Environmental Modelling & Software 58 (2014) 27–47

Contents lists available at ScienceDirect

Environmental Modelling & Software

journal homepage: www.elsevier.com/locate/envsoft

Representing situational knowledge acquired from sensor data for atmospheric phenomena

Markus Stocker [a,*], Elham Baranizadeh [b], Harri Portin [b,d], Mika Komppula [d], Mauno Rönkkö [a], Amar Hamed [b], Annele Virtanen [b], Kari Lehtinen [b,d], Ari Laaksonen [b,c], Mikko Kolehmainen [a]

[a] Environmental Informatics Group, Department of Environmental Science, University of Eastern Finland, P.O. Box 1627, 70211 Kuopio, Finland
[b] Aerosol Physics Group, Department of Applied Physics, University of Eastern Finland, P.O. Box 1627, 70211 Kuopio, Finland
[c] Finnish Meteorological Institute, P.O. Box 503, 00101 Helsinki, Finland
[d] Finnish Meteorological Institute, P.O. Box 1627, 70211 Kuopio, Finland

ARTICLE INFO

Article history:
Received 17 October 2013
Received in revised form
10 February 2014
Accepted 3 April 2014
Available online 8 May 2014

Keywords:
Environmental sensor networks
Sensor data
Knowledge acquisition
Knowledge representation
Atmospheric phenomena

ABSTRACT

A recurrent problem in applications that build on environmental sensor networks is that of sensor data organization and interpretation. Organization focuses on, for instance, resolving the syntactic and semantic heterogeneity of sensor data. The distinguishing factor between organization and interpretation is the abstraction from sensor data with information acquired from sensor data. Such information may be situational knowledge for environmental phenomena. We discuss a generic software framework for the organization and interpretation of sensor data and demonstrate its application to data of a large scale sensor network for the monitoring of atmospheric phenomena. The results show that software support for the organization and interpretation of sensor data is valuable to scientists in scientific computing workflows. Explicitly represented situational knowledge is also useful to client software systems as it can be queried, integrated, reasoned, visualized, or annotated.

Software availability

Wavellite is open source, is written in Java, and was released in 2013 under the Eclipse Public License (EPL 1.0). Wavellite was developed and is maintained by Markus Stocker. The Wavellite project page is at http://www.uef.fi/en/envi/projects/wavellite.

1. Introduction

Environmental sensor networks are an important research tool for earth and environmental science (Hart and Martinez, 2006). They play a key role in the monitoring of the natural environment and allow for unprecedented study of the dynamics of environmental systems and processes (Hill et al., 2011).

Over the past decades, many small and large scale environmental sensor networks have been deployed. The Finnish Station for Measuring Ecosystem-Atmosphere Relations (SMEAR) is an example for a large scale sensor network and Hart and Martinez (2006) present several others. SMEAR started its operations in 1991 with measurements for SO_2 in Eastern Lapland. It quickly grew to include several other locations and properties of environmental phenomena, including for weather, such as temperature, humidity, or wind speed; for atmospheric gases, such as the concentration of carbon dioxide or ozone; for aerosols, such as particle number concentration.

Today, SMEAR consists of four main stations: SMEAR I in Eastern Lapland, SMEAR II in Hyytiälä, SMEAR III in Helsinki, and SMEAR IV in Kuopio. The main stations consist of one or more substations. Substations consist of a set of sensing devices. For instance, SMEAR IV consists of two substations, at Puijo and at Savilahti. The substation SMEAR IV-Puijo resides on top of the Puijo observation tower (62°54′32″ N, 27°39′31″ E), 306 m above sea level and 224 m above the surrounding lake level. The Puijo observation tower is located in the city of Kuopio, in a semi-urban environment. Kuopio is situated in Eastern Finland, about 330 km to the northeast from Helsinki. SMEAR IV-Puijo consists of sensing devices for the monitoring of aerosols, weather, and atmospheric gases. Sensing devices are manufactured by various vendors, including Thermo Fisher Scientific Inc., TSI Inc., and Vaisala (Leskinen et al., 2009). In this study, we used data by sensors of SMEAR IV-Puijo.

* Corresponding author. Tel.: +358 44 500 3908; fax: +358 17 163 191.
E-mail addresses: markus.stocker@uef.fi (M. Stocker), elham.baranizadeh@uef.fi (E. Baranizadeh), harri.portin@fmi.fi (H. Portin), mika.komppula@fmi.fi (M. Komppula), mauno.ronkko@uef.fi (M. Rönkkö), amar.hamed@uef.fi (A. Hamed), annele.virtanen@uef.fi (A. Virtanen), kari.lehtinen@fmi.fi (K. Lehtinen), ari.laaksonen@fmi.fi (A. Laaksonen), mikko.kolehmainen@uef.fi (M. Kolehmainen).

http://dx.doi.org/10.1016/j.envsoft.2014.04.006
1364-8152/

Environmental sensor networks can produce large amounts of syntactically and semantically heterogeneous data. SMEAR IV-Puijo alone generates approximately 2.5 million data points every day, of which 1 million are by a single sensor (namely the optical cloud droplet spectrometer). To ensure its utility, such data must be managed with appropriate hardware and software systems (Hart and Martinez, 2006; Horsburgh et al., 2009). We distinguish the class of software systems that organize sensor data and the class of software systems that in addition *interpret* organized sensor data. The distinguishing feature between the two classes is the abstraction from organized sensor data with information acquired from sensor data, for instance information about a monitored environmental phenomenon.

In discussing the design of a software system for publishing environmental observations Horsburgh et al. (2009) underscore the challenges of persistent storage and management, data access and communication, data interoperability, and data discovery. These challenges are typical to software systems that organize sensor data. Organization of sensor data can be achieved in various ways. Systems may build on conventional relational database management systems (Horsburgh et al., 2009; Junninen et al., 2009) or so-called "NoSQL" databases, such as Apache Cassandra. Systems may be tailored for streamed data processing (Bonnet et al., 2001; Carney et al., 2002; Madden and Franklin, 2002). Systems may use advanced data and knowledge representation languages. We highlight Semantic Web (Berners-Lee et al., 2001) technologies, which have found their application in sensor networks and sensor data (Sheth et al., 2008) with ontologies (Compton, 2011) and software architectures and systems (Moraru and Mladenić, 2012) being developed for the purpose of organizing sensor data.

Software systems that interpret sensor data build on organized sensor data and include computational techniques in, e.g., machine learning, inference, or complex event processing, to acquire information from sensor data. In this study, information is for *situations* and the acquisition of information is automated and may occur in (near) real time. Information is represented explicitly. For example, given organized observations for mean hourly concentration of particulate matter with diameter less than 2.5 μm ($PM_{2.5}$), complex event processing can be used to automatically and continuously detect situations of unhealthy exposure. The semantics of situations are, typically, different from the semantics of observations. Specifically to situations of unhealthy exposure, 'exposure' entails a longer time interval than mean hourly concentration and 'unhealthy' requires mean hourly concentration to continuously exceed a certain threshold.

Software systems that interpret sensor data are interesting for several reasons. First, the problem is harder than mere sensor data organization. The problem is known to various domains and several software architectures have been proposed in the literature (Clemente et al., 2013; Gorrepati et al., 2013; Conroy et al., 2011; Gaglio et al., 2007; Liu and Zhao, 2005; Whitehouse et al., 2006; Vassev and Hinchey, 2012). However, to the best of our knowledge, the work in this area is fragmented. Second, information acquired from sensor data is typically of more value to people than sensor data (Barnaghi et al., 2012). Of specific interest in this study are scientists and scientific computing on environmental sensor data. Third, for applications that build on large sensor networks and/or high frequency sensors it may not be desirable, or practicable, to persist sensor data for offline analysis. For such applications it may be best to acquire information over streams of sensor data, discard the sensor data, and only retain the acquired information.

The problems addressed in this study are (1) the heterogeneity of sensor data and (2) the explicit representation of situational knowledge automatically acquired from heterogeneous sensor data for environmental phenomena, specifically for SMEAR. Our aim is to use Wavellite (Stocker et al., submitted for publication) and demonstrate with a concrete application how it addresses these problems. Wavellite is a generic software framework aimed at the organization and interpretation of sensor data. It supports the processing of heterogeneous sensor data to sensor observations with homogeneous syntax and semantics; the mapping of sensor observations to dataset observations and the processing of datasets; the acquisition of situational knowledge from datasets; and the representation of situational knowledge. Extending our previous work (Stocker et al., 2013), the environmental phenomena of interest in this study are new particle formation and clouds. Hence, situations of interest are events of new particle formation and cloud events, occurring at Puijo. Information for such situations is acquired from data by sensors used for the monitoring of aerosols and weather at Puijo. To the best of our knowledge, Wavellite is unique in its support for the representation of situational knowledge acquired from heterogeneous sensor data for environmental phenomena.

The contribution of this work is two-fold. First, for readers interested in generic (and practical) approaches to the problem of representing situational knowledge acquired from sensor data, this work presents Wavellite and its application for a concrete use case in aerosol science, with real sensors and sensor data as well as using various computational methods, including machine learning. Second, for aerosol scientists and, more generally, scientists in domains in which sensors play an important role, this work presents a software system that integrates the processes of sensor data organization and sensor data interpretation. Specifically to aerosol scientists who study new particle formation, this work describes a software system that could support their workflows.

The paper is structured as follows. In Section 2 we provide a brief overview of Wavellite, specifically the logical structure of its architecture. In Section 3 we present the concrete implementation of the architecture. In Section 4 we briefly discuss how the implementation can be used in applications. In Section 5 we present our experiment on SMEAR sensor data and the representation of situational knowledge for events of new particle formation and cloud events. In Section 6 we discuss the results of our experiments and Wavellite more generally. In Section 7 we present related work. Finally, Section 8 draws some concluding remarks.

2. Architecture

We describe the logical structure of the Wavellite architecture to provide an overview of the layers, components, and modules as well as their responsibilities and interactions. The logical structure consists of four layers: measurement, observation, derivation, and situation. The four layers build on each other, from measurement to situation. Each layer serves a purpose and abstracts from underlying complexity. Fig. 1 provides a graphical overview of the architecture. Figure D.7 (Appendix D) gives an overview of the most important interfaces, in particular component interfaces with emit and execute operations and operation parameters.

Layers consists of components. Components are categorized in three broad classes: engine, reader, and writer. Components may execute information entities received on incoming streams and emit information entities to outgoing streams. Information entities are messages, specifically measurements and their contextual information, sensor observations, dataset observations, and situations. Components and streams form the nodes and edges, respectively, of a directed acyclic graph, known as a topology. Associated to components, the architecture includes modules. Modules are categorized in three broad classes: processing, learning, and store. Modules implement computations for purposes such as digital signal processing, machine learning, complex event processing, inference, retrieval and storage. The knowledge base is a third-party system.

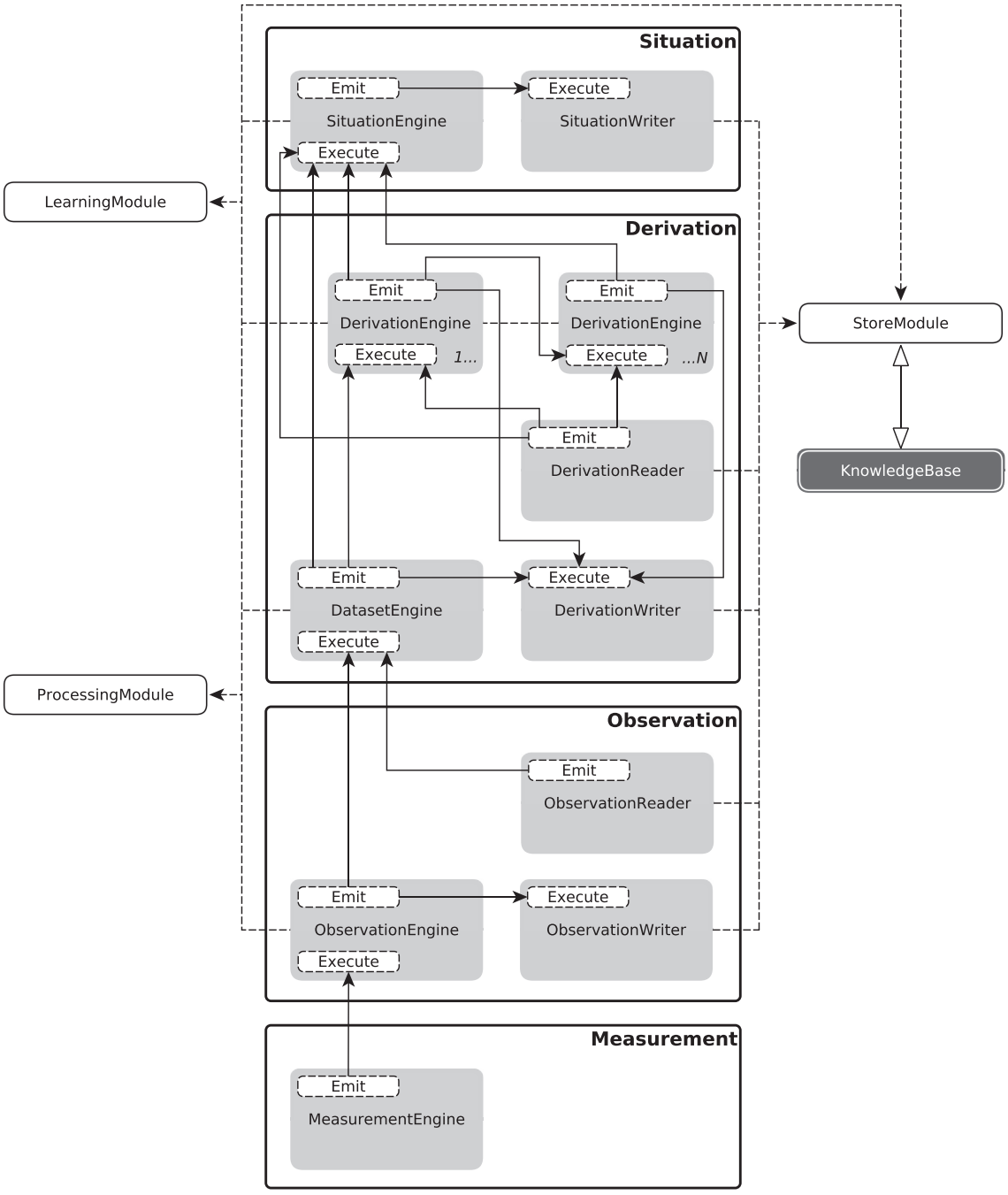


Fig. 1. Logical structure of the architecture with its four layers of measurement, observation, derivation, and situation as well as components and modules implementing software logic for data processing, knowledge acquisition and storage. Solid arrows represent streams. Dashed arrows represent the association between components and modules.

