Event dashboard: Capturing user-defined semantics events for event detection over real-time sensor data

Jonathan Yu[1] and Kerry Taylor[2]

[1] CSIRO Land and Water, Graham Road, Highett, Melbourne VIC 3190 Australia
[2] CSIRO ICT Centre GPO Box 664 Canberra ACT 2601 Australia

{Jonathan.Yu,Kerry.Taylor}@csiro.au

Abstract. Sensor networks provide the ability to observe physical phenomena in real-time and provide useful information to help conservation and management of environmental resources. However, sensor data meaning, format and interface heterogeneity are barriers to effective discovery and analysis of events of interest. We propose a web-based user application, the Event Dashboard, which facilitates user capture semantics for events of interest over a sensor network. The Event Dashboard user interface is driven by a set of ontologies, which provide metadata about relevant domain concepts and the sensor network. We utilise the SSN sensor ontology to capture constraints on the sensor metadata. We propose ontology extensions for capturing domain and event semantics using a case study in the water quality domain. Our approach allows the event description to be abstracted from specific interfaces of a sensor network and to be used for querying of sensor data. Event descriptions can subsequently be deployed through a semantic mediator to complex event processing and stream processing implementations over a sensor network.

Keywords: ontologies, sensor networks, semantic sensor web, semantic sensor network, user interface design

1 Introduction

Technological advancement has allowed real-time sensor data to be published and shared through increased connectivity with physical devices through web services. This is useful for informing policy and decision making in the management of the environment, infrastructure assets, and early warning systems. However, protocols and interfaces for discovery and access to sensors vary as they are not widely standardised. Service platforms and middleware technologies that provide standard (or defacto standard) protocols and Application Programming Interfaces (APIs) to allow consistent access to sensors are essential in handling the heterogeneous nature of sensor networks. Platforms, such as GSN [1], have provided the above features for enabling sensor data to be available via web services.

Despite the availability of such service platforms, finding useful information within the sensor data streams is still a challenge. Sensor data often provides parameter level information about events about a physical phenomenon, and as such it can be too fine-grained. Abstractions over the sensor data help to provide insight into significant events, e.g. a flood event occurring when heavy precipitation takes place over a short period of time.

However, users with domain knowledge and insight into these events are impeded by having to learn the syntax, languages, data formats, and APIs associated with sensor networks, their access protocols as well as middleware platforms such as GSN. Users may also need to consider data quality issues that arise from handling raw data streams. Without any appropriate user interfaces, the ability to directly interact and perform high level queries on sensor data is impeded.

Kwon et al. [10] present a system called SCONSTREAM for aggregating spatial data streams into spatial context streams based on spatial information maintained in a sensor catalogue. This system provides a user interface (UI) to allow the definition of spatial based queries, rather than on individual sensors. While this is a step towards lowering the barrier for users, this approach does not allow for event detection based on the semantics of the underlying sensor network expressed in a standard way, which restricts its interoperability and extensibility.

The semantic annotation of sensor data through a Semantic Sensor Web (SSW) is a proposed solution to the heterogeneity of sensor networks using ontologies and semantic web technologies [16]. This approach assumes three things: users are competent with both the semantic rule language and the task of formulating appropriate semantic rules; appropriate semantic rule engines are available; and standardised ontologies are available. At the time of the above work, sensor ontologies were not standardised, which meant that the semantics used lacked a common agreed representation of the semantics of a sensor network.

An alternative solution is to provide query languages to sensor data streams that have been formalized using RDF. Calbimonte et al. [4] propose a method of accessing sensor streams which have been mapped to an ontology via a data translation process. This process produces RDF data streams from the raw sensor data streams. They propose SPARQLstream, for capturing queries to streaming RDF data that have been translated from the raw sensor data. There are also other proposed solutions for querying semantic sensor data, such as C-SPARQL [18, 3], CQELS [11] and EP-SPARQL [2]. The limitations with such query-based approaches are that the set of query language extensions has not been standardised and a representation of the sensor data in RDF is assumed, which is certainly uncommon in practice. From a user's perspective, defining a query using such query-based approaches at the level of a user such as a domain scientist is a significant barrier. Thus a more appropriate mechanism is required for such users.

The approach proposed by Yu et al. [19] seeks to address the above challenges using an ontology-based approach for complex event processing over a sensor network (shown in Fig. 1). Yu et al. extends the prior work proposed in [17] using the SSN ontology developed by the W3C SSN-XG working group for describing sensors that is compatible with other standards [5], e.g. OGC Observations and Measurements. They also propose the use of GSN as a platform for interfacing with sensors and providing stream processing capabilities. This enables users to define complex events associated with the semantically annotated sensors. The current challenge in the above work is the appropriate methodology for facilitating user definition of event constraints. An *event constraint* relates to properties of the event of interest – that is, physical, spatial, and temporal properties of a set of observations over the sensor network, e.g. observed properties over a sensor network (physical) monitoring water quality parameters for a storage dam (spatial) over hourly data (temporal). An event constraint therefore captures the semantics of specific kinds of observation values from sensor data streams, e.g. when air temperature exceeds 25C. This presents two specific challenges: a) capture of event constraint semantics; and b) facilitating user-based definition of event constraints via ontology-driven interfaces.

