscdfs:SR_Container Container#2
rscdfs:SR_Statement Statement#1
Kettle#1
rscdfs:Description has_Container
rscdfs:SR_Container Container#1


RscDF Powered
SmartResource -"Device" Description
Kettle
max
min
t=[t1;t2]
T=max
max > V > min
Environment
rdfs:Resource
rscdfs:Time sysTime
rscdfs:SR_Property
rscdfs:TempMark TempMark#1
[t1;t2]
subject
predicate
object
rscdfs:Time
rscdfs:SR_Statement Statement#2
rscdfs:SR_Statement
sr_StatementType
context
member
rscdfs:SR_Container Container#2
rscdfs:SR_Container Container#3
rscdfs:Description
rscdfs:SR_Statement Statement#1
Resource Ontology
Kettle#1
rscdfs:Description has_Container
rscdfs:SR_Container Container#1
rscdfs:SR_Container Container#5
rscdfs:Measurement
rscdfs:SR_Statement Statement#3
Non-numerical Value Ontology
rscdfs:Measurement temperature
Measurement Ontology
rscdfs:SR_Container Container#4
rscdfs:SR_Statement Statement#4
max>V>min
rscdf:Measurement volume