1st International Workshop on the Semantic Sensor Web (SemSensWeb 2009)

Collocated with ESWC 2009
Heraklion, Crete, Greece
June 1st, 2009
Informal Proceedings

Oscar Corcho
Manfred Hauswirth
Manolis Koubarakis

Preface

Millions of sensors are currently been deployed in sensor networks around the globe and are actively collecting an enormous amount of data. Together with legacy data sources, specialized software modules (e.g., modules performing mathematical modeling and simulation) and current Web 2.0 technologies such as mashups, deployed sensor networks give us the opportunity to develop unique applications in a variety of sectors (environment, agriculture, health, transportation, surveillance, public security etc.). The terms Sensor Internet, Sensor Web and Sensor Grid have recently been used to refer to the combination of sensor networks and other technologies (Web, service-oriented, Grid and database) with the view of addressing this opportunity.

Previous Sensor Internet, Sensor Web and Sensor Grid proposals make very little use of semantics (e.g., they do not use semantic annotations, metadata, ontologies etc.) and, in fact, whenever these proposals do refer to semantic concepts, they do so in an unprincipled and non-systematic way. On the contrary, the use of explicit semantics for Web and Grid resources as pioneered by many Semantic Web and Semantic Grid projects enables us to overcome the heterogeneity of data and resources, and to improve tasks like data sharing, service discovery and composition etc.

The 1st International Workshop on the Semantic Sensor Web (SemSensWeb 2009) took place in Heraklion, Crete on June 1st, 2009 in the context of the European Semantic Web Conference 2009. The goal of the workshop was to explore whether the core ideas and technologies of the Semantic Web can also be applied to sensor networks to allow the development of an open information space which we call the Semantic Sensor Web. SemSensWeb 2009 addressed, among others, the following research questions that are fundamental for the realization of the Semantic Sensor Web:

- What extensions are needed to established Semantic Web data models and languages (e.g., RDF, SPARQL, OWL etc.) so that we can deal with sensor data and meta-data? How do we model the temporal, spatial and thematic dimensions that arise in sensor networks?

- What are appropriate ontologies for describing sensors, their processes and products? What are appropriate languages and tools for semantic annotation of sensors? How can we leverage existing standards developed by the Sensor Web Enablement Working Group of the Open Geospatial Consortium such as SensorML (`http://www.opengeospatial.org/standards/sensorml`) or the W3C Geospatial Incubator Group (`http://www.w3.org/2005/Incubator/geo/`)?

- What are appropriate principles and architectures for the semantics-based integration of sensor networks? What kind of middleware is appropriate for supporting the proposed architectures?

- What are appropriate techniques and tools for semantics-based data management over heterogeneous data streams coming from autonomously deployed sensor networks? Can we apply semantic data integration techniques as we know it from database and Semantic Web research? How do these techniques interact with existing ways of processing (continuous) queries over sensor networks e.g., in-network data processing?

- How do we develop open, scalable and fault-tolerant resource discovery mechanisms for the Semantic Sensor Web? Is there a role for successful technologies such as P2P networks and publish/subscribe systems here?

- Is it possible to combine existing techniques for developing mashups with semantic technologies and sensor networks to allow the flexible and rapid development of decision support systems for target application sectors? What are appropriate high-level APIs that ease the rapid development of such mashups? Can we build on already deployed tools such as SensorMap?

- What are interesting applications of Semantic Sensor Web in target sectors such as environment, agriculture, health, transportation, surveillance, public security etc.?

Heraklion, Crete
June 1st, 2009

Oscar Corcho
Manfred Hauswirth
Manolis Koubarakis

Organization

Workshop Chairs

Oscar Corcho, Ontology Engineering Group, Universidad Politecnica de Madrid
Manfred Hauswirth, Digital Enterprise Research Institute (DERI), Galway
Manolis Koubarakis, National and Kapodistrian University of Athens, Greece

Program Committee

Michael Compton, CSIRO
Dave de Roure, University of Southampton
Alvaro Fernandes, University of Manchester
Asun Gomez-Perez, Universidad Politecnica de Madrid
Dimitrios Gunopoulos, National and Kapodistrian University of Athens
Stathes Hadjiefthymiades, National and Kapodistrian University of Athens
Vana Kalogeraki, University of California, Riverside
Werner Kuhn, University of Muenster
Srdjan Krco, Ericsson
Kirk Martinez, University of Southampton
Sebastian Michel, EPFL
Norman Paton, University of Manchester
Amit Sheth, Wright State University
Ingo Simonis, Geospatial Research & Consulting
Heiner Stuckenschmidt, University of Mannheim
Kerry Taylor, CSIRO ICT Centre
Andreas Wombacher, University of Twente
Kai-Uwe Sattler, TU Illmenau

Table of Contents

Query Processing for the Semantic Sensor Web (Invited talk)

Antonis Deligiannakis[1]

[1]Department of Electronics and Computer Engineering
Technical University of Crete, 73100 Chania, Greece
adeli@softnet.tuc.gr

The vision of the Semantic Sensor Web promises to unify the real and the virtual world by integrating sensor technologies and Semantic Web technologies. Sensors and their data will be formally described and annotated in order to facilitate the common integration, discovery and querying of information. Since this semantic information ultimately needs to be communicated by the sensors themselves, one may wonder whether existing techniques for processing, querying and modeling sensor data are still applicable under this increased load of transmitted data. In our talk we revisit several techniques for query processing in sensor networks and discuss how they can be adapted to, and used by, applications in the Semantic Sensor Web.

Sensor Networks in the Wild: Challenges and opportunities for Semantic Web technologies (Invited talk)

Dave de Roure[1]

[1]School of Electronics and Computer Science
University of Southampton, Southampton, SO17 1BJ, United Kingdom
dder@ecs.soton.ac.uk

Real sensor net deployments reveal challenges that may benefit significantly from Semantic Web solutions. As data is collected we need to record its context so that it may be properly interpreted, and when it is published we need to know its provenance so that we can understand quality and trustworthiness. With more devices delivering more data more often we need to deal with a deluge of data through automation. Often neglected, we also need to be able to configure and monitor devices and networks, and to handle diagnostics normal running may be the exceptional case. In use, data is linked, annotated and integrated. Mashups and workflows can be assembled rapidly with assistance in finding and integrating data and the services that process it. And, as the data is used in data-intensive science, we can share not just the data but the methods that are used to handle it.

Semantic-Enabled Transformation Framework for Time Series

Robert Barta[1] and Thomas Bleier[2]

[1] rho information systems
rho@devc.at
[2] ARC Austrian Research Centers GmbH
thomas.bleier@arcs.ac.at

Abstract. Conventional processing of time series is done along a split horizon: on the one hand it has to handle quantitative data organized along the time axis, on the other hand meta data capturing circumstantial facts about the values, or about the time sequence as a whole. We propose to use an integrative approach using a domain specific language for the transformation of time sequences, covering arithmetic, temporal but also semantic aspects of such computations. In that we leverage Topic Maps as one existing semantic technology.

1 Introduction

Time series over sensor observations are the predominant data structure in many life science and geospatial applications. Whereever processes are observed, observation data is sampled and recorded together with environmental information when, how and under which circumstances a measurement is taken (or computation is performed).

Despite the simple concept of a chronologically ordered sequence of values, real-world time series can have a substantial variety, both in terms of the quantitative values involved, the number of values themselves, but also to which extent meta information is used. Processing needs also vary wildly, although there are some typical computation patterns:

- *Aggregation* is the condensation of quantitative information into more abstract, qualitative values. A gliding mean over SO2 values of the past half hour, for example, will aggregate over the time axis. The mean values do not simply stand for themselves; they have to incorporate the knowledge when and over which time frame they have been computed. Furthermore, aggregation methods will depend—among many other things—on the phenomenon itself or the procedure how the sensor came about the value in the first place.
  But aggregation can also be over geographical areas, or space in general, and also over semantic axes: If CO2 is known to be an instance of the class *greenhouse gas*, then an accumulated value of all greenhouse gases can be automatically created.
- *Disaggregation* is the inverse process to estimate quantitative information along time, space and semantic axes. One example are CO2 emissions of animals in a given area. If the total sum is known and so is the ratio of animals and an average emission per cow, sheep and camel, then the emissions of all sheep in the area can be factored out. This can be automatic if cow, sheep and camels are subclasses of animals in the background ontology.

It are these patterns which sit at the core of modern environmental monitoring and forecasting systems. For auditing, but also increasingly legal reasons more and more focus shifts from the data to the *meta data*, so that whole processing chains have to reliably keep track on that, how and why particular time series data is used for a particular decision. Recent EU regulations (INSPIRE) also mandate that environmental information is properly passed on between countries and public agencies and citizens. This must include meta information.

The challenge is that quantitative, temporal, spatial and semantic information has to be brought into one consolidated computational model. In this work we propose such a domain specific language, one which operates on time sequences. It should enable to specify transformations, not only based on the numerical data, but also any semantic data available, be that inside the time sequence or within an underlying semantic network. For practical reasons the language should be compatible with both predominant semantic technology stacks, RDF [12] and Topic Maps [7], and it should also degrade gracefully in the absence of any semantic network.

Our work is to be understood in the context of SWE [13] and is specifically targeted at enhancing sensor observation services (SOS [11]). The latter are not only instrumental to expose sensor measurements via a web service; also time series derived from original sensor values can be offered by specialized SOSes (virtual sensors). The necessary meta information to describe virtual sensors in SensorML [2] is directly linked to the processing model we propose.

