Large Scale Integrating Project

Grant Agreement no.: 257899

D3.1 – Semantic models of data streams

SMART VORTEX –WP3-D3.1

Project Number	FP7-ICT-257899
Due Date	June 30th 2011
Actual Date	March 3rd 2011
Document Author/s:	• Tore Risch, TR, UU • John Lindström, JL, LTU • Mathias Johanson, MJ, Alkit • Lars-Åke Johansson, LJ, Alkit • Mikael Lax, UU
Version:	1.0
Dissemination level	Confidential
Status	1st structural Draft
Contributing Sub-project and Work package	All partners in the project
Document approved by	

Co-funded by the European Union

Document Version Control			
Version	**Date**	**Change Made (and if appropriate reason for change)**	**Initials of Commentator(s) or Author(s)**
0.1	05.4.2011	Initial structural Draft	JL
0.2	2011-06-13	Introduction, RDF	TR
0.3	2011-06-14	More RDF	TR
0.4	2011-06-15	Minor changes/formatting	MJ
0.5	2011-06-15	Added explanations of RDF models	TR
0.6	2011-06-27	Modeling machines and sensors	TR
0.7	2011-06-28	Specific modeling of sensors and streams	TR
0.8	2011-06-28	Formatting, proof-reading	MJ
1.0	2011-06-28	Executive summary	TR

Document Change Commentator or Author		
Author Initials	**Name of Author**	**Institution**

Document Quality Control			
Version QA	**Date**	**Comments (and if appropriate reason for change)**	**Initials of QA Person**

Catalogue Entry

Title	Semantic models of data streams
Creators	Tore Risch, John Lindström, Mathias Johanson
Subject	Deliverable D3.1 Smart Vortex
Description	
Publisher	
Contributor	
Date	April 4th 2011
ISBN	
Type	Delivery SmartVortex
Format	
Language	English
Rights	SmartVortex Consortium

Citation Guidelines

EXECUTIVE SUMMARY

This document describes the structure of the ontology in Smart Vortex that models data streams and related concepts. First an upper ontology is presented modeling general data streams and programs operating on the data streams, followed by a model of how values produced in data streams are represented. Then a general model is presented of how validators and simulators operate on different kinds of data streams. To define how raw data streams are produced the overall modeling of machines and sensors is presented. The industrial scenarios require specializations of classes for specific kinds of sensors and streams. The structures of these specializations are exemplified by models of some of the different kinds of streams for one of our industrial partners. Analogous models are being made for the other industrial partner following the same structure, thereby creating a complete ontology for all scenarios.

TABLE OF CONTENTS

1 INTRODUCTION

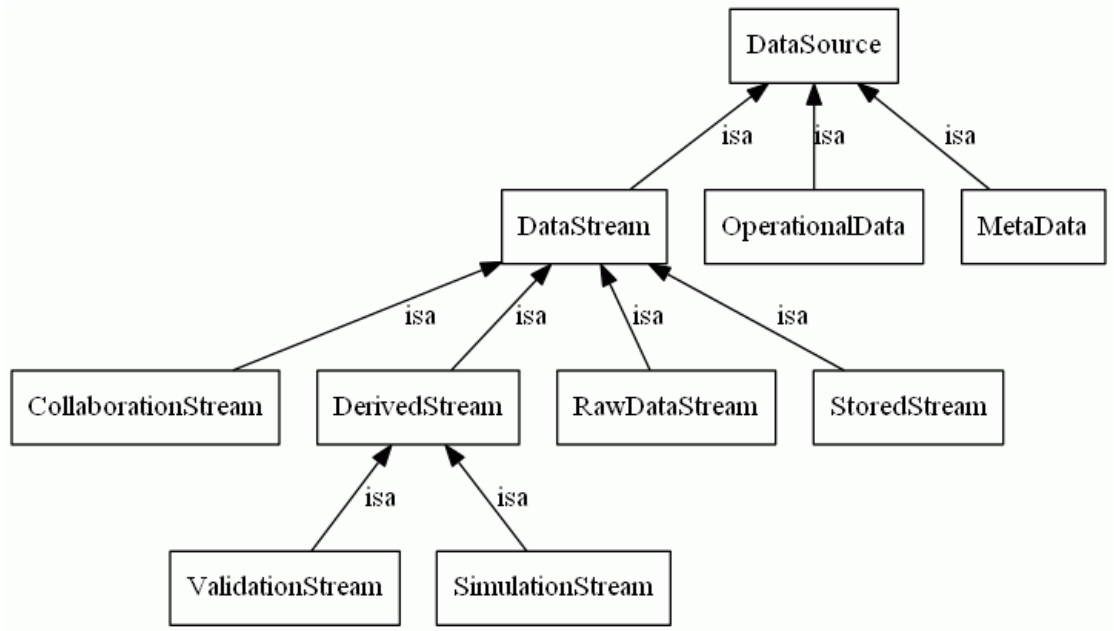
In the Smart Vortex project, efficient search and processing of data streams (and other data from the product life cycle) using a Data Stream Management System (DSMS) is a core feature. In order for this to be possible, the semantics of the data streams (and other data) need to be well defined. The aim of this document is to provide semantic models of data streams and related data and computations. The semantic model is represented using the RDF-Schema data modeling language [RDFS2004]. The model is developed using the Protégé tool from Stanford [Pro] version 3.4.6. Graphical visualizations of different details of the model have been automatically generated by the Ontoviz tool of Protégé 3.4.6. All presented models are stored in a machine readable RDF-Schema semantic model [Sem2011]. At this point we have deliberately limited the level of the ontology language to RDF-Schema, rather than e.g. OWL [OWL2009], in order to keep the representation as simple as possible and efficient to process while still being rich enough to express the desired semantics of our model.

