An Event Abstraction Layer for the Integration of Geosensor Data

Alejandro Llaves[1⋆] and Werner Kuhn[2]

[1] Ontology Engineering Group, Universidad Politécnica de Madrid, Madrid, Spain
[2] Center for Spatial Studies, University of California, Santa Barbara, USA

Abstract. Time series of observations reflect the status of environmental properties. Variations in these properties can be considered events when they potentially affect the stability of the monitored environment. Organisations dedicated to analyse environmental change use institutionalised descriptions of events to define the observable conditions under which events happen. This also applies to the methods used to classify and model changes in environmental monitoring. The heterogeneity of representations often causes interoperability problems when such communities exchange geospatial information. To enhance interoperability among diverse communities, it is required to develop models that do not restrict the representation of events, but allow integrating different perspectives on changes in the environment. The goal of the Event Abstraction Layer is to facilitate the analysis and integration of geosensor data by inferring events from time series of observations. For the analysis of geosensor data, we use event processing to detect event patterns in time series of observations. Spatio-temporal properties of the event are inferred from the geosensor location and the observation timestamps. For the data integration, we represent event-related information extracted from multiples sources under a common event model. Additionally, domain knowledge is modelled in a multilevel ontology structure.

Keywords: Spatio-temporal data modelling, Semantics, Interoperability, Event-oriented approaches

1 Introduction

In recent decades, science has moved from a data-poor to a data-rich environment because of the decrease in the cost of retrieving, storing, and processing digital data [39]. The sources of these data are mostly sensors that little by little are becoming common in our environment. Nowadays, sensors are integrated in some of the devices we use daily, such as smartphones, computers, or fridges; and they are also present in highways, traffic lights, buildings, and even in some natural landscapes [9]. Environmental monitoring is a critical process in areas

⋆ allaves@fi.upm.es

usually affected by natural disasters. It is aimed to ensure public safety, to set up continuous information services and to provide input for spatial decision support systems [45]. Here, the main challenge is the distributed processing and integration of vast amounts of heterogeneous sensor data in real-time.

One of the most relevant research questions today is *'How is the Earth's environment changing and what are the consequences for human civilisation?'* [13]. If we want to understand change and prevent potential consequences, we need to interpret observation data to extract relevant information. The reason for creating spatio-temporal models to represent change comes from the fact that space and time are two of the most important ordering relations used in human cognition and language [29]. Traditional snapshot modelling does not provide a full picture on how dynamic geospatial phenomena affect our environment because events, which are needed to capture the mechanisms of change, are not part of the model [7,52,24]. The term *event* refers in this paper to anything that happens or is observed as happening at an instant - or over an interval - of time, and which is relevant for the observer. If we want to obtain information from the continuous data flows produced by sensors, we need to provide meaning to relevant fragments of observations observed over time (patterns) and analyse where and when they appear.

The way we access geospatial information changed in the last years from local processing and storage to the migration of data and operations to the Web. Traditional methods of documentation in GIS do not include machine-readable encoding of metadata, thus causing the loss of context when these metadata is shared among systems [33]. Three different levels of heterogeneities are proposed by Bishr et al. [3]: syntactic, schematic, and semantic. Diversity of data syntax or structure may lead to syntactic or schematic interoperability problems, respectively, but semantic heterogeneity would be caused by different conceptualisations of the same real world entity [31, ch. 3]. Handling huge amounts of environmental geosensor data in form of time series can be addressed with event processing techniques, like Complex Event Processing (CEP). The principal problem we face in this context is the lack of semantic descriptions to define terms for event types and their properties to be understood by multiple applications [23].

Events are regarded as core geographic concepts that can answer questions about change [34]. The main goal of this research is to provide a method that facilitates the analysis and integration of sensor data by inferring events from time series of observations. The integration of event-related information from various communities is helpful to understand the behaviour of complex environmental phenomena, like floods. The classification criteria for environmental events strongly depends on the domain. Event descriptions are extracted from domain knowledge resources, like scientific papers or technical reports. The formalisation of descriptions of events as event patterns allows for creating an abstraction layer on top of observation data. An event pattern is defined as *'a template containing event templates, relational operators and variables. An event pattern can match sets of related events by replacing variables with values'* [37]. We assume that the

domain knowledge to decide what events are relevant for a specific use case is provided by experts. However, the inference of such events from continuous and heterogeneous data streams is a complex task for a person. For this reason, we investigate the different methods to perform this task automatically. The results of this research will answer the following question:

RQ1. How to infer events on the fly from time series of observations provided by in situ sensors?

Event processing allows detecting event patterns in time series of observations. However, semantic interoperability problems arise when information communities exchange event-related information. The reason is that information communities often use ambiguous terms and vocabularies to categorise events. In order to address these problems, event models have to accommodate the perspectives of different communities. The second research question analyses these issues:

RQ2. How to represent event-related information so that it can be shared among information communities?

We suggest organising knowledge in a multilevel ontology structure. The different levels contain conceptualisations specific to an application, a domain, or knowledge common to all domains. The event model, which captures the main properties of events derived from observations, is also part of this structure.

The next section introduces the necessary background discussed in the thesis and substantial related work. Section 3 includes the description of the Event Abstraction Layer. The application of our method to a flood monitoring case study is described in section 4. We describe the results of a case study experiment and compare our method to similar related work in the evaluation. In section 6, we summarise the main contributions of this paper and discuss about next steps.

2 Background Concepts and Related Work

The first part of this section describes some basic concepts to understand the context of this research work. In the second part, solutions that try to solve similar research problems are introduced. We compare some of these solutions to our method in the Evaluation section.

2.1 Background concepts

A sensor (or geosensor) is a device that measures detectable changes in our environment and can be geographically referenced [12]. Our research focuses on analysing observation time series obtained from in situ sensors. In situ sensing,

in contrast to remote sensing, deals with sensors which are in contact with the medium they are sensing.

Some applications require analysing several observation data sources to aggregate and extract higher level information. The components that carry out this kind of tasks are often called virtual sensors. Kabadayi et al. [30] defined a virtual sensor as *'a software sensor as opposed to a physical or hardware sensor. Virtual sensors provide indirect measurements of abstract conditions (that, by themselves, are not physically measurable) by combining sensed data from a group of heterogeneous physical sensors'*. The prototype presented in section 3.3 to abstract events from observations is, in general terms, a virtual sensor.

The concept of Sensor Web [15] refers to a network of spatially-distributed and interconnected sensing devices able to monitor uncertain environments. The OGC's Sensor Web Enablement (SWE) was created to provide a set of standards for managing online sensor networks and the data they produce [4]. The SWE working group defines data models, encodings, and Web service specifications to overcome issues raised by syntactic heterogeneities [4]. Yet, one of the biggest challenges in this field is dealing with semantic heterogeneities [6]. The Semantic Sensor Web provides a framework for the interoperable exchange and processing of observation data from sensor networks of different types. One of the goals of the Semantic Sensor Web is to solve the interoperability problems detected in the Sensor Web by enriching sensor data and sensor descriptions with spatial, temporal, and thematic metadata [46].

Event processing consists in working with representations of events. The Event Processing Glossary defines event processing as *'computing that performs operations on events, including reading, creating, transforming, or discarding events'* [37]. CEP is a specialisation of event processing and it deals with the processing of complex events [36]. The same glossary also defines a complex event as an event that represents a set of other events. CEP can be used to detect certain relations between events. Causal relations are common in CEP (e.g. 'caused by'), but temporal (e.g. 'before' or 'after') and spatial relations (e.g. 'within area') can also be used [36]. The rules for CEP are called event patterns and they are formalised using an Event Pattern Language (EPL).

2.2 Previous work on Semantic Event Processing

The first ideas on a semantic approach to event processing were given by Etzion and Niblett [20], who claimed that semantic structures common to different event processing systems exist, and that they can be used to create a semantic model of event processing. Later, the concept of *semantic event processing* was defined as the combination of event processing and semantic technologies which allows event processing engines to *"understand what is happening in terms of events and states"*, and to react appropriately [48]. Any system implementing semantic event processing is supposed to include a static knowledge module about the predefined event types, and a real-time analysis module which processes data streams searching for event patterns [48]. The research goal of Teymourian's

Ph.D. thesis is to develop a representation methodology for CEP which integrates domain and application ontologies for events, processes, states, actions, and other concepts that are related to change over time [49]. The hypothesis of that dissertation is that the use of ontological knowledge combined with event processing and stream processing techniques enhances the procedure of detection and processing of events.