Our contribution is

– to choose and customize a semantic framework to seamlessly host temporal information,
– to define a time sequence transformation language, Formula 3 (F3), and
– to demonstrate how underlying ontological information can be leveraged to perform informed semantic transformations.

In that we will proceed as follows. First we focus on Topic Maps as semantic technology and recapitulate the most important concepts together with a textual notation of our making. Then we turn to the query language (subsection 2.2). Using this as baseline, we defend our choice over the more main-stream RDF/S framework later in section 6.1 (Related Work). Why the TM data model is still suboptimal for our purposes and how it can be extended we cover in subsection 2.3.

The following larger section covers the language Formula 3 (F3). In order to keep this presentation compact, we traded formal grammar rules with canonical examples from the sensor web domain. We only hint at the fact that behind F3 a (process) algebra defines the formal semantics. The extension framework (new data types, operators, kernel functions, etc.) will be covered elsewhere. Section 5 finally demonstrates how F3 meta data management can be made *semantic* with the use of an underlying semantic network and a path expression language adopted from TMQL (query language for TMs).

2 Temporal Topic Maps

Topic Maps (TM, the ISO standard defines this in plural) is a knowledge representation framework quite comparable to the more main-stream RDF/S technology stack. While in the latter all information is couched in form of triples (subject, predicate, object), basic concepts in TM are designed in a more high-level, anthropocentric way. In the following we present these concepts in lockstep with a succinct text notation (AsTMa= [8]).

2.1 Factual Information

Topics represent subjects, which can be anything, physical or not. To further knowledge aggregation, topic identity can be supported by specially interpreted IRIs. In the case of objects which reside at certain network locations such identifiers will naturally be URLs. For example, a given SOS deployment can have its endpoint be used for identification:

```
demo-sos isa SOS-deployment = http://env05.arcs.ac.at/SOSsrv/
```

In the notation above such a *subject locator* IRI is symbolized by prefixing it with `=`. The topic identifier `demo-sos` is only local within the map and can be used there to refer to that topic. If a subject does not have a network address, then one (or several) *subject identifier*s can be used for identification:

```
arcs isa organisation ~ http://www.arcs.ac.at/
```

These identifiers are meant to *indirectly identify* the subject, such as web sites for organisations, images for persons, and so forth.

As also shown in the example above, topics can have types, i.e. are instances of a class. That itself is just another topic, to be elaborated on in this map or in some peripheral ontology. Topics can also have any number of *names* attached, signalled by `!`:

```
arcs isa organisation ~ http://www.arcs.ac.at/
 ! Austrian Research Centers
 ! acronym : ARCS
 ! branding: Austrian Institute of Technology
 ...
```

Names can be typed to allow to use different names for different purposes. While the first above is just a `name`, the next is an `acronym`, the other a `branding`. These types are again topics.

To attach values to topics *occurrences* can be used. To add, say, a `homepage` or the number of employees one would add to the above

```
 ...
 homepage    : http://www.arcs.ac.at/
 nrEmployees : 1000
```

The data types here are implicit (IRI and xsd:integer, respectively), but it can be made explicit as well. The types of occurrences themselves (`homepage` and `nrEmployees`) are further topics.

Relationships between topics are expressed via associations, whereby every involved topic is a player of a certain role. The fragment

```
provisioning (provider : arcs, service  : demo-sos)
```

means that `arcs` (in the role `provider`) provisions the `demo-sos` (in the role of a `service`). Obviously the whole association itself is also of a certain type (`provisioning`). Notably, an association has no intrinsic direction. It captures a certain fact, together with all involved parties. Other examples would be `marriages`, or—to stay within the theme—`observations` and `measurements`. The roles themselves (`provider`, `service`) are also topics to be detailed somewhere to the extent necessary.

2.2 TM Query Language

Instead of using an API into a consolidated topic map, we leverage TMQL [6] as access language. Like any other query language TMQL has two concerns: (a) to locate and detect certain information in the queried topic map, and (b) generate output based on the detected information. One familiar type of output is the tabular form and it can be requested using a SELECT syntax:

```
select $p / acronym, $s =
where
  provisioning (provider: $p, service : $s)
```

A query processor will first try to find all associations which follow the pattern above, i.e. have the required association type and the given roles. Once such an association is found, the variables `$p` and `$s` will be bound to the respective players in the captured association. On the outgoing side, `$p` and `$s` will be used in the SELECT clause to evaluate *path expressions*. The expression `$p / acronym` would evaluate to all acronyms of what `$p` is currently bound to. The expression `$s =` would return all subject locators of the topic bound to `$s`. The overall result would be:

```
"ARCS", "http://env05.arcs.ac.at/SOSsrv/"
```

The query language is flexible enough to also generate XML output, not as string via text templates, but in an internal representation (DOM). For this, one has to switch into FLWR (For, Let, Where, Return) style:

```
return
<services>{
  for $p in // organisation,
      $s in // web-service
  where
     provisioning (provider : $p, service  : $s)
  return
     <service href="{$s =}">{$p / acronym}</service>
}</services>
```

While the WHERE clause remains the same, the variables and the values over which they range are made explicit. In the case of `$p` it should be all instances of `organisation` and for `$s` all instances of `web-service`. The returned content is now organized as an XML structure. The expected output would then be:

```
<services>
   <service href="http://env05...at/SOSsrv/">ARCS</service>
</services>
```

Additionally we assumed here that a `SOS-deployment` is a subclass of a `web-service`. Only then taxonometric reasoning will deliver the above result.

Inside such an XML query string it is also trivial to embed further topic map information, such as the name of the service provider:

```
<ows:ProviderName>
        {$p / acronym || $p / name}
</ows:ProviderName>
```

The example also shows how TMQL expressions can be used to deal with incomplete or highly variable data. Above, for instance, we looked first for provider `acronyms`. If there were none, the query would fall back to the full `name` for the provider (`||` is the shortcut '`or`').

Naturally TMQL supports loops over repetitive items, so it is straightforward to include, say, a list of SOS offerings:

```
return
<ows:Parameter name="offering">
  <ows:AllowedValues>{

  for $o in // offering [ . <-> location == vienna ]
  return
     <ows:Value>{$o !}</ows:Value>

  }</ows:AllowedValues>
</ows:Parameter>
```

The path expression `// offering` will compute all instances of `offering` in the map; notably not only direct ones, but also instances along any subclass hierarchy existing the map. Then the path expression continues with a filter (indicated by `[]` brackets). It only passes those things (each thing referenced with `.`) which have an association of type `location` with a topic `vienna`.

One by one, each Viennese offering is bound to the variable `$o`. With such a binding the RETURN clause is evaluated. It will extract the topic identifier (via `$o !`) and embed that into the XML fragment.

2.3 Extending the TM Model

While the generic Topic Map Model (TMDM [7]) is sufficiently equipped to host all information we need for the experiment, it does not do it elegantly, or efficiently. Rather than to shoehorn measurement data, temporal and spatial information into the model, we decided to experiment with rather minimal extensions to the official TM data structure. Naturally, these extensions will propagate to the notation and further to the query language.

The first step is to allow numerical values to have physical units, such as `5 kg` or `20 mg / m^3`. Rather than to host quantity and the value inside a topic or a dedicated association, we prefer to define a new basic data type. Accordingly, the impact on the model is minimal. Only the notation to create map content has to allow units:

```
temperature-vienna isa temperature
value: 18 celsius
```

The very same notation is extended into TMQL, also to compare values and perform computations with units.

Another modification concerns how values and topics can be related. According to the standard model literal values can only be hosted inside occurrences. To make them take part in an association one would have to create a stub topic to hold the occurrence with the value. We lift this rather arbitrary restriction and allow values also to directly take part in associations, albeit only as players. As a secondary benefit we can now interpret occurrences as specialized associations, and names as specialized occurrences.

A more dramatic model extension is needed to naturally host *time sequences*. These are the most prevalent data structure in our targetted application domain, so an effective coverage greatly affects the scalability of any semantic system, both in terms of speed and complexity.

One particular value, say, a *measurement* inside such a sequence could be captured with an association:

```
measurement (value      : 30 mg / m^3,
             phenomenon : SO2)
```

Theoretically, the time aspect can be added using a predefined role `time`:

```
measurement (value      : 30 mg / m^3,
             phenomenon : SO2,
             time       : 2009-03-07T17:01)
```

The downside of this approach is that the lack of any temporal role inside an association leaves that time aspect open to interpretation. And so does the case when more than one such role exists. This makes interpretation by query processors difficult.

Another alternative would be to use the Topic Maps scope, an already existing mechanism to restrict the *validity* of an association. But *scope* is not a very well defined concept and is used for other contexts as well.

Instead, we redefined associations to have a time stamp, one which always exists. Such strict interpretation enables query processors to perform interval algebra operations (inside, outside, overlap, ...). If the timestamp is left undefined, then the association will range over all times. As physical events are never instantaneous but interval-based, an interval length can be added to the time stamp. We do allow the interval to be positive or negative to express subtle, but ultimately important information about when the value is created and whether its validity extends into the future or the past. Of course, the interval can be zero, covering the theoretical instantaneous case.

A typical example using time stamps with negative intervals is that of the gliding mean: The time when a mean value is computed will become its time stamp. The length of the time window over which the mean was built will be pointing into the past. Alternatively, mean values can also be computed over future time windows as is the case in non-causal systems.