It is important to point out that the models presented in this document will be continuously updated throughout the project, as new needs or new technical requirements arises. The example models show the structure of the ontologies for particular data sources.

2 SEMANTIC MODELS OF DATA AND DATA STREAMS IN SMART VORTEX

2.1 Data sources

The different types of data streams identified during the requirement specification work are classified according to the following RDF-Schema model created as an RDF Schema model using Protégé 3.4.3:



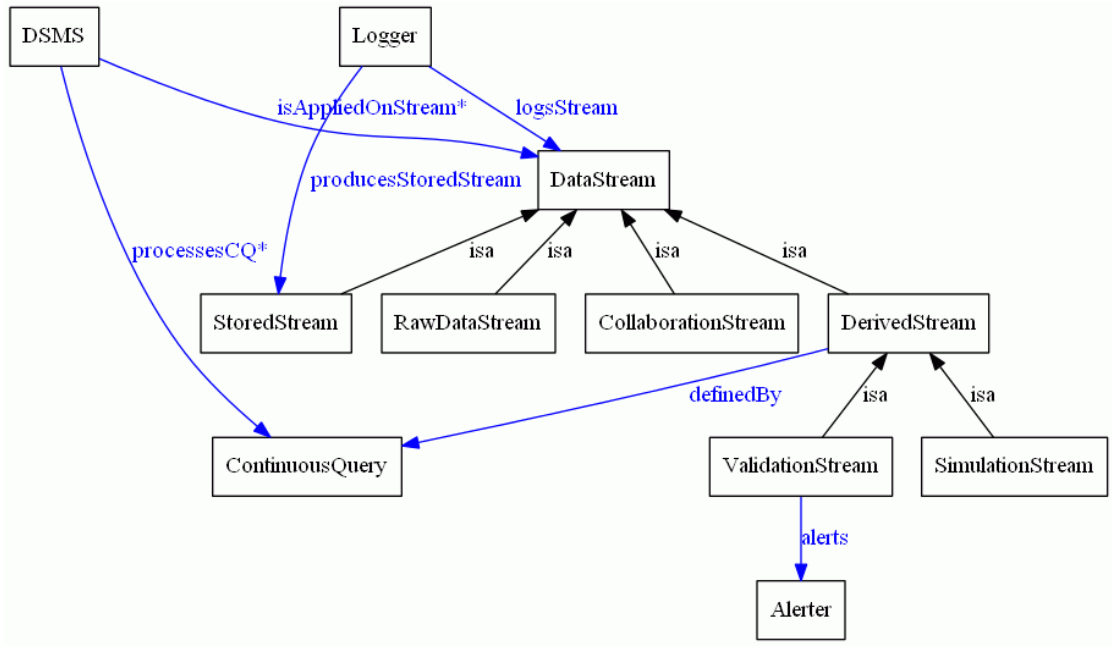
- A **data source** (class *DataSource*) is any kind of data used during the product life cycle.
- Product **meta-data** (class *MetaData*) are parameters describing properties of the monitored equipment, e.g. equipment locations, models, and what sensors are installed in each location.
- **Operational data** (class *OperationalData*) are continuously updated business data stored in regular databases, e.g. pricing, sales, customers, and personnel.
- **Data streams** (class *DataStream*) are continuously growing streams of data of different kinds. All kinds of streams are subclasses to the class *DataStream*.
- **Raw data streams** (class *RawDataStream*) are measurements from equipment-in-use usually provided by sensors on the equipment represented by the class *RawDataStream*. For example temperature readings, pressure readings, power consumption, etc.
- **Derived data streams** (class *DerivedStream*) are computed by a Data Stream Management System (DSMS) [GÖ2010] based on data from other data streams in combination with product meta-data and operational data. For example, the power consumption of some equipment in use may be estimated by a mathematical model over several data streams and product meta-data.
- **Simulated data streams** (class *SimulationStream*) are derived streams computed by the DSMS, usually by calling some computational system. For example, the desired limits for temperature readings over time may be computed by a simulator.
- **Validation data streams** (class *ValidationStream*) are derived streams computed by the DSMS for detecting significant deviations between observed streaming data and simulated streams or other data. For example, a validator may compare simulated temperature limits with the actual temperature readings from some equipment-in-use