The measurement layer abstracts from the physical sensor network and data communication links and protocols. It consists of measurement engines. A measurement engine implements the software logic required to process sensor data into measurements. Measurements are the numbers assigned to properties of objects or events of the real world in the process of measurement (Finkelstein, 1982). Such numbers are typically scalars, or may be vectors or matrices. Sensor data may be pulled by, or pushed to, a measurement engine. Sensor is understood in a broad sense. It is often a device but sensor data may also be sourced from files, databases, or other resources. Sources of sensor data may be local or remote.

A measurement engine implements an interface with `emit` method having a required parameter for the measurement and optional parameters for procedure, property, feature, temporal location, spatial location, and quality. The optional parameters form the contextual information of a measurement. Together with the measurement, the parameters form a measurement information entity, which is emitted to outgoing streams. Typically, procedure is a sensing device and refers to a deployed sensor. A procedure observes a property of a feature. Features refer to environmental phenomena. For example, a thermometer procedure may observe the temperature property of air, the latter being the feature. A measurement is located in time and space. For static procedures, the spatial location is modelled as metadata of procedures while the temporal location is modelled as metadata of measurements. In contrast, for mobile procedures the temporal location and the spatial location are both

modelled as metadata of measurements. Information for the quality of the measurement can be provided in contextual information.

The observation layer abstracts from the heterogeneity of sensor data. It consists of observation engines, observation writers, and observation readers. Observation engines may subscribe to streams of measurement information entities. An observation engine implements an interface with `execute` and `emit` methods. Thus, it executes measurement information entities and emits sensor observation information entities. Observation engines translate measurements and related contextual information into sensor observations. The semantics of sensor observations are defined by the Semantic Sensor Network Ontology (SSNO) (Compton, 2011). This translation overcomes the syntactic and semantic heterogeneity of sensor data by aligning sensor data to the syntax and semantics of the SSNO. Observation engines may make use of modules. For instance, a processing module may enrich sensor observations with information for their quality. Sensor observations are emitted as information entities to outgoing streams.

Observation writers may subscribe to streams of sensor observation information entities. An observation writer implements an interface with `execute` method. It executes sensor observation information entities and uses a store module to persist sensor observations. All sensor observations generated at the observation layer can thus be persisted. Persistence is implemented by the knowledge base. An observation writer does not further emit information entities.

Observation readers implement an interface with `emit` method. An observation reader typically uses a store module to retrieve sensor observations from the knowledge base. Sensor observations are emitted as information entities to outgoing streams. However, observation readers are not limited to retrieving sensor observations from the knowledge base. Indeed, implementations may retrieve data from any accessible resource and emit sensor observations.

The derivation layer abstracts from the sensor network dimension of sensor observations. It consists of dataset engines, derivation engines, derivation writers, and derivation readers. Dataset engines may subscribe to streams of sensor observation information entities. A dataset engine implements an interface with `execute` and `emit` methods. It executes sensor observation information entities and maps sensor observations to dataset observations. The semantics of dataset observations are defined by the RDF Data Cube Vocabulary (QB) (Cyganiak et al., 2013). Dataset observations are elements of datasets. Dataset observations are emitted as information entities to outgoing streams. Note that dataset observations emitted by a dataset engine may be streamed directly to the situation engine.

Derivation engines may subscribe to streams of dataset observation information entities. A derivation engine implements an interface with `execute` and `emit` methods. It performs computations on dataset observations. Such computation amounts to dataset processing, in that the dataset observations of one or more source datasets are processed to dataset observations of a target dataset. Derivation engines may associate to processing modules. Processing modules implement computations. Computations may be for, e.g., despiking, aggregation, interpolation, merging, filtering. Dataset observations are emitted as information entities to outgoing streams. Derivation engines can be chained in order to implement more complex dataset processing.

Derivation writers may subscribe to streams of dataset observation information entities. A derivation writer implements an interface with `execute` method. It executes dataset observation information entities and uses a store module to persist dataset observations. All dataset observations generated at the derivation layer can thus be persisted. A derivation writer does not further emit information entities.

Derivation readers implement an interface with `emit` method. A derivation reader typically uses a store module to retrieve dataset observations from the knowledge base. Dataset observations are emitted as information entities to outgoing streams. However, derivation readers are not limited to retrieving dataset observation from the knowledge base. Indeed, implementations may retrieve data from any accessible resource and emit dataset observations. Note that dataset observation emitted by a derivation reader may be streamed directly to the situation engine.

The situation layer abstracts from data. It consists of situation engines and situation writers. Situation engines may subscribe to streams of dataset observation information entities. A situation engine implements an interface with `execute` and `emit` methods. It employs learning modules to acquire situational knowledge from datasets. Learning modules perform computations over dataset observations. Computations may include methods in, e.g., machine learning or complex event processing. Situation engines represent situational knowledge as situations. The semantics of situations are defined by the Situation Theory Ontology (STO) (Kokar et al., 2009), which is grounded in Situation Theory (Barwise and Perry, 1983; Devlin, 1995). Situations are emitted as information entities to outgoing streams.

Situation writers may subscribe to streams of situation information entities. A situation writer implements an interface with `execute` method. It executes situation information entities and uses a store module to persist situations. All situations generated at the situation layer can thus be persisted. A situation writer does not further emit information entities.

3. Implementation

Wavellite is an implementation for the software architecture presented in Section 2. In this section we detail the adopted materials and methods. Wavellite builds on knowledge representation and reasoning, a distributed streamed data processing platform, as well as computational techniques in digital signal processing, complex event processing, and machine learning. Wavellite is implemented in Java.

Wavellite builds on the distributed and fault-tolerant real time computation system Storm to support (near) real time representation of situational knowledge and distributed processing of information entities. Storm provides three core abstractions: streams, spouts, and bolts. A stream is an unbounded sequence of tuples and a tuple is a named list of values.[1] Both spouts and bolts are Storm primitives that transform streams into new streams. A spout is a source of streams, i.e. a source of tuples for data (typically) read from an external source. Wavellite measurement engines, observation readers, and derivation readers are Storm spouts. In contrast to spouts, a bolt consumes one or more streams, processes tuples, and possibly emits one or more new streams. With the exception of measurement engines, Wavellite engines and writers are Storm bolts. Streams correspond to the edges while spouts and bolts correspond to the nodes of a graph, also known as Storm topology. A Storm topology is executed by a Storm cluster, which may operate in local or distributed mode. In local mode, the worker nodes of a Storm cluster are simulated with threads. In contrast, in distributed mode Storm operates as a cluster of machines. The Storm topology is generated at Wavellite application runtime from a Wavellite topology. A Wavellite topology is a user defined configuration for a Wavellite application. It specifies the components and modules, their interaction and configuration, that are relevant to a Wavellite application. A Wavellite topology is a document encoded in JavaScript Object Notation (JSON).

[1] https://github.com/nathanmarz/storm/wiki/.

The Wavellite observation, derivation, and situation layers build on materials in knowledge representation, specifically the Resource Description Framework (RDF) (Manola et al., 2004), RDF Schema (RDFS) (Brickley et al., 2004), and Web Ontology Language (OWL 2) (Hitzler et al., 2012) Semantic Web technologies. At the core, RDF is a language for representing information. RDF Schema is a semantic extension of RDF (Hayes and McBride, 2004) and provides the basic constructs to build RDF vocabularies, also called ontologies. Ontology, defined as a formal and explicit specification of a conceptualization (Gruber, 1993), is a means to represent knowledge of a domain, meaning the concepts of some area of interest and relations that hold among them. The Web Ontology Language, specifically OWL 2, builds on RDF and RDF Schema and introduces further constructs that allow for the building of ontologies with richer semantics.

Wavellite adopts an ontology for the representation of information at each layer above the measurement layer. The Semantic Sensor Network Ontology (SSNO) is adopted at the observation layer for the representation of sensor observations. The SSNO defines terms used for the description of sensors and observations. We highlight some that are of particular interest. The term `ssn:Sensor` is defined as an "entity that can follow a sensing method and thus observe some Property of a FeatureOfInterest".[2] Typically, sensors are devices. However, computational methods and people may also be sensors. According to the ontology, the term `ssn:Sensor` is an exact match with the term 'sensor' defined in the OGC Sensor Model Language standard. Fig. 4 (b) (Appendix B) is a representation of an example `ssn:Sensor`. A sensor observes a `ssn:Property`, which is defined as "an observable Quality of an Event or Object".[3] The term `ssn:Property` is an exact match with the term 'property' defined in the OGC Observations and Measurements standard. A `ssn:Property` is of a `ssn:FeatureOfInterest`, which is defined as "an abstraction of [environmental] phenomena".[4] The term `ssn:FeatureOfInterest` is an exact match with the term 'feature' defined in the OGC Observations and Measurements standard. Finally, the term `ssn:Observation` is defined in SSNO as "a Situation in which a Sensing method has been used to estimate or calculate a value of a Property of a FeatureOfInterest".[5] Fig. 4(a) (Appendix B) is the representation of an example `ssn:Observation`. The term `ssn:Observation` is a close match to the term 'observation' defined in the OGC Observations and Measurements standard. Overall, SSNO defines several dozen classes and properties. Sensor, property, feature, and observation are of specific interest to Wavellite.

At the derivation layer, Wavellite adopts the RDF Data Cube Vocabulary (QB). The term `qb:DataSet` is defined as "a collection of statistical data that corresponds to a defined structure"[6] and plays a central role in the modelling of datasets at the derivation layer. Fig. 5(b) (Appendix B) is the representation of an example `qb:DataSet`. A dataset relates to a `qb:DataStructureDefinition` with `qb:ComponentSpecification`(s). Component specifications relate to a `qb:ComponentProperty` and hold metadata for, e.g., the ordering of component properties. Component properties specify the dimensions of a dataset. The term `qb:Observation` is used to model dataset observations at the derivation layer. A dataset observation relates to a dataset and values for the component properties specified in the corresponding data structure definition. Thus, a dataset is a collection of dataset observations of a defined structure. Fig. 5(a) (Appendix B) is the representation of an example `qb:Observation`. QB builds on results from the Statistical Data and Metadata Exchange (SDMX) Initiative. Similar to QB, the SDMX Information Model defines an SDMX DataSet to be "a collection of a set of Observations that share the same dimensionality".[7]

At the situation layer, Wavellite adopts the Situation Theory Ontology (STO). STO relates to the Situation Theory developed by Barwise and Perry (1983) and extended by Devlin (1995). The theory formalizes the semantics of situations by means of the expression $s \vDash \sigma$ (read "s supports σ") meaning that the infon σ is "made factual" by the situation s. According to the definition by Devlin (1995), the object $\ll R,a_i,\ldots,a_m,i \gg$ is a well-defined infon if R is an n-place relation and $a_1,\ldots,a_m$ $(m \leq n)$ are objects appropriate for the argument places $i_1,\ldots,i_m$ of R, and if the filling of argument places $i_1,\ldots,i_m$is sufficient to satisfy the minimality conditions for R, and $i = 0,1$ is the polarity. Minimality conditions "determine which particular groups of argument roles need to be filled in order to produce an infon" (Devlin, 1995). The polarity is the 'truth value' of the infon. If $i = 1$ then the objects $a_1,\ldots,a_m$ stand in the relation R; else the objects do not stand in the relation R. Parameters, denoted as $\dot{a}$, make reference to *arbitrary* objects of a given type. For instance, $\dot{l}$ and $\dot{t}$ typically denote parameters for arbitrary objects of type spatial location and temporal location, respectively. Anchors are a mechanism to assign values to parameters. Hence, the parameter $\dot{t}$ may anchor the value for the current time. The STO follows the semantics of situations summarized here. It defines the term `sto:Situation` to model situations. Figure B.6 (Appendix B) is the representation of an example `sto:Situation`. A situation relates to one or more `sto:ElementaryInfon` via the property `sto:supportedInfon`. As in Situation Theory, an elementary infon consists of a `sto:Relation`, anchors one or more `sto:Object`, and relates to a `sto:Polarity`.