On the notational side, this extension is trivial; we simply allow time stamps and time intervals to be added to an association:

```
measurement (value      : 30 mg / m^3,
             phenomenon : SO2 )
      at 2009-03-07T17:09:37 - 3 hours
```

3 Formula 3

F3 is a functional language that transforms time sequences. Time sequence processors (TSP, Fig. 1) can consume any number of sequences on the incoming side.

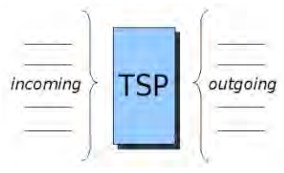


Fig. 1. Time Sequence Processor

TSPs are called a *source* if no sequence is expected. Typically these are constants or data fetched from a database backend. TSPs can produce any (finite) number of sequences on the outgoing side; *sinks* produce nothing and or are used for debugging, visualisation or again, database storage.

When a TSP is triggered into evaluation, it will consume a certain number of sequences on the incoming side. With these (and an additional variable binding to fine-control its behavior) the TSP will perform its computation. If there are still sequences left on the incoming side, then the computation will be repeated with those, continuing until the incoming side is exhausted. All partial results will be combined into one outgoing sequence of time sequences. A *greedy* TSP is one which consumes always all incoming sequences.

3.1 Time Series Abstract Data Model

While F3 makes no assumptions about the provenance of a time sequence, it has the abstract expectation that it is a linear array of chronologically ordered *slots*.

The time information within the slot is not just a time stamp (with a system specific precision). A time duration marks the temporal extension of the slot. That duration can be positive or negative, depending on whether the validity of the slot reaches into the future or the past. Slots also have a *logical time* which is the index in the sequence (starting with `0`).

The payload in the slot has the form of key/value pairs. Keys are either simple identifiers or take the form of QNames or IRIs. Values are either anything of the former or literals such as strings, integer, floats or application-specific objects such as images and matrices. They can also be time durations or time patterns.

When slots are combined into a sequence obviously their time stamp and their signed durations have to be honored. Any temporal overlaps have to be resolved to arrive at a *functional time sequence*, i.e. one which can deliver *one* slot for *one* particular time stamp.

3.2 Virtual Machine Operators

F3 defines a minimal set of primitive TSPs. As a whole they cover all possible computation patterns as all high-level language elements can be compiled into this set. Ignoring optimization, any implementation of F3 will only have to implement these operators.

- `Null`: This operator takes one sequence and creates none.
- `Nmap`: This operator takes one sequence and iterates over all slots. On each of them a lambda expression is evaluated returning a new slot. A new sequence is constructed from these slots.
- `Nreduce`: This operator takes one sequence and iterates over all slots. On each of them it will evaluate a lambda expression which aggregates the slot into an aggregate slot. That will be the only slot in the outgoing sequence.
- `Nfork`: This operator takes one sequence and evaluates a lambda expression on each slot. The result values are used for classification in that one outgoing sequence is generated for each different value, holding only those slots which produced exactly that value.
- `Ngrep`: This operator takes one sequence and evaluates a lambda expression on each slot. Slots for which that result is empty are discarded. With the others an outgoing sequence is constructed.
- `Tfork`: This operator takes one time sequence and slices it according to a time pattern and a window size. The time pattern (for instance *every 3 hours*) defines a number of time stamps, all computed relative to the start of the incoming time sequence. The time stamps are shifted along the window size, resulting in a number of individual time windows. These are used for slicing the incoming sequence into individual time sequences.
- `Tjoin`: This operator is the inverse of `Tfork`. It joins all incoming time sequences into one. Any temporal overlaps will be resolved.
- `Tee`: This is the identity operator. It echos all incoming sequences. It is used for debugging and visualisation.

3.3 Surface Syntax Operator

While applications can use an API to compose primitive TSPs for complex processing patterns, formulating transformation algorithms is more convenient using a compact syntax. Such a transformation can consist of (a) parameters for fine-control, (b) formal parameters for time sequences expected by the operator, and (c) blocks to generate values along a particular timeline.

At the beginning of a TSP definition a parameter block can define one or more modalities of that operator:

```
#-- modal parameters ----------
{ progress => 30 mins }
```

Each parameter can be associated with a default value which the invoking application can override.

This is following by a list of formal time sequence parameters. The example TSP expects two sequences, one which holds wind directions measured in degrees and another holding wind speeds:

```
#-- sequence parameters -------
@Direction { phenomenon => wind,
             unit       => degrees }
@Speed     { phenomenon => wind }
```

If a TSP is to return time sequences, then at least one block to generate values has to be declared. The first section of that handles the temporal aspect via the specification of a time pattern to be used, the second controls the quantitive aspect containing numerical computations, and the third aspect handles the meta data generation for that outgoing sequence.

```
#-- time pattern --------------
every $progress
#-- value generation ----------
< @Speed(t) if ( @Direction(t) < 30
            or   @Direction(t) > 330 ) >
#-- meta data -----------------
{ phenomenon => channel-0-speeds }
```

First the time pattern `every 30 mins` is used to compute time stamps starting from the earliest of the two incoming sequences. The time duration `30 mins` is taken from the modal parameter `progress` which is available as variable. For each of these times the `@Direction` sequence is sampled (under whatever interpolation regime that sequence is in) and that value is tested against the range `-30 .. 30`. If the direction falls within that range, the `@Speed` sequence is sampled at the same time and a value with the corresponding timestamp put into an outgoing slot. These slots are collected into an outgoing sequence. That is finally enriched by the meta information manifesting that these are speeds for a certain direction channel.

3.4 Modal Parameters

To fine-control the behavior of TSPs *modal parameters* can be declared using a simple key/value scheme. Key names are reinterpreted as variables to be used throughout the rest of a TSP definition. Values can be `undef`, any constant but also value expressions or time patterns.

The invoking application can optionally redefine the value of a modal parameter. If it does not, then the declared value will be used by default.

3.5 Time Patterns

The times at which new values have to be generated can be controlled via a time pattern language. That allows—in the simplest case—to enumerate individual times. But more general is the use of a declarative time pattern specification. That uses repeating temporal patterns, such as `every N hours` or `hourly at 12:00`. To increase the variability, time patterns can be hierarchical in that first a longer pattern is specified and within that a more fine-grained subpattern:

```
yearly:
   in May .. June : every 2nd week
   otherwise      : every 30 minutes
```

Starting with the start time of all involved time sequences, that pattern would create a yearly pattern whereby in the months May and June a time stamp will be computed every 14 days. In all other months a 30 minute rhythm will be used.

When using these patterns, then a special variable `t` is bound always to one timestamp, one at a time. Apart from generating physical times, there is also the option to use the index as logical time, such as in `every 2nd tick` to address every second time slot in the incoming sequence. If no pattern is provided the default is `every tick`. Using logical time, the variable `n` will always contain the current index, and `t` will be bound to the time in the current slot.

3.6 Sequence Parameters

Operators can use explicit sequence parameters to not only give an incoming time sequence a local name, but also to impose certain constraints on it. Only if all constraints are satisfied, evaluation will continue.

In the example

```
@Direction { phenomenon => wind,
             unit       => degrees }
@Speed     { phenomenon => wind }
```

for the first incoming sequence it is checked whether the phenomenon measured is actually wind and than in degrees. If this test passes, the sequence will be bound to a variable `@Direction`. The second incoming sequence will be bound to `@Speed` if it passes its own test.

The constraints themselves are given by key/value pairs. Only if the incoming sequence has that very key and that key links to the same value, then the constraint is satisfied. Values can be left `undef` to simply check for the existence of a certain key. In any case, the keys are again reinterpreted as variables. These can then be used throughout the rest of the TSP definition and is bound to the sequence value for that key.

The binding of incoming time sequences to sequence parameters is purely positional. Any unbound incoming sequence is left for another evaluation round of the same operator.

If no formal sequence parameter is declared, then a default one named `@_` will be used. It will always consume a single incoming sequence and it can be used implicitly within value expressions, i.e. `(t)` instead of `@_(t)` and `[n]` instead of `@_[n]`. If an operator is greedy and needs to consume all sequences, then the special `@...` must be used to indicate this. It cannot specify constraints.

3.7 Simple Value Generation

Value generation follows mostly the syntax and semantics of conventional programming languages such as FORTRAN, C or Java. This includes the notation for constant values, the usual prefix and infix operators, the general function invocations and the precedence grouping with parenthesis. An example would be

```
log ( @Speed(t) / 1000  + $speed )
```

There are, however, some language specifica. Conditionals, i.e. expressions where the evaluation depends on a condition, are not written with `if` cascades or a ternary operators, but instead use individual postfix `if` clauses:

```
 < [n]        if n.depth < -100 m
or [n] * 0.01 if n.depth <  100 m
or 0          otherwise >
```

Depending on the `depth` component different formulas will be used to generate a value. The `otherwise` is syntactic sugar as the lexical order is used for testing individual conditions.

When expressions inside a condition are evaluated, they actually do not return a `TRUE` or `FALSE` value as this data type per se does not exist in the language. Instead `undef` is used for `FALSE`, anything which is not `undef` is regarded to be `TRUE`.

3.8 Units

Life sciences being the main application domain for the language, all expressions are also aware of physical units, specifically those from the SI system. This starts with constants having units, such as `3kg` or `27.7 m/s`. But it also implies that all computations must respect units as well. In expressions such as `@Speed(t) - 100 km/h` the *physical dimensions* of all operands must match, i.e. the `@Speed` time sequence must have only values with *length per duration*.