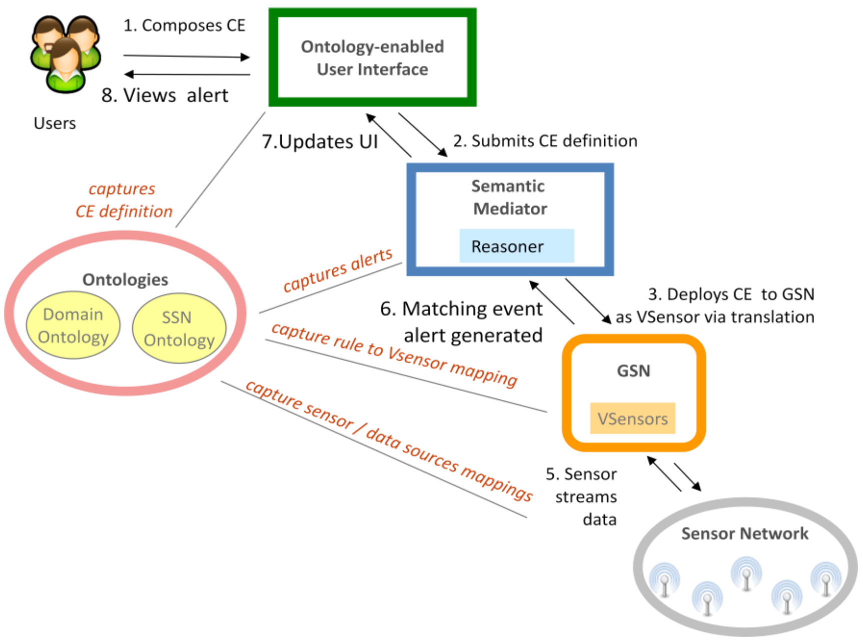


Fig. 1. Ontology-driven complex event processing over a sensor network

In this paper, we propose the *Event Dashboard*, which is an ontology-driven UI to address the challenge of facilitating user definition of machine-readable semantic descriptions of event constraints from the semantic descriptions of available sensors in the sensor network. Our approach aims to resolve the data heterogeneity of sensor networks by using a domain ontology, which extends the SSN ontology for representing semantics of a sensor network and relevant domain concepts. The *Event Dashboard* directly uses the aforementioned ontologies to populate relevant UI elements such as selection and input forms. We hypothesize that this approach helps to enable users to express event constraints over a sensor network. The captured constraints of event of interest can be then used for deploying complex event queries that is compatible with the approach proposed in [19].

We present this work in the context of a case study for detecting water quality events in water reservoirs in early warning systems, specifically algal bloom events. Nutrient availability, such as nitrogen and phosphorus, has been studied and determined as a factor in algal growth [15]. Detection can help authorities respond with appropriate preventative management to mitigate potential algal bloom events. In this paper, we will present a discussion of a domain ontology for representing event constraints to support this case study, limiting our discussion to observations around water quality chemical properties, such as *Total Nitrogen*, for which measurements are available. The structure of the paper is as follows. Sections 2 and 3 discuss the SSN ontology briefly, present our domain extensions in the context of annotating sensor observations of a Water feature, and propose a simple model for capturing event constraints. In Section 4, we present the ontology-driven UI design and implementation details for the *Event Dashboard* that enables the capture of events of interest. We then discuss this specific ontology-driven UI with other related work in Section 5 and present conclusions in Section 6.

2 Extending the SSN Ontology with Domain concepts

Ontologies are key components in our approach. They are used to drive the UI and allow users to select the appropriate set of event constraints. We propose domain extensions to the SSN ontology represent sensor observations for a water feature over a sensor providing a data stream and, more specifically, water quality chemical properties for Chaffey Dam sensor observations. We also reuse the Quantities, Units, Dimensions and Types (QUDT) ontologies [8] to represent the constraints around the observation values from a sensor. In this paper, we present the OWL ontologies as visualisations using the Cmap Ontology editor tool [6].

2.1 An observation-centric view of the SSN ontology

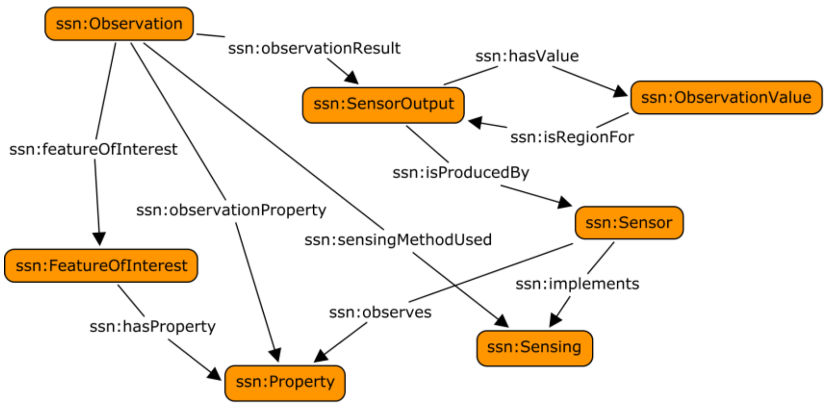


Fig. 2. Extract of the SSN ontology

The SSN ontology [5] models an observation with regards to a feature of interest, its observed property and the result (shown in Fig. 2). The `Observation` class relates to the `Sensor` class via two paths: via the `sensingMethodUsed` and the `observationResult` to the output of the sensor (i.e. the `SensorOutput` class). Focusing on the `SensorOutput` class, it is modelled with a relationship to an `ObservationValue` class, which is used to represent the value of the output from the sensor, e.g. a value of 21 degrees Celsius for the property, Air Temperature, produced by a temperature sensor. The rest of this section describes the specific extensions we have made to the SSN ontology relating to the above extract of the SSN ontology for enabling the representation of event constraints.