Aside SSNO, QB, and STO, Wavellite also adopts the WURVOC Ontology of units of Measure (OM) (Rijgersberg et al., 2011) to model quantities and dimensions as well as the Semantic Web for Earth and Environmental Terminology (SWEET) (Raskin and Pan, 2005), most importantly for the modelling of environmental phenomena, e.g. the feature related to a sensor observation or the object anchored by an infon of a situation, such as aerosol, air, or carbon dioxide.

The derivation layer performs transformations on datasets. Methods in digital signal processing (Rabiner and Gold, 1975), such as filtering, Fourier transform, interpolation or aggregation, are of interest at the derivation layer. Methods in digital signal processing may be used to, e.g., convert the observations of a dataset in time domain into observations of a dataset in frequency domain. Several Java libraries implement digital signal processing algorithms, e.g. the Modular Audio Recognition Framework (MARF) or JScience. Complex Event Processing (CEP) (Luckham, 2002) and related Java libraries such as Esper may also be of interest at the derivation layer to process datasets.

Of particular interest at the situation layer are methods in machine learning and data mining (Mitchell, 1997; Hand et al., 2001). Such methods play a key role in discovering situations from dataset observations. An example are Multilayer Perceptron artificial neural networks (MLP) (Haykin, 1999). An MLP network consists of a set of neurons that form the input layer, one or more hidden layers, and the output layer of the network. MLP is trained in a supervised manner, i.e. by means of a labelled training dataset, using error back-propagation learning, which consists of a forward pass and a backward pass through the layers. In the forward pass, the signal resulting from the application of an input vector is propagated through the network in a forward direction and the actual response of the network at the output layer is recorded. In the backward pass, the recorded response

[2] http://www.w3.org/2005/Incubator/ssn/ssnx/ssn#Sensor.
[3] http://www.w3.org/2005/Incubator/ssn/ssnx/ssn#Property.
[4] http://www.w3.org/2005/Incubator/ssn/ssnx/ssn#FeatureOfInterest.
[5] http://www.w3.org/2005/Incubator/ssn/ssnx/ssn#Observation.
[6] http://www.w3.org/TR/2013/CR-vocab-data-cube-20130625/#cubes-model-datasets.
[7] http://sdmx.org/docs/2_0/SDMX_2_0%20SECTION_02_InformationModel.pdf.

of the network at the output layer is subtracted from a desired response, i.e. the label, to produce an error signal, which is propagated through the network in a backward direction. In this pass the network is adjusted in order to align the actual response with the desired response. In supervised learning, the labelled training dataset is used to train (calibrate) the network. Once trained, the network can be used to classify input vectors for which the label is not known. At the situation layer, a learning module may use trained neural networks in order to classify input vectors represented as (multidimensional) dataset observations, elements of datasets of the derivation layer. The result of such classification allows for the detection of situations as well as for the acquisition of situational knowledge. There exist several Java libraries of interest to learning modules at the situation layer. A popular example is WEKA (Hall et al., 2009).

A Wavellite store module is for a specific knowledge base. We use the Stardog RDF database. The Stardog RDF database implements the persistence of RDF. Thus, all information entities generated at the observation, derivation, and situation layers may be persisted via store module to the Stardog RDF database. Wavellite can be extended with application specific store modules to support other RDF databases.

The Protégé Ontology Editor and Knowledge Acquisition System is useful for ontology management. It is particularly useful to the design of domain ontologies that extend the concepts and properties of ontologies discussed here, e.g. extend the SSNO concept for sensor with a domain specific hierarchy of sensor types and individuals representing deployed sensors.

4. Application

Wavellite can be implemented for specific applications. Figure E.8 (Appendix E) provides a schematic overview of Wavellite application implementation. The process begins with a domain ontology that imports SSNO, QB, and STO. The ontology is then populated with domain knowledge, for instance using Protégé. This may require a fair amount of expertise regarding the sensor network infrastructure, the sensed environmental phenomena, the purpose of sensing, and ontology engineering. The completed ontology is imported by the knowledge base.

Wavellite application implementation also requires extensions to the framework with custom (Java) program code. While any Wavellite component or module may be extended, most base implementations are intended to be reused. Main exceptions are the measurement engine and the situation engine. We recall that the responsibility of a measurement engine is to retrieve sensor data and to convert the data to measurements. The technicalities of retrieving and parsing sensor data very much depend on the sensing device and communication links and protocols. Thus, the base measurement engine expects the implementation of the program logic that is necessary to emit measurements and their contextual information.

The responsibility of a situation engine is to implement the program logic necessary for the acquisition of situational knowledge from dataset observations as well as the representation of situations. Situation engines are domain specific. Their implementation is determined by situational knowledge acquisition tasks of interest to the domain. Thus, the base situation engine expects the implementation of the program logic that is necessary for situational knowledge acquisition and representation.

Wavellite supports the implementation and reuse of processing and learning modules. However, the large number of methods for data processing and data driven learning, as well as the large number of domain specific problems, is likely to necessitate framework extension with modules required in applications.

There are two main modes of application execution. We may either create a Wavellite topology and execute it on a Storm cluster or we may use Wavellite in a (Java) application. They are aimed at different modes of operation. The former is designed for applications that require continuous (near) real time processing of current sensor data. In contrast, the latter is designed for applications that process historical sensor data. The use of Storm for historical data is technically possible but practically it is more direct to assemble the necessary program logic in a standalone application.

5. Experiment

We developed a Wavellite application for SMEAR IV-Puijo sensors for the acquisition and representation of situational knowledge for events of new particle formation and could events. The application was for data by the Differential Mobility Particle Sizer (DMPS) for particle size distribution and the Present Weather Sensor (PWS) for visibility and precipitation.

New Particle Formation (NPF) and the growth of newly formed particles have been well documented in a wide variety of environments all over the world (Kulmala et al., 2004). Aerosol particles are known to influence quality of life, for instance by affecting human health (Pope III et al., 2002). Newly formed nano-sized particles can grow, through condensation and coagulation processes, and directly effect on Earth's radiation balance by scattering sunlight. Indirectly, their potential to grow large enough to act as Cloud Condensation Nuclei (CCN) and their possible activation to cloud droplets results in more scattering of radiation. It is known that the scattering of radiation has a cooling effect on the climate (Solomon et al., 2007). However, the magnitude of indirect effects remains the single largest uncertainty in current estimates of anthropogenic radiative forcing (Solomon et al., 2007), leading to large uncertainties in the calculations of future climate change.

The study of NPF relies on methods for the identification and characterization of these atmospheric events. At the base of such methods is the measurement of particle size distribution for polydisperse aerosols. Of specific interest are particles with diameter size ranging 10^{-9}–10^{-6} m. The resulting data, as measured over the course of a day at a specific location, can be visualized to assess the presence of NPF (Dal Maso et al., 2005).

Different classifications have been proposed in order to characterize NPF (Dal Maso et al., 2005; Hamed et al., 2007; Vana et al., 2008). The criterion established by Hamed et al. (2007) is based on event clarity. According to the criterion, nucleation event classes 1, 2, and 3 indicate strong, intermediate, and weak nucleation event (E) days, respectively. Days during which no particle formation is observed are classified as non-event (NE). Days that are neither E nor NE are called undefined class, or class 0. The tasks of NPF identification and characterization are typically carried out visually by experts (Hamed et al., 2007).

We adopt the classification proposed by Hamed et al. (2007) and distinguish two (machine learning) classification tasks: NPF event identification and NPF event characterization. The former is for the classification of days as event (E) or non-event (NE). The latter is for the classification of event (E) days into classes 1, 2, or 3 for event clarity. We train, validate, and test MLP artificial neural networks for the two tasks of NPF event identification and characterization.

A could event at Puijo is considered to take place when average hourly visibility drops below 200 m. Cloud events are further classified as rainy if average hourly precipitation exceeds 0.2 mm h^{-1}. In contrast to NPF events, (rainy) cloud events are relatively straightforward to detect using complex event processing.

The sensors are installed on the Puijo observation tower. A DMPS consists of a Differential Mobility Analyser (DMA) and a Condensation Particle Counter (CPC). The particles of a polydisperse aerosol (source) are first classified according to diameter size by the DMA and then counted by the CPC (Kulkarni et al., 2011).

The DMPS measures the particle number concentration [cm^{-3}] for 40 discrete diameter sizes in the range 7–800 nm, on average 5 samples per hour. The PWS (Vaisala FD12P) measures visibility [m] and precipitation [mm h^{-1}], 1 sample per minute.

5.1. Implementation

In this section we describe the extensions made at the measurement, derivation, and situation layers in order to implement the application. We processed *historical* data for the period May 2007 through December 2011. Data was available in (daily) files. PWS data for visibility and precipitation was available as it was generated by the sensor. In contrast, DMPS data for particle number concentration was available post-processed. Post-processing is for DMPS sensor data inversion (Wiedensohler et al., 2012) and is performed in MATLAB. Thus, we used sets of sensor data for visibility and precipitation and sets of dataset observations for particle number concentration. We implemented a store module that streams generated RDF triples in N-Triples format to a plain text file. The motivation for this implementation is that the Stardog RDF Database is optimized for bulk load of RDF data at database creation time.

At the measurement layer, we processed PWS data for visibility and precipitation. Sensor data was converted to measurements and contextual information for sensor, property, feature, and temporal location. In contrast to PWS data, we did not process (the post-processed) DMPS data at the measurement layer. The motivation is three-folded. First, DMPS data requires a (non-trivial) inversion from sensor data in [V] to particle number concentration in [cm^{-3}]. The inversion was implemented in MATLAB. Second, the result of the inversion was readily available in daily files and could, thus, be used directly. Third, we can demonstrate that Wavellite is not restricted to inputting data at the measurement layer.

At the derivation layer, we processed sensor observations for visibility and precipitation to a dataset D_{pws} of dimensionality 3 for temporal location, and visibility and precipitation measurement. Furthermore, DMPS data were processed from daily files to a dataset D_{dmps}. D_{dmps} was a data matrix $m \times n$, where m was the number of samples and $n = 41$ was the temporal location plus the 40 discrete particle diameter sizes. We processed the data of D_{dmps} for each day between 6 AM and 6 PM using Singular Value Decomposition (SVD) to a vector, v_i, of size 40. Hyvönen et al. (2005) have argued that the time window between sunrise and sunset is a reasonable choice; we selected 6 AM through 6 PM as a rough approximation. Together with temporal location, v_i formed an observation of dataset D_{svd}. At this stage we also introduced the NPF label manually assigned for the historical data by experts, who assessed the label visually by plotting the particle size distribution over the course of each day. Thus, D_{svd} was a data matrix of dimensionality 42, for the temporal location, the NPF label, and v_i.

At the situation layer, we used the WEKA learning module implemented in Wavellite to classify observations of dataset D_{svd} using Multi-Layer Perceptron (MLP) artificial neural networks. To do so, we first needed to train MLP networks for the two classification tasks. For training and validation we used data from May 1, 2007 through December 31, 2010. The data for 2011 was used for test purposes. The generation of WEKA ARFF files from dataset D_{svd} is straightforward, as it amounts to the execution of SPARQL (Prud'hommeaux and Seaborne, 2008) queries and the processing of the result set to generate ARFF formatted files. Generated ARFF files were used for MLP networks training and validation as well as to train MLP networks used to classify test data.

For the acquisition of knowledge for events of NPF we trained two MLP networks, one for the identification (classes E and NE) and one for the characterization (classes 1, 2, and 3) of NPF events. Thus, we generated two ARFF files from D_{svd} for the period May 2007 through December 2010. The NPF labels were mapped to the corresponding training class for NPF event identification and characterization as shown in Table 1. The ARFF files were used to train MLP networks and evaluate their performance. The MLP networks consisted of 40 input neurons and one hidden layer with 21 hidden neurons. The number of output neurons was 2 and 3 for NPF identification and NPF characterization, respectively. The learning rate and momentum were set to 0.3 and 0.2, respectively. (Default WEKA configuration.) The MLP networks were validated using 10-fold cross validation, meaning that the training dataset was partitioned into 10 disjoint and equal-sized folds, and for each fold a classifier was trained using the other 9 folds and then validated on the fold (intermediate results were averaged). ARFF files were also used to test situational knowledge acquisition for events of NPF in 2011. For testing, observations of dataset D_{svd} for 2011 were classified by two WEKA learning modules, first to identify and, if identified, to characterize NPF events. For characterized NPF events, situations were generated and persisted. Such situations consisted of the infon

$$s \models \ll \text{npf}, \dot{c}, \dot{t}, 1 \gg$$

where 'npf' is the relation and $\dot{c}$ and $\dot{t}$ are parameters for the NPF event clarity classes 1, 2, or 3 and the temporal location for the day at which NPF was characterized, respectively.