Every expression can also be *unit-converted*. One way of conversion is to impose an additional unit onto the value of the expression. In

```
@A[n] <-< mg
```

every value would get `mg` as unit. If it already had a unit, that would be added as if the computation `@A[n] * 1 mg` had been used. In the other direction any existing unit can be relinquished:

```
@A[n] >-> m
```

The processor will convert any value with a length dimension into the number of meters, dropping the unit altogether from the value leaving a simple scalar. That mechanism can also be used to scale values. In `@A[n] >-> 1 km` the values are converted to kilometers, or even leaving the SI system with `@A[n] >-> inch` which converts into inches.

3.9 Slot Selection

When using time patterns which iterate over the incoming time sequence(s), a natural way to access a particular slot is to use an index. Intuitively, the first (and oldest) value is addressed via a subscript `[0]`, the next with `[1]` and so forth. To count from the last (and youngest) value one has to use negative values: `[-1]` retrieves the last value, `[-2]` the second-to-last, etc.

Referring to one particular value in the sequence is only moderately useful. If one needs to operate on each individual value one needs to address the current value `[n]`. Logical indices can also be used to timeshift sequences. In the same way as `[n]` always points to the current value, `[n-1]` refers to the previous, `[n-2]` refers to that before, and `[n+1]` to the next. To shift a sequence into its own future, one would write `< [n-1] >`, and to shift it into its past `< [n+2] >`.

In the case that the iteration over time sequences is based not on logical but a physical time, the current slot can be addressed via `(t)`, `t` symbolizing the current time. Similar to above a particular past or future can be referred to by providing a negative or positive duration, such as in `(t - 30 secs)` or `(t + 3 days)`.

3.10 Aggregations

Slots can also be addressed in groups using a logical range. This is only used together with aggregations, so that

```
< [n-2 .. n].sum >
```

produces sums of the last 3 values. Such ranges can be open to the left or to the right. This reflects the traditional interval notation where parentheses are used:

```
< (n-2 .. n].sum >
```

Now only the last two preceeding values are added.

Aggregation intervals can also be specified via the physical time, such as in `( t-3 hours .. t ].mean` to compute a 3 hour mean value. Again the interval can be open to either side.

The aggregation functions themselves are predefined (`mean`, `sum`, `prod`, `max`, `min`, `count`). All work type sensitive, in the sense that for the first two the plus operator for that data type is used, for `prod` the multiplication and for `max` and `min` the comparison. That list can only be extended outside the language. The same is true if the aggregation functions need a special treatment of undefined values.

If the selected interval does not contain any value, only `sum` and `count` render something defined (0), all other aggregates become undefined.

3.11 Property Management

According to the abstract data model of the language, properties are always key/value pairs. This is to ensure that properties can be easily mapped to RDF triples and Topic Map information items, such as occurrences, names and associations. Properties can also be virtually supplied by the programming environment in that pre-registered functions dynamically compute properties.

The syntax ensures that properties work equally on individual slots, whole time sequences and even sequences thereof. Some properties *inherit downwards* in that they apply to a single slot if the whole sequence has them. One example is `unit` where individual slots have to redefine this property to override any sequence-wide value. The same applies to `location`.

Slot Properties Only in simple cases a time sequence will have one single value in each slot. In general, slots will contain any number of properties, each with a key and a corresponding value, be that interpreted as data or meta data. To access a value, it has to be dereferenced via its key:

```
[n] . phenomenon
```

If such a property did not exist in that slot `undef` would be returned. That makes it simple to use as test for property existence as in `[n] if [n] . phenomenon`.

On the outgoing side, slots with their properties can be created using the following canonical syntax:

```
{
  value      => A[n],
  phenomenon => iso:SO2
}
```

Obviously one particular key can appear only once. The key must be an

identifier (or QName or IRI), the value can be specified via an arbitrary value expression. That expression is always evaluated in the current variable binding.

The key `value` is predefined and simplifies the syntax when a slot has one distinguished value, the rest being meta data. Then namely

```
< A[n] * 2 { phenomenon => iso:SO2 } >
```

can be used instead of the canonical

```
< { value      => A[n] * 2,
    phenomenon => iso:SO2 } >
```

The key `value` is also the one used by default when accessing slots within expressions. The syntax `A[n]` itself is a shortcut for the canonical `A[n].value`. Other properties of the slot have to be explicitly accessed via their corresponding keys.

Some properties are predefined as they have a special meaning in the language. For time aggregation, for instance, the `delta` property will cover the time interval over which was aggregated. If several slot values are involved in a computation, then the time interval they cover is used. Otherwise `delta` defaults to `undef`.

Also the `unit` property is handled by the language according to the computation. If an incoming value had `m/s` and is divided by a value with a time dimension, then the outgoing property for `unit` will be `m/s^2`. It is only defined if there is exactly one `value` component.

Time Sequence properties Also individual time sequences can have properties attached. Again, some have a predefined meaning, such as `start` and `end`. These, respectively, represent the start and the end time of the sequence. As every sequence is working under a particular interpolation regime, the property `interpolation` returns an identifier for that interpolation method.

On the outgoing side, time sequence properties can be added directly after the value generator:

```
@A @B < .... > { location   => vienna-stephansdom,
                 phenomenon => @A . phenomenon }
```

Some default handling here allows to reduce language noise: If the TSP is operating only on the default sequence, then all its properties are automatically propagated. Only if the sequences are explicitly declared, then the properties have to be as well.

3.12 Composite Values

In many cases the values within the properties will be numbers or strings, so that the language can directly operate on them. In general, though, values might be matrices, images or arbitrarily complex. These objects would be completely opaque to the language.

One way to handle this, would be to make the application developer write special accessor functions for these complex objects and overload the relevant arithmetic operators. But in practical cases it is much more convenient to give the language access to value components and let *it* handle the arithmetic.

Even though F3 cannot know anything about the internal structure of such objects, it can postulate a generic accessor syntax which allows to drill down into any object. In the example

```
[n] . value . columns [3] [4]
```

the dot notation is used to traverse an assumed tree structure within the `value` property. And indices are used to select by a number. Alternatively to the dot we also allow a slash `/` to insinuate a path language:

```
[n] . value / columns [3] [4]
```

At evaluation time, the language processor hands over such path expressions to the object which has to resolve the path to return a simple value.

4 Operator Algebra

To reuse operators and reduce the overall complexity, individual operators can be combined to form larger ones. One way is to pipeline them, so that the result of one operator becomes the input of the operator next in the pipeline. In the following example the incoming sequence is first incremented by one, then the results are doubled.

```
< [n] + 1 >  |  < [n] * 2 >
```

Pipelines can be extended to any number of stages, a single operator being just a trivial pipeline. If one stage produces more sequences than the next stage can consume, again the repetitive evaluation semantics from section 3 is used. Consequently, the expression

```
< [n] + 1 > < [n] - 1 >   |   < [n] * 2 >
```

is equivalent to

```
< ( [n] + 1 ) * 2 > < ( [n] - 1 ) * 2 >
```

Generators (section 3.3) can be used to stack time sequences on top of each other. But also for already existing operators it is possible to stack them. That is achieved by connecting them with `&` (or alternatively with commas):

```
< [n] + 1 > & < [n] - 1 >
```

When evaluating a stacked operator S, all incoming time sequences will be duplicated and subjected to each of the inside operators. The time sequences produced by those are then stacked on top of each other, honoring the lexical order in which the stacking was defined inside S. Consequently, the expression

```
< 2 * [n] > | < [n] + 1 > & < [n] - 1 >
```

is equivalent to the single operator

```
< 2 * [n] + 1 > < 2 * [n] - 1 >
```

As one would expect, the `&` binds stronger than the pipelining operator `|`. That precedence can be overridden by grouping inside `()` parentheses.

5 Semantic Properties

From here on it is assumed that the Formula 3 processor has access to an underlying topic map. That semantic network contains information pertinent to the application domain, in the geosemantic case generic concepts from O&M [3], SensorML but also necessary geographical information and background ontologies covering observable phenomena. Such a network may be *materialized* or it may be *virtual* where external resources are mapped dynamically into the map [1].

Depending on the needs, there are several levels of engagement with the semantic network.

5.1 Identification Regime

When testing for certain properties and when creating keys and identifiers one important aspect is a consistent identity management for any addressed subjects. Only with this quality assurance measure a robust and long-term management of data is feasible.

In this scenario a Formula 3 processor accesses the underlying topic map whenever an identifier, a QName or an IRI is used in the property handling. In the case of a simple identifier that is interpreted as topic identifier of an existing topic inside the map; when an IRI is used, then that must be one subject identifier for a topic there.

The resolution of QNames, such as `xsd:integer`, is more complex: First the QName prefix (`xsd`) is interpreted as topic identifier in the underlying map. That topic must be an instance of an `ontology` as—according to Topic Maps concepts—it has to reify a map with all the vocabulary in that corresponding namespace. That ontology is then consulted and in there a topic with the topic identifier `integer` must exist.