using some mathematical model implemented as a continuous query. If there are significant deviations for some time some production engineer may be alerted by software. When deviations occur the production engineer will further search and analyze different kinds of data to understand why the deviations occurred.

- **Collaboration data streams** (class *CollaborationStream*) are streams recording the interactions between humans participating in collaborations.
- **Stored data streams** (class *StoredStream*) are data streams that are saved in a file system or other repository. The contents of entire or partial live data streams can be saved as stored data streams by some software. A stored data stream can be later played back as a live data stream.

2.2 Data streams

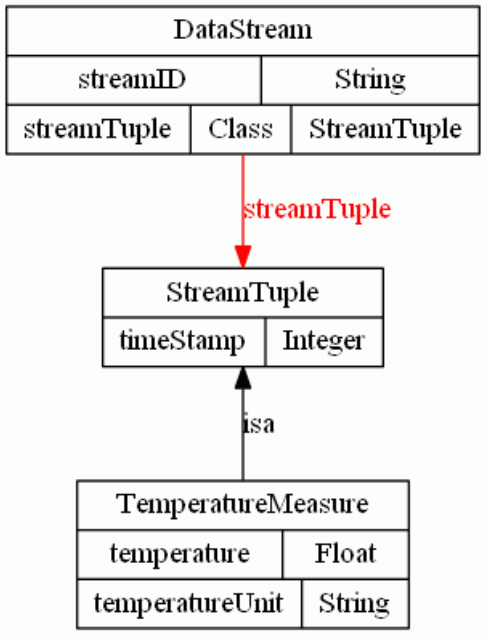
The following RDF-Schema diagram illustrates the relationships between the different kinds of data streams in Smart Vortex and the systems managing these data streams.



A **DSMS** produces one or several **derived streams** defined by a continuous query (relationship *definedBy*). If desirable and feasible any kind of data stream may be continuously saved on disk (class *StoredStream*) by running a **logger** (relationship *logsStream*). The output of a validation stream can be fed into an **alerter** (relationship *alerts*), which is a program notifying users and systems that some unexpected behavior has occurred.

2.2.1 Data stream tuples

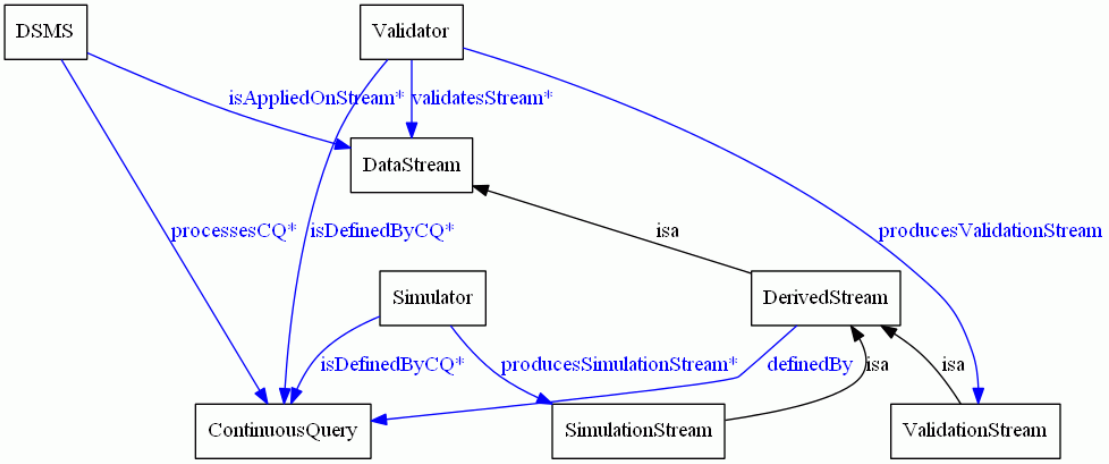
The following RDF-Schema view models details of the values generated in a data stream:



Each **data stream** has a unique identifier (attribute *streamID*), which is used in continuous queries to search and analyze the contents of the stream. A **data stream** continuously produces an ever growing sequence of **stream tuples** of measured stream data values (c.f. log records). A stream tuple (c.f. log record) is represented as an instance of the class *StreamTuple*. A stream tuple has a **time stamp** (attribute *timeStamp*) representing when the tuple was produced. Different kinds of streams produce tuples with different values that are specializations of *StreamTuple*. For example, a sensor measuring temperatures will produce objects of type *TemperatureMeasure*, which is a specialization of class *StreamTuple*. The relationship between a data stream and its tuples is modeled as a class relationship *streamTuple* from class *DataStream* to class *StreamTuple*.