The acquisition of knowledge for (rainy) cloud events occurred on observations of dataset D_{pws} for 2011. A complex event processing learning module was implemented to identify cloud events and their duration, i.e. time intervals during which visibility was continuously below 200 m. Furthermore, the module classified identified cloud events as rainy if mean precipitation during the time interval exceeded 0.2 mm h^{-1}. For (rainy) cloud events, situations were generated and persisted. Situations for cloud events consisted of the infon

$$s \models \ll \text{cloud} - \text{event}, \dot{t_1}, \dot{t_2}, \dot{v}, 1 \gg$$

where $\dot{t_1}$ and $\dot{t_2}$ are parameters for the temporal locations at which the cloud event begins and ends, respectively, and $\dot{v}$ is the parameter for mean visibility computed over the time interval $[\dot{t_1}, \dot{t_2}]$. Situations for rainy cloud events consisted of the infon

$$s \models \ll \text{rainy} - \text{cloud} - \text{event}, \dot{t_1}, \dot{t_2}, \dot{v}, \dot{p}, 1 \gg$$

where $\dot{p}$ is the parameter for mean precipitation computed over the time interval $[\dot{t_1}, \dot{t_2}]$.

5.2. Results

Executing the application generated approximately 100 million RDF triples for the period May 2007 through December 2011. At the observation layer, approximately 4.5 million sensor observations were generated from measurements for visibility and precipitation. These sensor observations were represented by approximately 67 million RDF triples. At the derivation layer, approximately 2.5 million dataset observations were generated for the DMPS and PWS

Table 1
The key characteristics of training datasets for NPF identification and NPF characterization. The table provides an overview of the mapping from (expert) label to training class. The number (#) of samples are included.

Label	Learning tasks			
	NPF identification		NPF characterization	
	Class	#	Class	#
NE	NE	195		
1	E	126	1	10
2			2	38
3			3	78

data. These dataset observations were represented by approximately 37 million RDF triples.

Figure B.4 (Appendix B) and Listing 1 (Appendix C) show the representation of an example sensor observation made on December 29, 2011 at 3:48 AM for the visibility of 161 m in air at Puijo by the Vaisala FD12P sensing device. For readability, we expand the sensing device separately. We highlight the modelling of the platform on which the sensing device resides. In fact, we reuse the GeoNames identifier for the Puijo Tower (in Finnish *Puijon torni*). We, thus, inherit the information for, e.g., the tower's longitude and latitude coordinates from GeoNames. Given that it observes three properties (namely temperature, precipitation and visibility) we underscore that the Vaisala FD12P may also be modelled as a system consisting of three sensors. We do not further expand here on the pros and cons of such modelling choices and leave them at the discretion of domain modellers. Figure B.5 and Listing 2 show the representation of an example dataset observation made on December 29, 2011 at 3:48 AM for the visibility and precipitation having values 165 m and 1.46 mm h^{-1}. The structure definition of the dataset is expanded separately. The example shows that the dataset observation consists of the values for the component properties defined in the structure of the dataset it relates to, namely for time, visibility, and precipitation. Finally, Figure B.6 and Listing 3 show the representation of an example situation for a rainy cloud event made on December 29, 2011 lasting between 3 and 5 AM with mean precipitation 1.18 mm h^{-1}. The situation supports an infon with `rainly-cloud-event` relation anchoring four attributes for the temporal locations at which the cloud event begins and ends (and $\dot{t_2}$), for mean visibility ($\dot{v}$) as well as for mean precipitation ($\dot{p}$) during $[\dot{t_1}, \dot{t_2}]$.

We validated MLP classifiers using WEKA. Classification performance (correctly classified instances) of the NPF identification learning task was 89%. Classification performance of the NPF characterization task was 58%. We recall that we used 10-fold cross-validation to validate MLP classifiers. Table 2 provides detailed precision, recall, and F-measure figures by class. As expect from classification performance, precision and recall for NPF identification and NPF characterization is good and mediocre, respectively. Indeed, all samples for class 1 were confused as either class 2 or class 3. The number of samples for the three classes of NPF characterization is insufficient, especially for class 1.

Using trained MLP classifiers, at the situational layer we identified and characterized 45 situations for events of NPF in 2011. Fig. 2 provides an overview. Most events are characterized as clarity 3, namely 34 (approximately 75%). The number of events characterized as clarity 1 and clarity 2 are 4 and 7, respectively. In contrast, experts have characterized 43 situations for events of NPF in 2011, of which 29 are of clarity 3, 3 are of clarity 1, and 11 are of clarity 2. Indeed, if we evaluate the classification performance (correctly classified instances) using trained classifiers on 2011 test sets the result is 88% for NPF identification and 63% for NPF characterization, which is slightly better than using 10-fold cross-validation (58%).

At the situational layer, we also identified and characterized 126 cloud events during 2011, of which 52 were rainy. The longest cloud event lasted 28 h between November 29 at 2 PM and November 30 at 6 PM with mean visibility 110 m. The rainy cloud event with maximum mean precipitation occurred on August 8 lasting from 3 AM to 4 AM with mean visibility 161 m and mean precipitation 6.4 mm h^{-1} Fig. 3 shows the (rainy) cloud events in December 2011. During December 2011 there were 31 cloud events, of which 8 were rainy. The cloud event with lowest visibility occurred on December 16 with mean visibility 101 m. The rainy cloud event with maximum precipitation occurred on December 29 with mean precipitation 1.2 mm h^{-1}. As shown in Fig. 3, some cloud events are longer than others.

Table 2
Precision, recall, and F-measure by class.

Measure	Learning tasks				
	NPF identification		NPF characterization		
	E	NE	1	2	3
Precision	0.875	0.896	0.000	0.333	0.698
Recall	0.833	0.923	0.000	0.342	0.769
F-Measure	0.854	0.909	0.000	0.338	0.732

6. Discussion

We provided an overview of Wavellite (Stocker et al., submitted for publication), a software framework for the representation of situational knowledge acquired from sensor data. We discussed in details the current version of its logical structure, in particular the four layers, the components, and modules. We presented the implementation of the architecture and the materials and methods the implementation builds on.

Extending our previous work (Stocker et al., 2013), the aim of this study was to apply Wavellite to SMEAR IV-Puijo sensor data and the representation of situational knowledge for events of atmospheric New Particle Formation (NPF) acquired from dataset observations by machine learning. In addition we processed Present Weather Sensor (PWS) data for visibility and precipitation from measurements and contextual information to sensor observations and dataset observations to represent situational knowledge for (rainy) cloud events. The results show a good classification performance for NPF identification (approximately 90%) and a mediocre classification performance for NPF characterization (approximately 60%). Thus, the system can be used for automated identification of NPF event days whereas expert review is required to determine NPF event clarity.

We demonstrated for (rainy) cloud events how explicitly represented situational knowledge can be of value to scientists. The term 'cloud event' has a determined interpretation in PWS data for visibility. This interpretation is shared between scientists and Wavellite. Thus, Wavellite is able to represent situational knowledge for cloud events and scientists can interact with Wavellite using the term. It is, hence, possible for scientists to query Wavellite for, e.g., the cloud event with lowest visibility and longest duration in a given time interval. Similar examples can be made for events of NPF.

The fundamental problem addressed in this work is known to other fields. For instance, Coradeschi and Saffiotti (2003) introduce 'anchoring' as the process of establishing and maintaining the

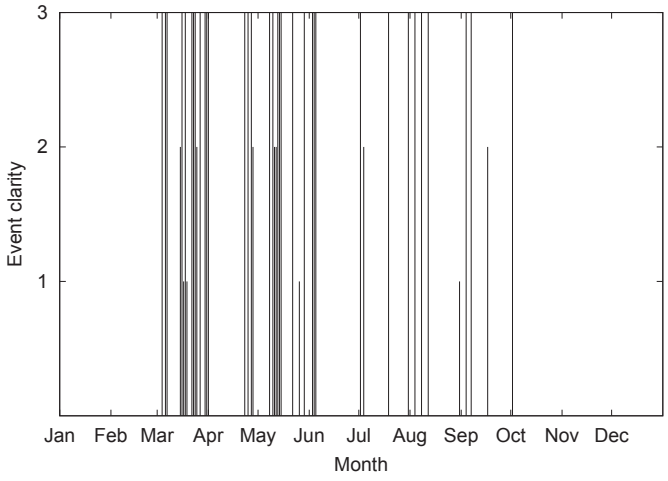


Fig. 2. Situations for events of NPF for 2011 acquired and represented at the situational layer. The clarity of the event is shown on the *y*-axis (1, 2, or 3).

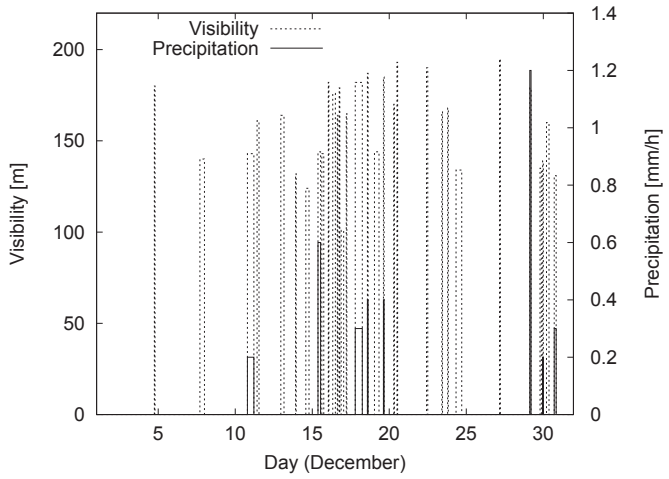


Fig. 3. Situations for (rainy) cloud events for December 2011. Visibility of cloud events and precipitation of rainy cloud events are shown. The width of each step represents the duration of the cloud event.

correspondence between abstract representations and perceptual (sensor) data "that refer to the same physical objects in the external world" (Coradeschi and Saffiotti, 2003). Their focus is on robotics and autonomous systems. The authors argue that "anchoring must necessarily take place in any robotic system that comprises a symbolic reasoning component" (Coradeschi and Saffiotti, 2003). We think that anchoring is more generally a useful concept to domains that employ sensors and process sensor data with the aim of improving the understanding of real world phenomena. Indeed, the process of anchoring is fundamental to Wavellite as the framework aims at drawing a correspondence between abstract representations for environmental phenomena and sensor data that explicitly or implicitly hold information about the phenomena. We deliberately use the term 'phenomenon' and contrast it with the 'object' of typical interest in robotics.

Wavellite supports the semantically rich modelling of information at each layer of abstraction. Such modelling is an interesting aspect for several reasons. First, Wavellite commits to reuse formal terminology and semantics that are explicitly defined in ontologies. Indeed, most of the terminology used by Wavellite is defined in SSNO, QB, and STO. Wavellite applications often only trivially extend these ontologies with domain knowledge. One of the benefits of this commitment is that the reuse of terminology explicitly represented in ontology can increase the interoperability between systems (Obrst, 2003; Uschold and Gruninger, 2004). However, in ontology based system it is not just software that commits to reuse terminology defined in ontology: human experts do so as well. In fact, knowledge provided by domain experts is formalized and extends SSNO, QB, and STO with information about the sensor network and observed environmental phenomena at the observation layer, information about datasets at the derivation layer, and information about situations of interest to the domain at the situation layer. Wavellite and third party software systems use domain expert knowledge to "learn" about, and adapt to, specific domains.

The three adopted ontologies—SSNO, QB, and STO—can serve as a modelling framework for the problem of sensor data organization and interpretation. This framework provides an organization of relevant generic concepts and relations. For instance, at the observation layer, the SSNO implements the so-called Stimulus-Sensor-Observation ontology design pattern (Janowicz and Compton, 2010; Compton, 2011). According to this pattern, sensors observe properties of features by detecting stimuli that are (directly or indirectly) related to properties. Moreover, sensors implement a procedure that transforms stimuli to results. Observations link sensors and stimuli. Thus, in committing to SSNO, domain experts adopt this pattern for the modelling of sensor networks and observations. Similarly, at the situation layer, Situation Theory (Barwise and Perry, 1983; Devlin, 1995) serves as an organization of generic concepts, relations, and their semantics for the modelling of situations. In committing to STO, domain experts adopt Situation Theory for the modelling of situations.

We have demonstrated the organization and interpretation of sensor data for SMEAR IV-Puijo, in particular for DMPS and PWS data. Organization was achieved at the observation and derivation layers with shared syntax and semantics for sensor observations and dataset observations as well as by supporting the persistence and retrieval of observations. Interpretation was achieved at the situation layer with functionality aimed at the acquisition and representation of situational knowledge for events of NPF and (rainy) cloud events.

Wavellite is not limited to the ontologies presented here. Indeed, other ontologies may be adopted, e.g. to represent time or space. For instance, the ontology used by PelletSpatial (Stocker and Sirin, 2009) may be adopted to represent qualitative spatial relations of the Region Connection Calculus (Randell et al., 1992) between regions, to allow for qualitative spatial reasoning. The WURVOC Ontology of units of Measure (OM) can be adopted to model quantities and dimensions and the Semantic Web for Earth and Environmental Terminology (SWEET) for the modelling of environmental phenomena.