One example of a semantic event processing system is ETALIS [1], which enables monitoring and specification of changes in near real-time and it is able to perform reasoning over streams of events with respect to background knowledge. In ETALIS, an event represents something that occurs, happens or changes the current state of affairs [1]. The example used to demonstrate ETALIS capabilities describes a system to identify traffic bottlenecks. Another example of a system is SCEPter [53], which performs Semantic Complex Event Processing (SCEP) of streaming and archived events using a hybrid solution which combines a CEP engine and a database. For demonstration purposes, that research addresses challenges arising in the domain of Smart Power Grids, where a vast amount of data is generated and exchanged among heterogeneous systems.

In the GIScience domain, the Semantic Sensor Observation Service (SemSOS) builds upon OGC's SOS[3] [40] to provide a more meaningful representation of sensor data [28]. The method proposed consists in various steps that include: the development of a set of ontologies to model the domain of sensors and sensor measurements, enriching sensor observations with semantic annotations, and using the ontology models to perform rule-based reasoning over the annotated observations in order to obtain new knowledge. For the implementation of SemSOS, the SOS is extended with a semantic knowledge base. SemSOS allows to execute temporal and thematic queries on historical sensor data by using SPARQL[4]. Results can be used to construct observation collection responses encoded in Observations and Measurements (O&M) [10].[5] Based on previous work on SemSOS, Patni built in his M.Sc. thesis [41] a system to convert real-time heterogeneous sensor data into an integrated stream of abstractions (also called features in the thesis), and to reason over them by using background knowledge. Patni's system is called Real-Time Feature Streams Infrastructure (RTFS) and converts raw sensor data to streams of Resource Description Framework (RDF)[6] triples.

Devaraju [17] used DOLCE [38] as a foundational ontology for her Sensing Geographic Occurrence Ontology (SEGO). This model represents the relations between observations and geographic occurrences reflected in them. In SEGO, geographic processes act as stimuli that trigger sensors, whereas geo-

[3] SOS is the OGC standard specification for sensor data retrieval. More information is available at http://www.opengeospatial.org/standards/sos.

[4] SPARQL is a W3C recommendation language to query RDF data. More details at http://www.w3.org/TR/rdf-sparql-query/.

[5] O&M is an OGC standard specification to enable sensor data encoding. The O&M website is available at http://www.opengeospatial.org/standards/om.

[6] RDF is a data model that allows defining statements in the form of subject-object-predicate triples. More information can be found at http://www.w3.org/RDF/.

graphic events are occurrences that are inferred from observations. Institutionalised descriptions [44] define geographic events based on observed properties. The use of institutionalised descriptions of events is an important part of my method, as described in section 3.

The description of other relevant event processing systems (like Cayuga [16] and Drools[7]), event query languages (like XChangeEQ [19]), and rule languages (like Prova[8]) has been excluded because of their non-explicit use of semantic technologies.

3 An Event Abstraction Layer

Some environmental monitoring applications require to sense and respond to certain changes. In these settings, it is required to shift the focus from analysing raw streams of data to the analysis of the higher level pieces of information they hide, namely events. Methods that infer and integrate event-related information from available data sources can reduce the response time in emergencies. The Event Abstraction Layer bridges the gap between sensor data services and event-driven applications, see figure 1. Therefore, event-driven applications do not need to deal with raw sensor data. On the one hand, the Event Abstraction Layer consumes time series of observations. It also handles mappings between event patterns and event types, so that inferred events are automatically classified according to the provided domain knowledge. On the other hand, the Event Abstraction Layer generates a stream of events that are represented under the same model. Data integration consists in providing users with a uniform view on data residing at different sources [35]. The Event Abstraction Layer enables data integration by representing event-related information extracted from multiple sources under a common event model.

3.1 Semantic annotation of event patterns

The *semantic annotation of event patterns* consists in labeling event patterns with event types defined in ontologies. Event types represent categories of a classification related to one or more observed properties, e.g. heavy rainfall. Event patterns describe events quantitatively based on observed properties, e.g. rainfall intensity above four millimetres per hour. The mapping between event patterns and event types is extracted from domain knowledge.

Defining the conditions for the classification of environmental events depends on various factors, such as the location of the region of interest or the availability of historical records of events. Two information communities may use the same event type to refer to different event patterns. Two communities may also use the same event pattern to describe different event types. The presented method allows for the two possibilities, but it focuses on the case of different event patterns

[7] More information about Drools can be found at http://www.jboss.org/drools.
[8] More information about Prova is available at https://prova.ws.

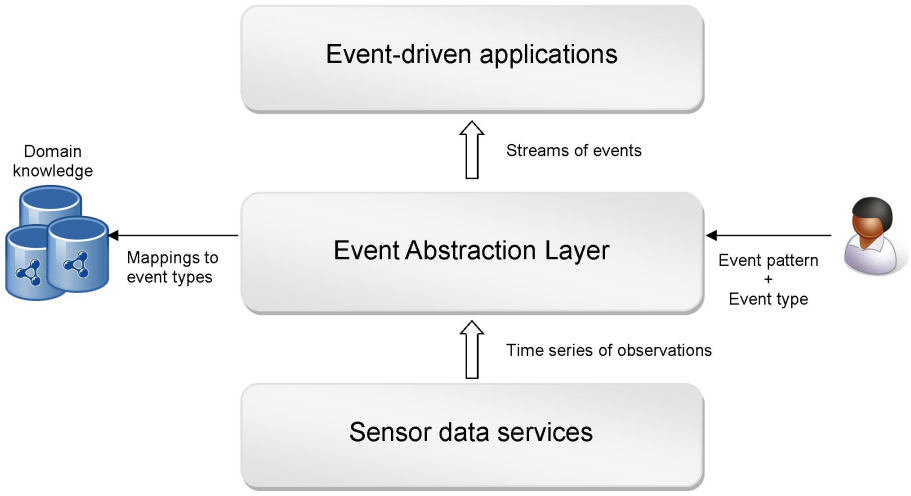

Fig. 1. The Event Abstraction Layer analyses time series of observations and generates streams of events.

annotated with the same event type, which seems to be more common in the environmental monitoring domain. The key lies on separating event types and event patterns in the formalisation of knowledge. Event patterns shall not be included in ontologies. This fosters the reusability of ontologies across information communities when they use similar event classifications.

Additionally, we organise application- and domain-specific knowledge in different levels as suggested by Klien and Probst [32]. At the bottom, application ontologies model event-related knowledge specific to information communities. In the middle level, domain ontologies represent a shared conceptualisation within a domain. On top, the foundational ontology defines concepts common to all domains, like *event*. Event types that are specific to an information community shall be represented in an application ontology, e.g. HEAVY RAINFALL event defined by a specific weather agency. Event types that are common to a domain shall be represented in a domain ontology, e.g. RAINFALL event in the meteorological domain. All concepts in application ontologies inherit from concepts defined in domain ontologies because application ontologies specialise domain knowledge. All concepts in domain ontologies inherit from the top-level concepts defined in the foundational ontology. The alignment to a foundational ontology provides a common set of top-level concepts as a reference across domains and applications. The mappings between application ontologies or domain ontologies are also possible, but out of the scope of this thesis.

3.2 The Event Abstraction ontology

Attempts at designing a foundational ontology for all types of events have failed [48]. The Event Abstraction ontology[9] does not aim at representing all types of events. It provides a set of concepts and relations to model events inferred from time series of sensor data. The Event Abstraction Layer uses these concepts and relations to represent the generated events.

The Event Abstraction ontology extends the Semantic Sensor Network (SSN) ontology [8]. We reused the SSN ontology because it describes the domain of sensors and sensor networks as a result of a revising seventeen existing sensor-centric and observation-centric ontologies.[10] The SSN ontology is supported by the W3C community and was created with the aim of being reused and extended. The SSN and the Event Abstraction ontologies are aligned to the DOLCE Ultra-lite (DUL) foundational ontology [25] to restrict the interpretation of concepts and relations. Observations are represented as subclasses of DUL:SITUATION.[11] A DUL:SITUATION is a view on a set of entities which is consistent with a description.[12] An SSN:OBSERVATION is a situation in which a sensing method has been used to estimate or calculate the value of a property of a feature of interest.[13]

The main concept of the Event Abstraction ontology is the *event abstraction*. To put it simply, an *event abstraction* is the representation of an event. An *event abstraction* represents the perception of certain conditions in time series of sensor data. These conditions are described by an *event abstraction rule* (or event pattern). *Event abstraction rules* are based on observed *properties*, like rainfall intensity. *Event abstractions* are inferred by *event processing agents*, which are virtual sensors. Etzion and Niblett [21] define an *event processing agent* as a software module that processes events. The *event detection procedure* describes the sensing method used by such agents to infer events. For instance, CEP is the procedure used in our implementation, as explained in next chapter. An *event abstraction* is attached to an *event type*. The semantic annotation of event patterns (see previous section) maps *event abstraction rules* to *event types*. This mapping is a formalisation of domain knowledge and allows inferring events automatically. *Event abstractions* are related to a *spatio-temporal region*. The spatial location of the *event abstraction* is extracted from the sensors providing the observations. The temporal location of the *event abstraction* is inferred from the observation timestamps. Moreover, the *event abstraction* is related to the observation *collection* that was used for the event inference.