2.2 Representation of streaming sensor data observations for a water feature with the SSN Ontology

We propose an an extension module to the SSN ontology by subclassing the following SSN classes - `Observation, Sensor, SensorOutput, and ObservationValue`, with classes defined in the `ext:` namespace to maintain the declarations in our extension module separate from original SSN definitions (shown in yellow in Fig. 3). We extend the `ObservationValue` class specifically with the `QuantityObservationValue` class for representing quantity values with reference to a Unit of measure ontology. In this case, we utilise the Unit class from the QUDT ontology [8], which is shown shaded in purple in the figure below. The intention is to allow the referencing of a subset of the defined units of measure instances from the QUDT ontology. We a set of SSN classes to represent domain concepts (shown in blue in Fig. 3). The domain extensions allow references to concepts relating to observations of a Water Feature to be made, i.e. we extended the `FeatureOfInterest` class to define a `WaterFeature` class. This allows us to represent particular instances of water features, such as lakes, rivers, and storage dams.

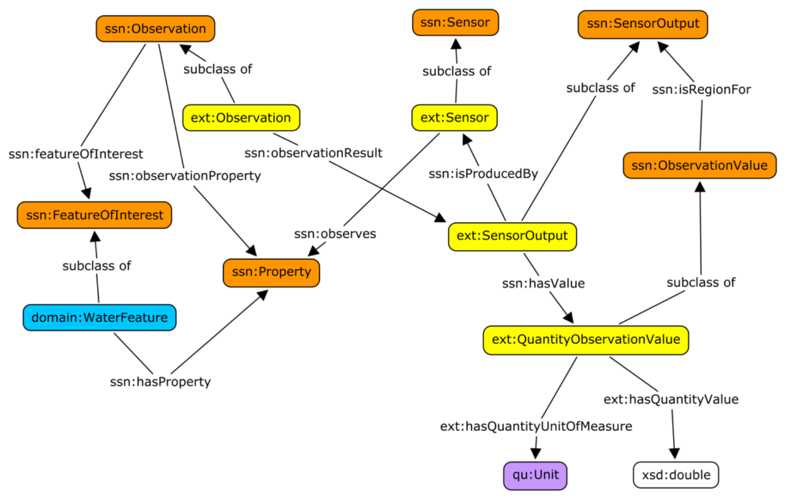


Fig. 3. Domain extensions to the SSN ontology to allow representation of the sensor data

2.3 Representation of domain-specific observations for a water feature

We further extend the proposed SSN extension module to describe domain-specific observation types. In the case of the Chaffey dam case study, we model observations around water quality chemical properties in the Chaffey dam by introducing new classes. We introduce the `WaterQualityChemicalObservation` class as a new observation type and add a restriction for the `observedBy` property associating it with the new sensor class, `WaterQualitySensor` (see Fig. 4). We add a property restriction between a `WaterFeature` class and a new class, `WaterQualityChemicalProperty` to associate the kinds of water quality properties that is possible for a water feature.

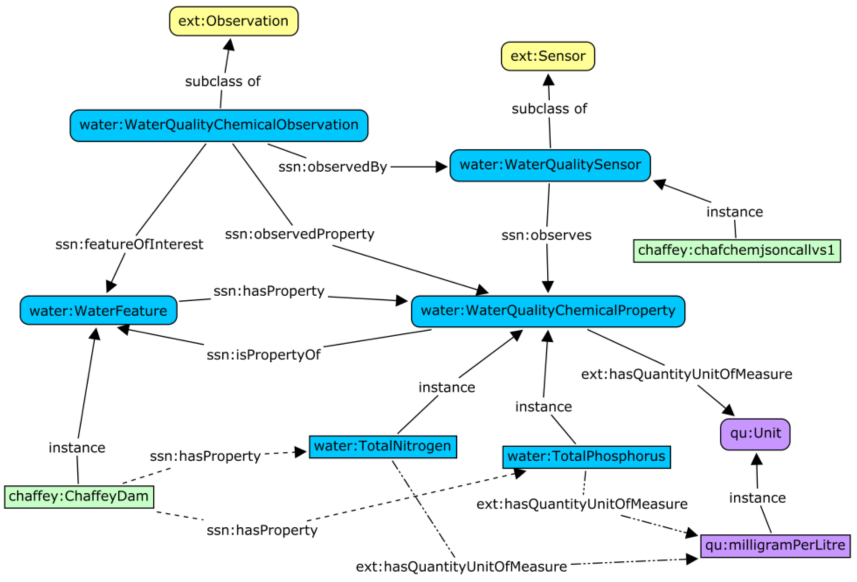


Fig. 4. Domain extensions to SSN ontology to represent Water Quality Chemical Observations

We define the following instances to support the definition of constraints around the water chemical observations: `TotalNitrogen`, `TotalPhosphorus`, and `Ph` as instances of the `WaterQualityChemicalProperty` class. Lastly, we include the `milligramPerLitre` instance of the QUDT Units ontology for specifying the units for the observation values and relate them to the respective property instances.

The above domain ontology allows us to model Chaffey Dam as an instance of a water feature and associate it with specific water quality property instances in our model, such as Total Nitrogen. The water quality instances in our domain model include relationships to the respective QUDT unit of measure instances. The instance level information will be used for constraining available property and unit selection in the UI. This is shown in green in the above figure.