Wavellite was designed with earth and environmental science as its target domain. It can be operated in a real time context to continuously represent situational knowledge acquired from processed sensor data. However, it can also serve as a computational platform to explore and prototype applications. In earth and environmental science, at the beginning of a workflow there may be historical sensor data. Experts may be unfamiliar with the data, unclear about the data processing, and perhaps unclear what situational knowledge to acquire. Wavellite may be used to explore and prototype such an application, layer by layer. Application developers first need to know what sensor data is available and how to retrieve and parse the data. These concerns matter at the measurement layer. Once these are known, at the observation layer the focus turns to applying the Stimulus-Sensor-Observation pattern to measurement data. At this stage, experts are more familiar with the available measurement data as well as the domain, including details about the observed environmental phenomena. This familiarity may also clarify the aims at the situational layer. Having the aims at the situational layer set, the derivation layer can be used to explore and prototype dataset processing chains that transform dataset observations in preparation for situational knowledge acquisition. This may often be a laborious task but systems such as Wavellite may support the process.

Wavellite can be used in combination with other software and scripts used by earth and environmental scientists to process data. Such software may provide input to Wavellite as well as consume output from Wavellite. Input can be provided at the measurement, observation, and derivation layers. Output can be retrieved from the observation, derivation, and situation layers. We think such features are of particular interest to earth and environmental science applications that build on sensor networks and process sensor data to, specifically, acquire and represent situational knowledge.

To the best of our knowledge, Wavellite is unique in its support for the representation of situational knowledge acquired from heterogeneous sensor data for environmental phenomena. It is a practical architecture with a concrete implementation that can be extended to meet the requirements of specific application.

7. Related work

Work related to Wavellite can be organized along several lines. First, the work relates to other software architectures that aim at the representation of knowledge acquired from sensor data. Second, Wavellite relates to the development of ontologies for sensor networks and sensor data as well as the development of systems for the semantic enrichment of sensor data. Third, it relates to work that applies the SSNO, QB, or STO to domains that build on sensor networks and sensor data. Fourth, our work relates to software systems that manage, process, or mine sensor data and were developed *ad hoc* for a specific domain and purpose, in particular the study of atmospheric new particle formation.

Clemente et al. (2013) propose a software architecture aimed at collision avoidance of ships in harbour areas. Their recent work relates to previous work on a software architecture aimed at the identification of, and reasoning about, situations (Furno et al., 2011) as well as the use of the STO to represent airport security situations (Fenza et al., 2010). The architecture presented by Clemente et al. (2013) relates to the one implemented by Wavellite in that, broadly speaking, both aim at the representation of situational knowledge acquired from sensor data. Moreover, both architectures leverage on the SSNO and Situation Theory. However, there are important differences. First, in contrast to Clemente et al. (2013) Wavellite suggests using a third ontology to model datasets and a corresponding layer at which arbitrary transformations on datasets can be computed. Second, our target domain of application is earth and environmental sciences. Different domains pose different requirements on software systems with comparable aims. It is important to identify and characterize such differences. Third, Wavellite proposes a concrete implementation of its architecture and is released open source.

Gorrepati et al. (2013) propose a software architecture for semantic modelling of bird ecology. The architecture consists of a physical layer (microphone array), an event layer (pattern matching, feature extraction, classification), a semantic layer (representation of location, species, numbers), an awareness layer (modelling of movement, location, interaction, health), and a service layer (for tracking, behaviour, richness and diversity). Their work relates to Wavellite in that raw sensor measurement data for the signal of a property (bird call) of a physical phenomenon (birds) is processed to knowledge for situations involving the phenomenon of interest. The work also relates to Wavellite in the target application domain. One of the main differences is that by building on generic terminology, Wavellite is reusable across applications. In fact, Gorrepati et al. (2013) directly extend the top level concept `owl:Thing` with domain specific terminology, e.g. `Bird`. Their modelling is, therefore, *ad hoc* for the semantic modelling of bird ecology.

As we briefly mentioned, our work also relates to the field of robotics, in particular to the notion of anchoring. In addition to Coradeschi and Saffiotti (2003) and Daoutis (2013), we highlight the work by Vassev and Hinchey (2012). The authors underscore that converting sensor data to symbolic knowledge is a challenge in cognitive robotic systems. According to Vassev and Hinchey (2012), the raw sensor data of a robotic system for the measured properties of physical phenomena must be converted to "programming variables or more complex data structures that represent collections of sensory data" which "must be labelled with [knowledge representation] symbols." Naturally, cognitive robotic systems have a substantially different set of requirements and challenges compared to similar software systems designed for the organization and interpretation of environmental sensor network data.

The relevance of ontologies to sensor networks and sensor data is well documented in the literature. Sheth et al. (2008) present the Semantic Sensor Web in which sensor data is annotated with spatial, temporal, and thematic semantic metadata. Terminologies to describe the characteristics of sensors and sensor networks have received considerable attention (Avancha et al., 2004; Eid et al., 2006). Compton et al. (2009) review eleven sensor ontologies for their range and expressive power.

The semantic enrichment of sensor data has been studied. Moraru and Mladenić (2012) propose a framework that resembles the Wavellite measurement and observation layers. Wanner et al. (2011) present an environmental information system in which environmental data, e.g. temperature measurements for a city, are stored in an OWL knowledge base. Barnaghi et al. (2009) propose a semantic model for (heterogeneous) sensor data representation, discussed using both XML and OWL. Wei and Barnaghi (2009) annotate sensor data with semantic metadata and relate sensor data with data of knowledge bases available online, such as DBpedia (Bizer et al., 2007), following the linked data principle (Berners-Lee, 2006). The principle has also been applied in works that aim at making sensor data accessible as linked data (Kessler and Janowicz, 2010).

An advantage of the semantic enrichment of sensor data is that systems can leverage ontology and rule based reasoning to infer new knowledge from sensor data. This practice is well documented, for instance by Sheth et al. (2008) to infer a 'blizzard condition'; Henson et al. (2009) to infer 'high winds' observations; Stocker et al. (2011) to infer the nutrient status of lakes; Wei and Barnaghi (2009) to infer the approximate temperature for a city neighbouring another city for which the temperature is known. If a knowledge acquisition task on sensor data can be expressed using rule languages, representing sensor data in ontology is attractive, as it allows us to leverage the powerful reasoning capabilities of inference engines. However, not all knowledge acquisition tasks on sensor data can be formalized using state of the art ontology and rule languages, e.g. the task of NPF identification presented in this study.

Using the SSNO, Barnaghi et al. (2012) describe a framework aimed at creating perception from sensor data. The authors motivate their work by underscoring how data consumers "are often interested in the higher-level concepts, such as events," rather than low-level sensor data. Contrary to Barnaghi et al. (2012) and our previous work (Stocker et al., 2012a,b) here we make use of the STO to represent knowledge at the most abstract level, leaving the SSNO at an intermediate level for the semantic enrichment of sensor data. Lefort et al. (2012) use the QB in combination with the SSNO to model a dataset with daily temperature sensor data. Their work relates to the Wavellite observation and derivation layers. The clear difference is the Wavellite situation layer. The STO has also been used in various studies. Aside Fenza et al. (2010) who use the STO to represent airport security situations, De Maio et al. (2012) present an approach to identify situations represented using the STO and Doulaverakis et al. (2011) use the STO in an architecture for intelligent information fusion in a sensor network environment, demonstrated for the domain of security and surveillance. In contrast, we suggest adopting the SSNO, QB, and STO at different levels of abstraction. Persistence of, and possibly inference on, sensor observations motivates the intermediate observation layer while arbitrary processing of dataset observations, as well as their persistence, motivates the intermediate derivation layer.

Wavellite relates to *ad hoc* systems that manage and process sensor data to organize such data, provide declarative query interfaces, and domain specific services, such as visualizations. An example related to this study is Smart-SMEAR (Junninen et al., 2009). Smart-SMEAR is a software system and Web interface for the visualization of data measured at the SMEAR II station in Hyytiälä, Finland. The measurement station has recorded data for gas, aerosol, and meteorological variables since 1996. Smart-SMEAR was designed and developed to manage the data, provide

visualizations, support querying and export, as well as some data mining functionality. Junninen et al. (2009) provide some implementation details. Of most interest here is that Smart-SMEAR builds on classical relational database management technologies, specifically a MySQL database. As we discussed earlier, one of the key advantages of ontology based systems such as Wavellite is that they reuse machine processable and interpretable terms and semantics defined explicitly in ontologies which can be shared across domains of application. Hence, they support rich semantic modelling of domain knowledge and increase the interoperability between human-machine systems.

The acquisition of knowledge for atmospheric new particle formation at the situation layer relates to work that applies data mining techniques on sensor data to study atmospheric new particle formation. Hyvönen et al. (2005) used clustering and classification of SMEAR II sensor data to identify the key parameters that explain new particle formation. The authors found that just two parameters, namely relative humidity and the condensation sink, are capable of explaining 88% of new particle formation events. Data mining techniques are relevant to Wavellite learning modules. However, with knowledge representation Wavellite goes clearly beyond mere knowledge acquisition.

8. Conclusions

Using the Wavellite software framework (Stocker et al., submitted for publication) for the organization and interpretation of sensor data and extending our previous work (Stocker et al., 2013), the aim of this study was to develop a Wavellite application for the representation of situational knowledge acquired from data by sensors of the Finnish Station for Measuring Ecosystem-Atmosphere Relations (SMEAR) for events of atmospheric new particle formation and (rainy) cloud events.

Wavellite supports the processing of heterogeneous sensor data to sensor observations with homogeneous syntax and semantics; the mapping of sensor observations to dataset observations and the processing of datasets; the acquisition of situational knowledge from datasets; and the representation of situational knowledge. We have provided an overview of the current version of the logical structure of its architecture and the materials and methods adopted by the implementation.

Our results show that Wavellite can be used to organize heterogeneous sensor data, interpret sensor data, and represent situational knowledge for atmospheric phenomena. Furthermore, the results show that Wavellite can support scientists in the identification and characterization of atmospheric new particle formation. Knowledge automatically acquired and represented by Wavellite can guide the manual review by scientists. The automated assessment can reduce the resources required in scientific computing workflows as well as reduce individual bias in manual assessment. Moreover, the knowledge layer provided by Wavellite can be a useful resource to scientists. We have shown how situational knowledge for (rainy) cloud events can be visualized and used to compute statistics, such as the maximum duration of cloud events or maximum precipitation of rainy cloud events. Similar conclusions can be drawn for events of new particle formation.

Together with our previous work on Wavellite, namely to represent situational knowledge acquired from road pavement vibration sensor data for vehicles (Stocker et al., 2012b) and situational knowledge for the exposure to carbon monoxide in a residential home (Stocker et al., 2012a), this study underscores that Wavellite provides a generic modelling framework for the problem of sensor data organization and interpretation. Its architecture includes the key aspects of a generic solution, in particular the modelling of sensor observations, the modelling of dataset observations, and the modelling of situational knowledge. Wavellite frames the addressed problem and proposes a concrete terminology, approach, and technologies to build on.

To support the requirements of applications, the framework can be extended with application specific program logic that includes algorithms in, e.g., digital signal processing, complex event processing, or machine learning. Application specific knowledge is accommodated by extension of three upper ontologies adopted in Wavellite for the representation of knowledge about sensor networks and sensor observations, the representation of datasets and dataset observations, and the representation of situational knowledge acquired from dataset observations. Knowledge is represented explicitly using the terms and semantics defined in ontologies. With Semantic Web technologies, Wavellite builds on open and *de facto* standard knowledge representation languages, on increasingly known and supported related software systems, and inherits the functionality for inference and query typically supported by systems.

Acknowledgement

We wish to thank the Finnish Funding Agency for Technology and Innovation (TEKES) CLEEN SHOK programme “Measurement, Monitoring and environmental Assessment” for supporting this work (MMEA, decision number 427/10).