---

[9] http://wsmls.googlecode.com/svn/trunk/global/Event-abstraction/0.2/.

[10] Reviews available at http://www.w3.org/2005/Incubator/ssn/XGR-ssn-20110628, section 4.

[11] To differentiate concepts of the different ontologies, we will use the ontology acronym as namespace: `dul` and `ssn`.

[12] A complete definition of `dul:Situation` is available at http://www.w3.org/2005/Incubator/ssn/wiki/DUL_ssn.

[13] More details at http://www.w3.org/2005/Incubator/ssn/wiki/Incubator_Report.

In the rest of the document we will use the abbreviation `eabs` as namespace to refer to the concepts and relations included in the Event-Abstraction ontology. The `rdfs` relations are reused from the RDF Schema vocabulary[14]. The main concepts and relations that compose the extension are depicted in figure 2.

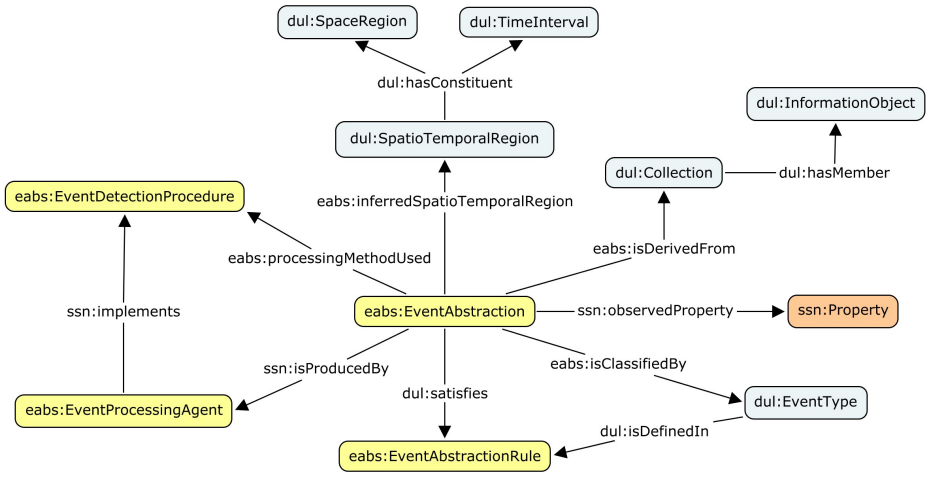

Fig. 2. Concept map of the Event-Abstraction ontology.

An EABS:EVENTABSTRACTION is an observed situation of change in one or various properties of a geographical entity that is relevant for the application purpose. Here, observing a situation of change means that an event is inferred from sensor data, which is assumed that reflects a situation of change in the environment being observed.

The spatio-temporal attributes of an EABS:EVENTABSTRACTION are specified by the concept DUL:SPATIOTEMPORALREGION. Probst [43] suggested that the spatial location of an event is defined indirectly by the location of its participants. The property EABS:INFERREDSPATIOTEMPORALREGION points to the region derived from the spatial location of the in situ sensor participating in the observation and the observation timestamps.

EABS:EVENTPROCESSINGAGENTS implement a method to infer events represented by the EABS:EVENTDETECTIONPROCEDURE. The product of the inference procedure is an EABS:EVENTABSTRACTION instance. Such instance satisfies the description of an EABS:EVENTABSTRACTIONRULE that defines an DUL:EVENTTYPE. The EABS:EVENTABSTRACTION represents changes in the observed SSN:PROPERTIES of the sensor data. Depending on the variety of data sources, the EABS:EVENTABSTRACTION can be related to more

[14] The RDF Schema Description Language 1.0 is available at http://www.w3.org/TR/2004/REC-rdf-schema-20040210/.

than one SSN:PROPERTY. A DUL:COLLECTION represents the set of sensor observations that were used to infer an event. This data set is formalised as DUL:INFORMATIONOBJECTS, i.e. pieces of information.

3.3 Implementation of the Event Abstraction Layer

This section presents an overview of the Event Abstraction Layer architecture and a description of the capabilities provided by the prototype implemented for this research, the Event Processing Service (EPS).

Architecture overview Event processing architectures are divided in three layers [21]: event producers, intermediary processing, and event consumers. Figure 3 depicts the interaction between these layers.

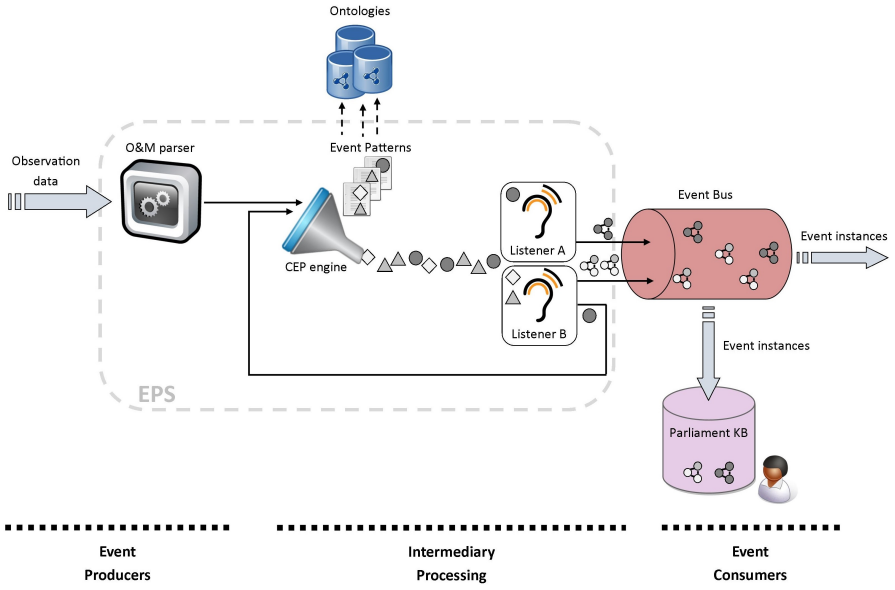

Fig. 3. Information flow diagram with the Event Abstraction Layer architecture components.

Event Producers generate events that are sent to the components in the processing layer. In our architecture the role of event producers is played by a parser that converts O&M observations into CEP objects that the CEP engine can process. These objects have the following properties: sampling time, sensor identifier, spatial location (of the sensor), observed property, observation value,

and unit of measurement. In figure 3, the O&M parser is included in included in the event producers' layer.

The *Intermediary Processing* layer of figure 3 contains a CEP engine that allows registering semantically annotated event patterns. That way, each pattern points to an ontology concept that represents an event type. Event patterns define conditions based on properties of CEP objects. Additionally, listeners are attached to event patterns. Listeners can be programmed, using the CEP engine API, to perform actions if an event pattern is matched. Figure 3 shows two examples of listeners. Listener A reacts on objects of type 'circle' and creates event instances encoded as RDF triples that are published to the Event Bus. Listener B reacts on situations where diamond and triangle objects are consecutive, and publish RDF event instances to the Event Bus. Moreover, Listener B constructs a 'circle' object that sends back to the CEP engine. This feedback to the CEP engine allows building complex patterns, e.g. diamond-triangle, upon simple event patterns.

Event consumers, such as a map to display events or other external applications, can consume the events from the Event Bus. The bus is not a consumer per se, but the channel on which the event instances are published. A copy of every event instance is stored into a knowledge base. In this implementation, I use the Parliament[15] triple store because it offers a SPARQL endpoint compatible with GeoSPARQL, a geographic query language for RDF data [42]. GeoSPARQL was developed to unify the access to geospatial data in the Semantic Web [2].

The Event Abstraction Layer overlaps the event producers and the intermediary processing layers of figure 3. The components included in the grey dashed box implement the functionalities of the Event Abstraction Layer. The EPS is the realisation of this layer and is described in next section.