2.3 Simulation and validation

The following RDF-Schema view illustrates how simulators and validators produce derived streams by calling the DSMS to be integrated in Smart Vortex:



A **DSMS** processes a number of **continuous queries** (relationship *processesCQ*). A continuous query (CQ) is an expression executed by a DSMS, which defines how the DSMS

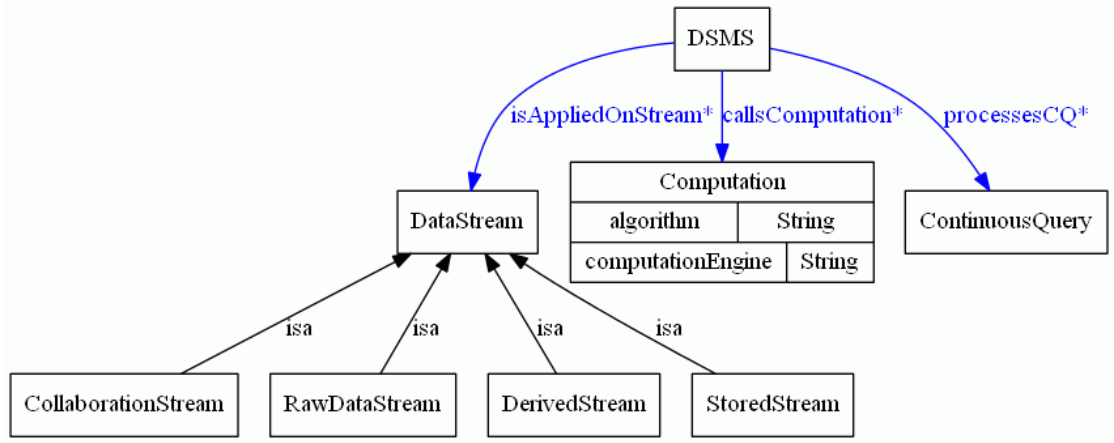
produces a **derived data streams** (relationship *producesDerivedStream*) based on a number of input data streams (relationship *isAppliedOnStream*) and data from other sources.

A **simulator** is a program defined by one or several CQs (relationship *isDefinedByCQ*) that produces one or several derived **simulation streams** (relationship *producesSimulationStream*).

A **validator** (or **verifier**) is a program defined by one or several CQs (relationship *isDefinedByCQ*) that compares one or several data streams (relationship *validateStream*) and produces one or several **validation streams** (class *ValidationStream* and relationship *producesValidationStream*).

2.4 DSMS computations

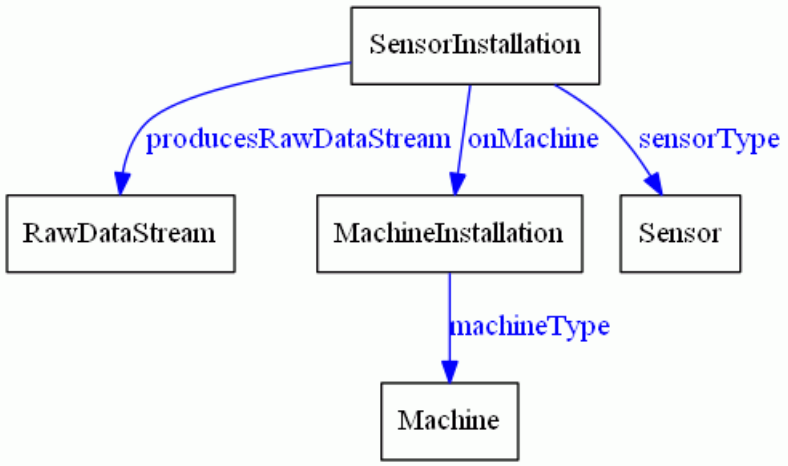
The following RDF-Schema describes the computations made by the DSMS:



Continuous queries may involve different kinds of **computations** over streams. The computations are called as plug-ins to the DSMS (relationship *callsComputation*). A computation implements some algorithm (attribute *algorithm*) by calling some computation engine (attribute *computationEngine*). For example, a simulation stream may be expressed as a continuous query executing some computation algorithm executed by a Python engine.