Appendix A. Internet Resources

Appendix A.1. Specifications

DAML + OIL: http://www.daml.org/2001/03/daml+oil-index
N-Triples: http://www.w3.org/2001/sw/RDFCore/ntriples/
OM: http://www.opengeospatial.org/standards/om
OWL: http://www.w3.org/TR/owl-features/
OWL 2: http://www.w3.org/TR/owl2-primer/
QB: http://www.w3.org/TR/vocab-data-cube/
RDF: http://www.w3.org/TR/rdf-primer/
RDFS: http://www.w3.org/TR/rdf-schema/
SDMX: htt://www.sdmx.org
SensorML: http://www.opengeospatial.org/standards/sensorml
SSNO: http://www.w3.org/2005/Incubator/ssn/ssnx/ssn
STO: http://vistology.com/ont/2008/STO/STO.owl
SWEET: http://sweet.jpl.nasa.gov/ontology/
WURVOC: http://www.wurvoc.org/vocabularies/om-1.8/

Appendix A.2. Software

Apache Cassandra: http://cassandra.apache.org
Esper: http://esper.codehaus.org/
JScience: http://jscience.org/
MARF: http://marf.sourceforge.net/
Protégé: http://protege.stanford.edu/
Smart-SMEAR: http://www.atm.helsinki.fi/smartSMEAR/
Stardog: http://stardog.com
Storm: http://storm-project.net/
WEKA: http://www.cs.waikato.ac.nz/ml/weka/

Appendix A.3. Other

SMEAR: http://www.atm.helsinki.fi/SMEAR/

Appendix B. RDF Graphs

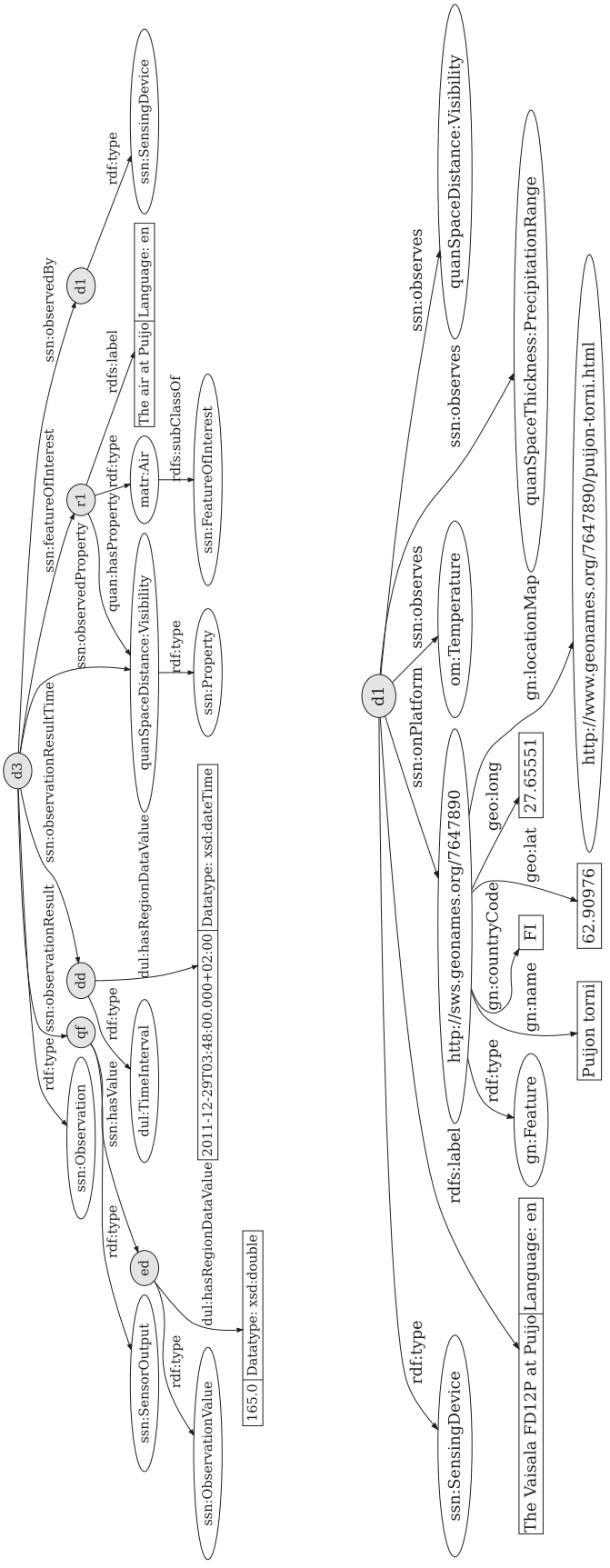
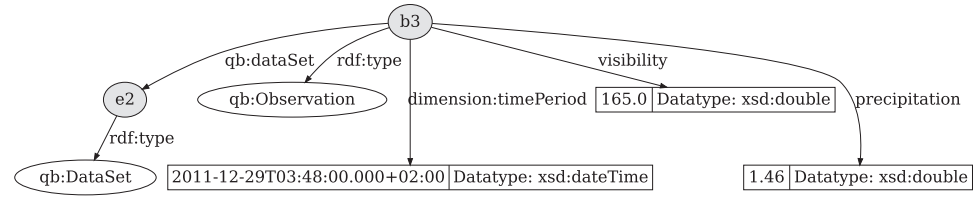
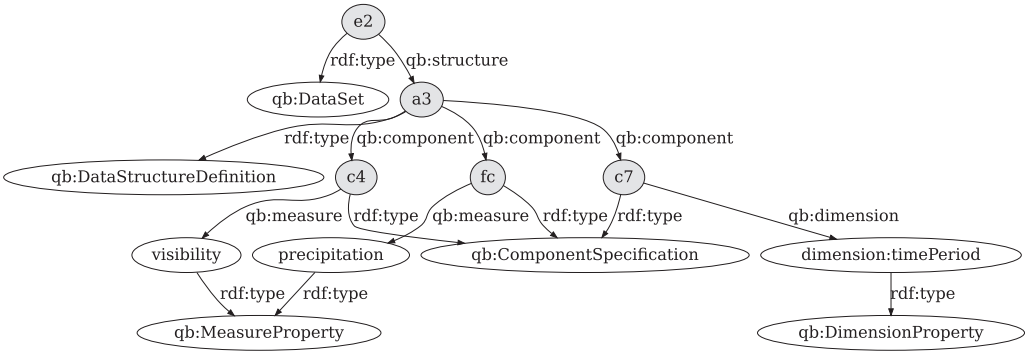


Figure B.4. RDF graphs showing the representation of a sensor observation and the sensor that made the observation. The greyed nodes correspond to named individuals with UUID fragment, here abbreviated for the sake of readability.



(a) Observation



(b) Dataset

Figure B.5. RDF graphs showing the representation of a dataset observation.

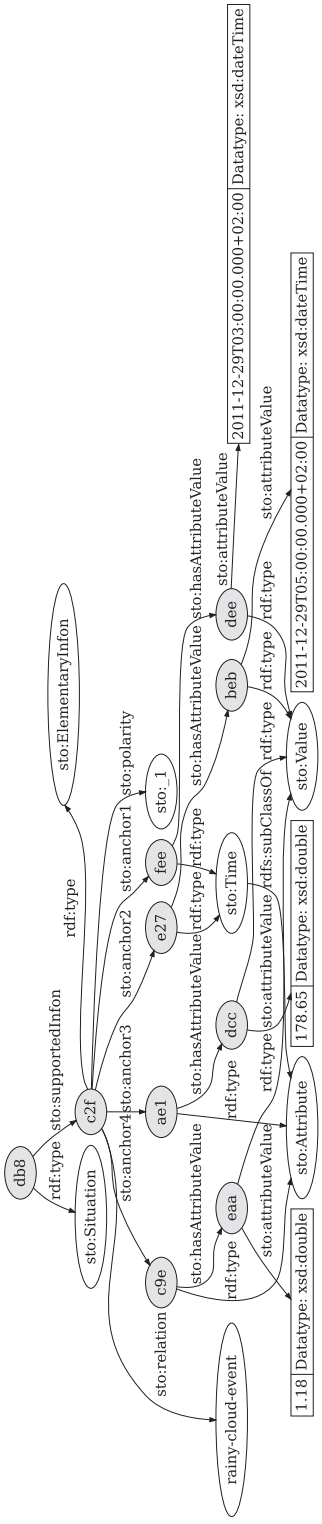


Figure B.6. RDF graph showing the representation of a situation for a rainy cloud event.

Appendix C. RDF Listings

Listing 1: RDF for the example sensor observation

```
@prefix : <http://envi.uef.fi/wavellite#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#> .
@prefix quanSpaceDistance:
        <http://sweet.jpl.nasa.gov/2.2/quanSpaceDistance.owl#> .
@prefix dul: <http://www.loa-cnr.it/ontologies/DUL.owl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix gn: <http://www.geonames.org/ontology#> .
@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix om: <http://www.wurvoc.org/vocabularies/om-1.8/> .
@prefix quan: <http://sweet.jpl.nasa.gov/2.2/quan.owl#> .
@prefix matr: <http://sweet.jpl.nasa.gov/2.2/matr.owl#> .
@prefix quanSpaceThickness:
        <http://sweet.jpl.nasa.gov/2.2/quanSpaceThickness.owl#> .

:d3 rdf:type ssn:Observation .
:d3 ssn:observationResult :qf .
:d3 ssn:observationResultTime :dd .
:d3 ssn:featureOfInterest :r1 .
:d3 ssn:observedProperty quanSpaceDistance:Visibility .
:d3 ssn:observedBy :d1 .

:qf rdf:type ssn:SensorOutput .
:qf ssn:hasValue :ed .
:ed rdf:type ssn:ObservationValue .
:ed dul:hasRegionDataValue "165.0"^^xsd:double .

:dd rdf:type dul:TimeInterval .
:dd dul:hasRegionDataValue "2011-12-29T03:48:00.000+02:00"^^xsd:dateTime .

:r1 rdf:type matr:Air .
:r1 rdfs:label "The␣air␣at␣Puijo"@en .
:r1 quan:hasProperty quanSpaceDistance:Visibility .
matr:Air rdfs:subClassOf ssn:FeatureOfInterest .

quanSpaceDistance:Visibility rdf:type ssn:Property .

:d1 rdf:type ssn:SensingDevice  .
:d1 rdfs:label "The␣Vaisala␣FD12P␣at␣Puijo"@en .
:d1 ssn:onPlatform <http://sws.geonames.org/7647890> .
:d1 ssn:observes om:Temperature .
:d1 ssn:observes quanSpaceThickness:PrecipitationRange .
:d1 ssn:observes quanSpaceDistance:Visibility .

<http://sws.geonames.org/7647890> rdf:type gn:Feature .
<http://sws.geonames.org/7647890> gn:name "Puijon␣torni" .
<http://sws.geonames.org/7647890> gn:countryCode "FI" .
<http://sws.geonames.org/7647890> geo:lat "62.90976" .
<http://sws.geonames.org/7647890> geo:long "27.65551" .
<http://sws.geonames.org/7647890> gn:locationMap
        <http://www.geonames.org/7647890/puijon-torni.html> .
```

Listing 2: RDF for the example dataset observation

```
@prefix : <http://envi.uef.fi/wavellite#> .
@prefix qb: <http://purl.org/linked-data/cube#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix terms: <http://purl.org/dc/terms/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix dimension: <http://purl.org/linked-data/sdmx/2009/dimension#> .

:b3 rdf:type qb:Observation .
:b3 dimension:timePeriod "2011-12-29T03:48:00.000+02:00"^^xsd:dateTime .
:b3 :visibility "165.0"^^xsd:double .
:b3 :precipitation "1.46"^^xsd:double .
:b3 qb:dataSet :e2 .

:e2 rdf:type qb:DataSet .
:e2 qb:structure :a38 .

:a38 rdf:type qb:DataStructureDefinition .
:a38 qb:component :c79 .
:a38 qb:component :c4b .
:a38 qb:component :fc4 .

:c79 rdf:type qb:ComponentSpecification .
:c79 qb:dimension dimension:timePeriod .

:c4b rdf:type qb:ComponentSpecification .
:c4b qb:measure :visibility .

:fc4 rdf:type qb:ComponentSpecification .
:fc4 qb:measure :precipitation .

dimension:timePeriod rdf:type qb:DimensionProperty .
:visibility rdf:type qb:MeasureProperty .
:precipitation rdf:type qb:MeasureProperty .
```

Listing 3: RDF for the example situation

```
@prefix : <http://envi.uef.fi/wavellite#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix sto: <http://vistology.com/ont/2008/STO/STO.owl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

:db8 rdf:type sto:Situation .
:db8 sto:supportedInfon :c2f .

:c2f rdf:type sto:ElementaryInfon .
:c2f sto:relation :rainy-cloud-event .
:c2f sto:anchor1 :fee .
:c2f sto:anchor2 :e27 .
:c2f sto:anchor3 :ae1 .
:c2f sto:anchor4 :c9e .
:c2f sto:polarity sto:_1 .

:fee rdf:type sto:Time .
:fee sto:hasAttributeValue :dee .
:dee rdf:type sto:Value .
:dee sto:attributeValue "2011-12-29T03:00:00.000+02:00"^^xsd:dateTime .

:e27 rdf:type sto:Time .
:e27 sto:hasAttributeValue :beb .
:beb rdf:type sto:Value .
:beb sto:attributeValue "2011-12-29T05:00:00.000+02:00"^^xsd:dateTime .

:ae1 rdf:type sto:Attribute .
:ae1 sto:hasAttributeValue :dcc .
:dcc rdf:type sto:Value .
:dcc sto:attributeValue "178.65"^^xsd:double .

:c9e rdf:type sto:Attribute .
:c9e sto:hasAttributeValue :eaa .
:eaa rdf:type sto:Value .
:eaa sto:attributeValue "1.18"^^xsd:double .

sto:Time rdfs:subClassOf sto:Attribute .
```