5.2 Path Expressions for Values

A further step is to allow TMQL path expressions everywhere where simple expressions in Formula 3 are allowed. At evaluation time these path expressions are evaluated against the underlying topic map. Any existing variable binding can be passed into the path expression. With that the following is possible:

```
< value     => [n] . amplitude,
  intensity => {   // wave-forms [ ./low  <= $value ]
                                 [ $value <= ./high ]
               } >
```

For every value in the one (anonymous) input sequence the
`amplitude` property is extracted and propagated as `value` property to the outgoing sequence. Additionally, the amplitude value is bound to the variable `$value`. The TMQL path expression is wrapped in {} brackets. It will first find all instances of `wave-forms` in the underlying map. It then will filter out those which have a high-low range inside which the `$value` lies. That remaining wave form topic will be returned in form of its topic identifier. With small modifications of the path expression also the subject identifier or topic name(s) can be requested.

Path expressions can also be used for properties of outgoing sequences as the following example demonstrates:

```
@A{ pheno => undef } # formal parameters
  < ..... >       # generating values
  { risk-level => { $pheno / risk } }
```

The `pheno` property of the incoming sequence `@A` is first bound to the variable `$pheno`. Then—when it comes to create the result properties—the phenomenon is used as starting topic to find a `risk` occurrence in the underlying map. Whatever is returned here (string, URL, ...) is embedded as value of the property.

5.3 Path Expressions as Properties

TMQL path expressions not only can provide a property value, but also implicitly define the key as well. All this depends on *what* a path expression actually returns.

In the most simplest case, a path expression can return an occurrence. That—according to the Topic Maps paradigm—consists of a type, a value (and the scope). Ignoring the scope, the type can be naturally interpreted as key and the occurrence value as the corresponding value. This is demonstrated with the following:

```
{                           # properties
  { tsunami / wikipedia }, # path expr
  { tsunami / homepage  }  # path expr
 }
```

When these properties are generated for a slot or for a whole sequence then first the `tsunami` topic is located in the underlying map. Then an occurrence of type `wikipedia` is looked for. If it exists, then the type `wikipedia` will be used as key and the WikiPedia URL as value. Similarily for the `homepage` occurrence.

What works for occurrences also works for names. Also here the name type will be used as key. The name itself is always a string and it will serve as property value. Also Topic Maps associations have such an embedding rule: Here per association role one property is generated. The role itself is used as key, the player for that role is the value.

6 Related Work

As the work here is rather architectural in nature, we touch several areas. First and foremost it is the choice of the semantic technology which impacts on the available infrastructure, feature sets and limitations. As we have chosen Topic Maps for our experiment, we will first argue this decision with some rationale. The remaining sections deal with similar processing models from which we have drawn ideas for the language F3, and with ways to extend a semantic network model with temporal information.

6.1 Topic Maps vs. RDF Rationale

While there has been some work in conceptually mediate between the RDF and the Topic Maps model ([5] [4]) the two semantic technology stacks differ in various aspects.

In contrast to RDF, TM have been designed to be *subject-centric*, rather than *resource-centric*. Accordingly, Topic Maps include a dedicated identification regime with subject locators and identifiers to control how to address resources, physical objects and abstract concepts. In that they avoid any discussion *what a URI actually means* or any need to resolve theme on the network (`httpRange-14` issue). One practical consequence thereof is that *merging* of maps is based not only on the equivalence of two node IRIs within graphs, but also whether these IRIs are used as locator or identifier. Map merging is then more robust as several such identifiers may exist for one and the same subject. The identification regime also does not make it necessary to resort to heavy-weight ontology-based mechanisms, such as `owl:equivalentClass` or `owl:sameAs`.

In terms of statements Topic Maps offer not only single-valued properties (equivalent to RDF triples with literals), but also multilateral associations involving more than two topics. Multilateral statements not only avoid the use of blank nodes, something which adds to inferencing complexity. They also allow to directly model N-ary relationships and therefore put a topic in a relationship *in that particular context* (relativistic modelling). All associations are symmetric in nature avoiding the need to keep multiple versions of a property only to constrain later explicitly in an ontology that one is the inverse of the other.

Any statement context can be further refined with the use of a *scope* (not mentioned earlier). While somewhat underdefined, it limits in a standardized way the *validity* of a statement, a feature so useful that many RDF programming frameworks offer it and that SPARQL mimicks with the GRAPH concept.

TMs have no limitations on the use of class/instance relationships. One and the same topic can be a class and an instance in the same map. While this may have theoretical implications in some reasoning scenarios, it drastically simplifies modelling of many (if not most) real-world scenarios where *sets of sets* are needed.

Like RDF, Topic Maps also allow to reify statements. The difference is that in TM only already asserted statements can be reified, staying consistent with a subject-centric approach.

In terms of the standards stack, Topic Maps use a fundamentally different layout. Instead of defining independently an ontology language (OWL) and a query language (SPARQL) directly on the model (RDF/S), Topic Maps first position the query and access language (TMQL) on top of the model (TMDM), committing hereby to closed world assumption and a particular inferencing regime. The constraint language (TMCL) is fully defined in terms of the query language; otherwise there is no ontology language for Topic Maps. With this setup the Topic Maps standards architecture limits the range of possible ontology languages, but it leads to a leaner overall model and a single point of definition for the semantics. That has a direct impact on the formal semantics and on optimization techniques when querying maps with known constraints.

There are also significant differences between the query languages:

- SPARQL only uses a pattern matching approach to detect certain node constellations in the underlying graph. TMQL offers that too, but additionally a path language to navigate to nearby corners of a map. The path expression language is powerful enough so that (almost) all queries can be expressed with it. It can also be used in SELECT clauses to further postprocess information bound to variables.
- TMQL can return customized XML content directly to be used by the invoking application, not just according to one particular standardized schema. This enables optimizations within TMQL and avoids situation where SPARQL query returns many NULL results which are eventually ignored by the application.
- As TMQL subscribes to the *closed world assumption* (CWA) it can offer a straightforward `NOT` operator within the WHERE clause. Many applications using SPARQL resort to postprocess the results.

The differences reflect that Topic Maps address rather controlled (and controllable) application scenarios, whereas RDF is more targeted to the (open) Semantic Web.

6.2 Temporal Extensions

There seem to be two schools how to embed temporal information into an existing semantic network model. The first, unobtrusive approach taken by OWL Time [9] is to define a dedicated vocabulary covering things *time events*, *durations* and *intervals* together with their relations (*contained-in*, *overlaps*, and so forth). Given an appropriate data type for the representation of dates, process information can be modeled as nodes labeled in this vocabulary.

The intrusive method is to modify the model itself. In the RDF space this has been proposed by [10]. While their background is to capture historical events and incremental changes, their line of argumentation is valid in the environmental monitoring domain and holds equally well for Topic Maps, as it does for RDF. Specifically the ability to reason over temporal aspects much more effectively than using a vocabulary approach is relevant for applications on a larger scale.

Extending the Topic Maps model by an intrinsic temporal component on associations—or in fact on any statement—we follow this approach. The variation we introduce is to also store the time interval (even together with a direction) to even better reflect the nature of our data corpus.

7 Summary

One of the driving factors for this work is to offer a consistent framework—conceptually and then in terms of programming languages—to manipulate time series of observation data. This is relevant for both, adhoc virtual sensors as well as for *from the craddle to the grave* long-term management.

Preliminary work on integrating semantic technologies into time series processing had shown that first not only concepts from the life sciences domain, but also that of the sensor web domain have to be aligned. Only then an integration of involved programming languages seemed feasible, something which also suggests global optimization opportunities.

One downside of our approach to extend a semantic technology model with temporal aspects is certainly that off-the-shelf software cannot be used. On the upside time sequences can now be hosted natively inside topic maps, make pathway for a range of other applications.

Our ontological coverage of O&M and SensorML terminology at this stage is rather incomplete and will have to be improved in further efforts. One goal is to be eventually able to mediate process descriptions between F3 and SensorML. In this step the strong identification regime of Topic Maps might prove helpful for discovery.

To evolve F3 itself a research prototype has been implemented. In that, the obvious intent was to benefit as much as possible from existing functional programming techniques and pave the way for large-scale deployments (such as in clouds).

The integration with semantic networks is also ongoing work. Still a number of ideas have not been explored yet. Path expressions, for instance, could also be used in incoming sequence properties, not just to compute values from the topic map, but also to provide the tests themselves. Also the component selector sublanguage could be mapped onto TMQL path expressions, furthering the integration and making it more obvious to host time sequence data itself inside the semantic network.

References

1. R. Barta and T. Bleier. Semantically enabled SOS with Topic Maps, 2008. FOSS4G 2008, Free and Open Source Software for GeoSpatial Conference.
2. M. Botts and A. Robin. Sensor Model Language (SensorML), Open Geospatial Consortium Inc., ogc 07-000. 2007.
3. S. Cox. Observations and Measurements, Part 1 - Observation Schema, Open Geospatial Consortium Inc., OGC 07-022r1. 2007. Open Geospatial Consortium Inc., OGC 07-022r1.
4. L. M. Garshol. Living with topic maps and rdf, 2003. Technical Report.
5. L. M. Garshol. Q: A model for topic maps: Unifying rdf and topic maps, 2005. Extreme Markup Languages 2005.
6. L. M. Garshol and R. Barta. TMQL, Topic Maps Query Language, working draft. ISO/IEC JTC1/SC34.
7. L. M. Garshol and G. Moore. ISO 13250-2: Topic Maps - data model, 2008-06-03. http://www.isotopicmaps.org/sam/sam-model/.
8. L. Heuer and R. Barta. AsTMa= 2.0 language definition. 2005. http://astma.it.bond.edu.au/astma=-spec-2.0r1.0.dbk.
9. J. R. Hobbs and F. Pan. Time ontology in owl, w3c working draft 27 september 2006, 2006. W3C.
10. T. Kauppinen, J. Väätäinen, and E. Hyvönen. Creating and using geospatial ontology time series in a semantic cultural heritage portal, 2008. ESWC 2008.
11. A. Na and M. Priest. Sensor Observation Service, Open Geospatial Consortium Inc., OGC 06-009r6. 2007.
12. O. Lassila and K. Swick. *Resource Description Framework (RDF) model and syntax specification, Technical report, W3C.* Camo AS, 1993.
13. G. Percivall and C. Reed. OGC Sensor Web Enablement Standard. *Sensors & Transducers Journal, Vol. 71, issue 9, pp.698-706*, 2006.