The EPS prototype The heart of the EPS is a CEP engine that allows analysing time series of observations and inferring new event instances.[16] The EPS is able to process time series of observations encoded in O&M 1.0 [10]. Users can register event patterns that are checked against the continuous flow of data in near real-time. CEP does not store data before processing. Data are pushed to the CEP engine and analysed on the fly. Every time a pattern is matched, a timestamped and geolocated representation of the event in RDF is published to the Event Bus. We use the Esper CEP engine (version 4.4.0) because it provides an open source and well documented API.[17] This section presents the main functionalities that can be performed by the EPS: scheduling of sensor data requests, registration of event patterns, and publication of event instances.

Scheduling sensor data requests The EPS offers a method called `registerService` to allow users scheduling requests of time series of sensor observations. The prototype supports instances of the SOS, version 1.0.

[15] More information at http://parliament.semwebcentral.org/.
[16] The source code is available at https://github.com/allaves/EPS.
[17] Esper for Java available at http://esper.codehaus.org/about/esper/esper.html.

When an SOS is registered, a new entry is created in the service registry. Then, a `getObservation` request is scheduled to be executed recurrently. The interval between data requests is specified in the parameters:

```
registerService(String serviceURL, String offering, String observedProperty,
   String timeUnit, String numberOfTimeUnits)
```

The `serviceURL` is the URL of the service without any additional parameters, e.g. 'http://v-swe.uni-muenster.de:8080/WeatherSOS/sos'. An observation `offering` is a thematic grouping of observations offered by a service [40]. It is similar to what we understand as a map layer in terms of sensor observations, e.g. 'urn:ifgi:uni-muenster:weatherSensor:2'. An `observedProperty` is the property of a phenomenon being observed by a sensor, e.g 'urn:ogc:def:phenomenon:OGC:1.0.30:waterflow'. The time interval to schedule the data requests is defined in `timeUnit`, e.g. 'minutes', and `numberOfTimeUnits`, e.g. 7.

The O&M parser converts each observation into a CEP object. The CEP engine processes the stream of objects produced by the consecutive requests. The service is easily adaptable to process sensor data streams in different encodings. This step would require the development of an additional parser for the target encoding.

Registering event patterns. In our system, event patterns are encoded in Esper's Event Pattern Language (EPL).[18] Event patterns are semantically annotated by users. The EPS method to register annotated event patterns is called `registerStatement`. The method has two inputs: `stm` and `eventType`:

```
registerStatement(String stm, String eventType)
```

The first parameter, `stm`, is the event pattern encoded in Esper's EPL. The second parameter, `eventType`, is a URL representing the event type. `eventType` points to an ontology. In that ontology, the event type is defined as a subclass of DUL:EventType. The mapping between event pattern and event type is stored in the EPS. If the streams of data match an event pattern, an event instance is inferred and published on the Event Bus. The mapping is used to assign a type to that event instance.

Publication of event instances. The Event Bus implements the publish-subscribe interaction paradigm [22]. It uses a RabbitMQ[19] server based on the Advanced Message Queuing Protocol (AMQP) [50]. When the EPS matches a set of events defined in a pattern, an `eabs:EventAbstraction` instance is created. The properties of an EABS:EventAbstraction instance are inferred from properties of CEP objects. Each event instance is encoded in Notation 3 (N3)[20] and sent to

[18] Esper's EPL is a SQL-like language, see http://esper.codehaus.org/esper-4.4.0/doc/reference/en/html_single/index.html.

[19] http://www.rabbitmq.com/

[20] Documentation for N3 is available at http://www.w3.org/TeamSubmission/n3/.

the Event Bus as a text message. I use this encoding because it is easier to read than RDF/XML. External applications can subscribe to events based on their type, e.g. a map application subscribing to heavy rainfall events.

This prototype infers events from time series of observations. The EPS realises the annotation of event patterns by using the Event Abstraction ontology and a CEP engine. In order to evaluate this tool, we test it with a data simulation in section 5.

4 Use case: Flood Monitoring in the Danube River

The countries located along the Danube River collect data to assess hydrological and meteorological conditions. We selected two Romanian organisations for the data tests: Romanian Waters National Administration and Hidroelectrica Romania. Romanian Waters National Administration is a governmental body responsible for water management in Romania. Hidroelectrica Romania manages hydroelectric power plants located at the South-West of Romania. Two dams are used by these plants to produce energy and help to protect the villages located downstream the river from floods. Both organisations are interested in obtaining high level information related to floods in order to use it for decision making, but they have two problems: i) there is no real-time management of data collected at the dams, and ii) there is no common model for sharing geospatial information about flood-related events. The former delays responses when flooding situations are detected. The latter leads to interoperability problems when information about such situations is exchanged between the two organisations.

A lightweight flood monitoring ontology (namespace `flood`) models a river with properties like water level or discharge (water flow).[21] Such properties are measured by a stream gauge which is deployed near the river. Our interest is on events that indicate changes on these properties. Event types are modelled as subclasses of FLOOD:FLOODMONITORINGEVENT.

4.1 Two Views on the River Floods: a Governmental Body and a Hydroelectric Power Plant

The Romanian Waters National Administration is the organisation in charge of the water management in Romania.[22] Information about the state of Romanian rivers can be accessed via its online GIS.[23] In this portal, gauging stations measuring water level, discharge, 24-hours precipitation, and air temperature are represented by symbols depending on the water level trend: a circle (stable), triangle pointing up (increasing), or a triangle pointing down (decreasing).

[21] Ontology available at http://wsmls.googlecode.com/svn/trunk/local/water/0.6/.

[22] Official website available at http://www.rowater.ro/default.aspx.

[23] Romanian Waters online GIS available at http://gis2.rowater.ro:8989/SituatieHidrologica.html.

Different colours classify the status of the river at a specific point based on water level thresholds.[24] Three thresholds for *Attention*, *Flooding*, and *Danger* are defined for each gauging station based on past events and historical data:

- Green - Normal situation.
- Yellow - Attention threshold exceeded: level at which the risk of flooding is possible after a relatively short time frame. It requires increased vigilance when carrying out activities exposed to flooding, e.g. 550 cm in Calafat station.
- Orange - Flooding threshold exceeded: level at which major floods occur. It can lead to flooding of households and socio-economic goods, e.g. 600 cm in Calafat station.
- Red - Danger threshold exceeded: level at which special measures are necessary for the evacuation of people and goods, the restriction on the use of bridges and roads, and the operation of hydraulic structures, e.g. 680 cm in Calafat station.

Hidroelectrica Romania manages two hydroelectric power plants: the Iron Gates I and II. The Iron Gates, located between Romania and Serbia, are the biggest dams in the Danube River. Each of them hosts two power plants, one at the Romanian side and another one at the Serbian side. Hidroelectrica Romania collects data and operates the discharge of the reservoirs in order to avoid upstream and downstream floodings [27]. Unfortunately, these tasks are usually carried out without the full support (in terms of input data) from local authorities, which are responsible for the management of emergency situations. Although sensor networks are in place, the systems collecting and analysing the observations are not fully interoperable and data is often exchanged by phone [27].

4.2 Application ontologies and event patterns

This section presents the event patterns that we defined for each event type. Additionally, we developed an application ontology with the event types for each information community described in section 4.1. The purpose of separating application-specific knowledge is to keep the domain ontology as a reusable resource for other communities. These application ontologies are aligned to the Flood Monitoring ontology.

Event types and patterns for Romanian Waters From the Romanian Waters flood stage classification of previous section, we have defined in this scenario six event patterns that are relevant for our experiment. Four of them

[24] Descriptions in Romanian are available in the Romanian Waters' Emergency Management Regulations http://www.rowater.ro/daprut/Documente%20Repository/Regulament%20%20gestionare%20situatii%20de%20urgenta%20.pdf, CAPITOLUL II, Art. 11, section (2) B.

correspond to crossings of two water level thresholds: *attention threshold exceeded* (below, a pattern example encoded in EPL),[25] *attention threshold deceeded*,[26] *flooding threshold exceeded*, and *flooding threshold deceeded*.

```
SELECT obs1, obs2
FROM pattern[every (obs1=ObservationEvent(
   observer.id = 'urn:ogc:object:feature:Sensor:CSRO:csro-sensor-0062',
   observedProperty = 'urn:ogc:def:phenomenon:OGC:1.0.30:waterlevel',
   value < 550)
   -> obs2=ObservationEvent(
   observer.id = 'urn:ogc:object:feature:Sensor:CSRO:csro-sensor-0062',
   observedProperty = 'urn:ogc:def:phenomenon:OGC:1.0.30:waterlevel',
   value > 550))]
```

When the EPS detects any of these event patterns, it creates an EABS:EVENTABSTRACTION instance. The spatial location is derived from the location of the gauging station providing the observations. The temporal location is the time instant corresponding to `obs2`. We built the other two event patterns upon the threshold crossings described above: *attention stage*[27] and *flooding stage*[28]. When two events match any of these two patterns, the EPS creates an EABS:EVENTABSTRACTION instance with a spatial location derived from the locations of `e1-e2`. The time interval of the event instance is defined by the `e1-e2` time intervals.