We can also extend the model with other relevant observation properties, such as meteorological properties relating to the spatial location of the water feature (e.g. temperature, wind speed, atmospheric pressure, humidity, precipitation), and also water quantity observation properties (e.g. flow and water level). To represent observations around these properties, the respective extensions would include appropriate sensor classes relating to the added observation properties.

3 Representing event constraints

A model is required for capturing machine-readable descriptions of event constraints. An **event constraint** in this context captures physical and spatial properties of a set of observation values over the sensor network. In terms of defining these properties using the SSN ontology, we needed to capture constraints around the respective classes: `ssn:FeatureOfInterest`, `ssn:Property`, and `ssn:ObservationValue` (see Fig. 5).

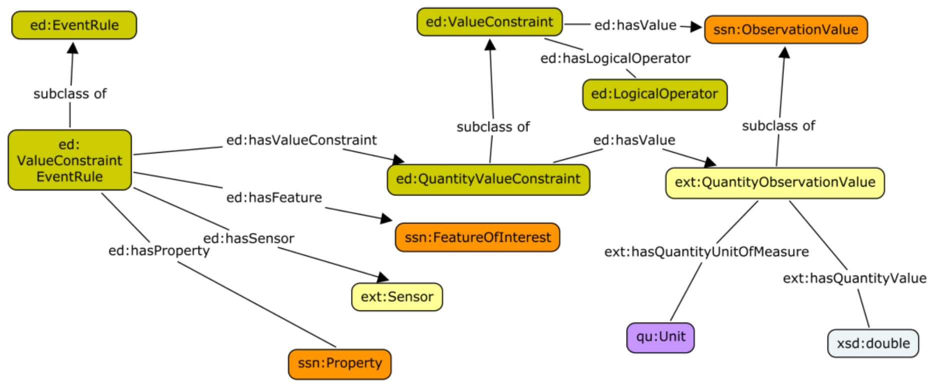


Fig. 5. A model for capturing **e**vent constraints

We have introduced two main classes for handling the event constraint - `ed:EventRule` and `ed:ValueConstraint`. Subsequently, we have defined two classes which specialise these classes – `ed:ValueConstraintEventRule` and `ed:QuantityValueConstraint` – to handle the definition of a class of event constraints based on quantity value constraints.

The Value Constraint classes allow the definition of a value constraint such as, a concentration value constraint with the milligrams per litre unit greater than the value of 10.0, with the following OWL 2 RDF representation of `concentrationConstraint` (shown in Fig. 6 using Turtle syntax).

```
:concentrationConstraint a ed:QuantityValueConstraint ;
   ed:hasValue [
     a   ext:QuantityObservationValue ;
     ext:hasQuantityValue "10.0"^^xsd:double ;
     ext:hasQuantityUnitOfMeasure qu:milligramsPerLitre;
   ] ;
   ed:hasLogicalOperator ">"^^xsd:string .
```

Fig. 6. OWL description of quantity value constraint

The `ed:ValueConstraintEventRule` class allows us to capture event constraints that associate the value constraints with the feature of interest (e.g. Chaffey Dam), sensor (e.g. water quality sensor - chafchemvm) and observed property (e.g. total phosphorus). This allows us to capture an event constraint such as `highPhosphorusEvent` as shown in Fig. 7.

```
:highPhosphorusEvent a ed:ValueConstraintEventRule ;
   ed:hasProperty water:TotalPhosphorusObservedProperty ;
   ed:hasFeature  chaffey:ChaffeyDam ;
   ed:hasSensor :chafchemvm ;
   ed:hasValueConstraint :concentrationConstraint .
```

Fig. 7. OWL description of an event constraint

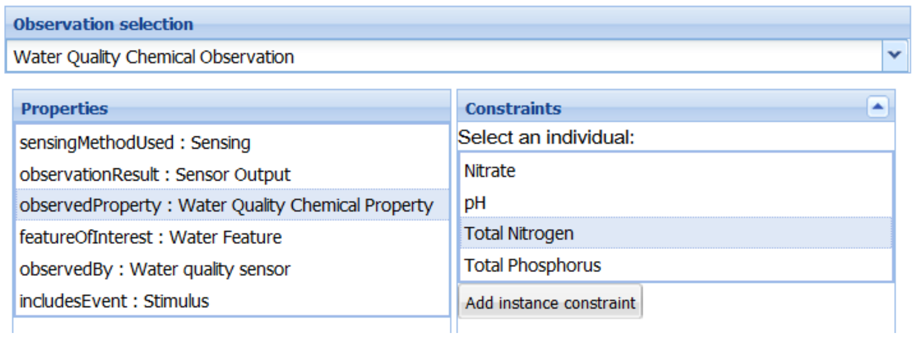
Using the above OWL model, therefore, enables the capture of machine-readable descriptions of event constraints in a declarative manner, whilst reusing appropriate SSN classes where possible. We have shown that we can use this model to capture simple physical and spatial properties, and constraints over the observation values relevant for a sensor network. We will use this model as the basis for proposing an ontology-driven UI that allows users to express these constraints.

4 The Event Dashboard

We present the design and implementation details for an ontology-driven UI to facilitate the capture of constraints over observations. The UI uses the domain extensions to the SSN ontology and event constraint model described in the previous two sections, which we refer to as the domain ontology. The domain ontology provides the metadata about relevant domain concepts and the sensor network. This allows for a simple method of capturing machine-readable descriptions of event constraints without the user having to learn the syntax and language of the ontology specification language. We seek to capture constraints over events of interest using ontological descriptions via our proposed UI.