2.5 Machine and sensor view

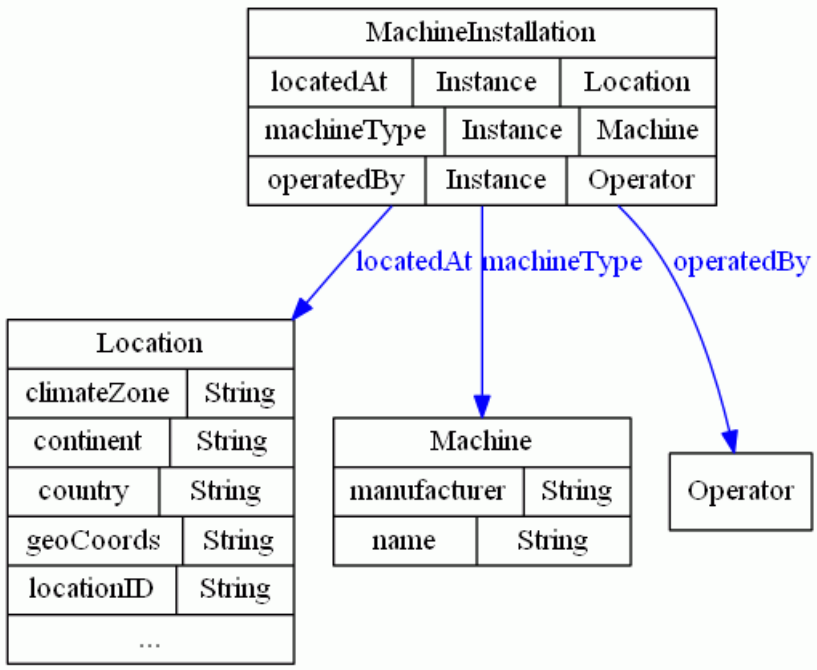
The following RDF-Schema view defines the relationships between data streams, machines, and sensors:



The class *SensorInstallation* represents **sensors** installed in a particular **machine** (class *MachineInstallation* and relationship *onMachine*). Each sensor is of a particular kind (class *Sensor* and relationship *sensorType*). A machine is also of a particular kind (class *Machine* and relationship *machineType*). A sensor produces a **raw data stream** (class *RawDataStream* and relationship *producesRawDataStream*).

2.5.1 General machine view

The following RDF-Schema view shows the details of generic modeling of **machines** and their **installation**s at different sites:

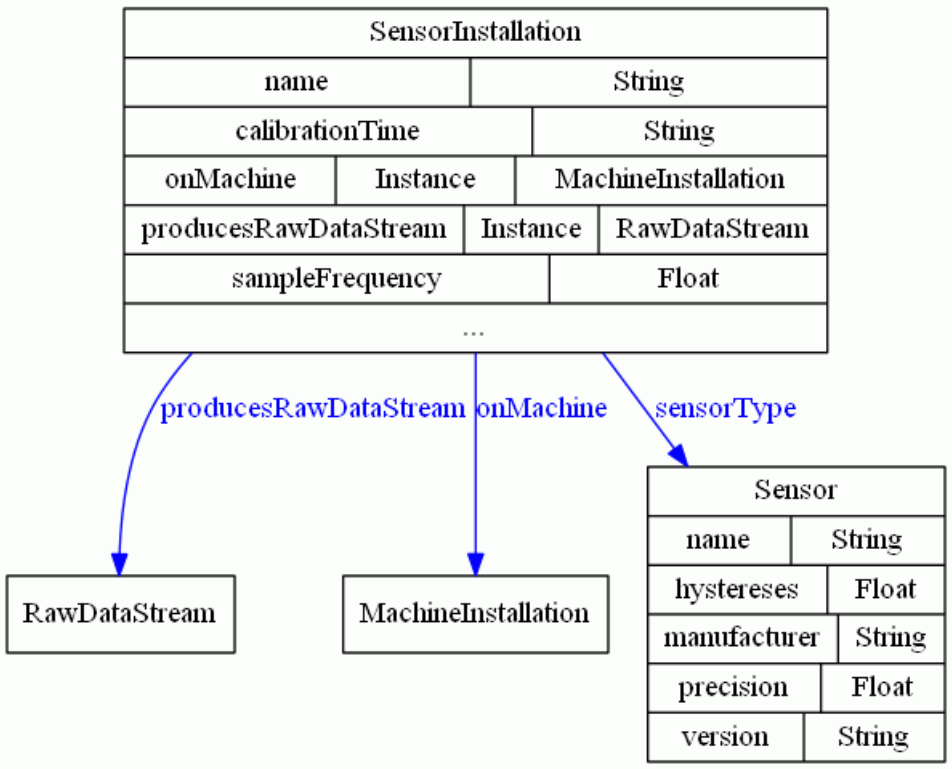


A **machine model** (class *Machine*) has a *manufacturer* and a *name*. Each **machine installation** (class *MachineInstallation*) has a particular machine model (relationship *machineType*), a location (relationship *locatedAt* to class *Location*), and is operated by some responsible operator (relationship *operatedBy* to class *Operator*).