We developed an application ontology with the event types of the Romanian Waters scenario, see concept map in figure 4.[29] This ontology is aligned to the Flood Monitoring ontology.

Event types and patterns for Hidroelectrica Romania Two types of events are relevant for Hidroelectrica Romania in the region of the Iron Gates: *low water level* and *high water level* events. The domain experts of Hidroelectrica Romania define thresholds for these events types. The conditions are based on observations taken at Iron Gates I and II, e.g. the water level at Iron Gates I must range between 63.00 mdMA[30] and 69.59 mdMA. Figure 5 depicts a concept

[25] Two observations `(obs1, obs2)` produced by the same sensor and ordered in time where the value of `obs1` is below the attention threshold and the value of `obs2` is above.

[26] 'Deceed' is a neologism that corresponds to the antonym of 'exceed'. Two observations `(obs1, obs2)` produced by the same sensor and ordered in time where the value of `obs1` is above the attention threshold and the value of `obs2` is below.

[27] Two events `(e1, e2)` related to the same gauging station and ordered in time where `e1` is of type *attention threshold exceeded* and `e2` is of type *attention threshold deceeded*.

[28] Two events `(e1, e2)` related to the same gauging station and ordered in time where `e1` is of type *flooding threshold exceeded* and `e2` is of type *flooding threshold deceeded*.

[29] Ontology available at http://wsmls.googlecode.com/svn/trunk/application/EventType/RomanianWaters/.

[30] Meters above the Adriatic Sea.

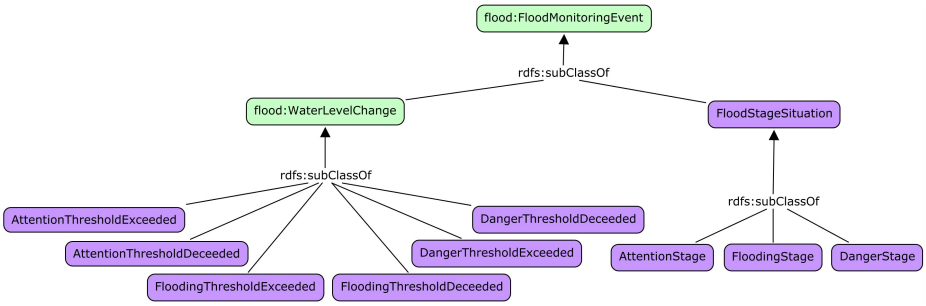

Fig. 4. Event types for the Romanian Waters scenario.

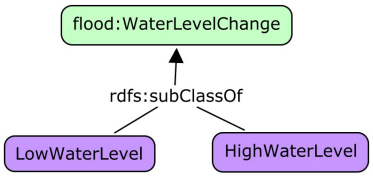

Fig. 5. Event types for the Hidroelectrica Romania scenario.

map with the corresponding application ontology, which is aligned to the Flood Monitoring ontology.[31]

We have described a flood monitoring scenario to apply the method proposed in section 3. The challenges in the flood monitoring scenario are motivated by the necessity of sharing in near real-time geospatial information inferred from time series of observations. To solve the interoperability problems that arise when different information communities interact, we suggested structuring domain- and application-specific knowledge in ad-hoc ontologies. Moreover, we showed how to extract and model event types from text descriptions. Finally, we described some examples of patterns for the target event types. The method is evaluated in the next chapter by using the developed ontologies, the event patterns, and the EPS prototype with simulated data.

5 Evaluation

This section presents the evaluation of the methodology to infer events from time series of observations. First, we describe a experiment with simulated data and its results. Second, we compare the method to other existing solutions introduced in section 2.2.

[31] The application ontology for Hidroelectrica Romania, namespace HR, is available at http://wsmls.googlecode.com/svn/trunk/application/EventType/IronGates/.

5.1 Simulating flood conditions

We designed two experiments to demonstrate that our system is able to infer events from time series of observations. The first experiment will be documented at the thesis (forthcoming) and deals with historical data from 2006. In the second experiment, included in this paper, we simulated flood conditions at the Danube River.

This experiment simulates a water level increase and decrease at a particular point of the Danube River. The water flow is only increased (and not decreased) to simulate what happens at the Iron Gates when one of the dam gates is opened. We selected a gauging station located at Pancevo, Serbia, because we have access to insert observations remotely.[32] The station is located at the Danube River, near Belgrade. The simulation is split in two parts: i) water flow increase and ii) water level increase and decrease. The former consists in increasing the initial water flow value, set to 2900 m^3/s, by adding 40 m^3/s to each new simulated observation every ten seconds during three minutes. The simulation of water level variations consists in increasing the initial water level value, set to 65.9 cm, at a rate of 0.5 cm every ten seconds during six minutes. Table 1 contains the simulated data inserted for the various time steps (t_0, t_1,..., t_{35}).

Time step / Observed property	**Water flow (m^3/s)**	**Water level (cm)**	**Time step / Observed property**	**Water flow (m^3/s)**	**Water level (cm)**
t_0	2900	65,9	t_{18}	3620	74,9
t_1	2940	66,4	t_{19}	3620	74,4
t_2	2980	66,9	t_{20}	3620	73,9
t_3	3020	67,4	t_{21}	3620	73,4
t_4	3060	67,9	t_{22}	3620	72,9
t_5	3100	68,4	t_{23}	3620	72,4
t_6	3140	68,9	t_{24}	3620	71,9
t_7	3180	69,4	t_{25}	3620	71,4
t_8	3220	69,9	t_{26}	3620	70,9
t_9	3260	70,4	t_{27}	3620	70,4
t_{10}	3300	70,9	t_{28}	3620	69,9
t_{11}	3340	71,4	t_{29}	3620	69,4
t_{12}	3380	71,9	t_{30}	3620	68,9
t_{13}	3420	72,4	t_{31}	3620	68,4
t_{14}	3460	72,9	t_{32}	3620	67,9
t_{15}	3500	73,4	t_{33}	3620	67,4
t_{16}	3540	73,9	t_{34}	3620	66,9
t_{17}	3580	74,4	t_{35}	3620	66,4

Table 1. Simulated data inserted in the SOS for the real-time experiment.

[32] The gauging station of Pancevo is located at coordinates 4447'53"N 2038'13"E.

In total, we created ten event patterns for the following event types: *low* and *high water level*, *attention threshold exceeded*, *attention threshold deceeded*, *flooding threshold exceeded*, *flooding threshold deceeded*, *attention stage*, and *flooding stage*. The formalisation of these event patterns can be found in the application that we developed for this experiment.[33] First, the application registers all event patterns with the mappings to event types. Then, the application schedules two recurrent SOS requests for water level and water flow, respectively. The method used is `registerService`, which returns the latest observations for all available sensors every ten seconds. After six minutes, the service is stopped. Below, they are described using natural language. The event types used to annotate the patterns are the ones included in the two application ontologies described in section 4.2 (`rw` is the namespace for the Romanian Waters ontology and `hr` corresponds to the concepts of the Hidroelectrica Romania scenario):

- IF the water flow is below 3000 m^3/s AND the water level drops below 65.5 cm in the following 10 seconds THEN a HR:LOWWATERLEVEL instance is created.
- IF the water flow is between 3000 and 3500 m^3/s AND the water level exceeds 69.81 cm in the following 10 seconds THEN a HR:HIGHWATERLEVEL instance is created.
- IF the water flow is above 3500 m^3/s AND the water level exceeds 71.0 cm in the following 10 seconds THEN a HR:HIGHWATERLEVEL instance is created.
- IF the water flow is above 3500 m^3/s AND the water level drops below 66.5 cm in the following 10 seconds THEN a HR:LOWWATERLEVEL instance is created.
- IF the attention threshold (70.17 cm) is exceeded THEN an instance of type RW:ATTENTIONTHRESHOLDEXCEEDED is created.
- IF the attention threshold (70.17 cm) is deceeded THEN an instance of type RW:ATTENTIONTHRESHOLDDECEEDED is created.
- IF the flood threshold (71.07 cm) is exceeded THEN an instance of type RW:FLOODINGTHRESHOLDEXCEEDED is created.
- IF the flood threshold (71.07 cm) is deceeded THEN an instance of type RW:FLOODINGTHRESHOLDDECEEDED is created.
- IF an RW:ATTENTIONTHRESHOLDEXCEEDED event is followed by an event of type RW:ATTENTIONTHRESHOLDDECEEDED event THEN an RW:ATTENTIONSTAGE instance is created.
- IF an RW:FLOODINGTHRESHOLDEXCEEDED event is followed by an event of type RW:FLOODINGTHRESHOLDDECEEDED THEN an RW:FLOODINGSTAGE instance is created.