4.1 Leveraging the ontology in the UI

Classes in the SSN ontology are used to bind the various UI panels with appropriate details. Observation constraints are key components in specifying events of interest. Thus, we propose a UI design with three panels (see Fig. 8).



Observation selection

Water Quality Chemical Observation

Properties

sensingMethodUsed : Sensing
observationResult : Sensor Output
observedProperty : Water Quality Chemical Property
featureOfInterest : Water Feature
observedBy : Water quality sensor
includesEvent : Stimulus

Constraints

Select an individual:

Nitrate
pH
Total Nitrogen
Total Phosphorus

Add instance constraint

Fig. 8. UI for specifying constraints for the observed property

The first panel, the Observation selection panel (shown in the top of Fig. 8), captures the various constraints that require a selection of the observation class. The second panel, the Properties panel (shown on the left side of Fig. 8), presents the user with the property restrictions for the selected observation. The third and last panel, the Constraints panel (shown on the right side of Fig. 8), presents to the user an appropriate form for users to input the appropriate constraints in context of the selected observation and property restriction (Constraints panel). The three panels allow the user to add a constraint definition. Users can append constraints one after another by interacting with this form such that users can define conjunctive constraints on the set of properties listed in the Properties panel.

OWL API [9] is used to interface with the ontologies and is used, with reasoning capabilities provided by the Pellet reasoner [12], to populate components such as the list of observation subclasses for the observation selection drop-down box, the list of property restrictions for the selected observation class for the properties panel and the appropriate forms and input widgets for the constraints panel (e.g. list of instances for the appropriate class of the property restriction selection).

The UI is implemented with a binding to the URIs of classes in the SSN ontology. For example, the Observation selection panel is bound to the URI of the `Observation` class in the SSN ontology. This allows for the UI to be unchanged for other domain extensions of the SSN ontology. It also allows for the UI to incorporate any additional extensions to the domain ontology.

The UI is implemented using the following APIs: Google Web Toolkit API, OWL API, and the Pellet reasoner. The Google Web Toolkit (GWT) allows us to develop UIs with the look and feel of standard HTML forms and allows our application to be deployed on servers such as Apache Tomcat for access via standard web browsers. Due to technical challenges, we have developed additional software libraries that bridge GWT and ontology tools, such as, OWL API, for building the ontology driven UIs. The bridging software libraries are available via GitHub[1].

4.2 Specifying observed property constraints

In Fig. 8, we show how constraints for an observed property of the `WaterQualityChemicalObservation` class are specified via the UI. The `observedProperty` property restriction is selected in the Properties panel. The UI uses OWL API and the reasoner to evaluate an appropriate Property class for the observedProperty of the selected observation class. In this case, the `observedProperty` of a `WaterQualityChemicalObservation` is `WaterQualityChemicalProperty`. For this particular case, to specify an `observedProperty` property constraint, the Constraints panel displays the list of available instances corresponding to the modeled class, `WaterQualityChemicalProperty`. In the example shown in Fig. 8, the user has selected `TotalNitrogen` is selected as the constraint for the `observedProperty`. The constraint captured with this UI populates the respective information in the event constraint definition as per the following:

```
:exampleEventConstraint a ed:ValueConstraintEventRule ;
   ed:hasProperty water:TotalNitrogenObservedProperty .
```

4.3 Specifying observedResult constraint

Another key requirement in specifying an observation constraint is defining a constraint on the observation result. The interface design for capturing the observed result constraint is more complex than the previous UI presented in Section 4.1, due to the way the SSN ontology models the result.

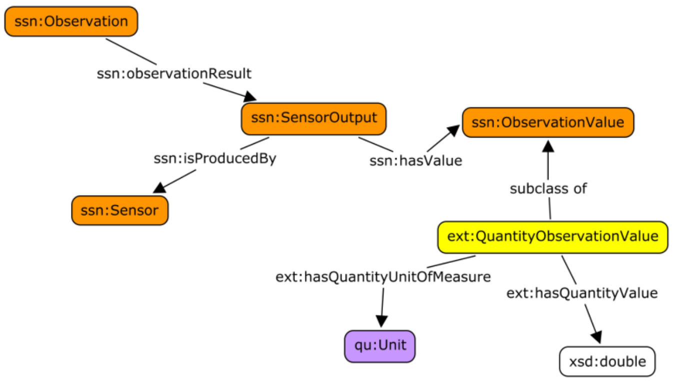


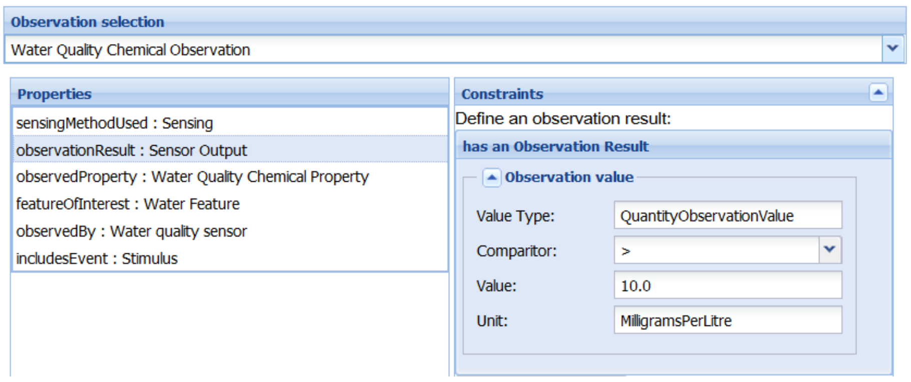
Fig. 9. Extract of the domain ontology around observation result

[1] https://github.com/jyucsiro/gwt-ontology-lib

In the SSN ontology, an `Observation` has an associated `observationResult` property with a `SensorOutput` class. The `SensorOutput` class has an association with a `QuantityObservationValue` class. The domain ontology extensions we propose allow for quantity observation values to be represented (shown in Fig. 9).