The **geographical location** of a machine is of interest e.g. to find all installations in a geographic area or the climate zone of a sensor. Therefore we need to model geographic locations as class *Location*. A geographic location has attributes *climateZone*, *continent*, *country*, *geoCoords*, and *locationID* (i.e. a unique name).

2.5.2 General sensor view

The following RDF-Schema view shows the details of generic modeling of **sensors** and their **installations** on machines:



Each **sensor model** (class *Sensor*) has the attributes *name*, *hysteresis* (its reaction time), a *manufacturer*, *precision* (of the values it measures), and a *version*.

Each **installation** of a sensor on a machine (class *SensorInstallation*) has a *name* (e.g. “particle counter 2”), a sampling frequency (attribute *sampleFrequency*), and a calibration time (attribute *calibrationTime*).

Each particular installation of a sensor on a given machine installation will have a number of more specialized properties, modeling the details of the installation. In the following subsection the details of these **specific sensor installations** for our industrial partners are modeled. The detailed models of specific sensor installations are all subclasses of class *SensorInstallation*.

A sensor installation produces a **raw data stream** (relationship *producesRawDataStream*) of data stream tuples (Section 2.2.1).

2.6 Specific site views

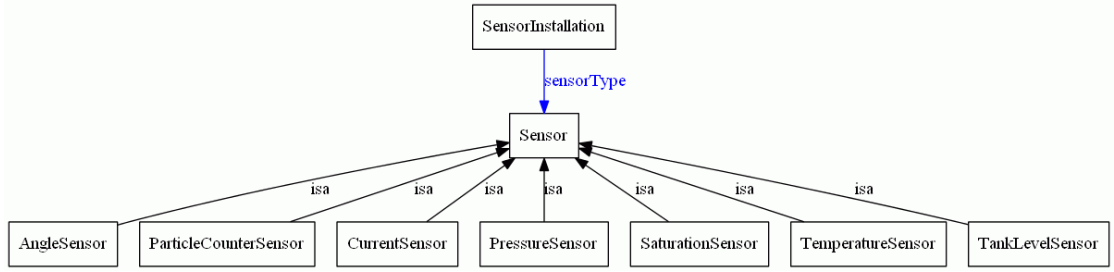
Here the specific installations of particular sensors of particular machines for some of the sites of our industrial partners are described. The views illustrate how specific site views are modeled. Analogous views will be defined for all sites of all industrial partners.

2.6.1 Hägglunds sensor installations

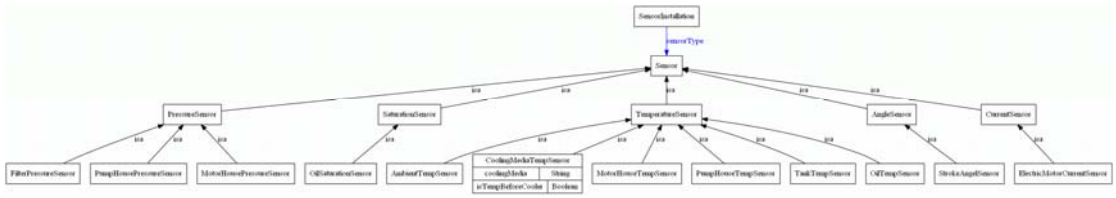
Some examples are given of the specific kind of data used by Hägglunds Drives AB. Each kind of sensor is modeled as a special class. Analogously each stream produced by a sensor is modeled as a special class.