According to the event patterns registered for this experiment and the simulated data, the EPS should generate the following twenty-five event instances located at Pancevo:

[33] The code of the real-time experiment is available at https://github.com/allaves/EPS/blob/master/EventProcessingServiceClient/test/de/ifgi/envision/eps/thesis/Danube/IGDEALThesisRealTimeDataTest.java.

- Eighteen `hr:HighWaterLevel` events created consecutively from t_8 to t_{25}, both included.
- One `rw:AttentionStage` event started by an `rw:AttentionThresholdExceeded` event at t_9 and finished by an `rw:AttentionThresholdDeceeeded` event at t_{28}.
- One `rw:FloodingStage` event started by an `rw:FloodingThresholdExceeded` event at t_{11} and finished by an `rw:FloodingThresholdDeceeded` event at t_{26}.
- One `hr:LowWaterLevel` event at t_{35}.

The execution of the experiment delivers to the bus the expected number and type of events as they are detected. To get a list of all the inferred event instances, the knowledge base can be queried via the SPARQL endpoint. The SPARQL query below requests all the event instances available in the knowledge base, including the instance URL, the event type, and the spatio-temporal location of the event.[34] The resulting event instances of this experiment are available online encoded as N3 triples.[35]

```
SELECT ?event ?eventType ?geometry ?begin ?end
WHERE {
   ?event a eabs:EventAbstraction .
   ?event eabs:isClassifiedBy ?type .
   ?type a ?eventType .
   ?event geo:hasGeometry ?spatialLocation .
   ?spatialLocation geo:asWKT ?geometry .
   ?event dul:isObservableAt ?temporalLocation .
   ?temporalLocation dul:hasBeginning ?begin .
   ?temporalLocation dul:hasEnd ?end
}
```

The Event Bus API does not provide a way to register the timestamp of the message arrival at the queue and it is not our purpose to test the performance of the bus. However, we can estimate the average delay of the message detection. First, we can give values to (t_0, t_1,..., t_{35}) by adding ten seconds to the initial timestamp (t_0 is returned when the experiment is executed). Additionally, we added a timestamp to the header of each event message just after its detection. Since we know what events should be detected and when, we can measure the difference between the corresponding time step and the event that should be detected. After several executions of the real-time experiment in the current server settings, the estimated average delay was always between three and four seconds[36]. The delay can vary depending on the features of the server but with

[34] Corresponding namespaces must be added above the query with the format 'PREFIX [namespace] <namespace_URI>').

[35] The event instances exported from Parliament are available at https://www.dropbox.com/s/gkes69fod08kx1s/realTimeExperiment_parliamentExport.n3.

[36] The EPS is running on Ubuntu 12.04.2 LTS, with an Intel Xeon CPU E5530 @ 2,40 GHz processor and a cache size of 8KB. The RAM memory has 1 GB.

low message rates, similar to the ones in our experiment and in the Danube's flood monitoring scenario, the delay should be in the order of a few seconds.

With this experiment we showed that the developed EPS prototype is able to infer events from real-time sensor data. The EPS can infer punctual events (e.g. HR:HIGHWATERLEVEL). Moreover, the event pattern language used allows for building patterns for more complex events (e.g. RW:ATTENTIONSTAGE) upon punctual event patterns. The EPS creates instances of punctual and durative events using the Event Abstraction model. Instances are published to the Event Bus in near real-time. Storing the event instances in a triple store allows executing SPARQL queries against the knowledge base. For instance, these kind of queries can be used to request events related to a specific location, or events of a specific type. Geospatial information that was not available before can now be queried via a SPARQL endpoint. The benefit of integrating different views from information communities via a common event model is that users can formulate questions about changes involving different applications on the same domain. Yet, the proposed methodology does not impose domain- or application-specific ontologies.

5.2 Multi-criteria comparison to similar methods

This section defines a set of comparison criteria and compares the presented methodology to similar solutions. The problem that our method addresses is about inferring event-related information from observations. For the comparison, we selected the solutions from section 2.2 that offer a methodology for inferring events from time series of observations. The goal of this research is to manage interoperability problems among information communities exchanging geospatial information, thus the evaluation of performance measures, such as throughput, is out of the scope of this section.

The criteria for comparison are a mixture of requirements extracted from section 1 and related work. Results are included in table 2. The content of each criterion/method cell is the answer to the criterion's question applied to the method. Our solution is located at the bottom of the table.

Methods / Criteria	Near real-time	Multi-perspective	Domain-independent	Query-ability	Decoupling
Teymourian's thesis [49]	YES	N/A	YES	YES (SPARQL)	YES
ETALIS [1]	YES	N/A	YES	YES (Event Processing SPARQL)	YES
SCEPter [53]	YES	NO	YES	YES (SCEP query model)	YES
SemSOS [28]	NO	NO	YES	YES (SOS)	YES
RTFS [41]	YES	NO	YES	YES (SPARQL)	YES
SEGO [17]	NO	NO	YES	YES (SQWRL + SPARQL)	NO
Event Abstraction Layer	YES	YES	YES	YES (SPARQL)	YES

Table 2. Comparison of the Event Abstraction Layer to similar solutions.

Near real-time: Can this method be scheduled in order to analyse streaming data in near real-time? The compared solutions are conceived to analyse vast amounts of data. In the context of the Digital Earth [26], these solutions would be implemented as part of the *nervous system* [14]. Reacting to the information extracted from the data in a timely manner requires real-time processing capabilities [45].

Multi-perspective: Is it possible to define various patterns for the same event type using different property conditions? Previous work in the area of spatial cognition and sensor data analysis call for models and methods that support multiple definitions of domain-specific concepts [5,18]. This request applied to event inference is translated to supporting multiple definitions of the same event type.

Domain-independent: Is this method applicable to other domains? Separating domain knowledge from application-specific and foundational knowledge, see section 3.1, is essential to build consistent models independently of their implementation [32].

Query-ability: Is it possible to query the generated event-related information? The capability of processing queries is considered fundamental in data integration solutions [35]. The Digital Earth also accounts for platforms that are able to answer domain-specific questions by multiple information communities [12].

Decoupling: Is the system loosely coupled in terms of event producers and event consumers? In our architecture, publishers (event producers) only deal with the production of the event and do not care about receivers (event consumers), see section 3.3. Similarly, receivers focus on listening to events, but not on who is publishing them. This decoupling is a desirable property in event-driven architectures because it enables scalability at the abstraction level, and allows producers and consumers operating independently [22].

This comparison does not intend to rank solutions for event inference from sensor data. In some cases, the compared solutions do not aim at the same research goal than the Event Abstraction Layer does. Nevertheless, the comparison shows that, for the selected criteria, the methodology developed for this thesis offers similar or better capabilities than the rest described above.

6 Conclusion

This paper investigates how to infer and represent events from time series of in situ sensor observations. The focus is on solving semantic interoperability problems among information communities exchanging event-related information.

We formulated two research questions in section 1. For the answer to RQ1, we proposed to use event processing to infer events from time series of observations.

Information communities use descriptions of events to define the observable conditions under which events happen. Event processing allows using descriptions of events, formalised as event patterns, to extract fragments of sensor data that fit the event pattern. We consider this extraction of data the event inference. As a result of the inference, an event instance is created. Spatio-temporal properties of the event are implicit in the extracted observations. We derive the spatial location of the event from the location of the in situ sensors providing the data. For the temporal location of the event, we use the observation timestamps. The type of the event is provided by the domain expert together with the event description. Other properties of the event, such as the data provenance, are useful to make the inference procedure more transparent. As a proof of concept, we developed a prototype called Event Processing Service (section 3.3).

RQ2 addresses the exchange of event-related information among information communities. We propose to model event-related information in a multilevel ontology structure, as suggested by Klien and Probst [32], see section 3.1. To model events inferred from observations, we developed the Event Abstraction ontology (section 3.2). This model is an extension of the SSN ontology [8]. The EPS creates EventAbstraction instances that are classified with a specific event type. The proposed approach allows for representing different types of events derived from the same data. Additionally, it also supports the formalisation of multiple event descriptions annotated with the same event type, which is a common problem when information communities share event-related information.