To adequately capture the constraint of a sensor output, the interface design requires a specification of an instance of `SensorOutput` class to encapsulate the modelling of the domain ontology as shown in the above figure. An example of the UI selection to facilitate the input of a logical constraint for the `QuantityObservationValue` is shown in Fig. 10.

For this particular case, as shown in the below figure, we needed to create a customised form layout using the ontology-driven UI APIs to flatten the property chains from an Observation class to a definition of a constraint on a `QuantityObservationValue` class. We bind the cu stomised form input elements to corresponding elements in the SSN ontology and our extension module, rather than the domain ontology to enable reusability in other domains. Based on the context of the user selection, the OWL API and Pellet reasoner is used to evaluate the appropriate domain specific `Sensor` and `SensorOutput` classes. The figure below shows an example where the reasoner has evaluated the appropriate `Sensor` and `SensorOutput` classes for an `observationResult` of a `WaterQualityChemicalObservation` class `SensorOutput` and `WaterQualitySensor` respectively.


Observation selection
Water Quality Chemical Observation

Properties
sensingMethodUsed : Sensing
observationResult : Sensor Output
observedProperty : Water Quality Chemical Property
featureOfInterest : Water Feature
observedBy : Water quality sensor
includesEvent : Stimulus

Constraints
Define an observation result:
has an Observation Result
Observation value
Value Type: QuantityObservationValue
Comparitor: >
Value: 10.0
Unit: MilligramsPerLitre

Fig. 10. User interface for specifying an observation result

We define a datatype for expressing logical comparison operators (shown in Fig. 11).

```
ed:LogicalOperator  a  rdfs:Datatype ;
    owl:oneOf (  ">"^^xsd:string
                "<="^^xsd:string
                 ">="^^xsd:string
                 "=="^^xsd:string
                 "<"^^xsd:string)  .
```

Fig. 11. OWL Datatype definition of Logical Operator

This is used to populate the list of available logical-operator constraints on the QuantityObservationValue section of the customised form. In the example in the above figure, the constraint on the QuantityObservationValue is "greater-than a value of 10.0". This updates the event constraint as shown in Fig. 12.

```
:exampleEventConstraint a ed:ValueConstraintEventRule ;
   ed:hasProperty water:TotalNitrogenObservedProperty ;
   ed:hasValueConstraint [
      a ed:QuantityValueConstraint ;
      ed:hasLogicalOperator ">"^^xsd:string .
      ed:hasValue [
         a   ext:QuantityObservationValue ;
         ext:hasQuantityValue "10.0"^^xsd:double ;
         ext:hasQuantityUnitOfMeasure
                    qu:milligramsPerLitre;
      ] ;
   ].
```

Fig. 12. Example event constraint OWL description expressed using Turtle syntax

4.4 Specifying observedBy constraint

Constraining the sensor selection is a key requirement. The observedBy property restriction for an Observation class associates directly with a Sensor class. The UI design to capture the constraint allows users to be able to select from a list of appropriate Sensor instances. Users can select Sensor instances for the selected Observation class using the right hand side "Constraints" panel of the UI (shown in Fig. 13).

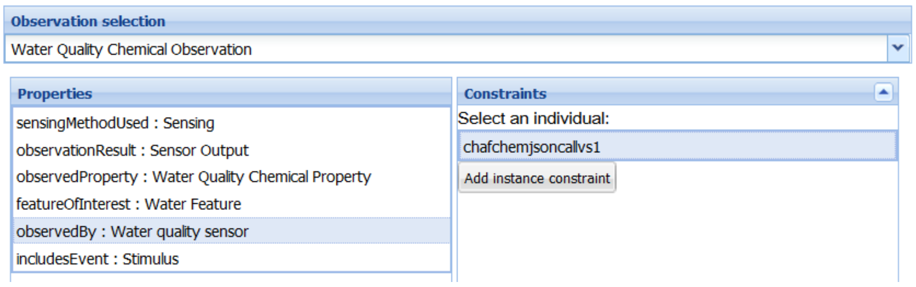

Observation selection
Water Quality Chemical Observation
Properties
sensingMethodUsed : Sensing
observationResult : Sensor Output
observedProperty : Water Quality Chemical Property
featureOfInterest : Water Feature
observedBy : Water quality sensor
includesEvent : Stimulus
Constraints
Select an individual:
chafchemjsoncallvs1
Add instance constraint

Fig. 13. User interface for specifying the observed by constraint

This yields the following expression:

```
:exampleEventConstraint a ed:ValueConstraintEventRule ;
   ed:hasSensor chaffey:chafchemjsoncallvs1   .
```

A fuller example of the capability offered with the above UIs is shown below in Fig. 14 for defining a high total nitrogen event: which is defined to be for WaterQuali-

`tyChemicalObservation` on a `Sensor`, which observes the `TotalNitrogen` property with a value greater than 10.0.

```
:HighNitrogenEventConstraint a ed:ValueConstraintEventRule ;
   ed:hasProperty water:TotalNitrogenObservedProperty ;
   ed:hasFeature  chaffey:ChaffeyDam ;
   ed:hasSensor :chafchemvm ;
   ed:hasValueConstraint [
      a ed:QuantityValueConstraint ;
      ed:hasLogicalOperator ">"^^xsd:string .
      ed:hasValue [
         a   ext:QuantityObservationValue ;
         ext:hasQuantityValue "10.0"^^xsd:double ;
         ext:hasQuantityUnitOfMeasure
                    qu:milligramsPerLitre;
      ] ;
   ];
].
```