2.6.1.1 Sensors

The following RDF-Schema view shows the kind of sensors used by our industrial partner Hägglunds Drives AB:



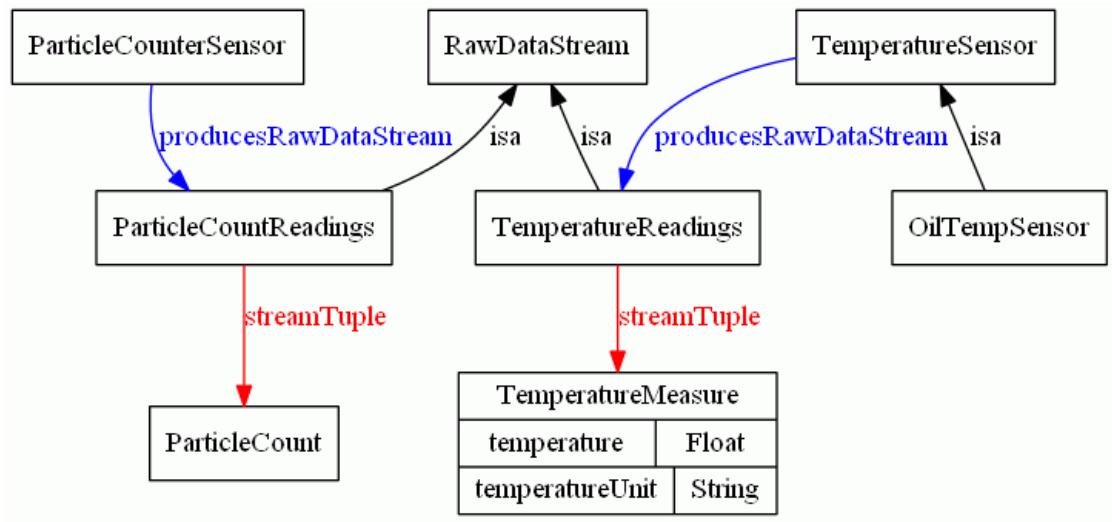
The particular kinds of sensors used by Hägglunds are subclasses of type *Sensor*. The different kinds of sensors will be further specialized by subclassing each specific sensor class. Thus, all kinds of sensors used by Hägglunds are modeled by this RDF-Schema view:



For example, there is a special **cooling media temperature sensor** (class *CoolingMediaTemperatureSensor*) measuring the temperature of cooling media. It has attributes *coolingMedia* (kind of cooling media used) and *isTempBeforeCooler* (whether the temperature is measured before cooling or not).

2.6.1.2 Raw Data Streams

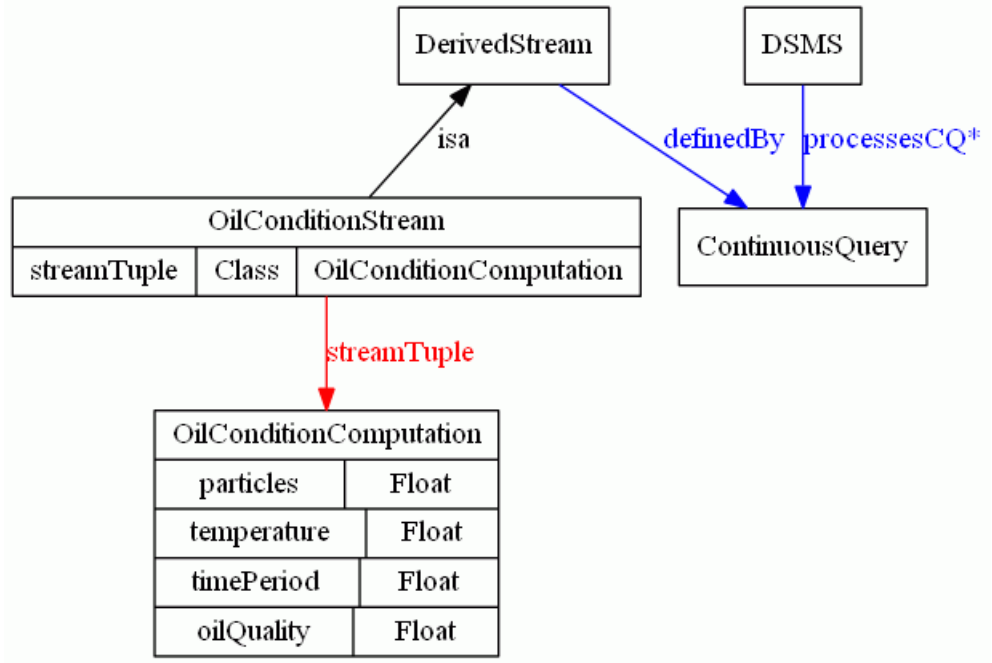
As described in Section 2.2.1, each data stream is modeled as an instance of class *DataStream* having a property *streamTuple* that describes the elements of the tuples produced in the stream as a class. This class serves as a *schema* for a particular stream. This is sufficient to describe the data tuples in the different kinds of streams produced by the machines. For example, a **particle counter sensor** (*ParticleCounterSensor*) will produce a raw data stream **particle counter readings** (*ParticleCountReadings*) where each tuple is a measurement of a **particle count** (*ParticleCount*), as illustrated by this view:



Analogously a **temperature sensor** (*TemperatureSensor*) measures the current temperature of some artifact. It produces a raw data stream of **temperature readings** (*TemperatureReadings*), which is a stream of **temperature measurements** (*TemperatureMeasure)*. A temperature measurement has attributes *temperature* and *temperatureUnit* (Celsius, Fahrenheit, Kelvin).