SEMbySEM: a Framework for Sensors Management

Jean-Sébastien Brunner, Jean-François Goudou, Patrick Gatellier, Jérôme Beck, Charles-Eric Laporte

Theresis Innovation Center - Thales Security Solutions & Services
Campus Polytechnique - 1, avenue Augustin Fresnel
91767 Palaiseau cedex - France
jean-sebastien.brunner@thalesgroup.com
jean-francois.goudou@thalesgroup.com
patrick.gatellier@thalesgroup.com
jerome.beck@thalesgroup.com
charles-eric.laporte@thalesgroup.com

Abstract. This document presents the SEMbySEM project aiming to provide a framework for universal sensors management by semantics. The entire scope from the sensors description to the End-users display is addressed, including sensors connection and events handling, system ontology, business rules design, graphical models and End-users display. Within the course of the project a new semantic standard dedicated to system management is defined according to business requirements and addressing the semantic description of managed objects as well as the means to bind the actual entities to their conceptual counterparts.

Keywords: Semantic Web Technologies, Sensor Web, Ontologies, Rules, Sensors, Internet of things.

1 Introduction

With the advent of what is commonly described as the “Internet of things”, the trend toward a world of sensors is becoming everyday more obvious as many current life objects become equipped with embedded data and communication capabilities (like RFID tags). In this “world of sensors”, the semantic sensor web is a framework aiming to provide ways to process the huge amount of data they will produce.

Our work targets the end-user point of view. From an end-user point of view, the information provided by a set of sensors is only meaningful within the scope of some end-user activity, targeting a defined goal achievable via a dedicated scenario.

The SEMbySEM project aims at defining tools and standards for the management of systems defined as coherent set of objects and grounded on a semantic abstract representation of the system to be supervised or managed.

This abstract representation has two purposes. The first one is to isolate the technical issues related to the communications with the various sensors, in what we call a Façade Layer. This Facade layer transforms the data coming from these sensors into semantic information and allows end-users to focus only on their activity while ignoring the technical details of each sensor. The second purpose is to be able to work directly on a semantic model of the system consisting of dynamically updated ontology plus related business rules (i.e. production rules). In this way, the multiple sensors data is linked to concepts of the system using a well-defined level of granularity. For instance, sensors will be grouped together if they belong to the same object, or if they are in the same location.

In order to define the ontology and the business rules a need for a new semantic representation appears, as the systems to be managed are intrinsically dynamic. A main need in the semantic model is the possible actions on real-life objects, as sensors may also be linked to actuators.

2 Related work

Sensor Web has gained interest due to hardware and communication advances (generalization of technologies such as RFID, geo-localisation, extension of internet-connected devices) and needs for standards to allow more interoperability between the various types of sensors. The Open Geospatial Consortium[1] developed a framework of standards for Sensor Web Enablement (SWE). This standardization effort enables the use of a neutral format to define the various sensors and systems, their interfaces, the type of information they convey and their communications. However, SWE standards are syntactic and do not embed logical expressivity for inference. Therefore the logic of the managed system, defining how the various sensors combine their information together to represent complex objects, needs to be embedded in the core of applications.

On another hand, Semantic Web standards, developed by the World Wide Web consortium[2], are able to represent complex knowledge, including logic associated to the data. RDF [5], as a neutral format for data representation, enables communication and storage in a neutral format. Based on this format, OWL [4] permits to define ontologies, i.e. the conceptualization of a given domain. While this format allows the definition of a model, it also enables the use of Description Logic (DL) to partly defines the behaviour of the system. For instance, Description Logic defines the notion of *Restriction*, allowing the definition of dynamic classification; instances are classified in a class as soon as they match given criteria (e.g. a given *train* is classified in the *Late Train* class as soon as it has some *Delay*).

Since DL is sometimes below the expressivity needs for real systems representations, several proposals were developed to extend it with rules in order to embed more business logic in the model itself and not spread this additional logic in software code. SWRL [6] was proposed as extensions to this model, but is felt insufficient since the expressivity of the rule and the expressivity of the DL model can

[1] OGC, http://www.opengeospatial.org/
[2] W3C, http://w3c.org/

lead to undecidability [16]. These standards also suffer from lack of skill from users who are not familiar with knowledge representation and Description Logics.

From a corporate point of view, while production rule engines are already widely spread in enterprise applications they are not yet fully integrated with semantic models. Moreover, rules suffer from heterogeneity of expressivity (Production Rule, Logic Programs) and heterogeneity of formats. Several standardization processes are on-going, such as the JSR94 standard (addressing rule interoperability at Java level) and, at a more general level, the OMG Production Rule Representation (PRR) [9] and W3C Rule Interchange Format[3] (RIF) proposal for a rule interoperability language[4].

In term of general framework for Semantic Sensor Web, different works highlight the added value of semantics, such as [1,2,3]. They propose different architectures gathering SWE, Ontology and Rules to process sensor data. These standard-based prototypes illustrate the added-value of such architecture to answer concrete use-cases. However they not address the soundness of the system, the scalability issue and the user interaction in the system.

Scalability issue mainly comes from the reasoning engine, able to apply the logic of the model. This issue comes from the complexity of the algorithms based on DL (e.g. NExpTime-complete) and of logic programming rule systems.

Regarding user interaction, these systems focuses on monitoring applications and do not allow to perform action on the underlying systems linked by sensors. Sensors can be available as Web Service, but current SSW architecture does not take into account their potential operations. In particular ontologies do not include the notion of action. In this area, Semantic Web Service attempts to add semantic metadata to the Web Services standards. Some standards such as SAWSDL[7] or OWL-S[8] propose different supports of the semantics in Web Services. The first one allows semantically annotating the service when OWL-S allows to entirely define the service using semantic concepts. In the case of OWL-S it is then possible to define the goal of the service and how to perform some processes.

Based on these assessments, we propose a framework able to go beyond the observed limitations, that is to say able (1) to provide a generic communication layer with sensors, (2) to semantically define a model and its logic to aggregate information from various sensors, (3) to allow the definition of the model of the managed system by business experts thanks to a targeted standard, (4) to deal with large scale systems, (5) to perform actions on objects connected to sensors and (6) to display a pertinent interface to End-users.

[3] W3C RIF (Rule Interchange Format) working group, http://www.w3.org/2005/rules/wiki/RIF_Working_Group

[4] There is an overlap in scope between W3C RIF and PRR. While PRR focuses on the standard metamodel definition and modeling of production rules with an XMI format, RIF focuses on a rule interchange format based on XML for web applications and also defines interactions between ontologies and rules, see [9] for more details.

3 The SEMbySEM Project

3.1 Project Overview

SEMbySEM (SErvices Management by Semantics) is a 30-months European project carried out under the EUREKA ITEA2 framework and begun July 1st, 2008. This project aims at creating a lightweight, adaptive monitoring software system dedicated to the management of systems of all sizes. The Human-Machine Interface (HMI) will be dedicated for each End-user's "business role", displaying to each End-user only the pertinent information about the monitored system.

The software core of SEMbySEM will constitute the initial contribution of an Open Source project aiming to promote the use of domain specific semantics for the management of large systems in various domains like logistics, computing and system monitoring.

Supervision software dedicated to future systems need to be easier to deploy and to maintain than the present ones, while addressing the increasing complexity of "systems of systems" and keeping an overall management capability for the users. The approach envisioned for SEMbySEM to address this issue is the extensive use of semantics in the system description allowing the active contribution of expert users for the monitoring system design and configuration.

The SEMbySEM project is based on the definition of two standards and several tools:

- A MicroConcepts standard for the semantic description of manageable objects and a standard allowing the mapping of real world Manageable Objects to MicroConcepts;
- A consistent set of tools including a common software framework comprising runtime tools and authoring software.

The targeted managed system size is between one thousand and one hundred thousand of concepts instances with ten thousands rules.

3.2 Project Limitations

The project is mainly dedicated to event-based supervision, aiming at hiding any technological issue under a semantic abstraction layer and specific HMI for each End-user. This framework is very flexible and can be extended for further applications depending on specific needs.

Some limitations will appear in the first version of the project. This one will mainly focus on ontological system representation and rules reasoning. For instance planning or workflow processing are not included in the SEMbySEM framework. The second drawback, common to all event-based systems, is that commands from End-users may not be available to sensors as they will not be connected or available at any time.

3.3 Illustrative Use-Case

An illustrative Use-Case is the management of sensors in a railways station. In a station, several sensors may exist notably for building management and security (smoke sensors, doors sensors, ...) or for the operations of the station (sensors in the engines and wagons,...) Managed objects also exist, that are linked to sensors and on which actions are also possible: escalators, lifts, cameras, live departure boards, TV screens and the station announcement system.