All the compared solutions in section 5.2 support analysing data from different domains. This fact highlights the trend of separating domain modelling issues from the event model and the application-specific details. One of the novelties of the presented approach is the connection between event patterns and event types. Keeping the event patterns out of the ontologies instead of including them as ontology rules allows for multiple descriptions of event types. The possibility of defining multiple patterns for the same event type is either not supported, considered as future work, or not addressed in most of the compared methods in table 2. The key is on the relationship between quantitative (event patterns) and qualitative (event types) descriptions of occurrences which are represented separately, but linked by (many to one) semantic annotations. Another interesting aspect of this approach is the semi-automatic population of ontologies. The capability of performing near real-time processing of sensor data is important in environmental monitoring domains [45,14]. Using the Event Abstraction Layer, users can schedule sensor data requests and register their patterns to infer events. This is a step forward to ground the Semantic Web in the Sensor Web [29]. This research work is a contribution to the mapping of objective measurements, made by sensors, with subjective judgements [33], in the form of event classifications within information communities. The final purpose of these mappings is to support the creation of more meaningful geospatial information.

For next steps in this research line, first we would recommend to consider additional observation sources and formats. Supporting alternative sensor data sources, such as mobile devices, could affect the inference of the event location,

since a mobile sensor is still in contact with the medium it is sensing. Regarding the input format, the implemented prototype only supports data streams encoded as O&M 1.0. For full coverage of all SOS versions, a parser for O&M 2.0 [11] should be written. Adding alternative parsers would not involve substantial changes in the architecture of the Event Abstraction Layer. Moreover, the presented approach does not perform reasoning on events. The EPS prototype manages event patterns that can involve multiple sensors, but no spatio-temporal awareness is used. For instance, in the case of flood monitoring it would be interesting to model a *flood wave* event as a result of *high water level* events in sensors located downstream consecutively. In this research direction, Worboys and Duckham [51] proposed a modelling framework for the inference of global events based on local observations. Another issue we found interesting for future work is that most of sensor data services providing environmental data lack machine-readable uncertainty medatada. The Event Abstraction Layer does not manage the propagation of errors from the sensor data to the inferred events. Uncertainty-enabled sensor data services can help to propagate uncertainty when information is inferred from data [47]. This would require changes in the sensor data parser, the Event Abstraction ontology, and the inference procedure. Finally, the Event Abstraction Layer generates event instances following the Linked Data principles.[37] Finding the right data sets in the Linked Data Cloud[38] to link new inferred events would allow event consumers navigating to other data sets and, potentially, infer new information. Some of the candidate data sets could be: Linked Sensor Data,[39] GeoLinked Data,[40] DBpedia,[41] and GeoNames[42].

Acknowledgements

This research was carried out at the Institute for Geoinformatics, University of Muenster and it was funded by the European project *ENVISION* (FP7-249170). We are thankful for discussions with members of the Muenster Semantic Interoperability Lab (*MUSIL*) and the International Research Training Group on Semantic Integration of Geospatial Information (*IRTG-SIGI*).

[37] Tim Berners-Lee defined the Linked Data Principles at http://www.w3.org/DesignIssues/LinkedData.html.
[38] http://richard.cyganiak.de/2007/10/lod/
[39] http://wiki.knoesis.org/index.php/LinkedSensorData
[40] http://geo.linkeddata.es/web/guest
[41] http://dbpedia.org/About
[42] http://www.geonames.org/

References

1. Anicic, D., Rudolph, S., Fodor, P., Stojanovic, N.: Stream Reasoning and Complex Event Processing in ETALIS. Semantic Web Journal 3(4), 1–5 (2009), http://www.semantic-web-journal.net/content/stream-reasoning-and-complex-event-processing-etalis
2. Battle, R., Kolas, D.: Enabling the Geospatial Semantic Web with Parliament and GeoSPARQL. Semantic Web Journal (SWJ) 3(4), 355–370 (2012), http://iospress.metapress.com/content/h470t26742n7x506/
3. Bishr, A.Y., Pundt, H., Kuhn, W., Radwan, M.: Probing the Concept of Information Communities - A First Step Toward Semantic Interoperability. In: Goodchild, M., Egenhofer, M., Fegeas, R., Kottman, C. (eds.) Interoperating Geographic Information Systems, The Springer International Series in Engineering and Computer Science, vol. 495, pp. 55–69. Springer US (1999), http://dx.doi.org/10.1007/978-1-4615-5189-8_5
4. Botts, M., Percivall, G., Reed, C., Davidson, J.: Ogc® sensor web enablement: Overview and high level architecture. In: Nittel, S., Labrinidis, A., Stefanidis, A. (eds.) GeoSensor Networks, pp. 175–190. Lecture Notes in Computer Science, Springer Berlin Heidelberg (2008)
5. Brodaric, B., Gahegan, M.: Experiments to Examine the Situated Nature of Geoscientific Concepts. Spatial Cognition & Computation 7(1), 61–95 (2007), http://www.tandfonline.com/doi/abs/10.1080/13875860701337934
6. Broering, A., Janowicz, K., Stasch, C., Kuhn, W.: Semantic challenges for sensor plug and play. In: Carswell, J., Fotheringham, A., McArdle, G. (eds.) Web and Wireless Geographical Information Systems, Lecture Notes in Computer Science, vol. 5886, pp. 72–86. Springer Berlin Heidelberg (2009), http://dx.doi.org/10.1007/978-3-642-10601-9_6
7. Chrisman, N.: Beyond the snapshot: changing the approach to change, error, and process. In: Egenhofer, M., Golledge, R.G. (eds.) Spatial and temporal reasoning in geographic information systems, chap. 6, pp. 85–93. Oxford University Press, USA (1998)
8. Compton, M., Barnaghi, P., Bermudez, L., Garcia-Castro, R., Corcho, O., Cox, S., Graybeal, J., Hauswirth, M., Henson, C., Herzog, A., Huang, V., Janowicz, K., Kelsey, W.D., Phuoc, D.L., Lefort, L., Leggieri, M., Neuhaus, H., Nikolov, A., Page, K., Passant, A., Sheth, A., Taylor, K.: The ssn ontology of the w3c semantic sensor network incubator group. Web Semantics: Science, Services and Agents on the World Wide Web 17(0) (2012), http://www.websemanticsjournal.org/index.php/ps/article/view/292
9. Compton, M., Henson, C., Neuhaus, H., Lefort, L., Sheth, A.: A Survey of the Semantic Specification of Sensors. In: 2nd International Workshop on Semantic Sensor Networks, at 8th International Semantic Web Conference. vol. 17 (2009)
10. Cox, S.: Observations and Measurements - Part 1 - Observation Schema (2007), http://portal.opengeospatial.org/files/?artifact_id=22466, available from: http://portal.opengeospatial.org/files/?artifact_id=22466
11. Cox, S.: Observations and Measurements - XML Implementation (version 2.0). Tech. rep., Open Geospatial Consortium (2011), http://portal.opengeospatial.org/files/?artifact_id=41510
12. Craglia, M., Goodchild, M.F., Annoni, A., Camara, G., Gould, M., Kuhn, W., Mark, D., Masser, I., Maguire, D., Liang, S., Parsons, E.: Next-generation Digital