Fig. 14. OWL description of complete example of an event constraint

5 Discussion

The proposed *Event Dashboard* facilitates the capture of user-defined constraints around observation events as machine-readable descriptions based on the domain ontology and the SSN ontology. This allows the event definition to be abstracted from the UI implementation while retaining machine-readable domain and sensor semantics. With the proposed UI, users are able to capture a range of domain specific event constraints over a sensor network. The captured constraints of event of interest can be then be used for deploying complex event queries that is compatible with the approach proposed in [19].

Table 1. Characteristics of our use of ontologies for user interfaces as defined by [13]

Characteristics	Classifications				
Domain	[✓ Real world]	[✓ IT Systems]	Users and roles		
Complexity	Informal	Low	Medium	[✓ High]	
Usage	Design time	[✓ Run time]			
Visualization	None	[✓ List]	[✓ Graphical]	[✓ Verbalized]	Source code
Interaction	None	View	[✓ View and edit]		

In Table 1, we codify our usage of ontologies in the interface according to the classification system presented by [13]. The proposed ontology-driven UI has a high level of interaction, as it allows users to view the ontology base and edit the ontology base for adding and deleting OWL individuals for event constraints. The set of ontologies used to drive the UI has a relatively high degree of complexity, according to the classification system, as we are utilizing OWL DL to capture the appropriate semantics and constraints. There are also a number of features that possess a moderate to high level of visualization as the ontology is used to populate lists, drop-boxes, tree structures and textual descriptions.

It is highlighted that most approaches to ontology-enabled UI applications use static ontologies due to lack of useful scenarios for editing ontologies, and the ability of users to edit ontologies [13]. This work demonstrates a scenario for using an application for using ontologies beyond the static scenarios. We have shown that a key aspect of the proposed UI is the ability to create and edit semantic descriptions of domain specific event constraints using ontologies, although this is currently limited to adding and deleting individuals as opposed to other kinds of ontological descriptions, e.g. class descriptions. The issue around users lacking the ability to edit ontology and preventing the use of alterable ontologies may be due to the fact that describing semantics using ontology languages, such as OWL, is difficult and beyond user groups like domain scientists and managers. That is not to say that the scenarios and use cases do not exist, as we discussed earlier. The reason could be linked to barriers in terms of suitable tools to allow these users to specify semantics using familiar interfaces, rather than the need for specifying the semantics. Our approach seeks to overcome these barriers and facilitate user-definition of semantics of event constraints by building an ontology UI library on which our ontology-driven UI is based on. A user study to comparing our approach with existing ontology editing tools is required to test this hypothesis.

An alternative approach to our ontology-driven UI is presented in [7]. They present a generalized approach for generating ontology-driven UIs for capturing descriptions of instances and related property instances. Our work differs as we specifically bind our UI to the SSN ontology and our extension module. The drawback is that as our UI is tightly coupled to the SSN ontology, and is sensitive to any significant changes to the design of the SSN ontology and our extension module. The advantage of our approach is that our UI is tailored specifically for the creating semantic descriptions of event constraints over a sensor network and thus provides a more customized look and feel to the user experience in this context.

In Sections 2 and 3, we presented an extension module to the SSN ontology, other domain extensions and a model for describing the semantics of an event constraint. The extensions and model for event constraints allow the description of domain of interest and events to be used for the respective domain users, with specific modeling around the sensors, observations, and observed properties. The methodology used to extend the SSN ontology can be used for modeling other domains so that the semantics captured are modeled have the same level of granularity, for example, modeling other domains such as soil observations. Because the UI is not bound to a specific domain, it allows the UI to be reused based on a given domain ontology.

In this work, we have explored how event constraints are defined for a simple set of observation use cases. We presented an OWL model for capturing machine-readable descriptions of event constraints. Although, capturing event constraints using OWL presents an overhead over defining queries in a complex event processing engine, it allows a level of flexibility as the semantics are abstracted. Event constraints can potentially be deployed over a wider range of complex event processing implementations, provided that the appropriate semantic mediators exist or are developed.

We chose to capture the event constraints using OWL over rule-based languages, such as RIF, SWRL and SPIN, and query languages, such as SPARQL, as they restrict further reasoning and processing that our approach allows. Using OWL to define event constraints allows us to use URIs for tying subsequent event notifications to the event rule, i.e. for provenance. It also allows for semantic optimization of translation processes between an event constraint and a complex event processing implementation, e.g. subsumed relationships between event constraints can be collated and processed together. The current UI design is not exhaustive and does not allow for additional constraints parameters to be captured, such as user-defined spatial geometries and temporal aspects. The current UI can be extended to provide the ability for users to specify windowing over a sensor output, for example, specifying a window of observations over a 24-hour period, by modeling the windowing semantics in the domain ontology. The UI can also be extended to describe more sophisticated event semantics, for example, for allowing a user to express events arising from causality or correlation by incorporating additional event semantics such as that proposed by [14]. We plan to address the above UI extensions in future work.