Sensors are accessed by End-users through a representation of managed objects and local grouping: security officers consider rooms or areas more than sensors. A train is not a physical object, it is a railways-domain concept composed of engine(s) and wagons and having its own properties (number, schedule, etc.). Therefore sensors composition and abstraction are mandatory from a business point of view.

Actions may be done on managed objects. Cameras can be rotated, doors can be closed, live departure boards are regularly modified. Therefore the Actions on managed objects are also to be considered when we design such a system. Sensors are not enough to describe this system, as only bottom-up information collection is insufficient.

Any automatic procedures that are embedded in the existing information system can be expressively described in rules. For example, when a fire alarm is triggered, the fire doors close automatically. Describing such rules in the system is interesting from a business point of view.

4 Semantics of the system

4.1 MicroConcepts, a business-driven standard for representation of objects

The definition of a semantic model able to deal with the specificity of Sensor Web is important. As mentioned earlier there are two trends to model sensor web data. First is to use OGC syntactic standards, which are specifically designed for sensors but lack for semantics, and other trend is to use Semantic Web standards such as OWL to bring semantics to the definition.

Before choosing any standard we started a bottom-up analysis of the business needs to propose a business-driven solution and eventually chose or design an appropriate standard. We firstly pointed out the need of a high level standard to allow easy system management by end-users, receiving semantic information from the Façade, itself connected to sensors. Our need was then to define the semantics used by experts compared to the needs.

Our study shown that OWL and the use of Description Logic are difficult to handle by business experts. In particular, users familiar with enterprise data management and more specifically databases are confused with the Open World Assumption[5]

[5] Definition from Wikipedia: In formal logic, the **Open World Assumption** is the assumption that the truth-value of a statement is independent of whether or not it is *known* by any single

principle. The use of Close World Assumption and Unique Name Assumption[6] enables a better adoption of this standard since it is closer to databases and more generally to enterprise data management, compared to Open-World-Assumption which targets open web environment. In this context, various OWL axioms can be transformed in DB-like constraints as proposed in [10] and experimented in [11] to ensure the consistency of the model.

Additionally, OWL expressiveness is somewhat limited to express some business needs because models are often very sophisticated. In particular qualified cardinality restrictions, property composition roles and efficient management of n-ary relationships and meta-modelling are missing compared to some real business needs. At the time of our study, OWL 2 working group published a working draft of the next OWL standard [12], extending the language by a number of new features such as qualified cardinality restrictions, property composition roles, definition of interval restriction for literals, etc. and then answering to several of our needs.

Compared to our needs, further extensions can be proposed, notably Advanced Property Composition[7] (saying for example that a property value of an instance equals the average/min/max/sum of some of the value of its components), Actions enabling acting on objects (for example "start" or "stop" a device managed by the system) and Parameters.

We then defined a business-oriented model, named MicroConcept, developed in the scope of the SEMbySEM project. This is a business-driven standard to be publicly released, and comprising a limited set of axioms. The main ones are the following:

- Ontology, as container of all objects of a given domain.
- Concept, as classifier for objects sharing some common features.
- Property (with object or literal value), defined independently from concepts and then able to be used in different classes. Property can use:
    - Domain.
    - Range.
    - Cardinality restrictions.
    - Qualified Cardinality Restrictions.
    - Properties of properties (transitive, symmetric, etc.)

observer or agent to be true. It is the opposite of the closed world assumption which holds that any statement that is not known to be true is false. […] Semantic Web languages such as RDF(S) and OWL make the open world assumption. The absence of a particular statement within the web means, in principle, that the statement has not been made explicitly yet, irrespectively of whether it would be true or not, and irrespectively of whether we believe (or would believe) that it is (or would be) true or not. In essence, from the absence of a statement alone, a deductive reasoner cannot (and must not) infer that the statement is false.

[6] Definition from Wikipedia: The **Unique Name Assumption** is a concept from ontology languages and Description Logics. In logics with the unique name assumption, different names always refer to different entities in the world. The ontology language OWL does not make this assumption, but provides explicit constructs to express that two names denote distinct entities [4].

[7] Advanced Property Composition was part of OWL 2 discussions but seems not appear in latest working drafts.

    - Default value for properties.
    - Static values (values shared by all instances of a concept).
    - Property composition (a property value is equal to the property value of a linked component).
    - Advanced property composition (similar to previous one but using mathematical functions).
- Concept and property subsumption to define inheritance.
- Instance of a concept.
- Enumeration.
- Action, defining the way to act on the real object represented by its instances.
    - Actions have input and output parameters.
- All elements contain identification (unique ID), versioning, localized name and description.

4.2 Adding rules to MicroConcepts

Rules bring added-value by avoiding spreading the business logic across company models, code and documentation. It ensures the uniqueness of the behaviour attached to semantic objects. A drawback of this approach is that the addition of a rule language on top of an ontology language (such as OWL) can lead to inconsistency because axioms of the language and rules can affect each others. Different approaches were proposed such as Semantic Web Rule Language (SWRL)[6] and Description Logic Program (DLP) [13]. SWRL extends OWL with rules in a non-native way; in the DLP approach, the intersection of Description Logic and Logic Program is used, using only a subset of DL but providing a better computability.

For higher scalability we developed the MicroConcept standard in order to be used with a production rule engine (such as JESS or DROOLS) implementing RETE [14] algorithm. The scalability of such approach was proven and enables its use in industrial environment as RETE-based algorithms are already used in many enterprises.

In order to cope with the heterogeneity of rule standards, we define rules in a neutral format linked to the MicroConcept standard. In particular, the rules are able to directly address the semantic objects of the model (concepts, instances, properties) and benefit from the logic of the model: for instance, if a rule uses a concept, the matching is done for the more general concept as well.

Integration of a RETE-based rule engine, giving good performances is then smooth.

4.3 Implementation strategy of the MicroConcept standard

Our studies enable us to design specifications and language semantics of the MicroConcept standard, based on the needs expressed by real use-cases, without limitation to existing standards. Compared, for example, to OWL 2, MicroConcept adds several axioms (notably Action and Advanced Property Composition) and moreover uses the closed-world and unique-name assumptions.

Besides these differences, we want to leverage existing standards for the implementation in order to benefit from existing design tools, API, serialization forms and repositories. We identified two strategies of implementation:

(1) Directly define MicroConcept based on MOF 2 [15] models (an OMG recommendation). Similar to OWL2, whose structural definition is based on the MOF, our language can be expressed in term of MOF meta-model, giving it a formal, computable definition.
As a result, MicroConcept is a meta-model and can be serialized in XMI format, edited with compliant editors (such as UML tools with an appropriate profile), and moreover can benefit from a powerful programmatic environment. In particular we can benefit from technologies such as model transformation implemented in the Eclipse Modeling Framework[8]. This ensures to limit specific code to the minimum and to be able to maintain the standard in the future.

(2) Define MicroConcept based on OWL 2 meta-model. In this case, our standard represents a meta-ontology which can be instantiated by business ontology taking the benefits from all the logic of the standard and from all the tools developed around this language: parsers, inference engines (e.g. Pellet[9]), programmatic environment (such as Jena[10] or OWL API[11]) and repositories.
Extensions proposed in our standards (in particular Action) are addressed by the rule engine and by a set of rules not editable by users, given a way to easily maintain the standard and be able to make some evolution. Closed-World-Assumption is addressed by the specific architecture of the core of the application (Cf. subsection 5.4).

4.4 Illustrative examples

We give here Micro-Concepts and rules for the illustrative use-case presented in section 3.3. Full specifications of these languages will be published later on the project website[12].

4.4.1 Micro-Concepts

The following Micro-Concepts are defined:

- Train
- Engine
- Wagon
- Station
- Camera

[8] http://www.eclipse.org/modeling/emf/
[9] http://clarkparsia.com/pellet
[10] http://jena.sourceforge.net/
[11] http://owlapi.sourceforge.net/
[12] http://www.sembysem.org

- Light

4.4.2 Properties

The following Properties are defined:

- **speed**: relation possessed by a Train or an Engine with a decimal value.
- **serialNumber**: relation possessed by an Engine or a Wagon with a string value.
- **trainNumber**: relation possessed by a Train with an integer value.
- **hasEngine**: relation possessed by a Train with a value that is an instance of Engine.
- **hasWagon**: relation possessed by a Train with values that are instances of Wagon.
- **inPlatform**: relation possessed by a Train with a value that is an instance of the Platform.
- **hasLight**: relation possessed by a Platform with values that are instances of Lights.
- **hasCamera**: relation possessed by a Platform with values that are instances of Camera.

4.4.3 Actions

The following Actions are defined:

- Engine has **'Start'** and **'Stop'** actions.
- Camera has a '**Focus_on_platform**' action. This action has a parameter '**to_platform**' taking an instance of '**Platform**' as parameter.
- Light has '**Switch_On**' and '**Switch_Off**' actions.