Earth: A position paper from the Vespucci Initiative for the advancement of geographic information science. International Journal of Spatial Data Infrastructures Research (IJSDIR) 3, 146–167 (2008)
13. Davis, C.A., Fonseca, F.T., Camara, G.: Understanding global change: The role of geographic information science in the integration of people and nature. In: Nyerges, T., Couclelis, H., McMaster, R. (eds.) The SAGE Handbook of GIS and Society, pp. 123–137. SAGE Publications Limited, Thousand Oaks, CA (2009)
14. De Longueville, B., Annoni, A., Schade, S., Ostlaender, N., Whitmore, C.: Digital Earth's Nervous System for Crisis Events: Real-Time Sensor Web Enablement of Volunteered Geographic Information. International Journal of Digital Earth 3(3), 242–259 (2010)
15. Delin, K.A., Jackson, S.P.: The Sensor Web: A New Instrument Concept. In: Descour, M.R., Rantala, J.T. (eds.) Proceedings of SPIE, the International Society for Optical Engineering. vol. 4284, pp. 1–9. SPIE, San Jose, CA (2001)
16. Demers, A., Gehrke, J., Panda, B., Riedewald, M., Sharma, V., White, W.: Cayuga: A General Purpose Event Monitoring System. In: CIDR 2007. pp. 412–422 (2007), http://www.ccs.neu.edu/home/mirek/papers/2007-CIDR-CayugaImp.pdf
17. Devaraju, A.: Representing and Reasoning about Geographic Occurrences in the Sensor Web. Ph.D. thesis, Institute for Geoinformatics, University of Muenster. (2012)
18. Devaraju, A., Kauppinen, T.: Sensors Tell More than They Sense: Modeling and Reasoning about Sensor Observations for Understanding Weather Events. Special Issue on Semantic Sensor Networks, International Journal of Sensors, Wireless Communications and Control 2(1) (2012), http://kauppinen.net/tomi/devaraju-kauppinen-sensors-2011.pdf
19. Eckert, M.: Complex Event Processing with XChangeEQ. Ph.D. thesis, Ludwig-Maximilians-Universität München. (2008)
20. Etzion, O.: Semantic approach to event processing. In: Proceedings of the 2007 inaugural international conference on Distributed event-based systems. pp. 139–139. DEBS '07, ACM, New York, NY, USA (2007), http://doi.acm.org/10.1145/1266894.1266920
21. Etzion, O., Niblett, P.: Event Processing in Action. Manning Publications Co., Greenwich, CT, USA (2010)
22. Eugster, P.T.H., Felber, P.A., Guerraoui, R., Kermarrec, A.m.: The Many Faces of Publish/Subscribe. ACM Computing Surveys 35(2), 114–131 (2003), http://doi.acm.org/10.1145/857076.857078
23. Everding, T., Echterhoff, J.: OGC OWS-6 SWE Event Architecture Engineering Report (2009), available from: https://portal.opengeospatial.org/files/?artifact_id=33347
24. Galton, A., Worboys, M.: Processes and events in dynamic geo-networks. In: Rodrguez, M., Cruz, I., Levashkin, S., Egenhofer, M. (eds.) GeoSpatial Semantics, Lecture Notes in Computer Science, vol. 3799, pp. 45–59. Springer Berlin Heidelberg (2005), http://dx.doi.org/10.1007/11586180_4
25. Gangemi, A.: DOLCE+DnS Ultralite ontology (2010), available from: http://www.loa.istc.cnr.it/ontologies/DUL.owl
26. Gore, A.: The digital earth: understanding our planet in the 21st century. Australian surveyor 43(2), 89–91 (1998)
27. Grosu, M., Trasca, S., Udroiu, C., Ionescu, M., Grosu, D., Vilcu, M.: ENVISION Deliverable 1.7 - Real-time Pilot Case Definition and Requirements Specification. Tech. rep., ENVISION project (2012), http://www.envision-project.eu/wp-content/uploads/2012/05/D1-7_v1-1.pdf

28. Henson, C.A., Pschorr, J.K., Sheth, A.P., Thirunarayan, K.: Semsos: Semantic sensor observation service. In: Proceedings of the 2009 International Symposium on Collaborative Technologies and Systems (CTS '09). pp. 44–53. IEEE Computer Society, Washington, DC, USA (2009), http://dx.doi.org/10.1109/CTS.2009.5067461
29. Janowicz, K.: The role of space and time for knowledge organization on the semantic web. Semantic Web Journal 1(1), 25–32 (2010)
30. Kabadayi, S., Pridgen, A., Julien, C.: Virtual Sensors: Abstracting Data from Physical Sensors. In: Proceedings of the 2006 International Symposium on World of Wireless, Mobile and Multimedia Networks. pp. 587–592. WOWMOM '06, IEEE Computer Society, Washington, DC, USA (2006)
31. Kavouras, M., Kokla, M.: Theories of geographic concepts: ontological approaches to semantic integration. CRC Press, Boca Raton (2007)
32. Klien, E., Probst, F.: Requirements for Geospatial Ontology Engineering. In: Toppen, F., Painho, M. (eds.) 8th Conference on Geographic Information Science (AGILE 2005). pp. 251–260. Estoril, Portugal (2005), http://www.agile-online.org/Conference_Paper/CDs/agile_2005/papers/79_Eva%20Klien.pdf
33. Kuhn, W.: Geospatial semantics: why, of what, and how? Journal on data semantics III 3534, 1–24 (2005)
34. Kuhn, W.: Core concepts of spatial information for transdisciplinary research. International Journal of Geographical Information Science 26(12), 2267–2276 (2012)
35. Lenzerini, M.: Data integration: a theoretical perspective. In: Proceedings of the 21st ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. pp. 233–246. PODS '02, ACM, New York, NY, USA (2002), http://doi.acm.org/10.1145/543613.543644
36. Luckham, D.: The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. Addison-Wesley Professional, Boston (2002)
37. Luckham, D., Schulte, R., Adkins, J., Bizarro, P., Mavashev, A., Niblett, P.: Event Processing Glossary - Version 2.0 (2011), available from: http://www.complexevents.com/wp-content/uploads/2011/08/EPTS_Event_Processing_Glossary_v2.pdf
38. Masolo, C., Borgo, S., Gangemi, A., Guarino, N., Oltramari, A., Horrocks, I.: WonderWeb Deliverable D18 - Ontology Library (final). Tech. rep., WonderWeb project (2003), http://wonderweb.man.ac.uk/deliverables/documents/D18.pdf
39. Miller, H.J.: The data avalanche is here. Shouldn't we be digging? Journal of Regional Science 50(1), 181–201 (2010)
40. Na, A., Priest, M.: OpenGIS Sensor Observation Service (2007), available from: http://portal.opengeospatial.org/files/?artifact_id=26667
41. Patni, H.: Real Time Semantic Analysis of Streaming Sensor Data. Master's thesis, Wright State University, Ohio (2011)
42. Perry, M., Herring, J.: OGC GeoSPARQL - A Geographic Query Language for RDF Data. Tech. rep., Open Geospatial Consortium (2012), https://portal.opengeospatial.org/files/?artifact_id=47664
43. Probst, F.: Ontological analysis of observations and measurements. In: Raubal, M., Miller, H., Frank, A., Goodchild, M. (eds.) Geographic Information Science, Lecture Notes in Computer Science, vol. 4197, pp. 304–320. Springer Berlin Heidelberg (2006), http://dx.doi.org/10.1007/11863939_20
44. Reitsma, F.: A New Geographic Process Data Model. Ph.D. thesis, University of Maryland. (2004)

45. Resch, B., Mittlboeck, M., Girardin, F., Britter, R., Ratti, C.: Live Geography - Embedded Sensing for Standardised Urban Environmental Monitoring. International Journal on Advances in Systems and Measurements 2(2&3), 156–167 (2009), http://www.berndresch.com/download/work/publications/resch_et_al_asm_urban_environmental_monitoring.pdf
46. Sheth, A., Henson, C., Sahoo, S.S.: Semantic Sensor Web. IEEE Internet Computing 12(4), 78–83 (2008), http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4557983
47. Stasch, C., Jones, R., Cornford, D., Kiesow, M., Williams, M., Pebesma, E.: Representing Uncertainties in the Sensor Web. In: Workshop Sensing A Changing World. pp. 1–7 (2012)
48. Teymourian, K., Paschke, A.: Towards semantic event processing. In: Proceedings of the Third ACM International Conference on Distributed Event-Based Systems. pp. 29:1–29:2. DEBS '09, ACM, New York, NY, USA (2009), http://doi.acm.org/10.1145/1619258.1619296
49. Teymourian, K., Paschke, A.: Enabling knowledge-based complex event processing. In: Proceedings of the 2010 EDBT/ICDT Workshops. pp. 37:1–37:7. EDBT '10, ACM, New York, NY, USA (2010), http://doi.acm.org/10.1145/1754239.1754281
50. Vinoski, S.: Advanced Message Queuing Protocol. Internet Computing, IEEE 10(6), 87–89 (2006)
51. Worboys, M., Duckham, M.: Monitoring qualitative spatiotemporal change for geosensor networks. International Journal of Geographical Information Science 20(10), 1087–1108 (Nov 2006), http://www.informaworld.com/openurl?genre=article&doi=10.1080/13658810600852180&magic=crossref||D404A21C5BB053405B1A640AFFD44AE3
52. Worboys, M., Hornsby, K.: From objects to events: GEM, the geospatial event model. In: Egenhofer, M., Freksa, C., Miller, H. (eds.) Third International Conference on GIScience 2004. pp. 327–344. SpringerVerlag, Berlin (2004)
53. Zhou, Q., Simmhan, Y., Prasanna, V.: Scepter: Semantic complex event processing over end-to-end data flows (2012), available from: http://www.cs.usc.edu/assets/001/82856.pdf