6 Conclusion

We have presented an ontology-driven UI to facilitate the capture of event constraints on observations in a sensor network. The proposed UI does not require users to learn query languages, sensor network APIs, or ontology languages. Rather, the UI directly uses the domain ontology to populate selection and input forms to allow users such as, domain scientists, to express event constraints over a sensor network.

We extended the SSN ontology with an extension module for specifying quantities and domain concepts as well as a simple event model for enabling the capture event constraints, such as, water quality event constraints for the Chaffey Dam. The methodology of extending the SSN ontology with domain concepts can be applied to represent other kinds of observations, such as water quantity observations. Because the UI is coupled to the SSN ontology and our extension module, it allows the UI to be unchanged for other domain extensions of the SSN ontology. We have also proposed an OWL 2 model for capturing event constraints.

Although the set of UIs presented in this paper is not exhaustive, it offers a capability for users to specify a range of events constraints over sensor observations and their properties. For future work, we propose expanding the UIs for sensor mapping and model-driven generation of sensor stream processing code through semantic mediation for deployment to sensor network middleware technologies, such as GSN.

We also propose UI and ontology extensions to allow users to represent data stream windowing and other event semantics in future work. The GSN semantic mediator requires further development for generating a broader range of virtual sensor descriptions. Lastly, we suggest the application our approach for wider range of event detection scenarios in the water and hydrology domain, such flash flood detection.

Acknowledgements. We thank our colleagues at CSIRO: Bradford Sherman for his expertise in the water quality, and Laurent Lefort for his helpful review of this work.

7 References

[1] K. Aberer, M. Hauswirth, and A. Salehi. A middleware for fast and flexible sensor network deployment. In *Proc. Intl. Conf. Very Large Data bases*, VLDB '06, pp. 1199–1202, 2006.
[2] D. Anicic, P. Fodor, S. Rudolph, and N. Stojanovic. EP-SPARQL: a unified language for event processing and stream reasoning. In *Proc. of the 20th Intl. Conf. on World Wide Web*, WWW'11, pages 635–644, NY, USA, 2011. ACM.
[3] D.F. Barbieri, D. Braga, S. Ceri, E.D. Valle, and M. Grossniklaus. Querying RDF streams with C-SPARQL. *SIGMOD Rec.*, 39(1):20–26, Sept 2010.
[4] J.P. Calbimonte, O. Corcho, and A.J. Gray. Enabling ontology-based access to streaming data sources. In *The Semantic Web (ISWC)*, *LNCS* 6496:96–111. Springer, 2010.
[5] M. Compton, P. Barnaghi, L. Bermudez, R. Garcia-Castro, et al. The SSN Ontology of the W3C Semantic Sensor Network Incubator Group. *Web Semantics: Science, Services and Agents on the World Wide Web*, 17:25–32, Dec 2012.
[6] T. Eskridge, P. Hayes, R. Hoffman, et al. Formalizing the informal: A confluence of concept mapping and the semantic web. In *Proc. 2nd Intl. Conf. Concept Mapping*, Vol.1, 2006.
[7] A. Haller, T. Groza, and F. Rosenberg. Interacting with linked data via semantically annotated widgets. In *Proc. Joint Intl. Semantic Tech. Conf. (JIST)*, December 2011.
[8] R. Hodgson and P.J. Keller. QUDT - quantities, units, dimensions and data types in OWL and XML, September 2011. http://www.qudt.org (Accessed Dec 2012).
[9] M. Horridge and S. Bechhofer. The OWL API: a Java API for working with OWL 2 ontologies. *Proc. of OWL Experiences and Directions*, 2009.
[10] O. Kwon, Y.S. Song, J.H. Kim, and K.J. Li. SCONSTREAM: A spatial context stream processing system. In *Proc. Intl. Conf. Comp.Sci. & Applications (ICCSA)*, pp. 165–170, 2010.
[11] D. Le-Phuoc, M. Dao-Tran, J. Xavier Parreira, and M. Hauswirth. A native and adaptive approach for unified processing of linked streams and linked data. In *The Semantic Web – ISWC 2011*, *LNCS* 7031:370–388. Springer Berlin Heidelberg, 2011.
[12] B. Parsia and E. Sirin. Pellet: An OWL DL reasoner. In *The Semantic Web Conference-Poster*, p.18, 2004.
[13] H. Paulheim and F. Probst. Ontology-enhanced user interfaces: A survey. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 6(2):36–59, 2010.
[14] A. Scherp, T. Franz, C. Saathoff, and S. Staab. F–a model of events based on the foundational ontology dolce+ DnS ultralight. In *Proc. Intl. Conf. Knowledge capture*, pp. 137–144. ACM, 2009.
[15] B. Sherman. The Chaffey Dam Project - nutrient supply and algal response. In *Proc. Intl. Conf. Reservoir Limnology and Water Quality*. ICARIS Ltd., 2002.
[16] A. Sheth, C. Henson, and S.S. Sahoo. Semantic sensor web. *Internet Computing, IEEE*, 12(4):78–83, 2008.
[17] K. Taylor and L. Leidinger. Ontology-driven complex event processing in heterogeneous sensor networks. In *Proc. 8th Extended Semantic Web Conference*, pages 285–299, Crete, Greece, May 2011. Springer.
[18] E.D. Valle, S. Ceri, D.F. Barbieri, D. Braga, and A. Campi. A first step towards stream reasoning. In *FIS*, pp. 72–81, 2008.
[19] J. Yu, K. Taylor, and B. Sherman. Ontology-driven complex event processing for real-time algal bloom detection. In *Proc. Aust. Ontology Workshop, CRPIT, 132:*35–36. ACS, 2011.