2.6.1.3 Derived Data Streams

The following is a model of the derived data stream **oil condition stream** (class *OilConditionStream*) produced as a continuous query by the DSMS:

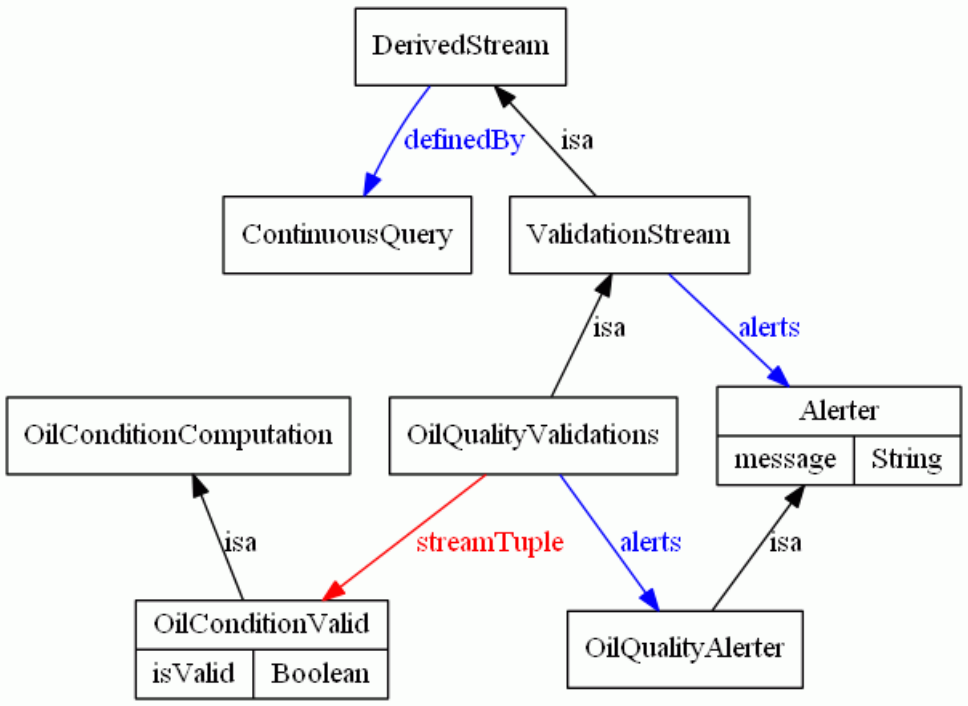


A tuple in the oil condition stream contains a value estimating the oil quality based on the temperature readings and particle counts during a time period. The values are computed as a continuous query (relationship *definedBy*) based on two other raw data streams, a particle count stream and a temperature reading stream. To enable explaining the estimated value of

the oil quality both the average particle count over a time period and the average temperature is included, as well as the time period itself.

2.6.1.4 Validation streams

When the oil quality is lower than a threshold an **alerter** is invoked to notify the operator or some system, as illustrated by this view:



A validation stream detecting oil quality validations, *OilQualityValidations*, is defined by a continuous query that produces tuples of class *OilConditionValid*, which extends the derived stream *OilConditionComputation* with a flag *isValid* indicating whether or not the oil quality is satisfactory. If the oil is not of satisfactory quality an alerter is invoked that sends a message to the responsible operator.

2.6.2 Volvo and Sandvik sensor installations

Analogous models of the sensors and streams for Volvo Construction Equipment and Sandvik Coromant are constructed following the same principles are the data streams of Hägglunds Drives AB.

3 REFERENCES

[GÖ2010] L. Golab, M.T. Özsu: Data Stream Management. Synthesis Lectures on Data Management, Morgan & Claypool Publishers, 2010.

[OWL2009] OWL 2 Web Ontology Language Document Overview, W3C Recommendation, 27 October, 2009, http://www.w3.org/TR/owl2-overview/

[Pro] Protégé home page: http://protege.stanford.edu/

[RDFS2004] RDF Vocabulary Description Language 1.0: RDF Schema, W3C Recommendation, 10 February 2004, http://www.w3.org/TR/rdf-schema/

[Sem2011] RDF-Schema model of Semantic Streams for Smart Vortex, June 28, 2011, https://documents.smartvortex.eu/WP03/SemanticModel/