Appendix D. Component Interfaces

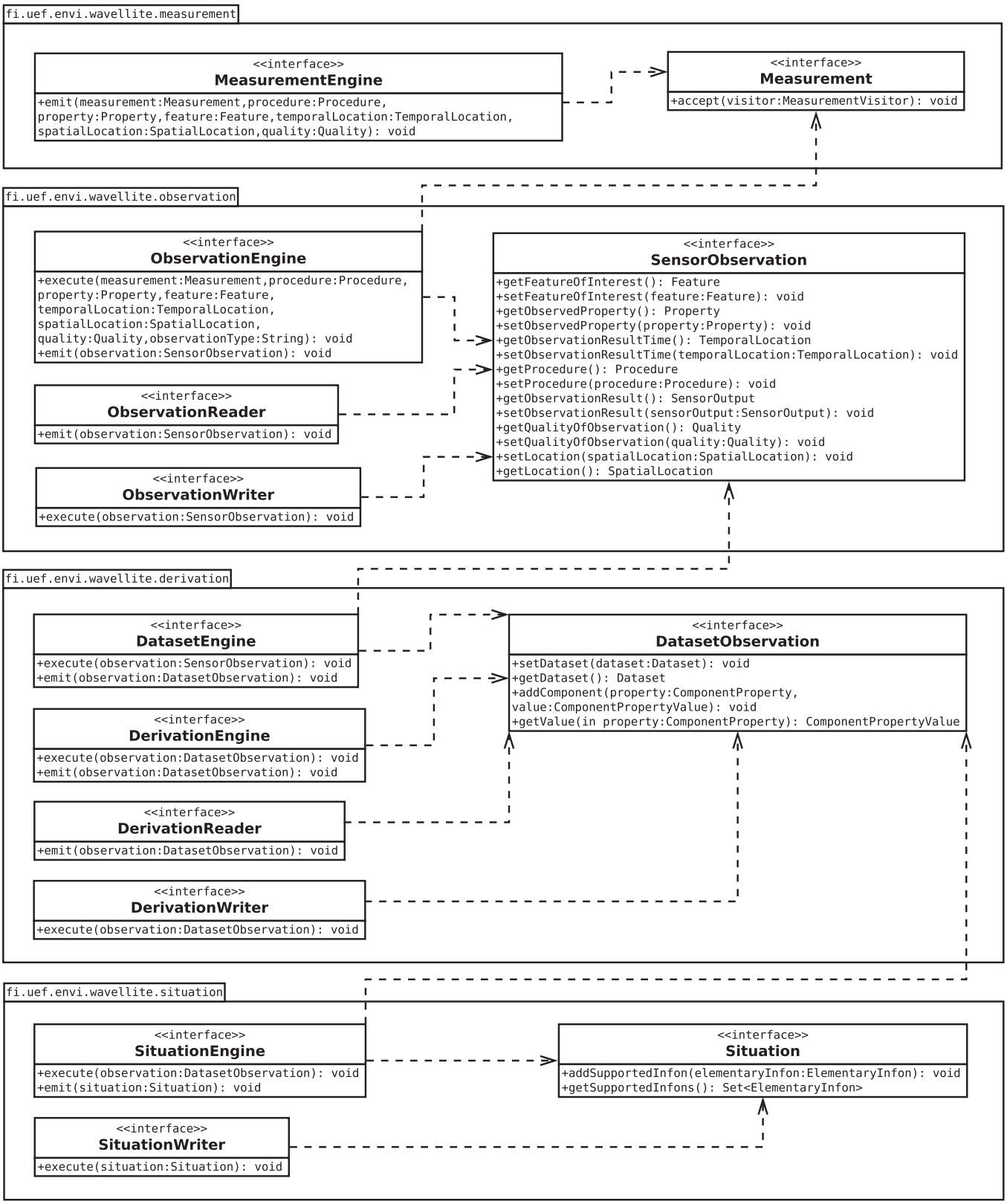


Figure D.7. Interfaces of Wavellite components with emit and execute operations.

Appendix E. Application Implementation

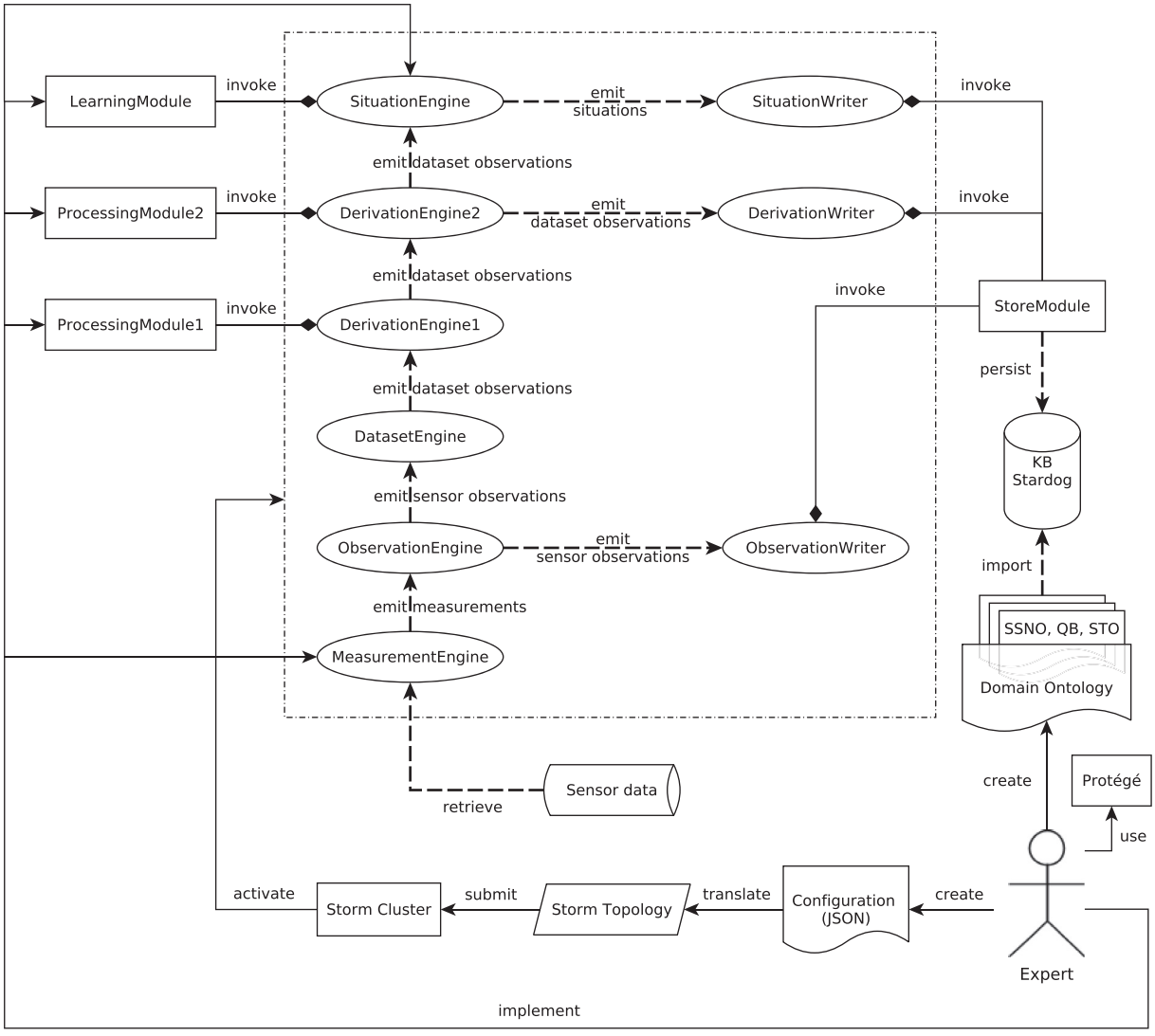


Figure E.8. Schematic overview of Wavellite application implementation and execution. Experts extend upper ontologies (SSNO, QB, STO) with domain knowledge. These ontologies are imported to the Knowledge Base (KB). Experts create the (JSON) Wavellite application configuration for components, their settings, incoming and outgoing streams, and class implementations. Finally, experts implement measurement engines and learning modules, as well as processing modules (if required by the application). At runtime, the Wavellite application configuration is translated to a Storm topology which is submitted to a Storm cluster. The Storm cluster then initializes the topology and activates spouts (measurement engines and readers) and bolts (other components). In activating measurement engines, sensor data is retrieved and processed to measurements, which are emitted by measurement engines. Data is, henceforth, processed to sensor observations and dataset observations, which may be persisted by accordingly configuring writers. Situations are acquired from dataset observations by situation engines. Sensor observations, dataset observations, and situations are represented according to the SSNO, QB, and STO upper ontologies, respectively, and are persisted to the KB by store modules.

References

Avancha, S., Patel, C., Joshi, A., 2004. Ontology-driven adaptive sensor networks. In: Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004. The First Annual International Conference on, pp. 194–202. http://dx.doi.org/10.1109/MOBIQ.2004.1331726.

Barnaghi, P., Ganz, F., Henson, C., Sheth, A., 2012. Computing Perception from Sensor Data. Technical Report. knoesis.org.

Barnaghi, P., Meissner, S., Presser, M., Moessner, K., 2009. Sense and sens'ability: semantic data modelling for sensor networks. In: Proceedings of the ICT Mobile Summit 2009.

Barwise, J., Perry, J., 1983. Situations and attitudes. Bradford books. MIT Press.

Berners-Lee, T., 2006. Linked Data—Design Issues. URL: http://www.w3.org/DesignIssues/LinkedData.html.

Berners-Lee, T., Hendler, J., Lassila, O., 2001. The semantic web. Sci. Am. 284, 28–37.

Bizer, C., Auer, S., Kobilarov, G., Lehmann, J., Cyganiak, R., 2007. DBpedia—Querying Wikipedia like a database. In: Developers Track Presentation at the 16th International Conference on World Wide Web, WWW, pp. 8–12.

Bonnet, P., Gehrke, J., Seshadri, P., 2001. Towards sensor database systems. In: Mobile Data Management. LNCS, vol. 1987. Springer Berlin/Heidelberg, pp. 3–14.

Brickley, D., Guha, R., McBride, B., 2004. RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation. W3C.

Carney, D., Çetintemel, U., Cherniack, M., Convey, C., Lee, S., Seidman, G., Stonebraker, M., Tatbul, N., Zdonik, S., 2002. Monitoring streams: a new class of data management applications. In: Proceedings of the 28th international conference on Very Large Data Bases, VLDB Endowment, pp. 215–226.

Clemente, S., Loia, V., Veniero, M., 2013. Applying cognitive situation awareness to collision avoidance for harbour last-mile area safety. J. Ambient Intell. Humaniz. Comput., 1–5 http://dx.doi.org/10.1007/s12652-013-0187-6. URL:

Compton, M., 2011. What now and where next for the W3C semantic sensor networks incubator group sensor ontology. In: Proceedings of the 4th International Workshop on Semantic Sensor Networks 2011 (SSN11), CEUR-WS, pp. 1–8.

Compton, M., Henson, C., Neuhaus, H., Lefort, L., Sheth, A., 2009. A survey of the semantic specification of sensors. In: 2nd International Workshop on Semantic Sensor Networks, at 8th International Semantic Web Conference.

Conroy, K., May, G., Roantree, M., Warrington, G., Cullen, S.J., McGoldrick, A., 2011. Knowledge acquisition from sensor data in an equine environment. In: Proceedings of the 13th international conference on data warehousing and knowledge discovery. Springer-Verlag, Berlin, Heidelberg, pp. 432–444.

Coradeschi, S., Saffiotti, A., 2003. An introduction to the anchoring problem. Robotics Aut. Syst. 43, 85–96. URL: http://www.sciencedirect.com/science/article/pii/S0921889003000216 http://dx.doi.org/10.1016/S0921-8890(03)00021-6.

Cyganiak, R., Reynolds, D., Tennison, J., 2013. The RDF Data Cube Vocabulary. W3C Candidate Recommendation. W3C.

Dal Maso, M., Kulmala, M., Riipinen, I., Wagner, R., Hussein, T., Aalto, P., Lehtinen, K., 2005. Formation and growth of fresh atmospheric aerosols: eight years of aerosol size distribution data from SMEAR II, Hyytiala, Finland. Boreal Environ. Res. 10, 323–336.

Daoutis, M., 2013. Knowledge based perceptual anchoring. KI-Künstl. Intell., 1–4.

De Maio, C., Fenza, G., Furno, D., Loia, V., 2012. Swarm-based semantic fuzzy reasoning for situation awareness computing. In: Fuzzy Systems (FUZZ-IEEE), 2012 IEEE International Conference on, pp. 1–7 http://dx.doi.org/10.1109/FUZZ-IEEE.2012.6251159.

Devlin, K., 1995. Logic and Information. Cambridge University Press.

Doulaverakis, C., Konstantinou, N., Knape, T., Kompatsiaris, I., Soldatos, J., 2011. An approach to intelligent information fusion in sensor saturated urban environments. In: Intelligence and Security Informatics Conference (EISIC), 2011 European, pp. 108–115.

Eid, M., Liscano, R., El Saddik, A., 2006. A novel ontology for sensor networks data. In: Computational Intelligence for Measurement Systems and Applications, Proceedings of 2006 IEEE International Conference on, pp. 75–79. http://dx.doi.org/10.1109/CIMSA.2006.250753.

Fenza, G., Furno, D., Loia, V., Veniero, M., 2010. Agent-based cognitive approach to airport security situation awareness. In: Proceedings of the 2010 International Conference on Complex, Intelligent and Software Intensive Systems, IEEE Computer Society, pp. 1057–1062.

Finkelstein, L., 1982. Theory and Philosophy of Measurement. John Wiley & Sons.

Furno, D., Loia, V., Veniero, M., Anisetti, M., Bellandi, V., Ceravolo, P., Damiani, E., 2011. Towards an agent-based architecture for managing uncertainty in situation awareness. In: Intelligent Agent (IA), 2011 IEEE Symposium on, pp. 1–6. http://dx.doi.org/10.1109/IA.2011.5953605.

Gaglio, S., Gatani, L., Lo Re, G., Ortolani, M., 2007. Understanding the environment through wireless sensor networks. In: Proceedings of the 10th Congress of the Italian Association for Artificial Intelligence on AI*IA 2007: Artificial Intelligence and Human-Oriented Computing. Springer-Verlag, Berlin, Heidelberg, pp. 72–83.

Gorrepati, R., Ali, S., Kim, D.H., 2013. Hierarchical semantic information modeling and ontology for bird ecology. Clust. Comput., 1–8 http://dx.doi.org/10.1007/s10586-013-0269-4. URL:

Gruber, T., 1993. A translation approach to portable ontology specifications. Knowl. Acquis. 5, 199–220.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H., 2009. The WEKA data mining software: an update. In: SIGKDD Explorations.

Hamed, A., Joutsensaari, J., Mikkonen, S., Sogacheva, L., Dal Maso, M., Kulmala, M., Cavalli, F., Fuzzi, S., Facchini, M., Decesari, S., et al., 2007. Nucleation and growth of new particles in Po Valley, Italy. Atmos. Chem. Phys. 7, 355–376.

Hand, D., Mannila, H., Smyth, P., 2001. Principles of Data Mining. A Bradford Book, Mit Press.

Hart, J.K., Martinez, K., 2006. Environmental sensor networks: a revolution in the earth system science? Earth-Sci. Rev. 78, 177–191. http://dx.doi.org/10.1016/j.earscirev.2006.05.001.

Hayes, P., McBride, B., 2004. RDF Semantics. W3C Recommendation. W3C.

Haykin, S., 1999. Neural Networks: A Comprehensive Foundation. In: Prentice Hall International Editions Series. Prentice Hall.

Henson, C.A., Pschorr, J.K., Sheth, A.P., Thirunarayan, K., 2009. SemSOS: semantic sensor observation service. In: Proceedings of the 2009 International Symposium on Collaborative Technologies and Systems (CTS 2009), Baltimore, MD.

Hill, D.J., Liu, Y., Marini, L., Kooper, R., Rodriguez, A., Futrelle, J., Minsker, B.S., Myers, J., McLaren, T., 2011. A virtual sensor system for user-generated, real-time environmental data products. Environ. Model. Softw. 26, 1710–1724 http://dx.doi.org/10.1016/j.envsoft.2011.09.001. URL: http://www.sciencedirect.com/science/article/pii/S1364815211001988.

Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S., 2012. OWL 2 Web Ontology Language Primer, second ed. W3C Recommendation. W3C.

Horsburgh, J.S., Tarboton, D.G., Piasecki, M., Maidment, D.R., Zaslavsky, I., Valentine, D., Whitenack, T., 2009. An integrated system for publishing environmental observations data. Environ. Model. Softw. 24, 879–888 http://dx.doi.org/10.1016/j.envsoft.2009.01.002. URL: http://www.sciencedirect.com/science/article/pii/S1364815209000036.

Hyvönen, S., Junninen, H., Laakso, L., Dal Maso, M., Grönholm, T., Bonn, B., Keronen, P., Aalto, P., Hiltunen, V., Pohja, T., Launiainen, S., Hari, P., Mannila, H., Kulmala, M., 2005. A look at aerosol formation using data mining techniques. Atmos. Chem. Phys. 5, 3345–3356 http://dx.doi.org/10.5194/acp-5-3345-2005. URL: http://www.atmos-chem-phys.net/5/3345/2005/.

Janowicz, K., Compton, M., 2010. The stimulus-sensor-observation ontology design pattern and its integration into the semantic sensor network ontology. In: The 3rd International workshop on Semantic Sensor Networks, pp. 7–11.

Junninen, H., Lauri, A., Keronen, P., Aalto, P., 2009. Smart-SMEAR: on-line data exploration and visualization tool for SMEAR stations. Boreal Environ. Res. 14, 447–457.

Kessler, C., Janowicz, K., 2010. Linking sensor data – why, to what, and how? In: Taylor, K., Ayyagari, A., Roure, D.D. (Eds.), Proceedings of the 3rd International Workshop on Semantic Sensor Networks (SSN10), CEUR-WS, Shanghai, China.

Kokar, M.M., Matheus, C.J., Baclawski, K., 2009. Ontology-based situation awareness. Inf. Fusion 10, 83–98.

Kulkarni, P., Baron, P.A., Willeke, K., 2011. Aerosol Measurement: Principles, Techniques, and Applications. Wiley.

Kulmala, M., Vehkamäki, H., Petäjä, T., Dal Maso, M., Lauri, A., Kerminen, V., Birmili, W., McMurry, P., 2004. Formation and growth rates of ultrafine atmospheric particles: a review of observations. J. Aerosol Sci. 35, 143–176.

Lefort, L., Bobruk, J., Haller, A., Taylor, K., Woolf, A., 2012. A linked sensor data cube for a 100 year homogenised daily temperature dataset. In: Henson, C., Taylor, K., Corcho, O. (Eds.), Proceedings of the 5th International Workshop on Semantic Sensor Networks, CEUR-WS, Boston, Massachusetts, pp. 1–16.

Leskinen, A., Portin, H., Komppula, M., Miettinen, P., Arola, A., Lihavainen, H., Hatakka, J., Laaksonen, A., Lehtinen, K., 2009. Overview of the research activities and results at Puijo semi-urban measurement station. Boreal Env. Res. 14, 576–590.

Liu, J., Zhao, F., 2005. Towards semantic services for sensor-rich information systems. In: Broadband Networks, 2005. BroadNets 2005. 2nd International Conference on, pp. 967–974 http://dx.doi.org/10.1109/ICBN.2005.1589709.

Luckham, D.C., 2002. The Power of Events, vol. 204. Addison-Wesley Reading.

Madden, S., Franklin, M., 2002. Fjording the stream: an architecture for queries over streaming sensor data. In: Data Engineering, 2002. Proceedings. 18th International Conference on, pp. 555–566.

Manola, F., Miller, E., McBride, B., 2004. RDF Primer. W3C Recommendation. W3C.

Mitchell, T., 1997. Machine Learning. McGraw-Hill, Boston, MA.

Moraru, A., Mladenić, D., 2012. A framework for semantic enrichment of sensor data. In: Information Technology Interfaces (ITI), Proceedings of the ITI 2012 34th International Conference on, pp. 155–160. http://dx.doi.org/10.2498/iti.2012.0481.

Obrst, L., 2003. Ontologies for semantically interoperable systems. In: Proceedings of the twelfth international conference on Information and knowledge management, ACM, New York, NY, USA, pp. 366–369. http://dx.doi.org/10.1145/956863.956932. URL.

Pope III, C.A., Burnett, R.T., Thun, M.J., Calle, E.E., Krewski, D., Ito, K., Thurston, G.D., 2002. Lung cancer, cardiopulmonary mortality, and long-term exposure to fine particulate air pollution. JAMA: J. Am. Med. Assoc. 287, 1132–1141.

Prud'hommeaux, E., Seaborne, A., 2008. SPARQL Query Language for RDF. Technical Report W3C Recommendation. W3C.

Rabiner, L., Gold, B., 1975. Theory and Application of Digital Signal Processing. In: Prentice-Hall Signal Processing Series. Prentice-Hall.

Randell, D., Cui, Z., Cohn, A., 1992. A spatial logic based on regions and connections. In: Nebel, B., Rich, C., Swartout, W. (Eds.), Principles of Knowledge Representation and Reasoning. Morgan Kaufmann, San Mateo, CA, USA, pp. 165–176.

Raskin, R.G., Pan, M.J., 2005. Knowledge representation in the semantic web for Earth and environmental terminology (SWEET). Comput. Geosci. 31, 1119–1125.

Rijgersberg, H., Wigham, M., Top, J., 2011. How semantics can improve engineering processes: a case of units of measure and quantities. Adv. Eng. Inform. 25, 276–287.

Sheth, A., Henson, C., Sahoo, S., 2008. Semantic sensor web. Internet computing. IEEE 12, 78–83. http://dx.doi.org/10.1109/MIC.2008.87.

Solomon, S., Qin, D., Manning, M., Chen, Z., Marquis, M., Averyt, K., Tignor, M., M, H.L., 2007. Climate Change 2007: the Physical Science Basis. Contribution of Working Group I to the Fourth Assessment Report of the Intergovernmental Panel on Climate Change. Cambridge University Press, Cambridge, United Kingdom and New York, NY, USA.

Stocker, M., Baranizadeh, E., Hamed, A., Rönkkö, M., Virtanen, A., Laaksonen, A., Portin, H., Komppula, M., Kolehmainen, M., 2013. Acquisition and representation of knowledge for atmospheric new particle formation. In: Hřebíček, J., Schimak, G., Kubásek, M., Rizzoli, A.E. (Eds.), Environmental Software Systems, Fostering Information Sharing, IFIP Advances in Information and Communication Technology, vol. 413. Springer, Boston, pp. 98–108.

Stocker, M., Rönkkö, M., Kolehmainen, M., 2012a. Making sense of sensor data using ontology: a discussion for residential building monitoring. In: Iliadis, L., Maglogiannis, I., Papadopoulos, H., Karatzas, K., Sioutas, S. (Eds.), Artificial Intelligence Applications and Innovations, IFIP Advances in Information and Communication Technology, vol. 382. Springer Boston, pp. 341–350.

Stocker, M., Rönkkö, M., Kolehmainen, M., 2012b. Making sense of sensor data using ontology: a discussion for road vehicle classification. In: Seppelt, R., Voinov, A., Lange, S., Bankamp, D. (Eds.), 2012 International Congress on Environmental Modelling and Software, iEMSs, Leipzig, Germany, pp. 2387–2394.

Stocker, M., Rönkkö, M., Kolehmainen, M., 2013b. The wavellite modelling and software framework for situation awareness in environmental monitoring. Environ. Monit. Assess. (submitted for publication).

Stocker, M., Rönkkö, M., Villa, F., Kolehmainen, M., 2011. The relevance of measurement data in environmental ontology learning. In: Environmental Software Systems. Frameworks of Environment. IFIP Advances in Information and Communication Technology, vol. 359. Springer Boston, pp. 445–453.

Stocker, M., Sirin, E., 2009. PelletSpatial: a hybrid RCC-8 and RDF/OWL reasoning and query engine. In: Hoekstra, R., Patel-Schneider, P.F. (Eds.), Proceedings of the 6th International Workshop on OWL: Experiences and Directions (OWLED 2009), CEUR Workshop Proceedings, Chantilly, Virginia, USA.

Uschold, M., Gruninger, M., 2004. Ontologies and semantics for seamless connectivity. SIGMOD Rec. 33, 58–64 http://dx.doi.org/10.1145/1041410.1041420. URL:

Vana, M., Ehn, M., Petäjä, T., Vuollekoski, H., Aalto, P., de Leeuw, G., Ceburnis, D., O'Dowd, C.D., Kulmala, M., 2008. Characteristic features of air ions at Mace Head on the west coast of Ireland. Atmos. Res. 90, 278–286.

Vassev, E., Hinchey, M., 2012. Knowledge Representation for Cognitive Robotic Systems. In: 2012 15th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops (ISORCW). IEEE, pp. 156–163.

Wanner, L., Vrochidis, S., Tonelli, S., Moßgraber, J., Bosch, H., Karppinen, A., Myllynen, M., Rospocher, M., Bouayad-Agha, N., Bügel, U., Casamayor, G., Ertl, T., Kompatsiaris, I., Koskentalo, T., Mille, S., Moumtzidou, A., Pianta, E., Saggion, H., Serafini, L., Tarvainen, V., 2011. Building an environmental information system for personalized content delivery. In: Hrebícek, J., Schimak, G., Denzer, R. (Eds.), Environmental Software Systems. Frameworks of Environment, IFIP Advances in Information and Communication Technology, vol. 359. Springer Boston, pp. 169–176.

Wei, W., Barnaghi, P., 2009. Semantic annotation and reasoning for sensor data. Smart Sens. Context, 66–76.

Whitehouse, K., Zhao, F., Liu, J., 2006. Semantic streams: a framework for composable semantic interpretation of sensor data. In: Römer, K., Karl, H., Mattern, F. (Eds.), Wireless Sensor Networks, LNCS, vol. 3868. Springer Berlin/Heidelberg, pp. 5–20.

Wiedensohler, A., Birmili, W., Nowak, A., Sonntag, A., Weinhold, K., Merkel, M., Wehner, B., Tuch, T., Pfeifer, S., Fiebig, M., Fjäraa, A.M., Asmi, E., Sellegri, K., Depuy, R., Venzac, H., Villani, P., Laj, P., Aalto, P., Ogren, J.A., Swietlicki, E., Williams, P., Roldin, P., Quincey, P., Hüglin, C., Fierz-Schmidhauser, R., Gysel, M., Weingartner, E., Riccobono, F., Santos, S., Grüning, C., Faloon, K., Beddows, D., Harrison, R., Monahan, C., Jennings, S.G., O'Dowd, C.D., Marinoni, A., Horn, H.G., Keck, L., Jiang, J., Scheckman, J., McMurry, P.H., Deng, Z., Zhao, C.S., Moerman, M., Henzing, B., de Leeuw, G., Löschau, G., Bastian, S., 2012. Mobility particle size spectrometers: harmonization of technical standards and data structure to facilitate high quality long-term observations of atmospheric particle number size distributions. Atmos. Meas. Tech. 5, 657–685 http://dx.doi.org/10.5194/amt-5-657-2012. URL: http://www.atmos-meas-tech.net/5/657/2012/.