4.4.4 Rules

Rules can be defined directly on top of MicroConcepts. We give as example the expression of the rule "If a train arrives at a given platform, turn the camera to that platform and switch on all the lights on this platform". This rule used the proposed rule serialization.

```
rule "TrainInPlatform"
if
{
// If a train arrives at a given platform
?t := Train (?tPlatform := inPlatform, ?cams := one(hasCamera), ?lights :=
one(hasLight) )
}
then
{
```

```
// Then turn the camera to the given platform
?CameraFocusAction := createAction(?cams, Camera/Focus_on_platform);
?CameraFocusAction->to_platform := ?tPlatform;
execute(?CameraFocusAction);

//Then switch on the lights of this platform
excecute(?lights, Light/Switch_On);

}
```

5 SEMbySEM general architecture

Let us consider an existing set of communicating objects or elements, constituting what from now we will call indifferently a *universe* or a *managed system*. This universe will be monitored with sensors dispatched on several fixed locations and on some moving objects. The deployment and operational use of a management system for this universe will be done in two phases, design time and runtime. Design time operations will encompass the detailed definition of all the sensors which can contribute to the universe, the ontology of the universe including all the existing and required concepts related to the universe sensors and their associated business rules, and the viewpoints of each stakeholder including a display HMI. Runtime will be the operational use of the management system controlling this universe.

5.1 Design time

The design is intended to be done by expert users in the domain, assisted by ontology designers, rules designers and sensors communications designers.

Firstly, the ontology definition concerns the mandatory concepts defining and operating the universe, including first the objects that are managed and on which sensors acquire data, objects composed from several elementary objects and abstract objects that correspond to business concepts. The associated rules to permanently update the ontology are a whole part of the universe dynamic model. The ontology must also support the actions defined on the concepts and linked to actuators on the real managed objects.

The sensors definition includes all the sensors that can be encountered within the universe from an operational point of view, meaning the communication protocols to access them, the type of communication they support, the kind of message they deliver, the potential actions on the managed objects, the operational flow rate of data, an identifier to the associated concepts in the ontology, etc.

The stakeholders' viewpoints definition includes all the graphic data (icons, widgets, buttons, etc.) and the links to the related semantic data (in the ontology). These two features are grouped in several HMI models, each model containing one or several different views. Each model corresponds to a set of End-users and will present only pertinent information for this set of users.

5.2 Runtime

During runtime the dynamicity of the managed system is very important. The fixed and mobile sensors emit messages when an event occurs or when they are scheduled for it, while End-users connect and disconnect through their interfaces, act on the managed objects or on virtual objects in the semantic model. Each event from sensors is registered and processed in order to update the semantic model through direct modification and modifications triggered by the business rules. The display of the connected End-users must be updated accordingly when the semantic model updates are pertinent for them. Each action from an End-user or from rules is processed internally and sent to the right managed object when necessary.

5.3 Overall architecture

The architecture we have retained to address these issues is composed of three layers: the Façade layer, the Core layer and the Visualisation layer. The Façade layer is the interface with sensors and the Visualisation layer is the interface with End-users. The Core layer contains the semantic model.

The goal of the Façade layer is to be the interface between the sensors and the semantic model. All the technical diversity concerning protocols, communication matters, sensor types and so on is addressed in this layer. The Façade transforms heterogeneous messages and events from sensors to standardized messages addressed to one or several concepts transmitted to the Core layer. The Façade also transforms actions messages from the Core to the actuators.

The Core processes the events from the Façade in order to maintain an up-to-date semantic model of the universe. For this the arrival of a message from the Façade triggers a short process: identification of the concept instance related to the message or creation of this instance if it does not exist, consistency validation of the update with regards to the model requirements and update of the semantic model. Afterwards, the rule engine is called, taking as input the successful model changes and processing until no rule is left to trigger. The second main task of the core layer is to send the pertinent semantic data to the Visualisation layer. Each time an End-user connects to the system, the Core layer is notified of the semantic concepts instances requiring data display. Then each time these instances are updated the data is also sent to the Visualisation layer until the End-user disconnects.

The Visualisation layer aims at displaying to the End-users the pertinent information they require to perform their task. Therefore each End-user has access to tailored viewpoints, designed by expert users and HMI experts and displaying data from the semantic model. This information is continuously updated each time an event occurs. The End-users may also perform actions on the instances of the semantic model through their HMI. The Visualisation layer performs several tasks: it gets all the semantic data that is of interest for the End-user and links it to graphical components for display, according to HMI models.

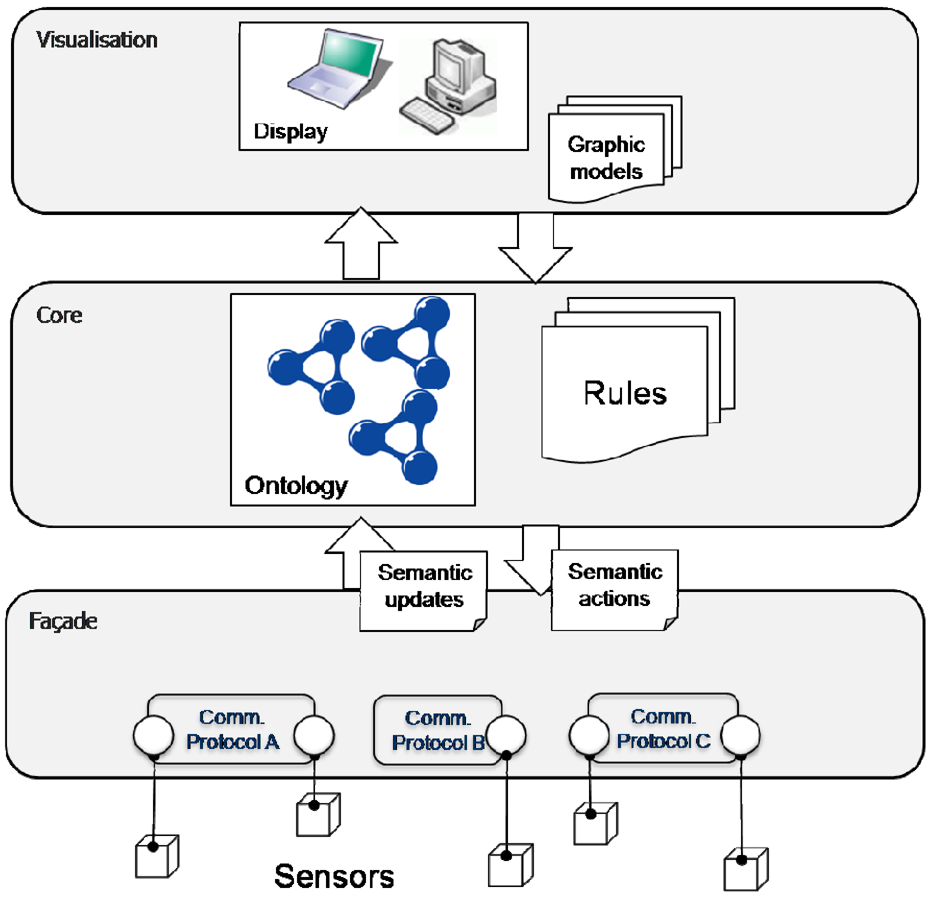


Fig. 1. Overall SEMbySEM architecture

5.4 Architecture of the semantic processing layer

In the two implementation strategies described at section 4.3, the embedded logic is not exactly the same as OWL, especially regarding the concept of Closed-World-Assumption. In this context, we use three levels to process the logic of our model.

First, a Constraint Checking module is responsible for consistency checks similarly to DB-style constraints [10]. This module ensures the consistency of the model in the Closed-World-Assumption, it is applied, in particular, on Cardinality (e.g. if a property has a *MaxCardinality* of 1 and has already one value), and Functional Property.

Secondly a reasoner is responsible to apply the general logic of the MicroConcept model. This module expands the asserted data with inferred data resulting of classification, use of property composition, symmetric, inverse property, etc.

Finally, a rule engine based on RETE algorithm, applies the additional business logic defined by the business user.

Additionally, a query engine is responsible to handle queries received from the visualization layer. It interprets the query and answer according to the logic of the model (already inferred by the 3 previously described modules since we use forward chaining inference).

The model itself benefit from the advantage of the chosen implementation strategy. In particular memory, disk representation, serialization and persistency use state-of-the-art standards to provide a powerful and maintainable solution.

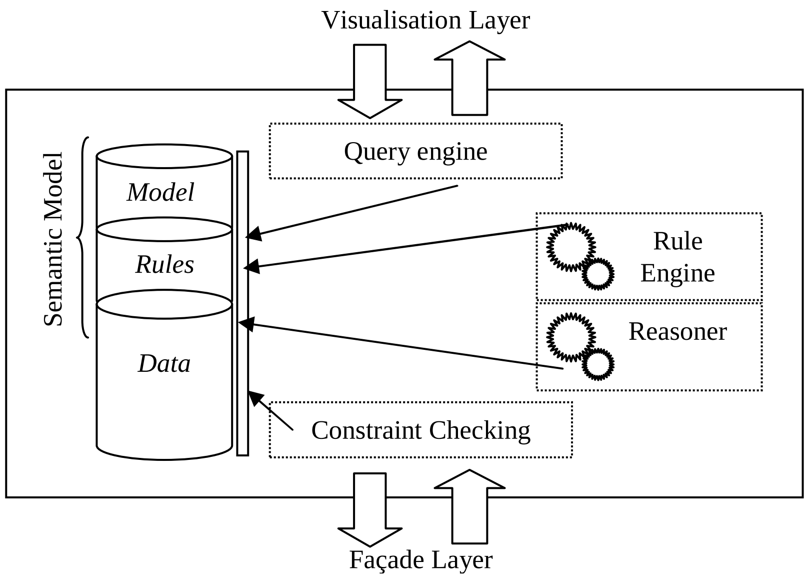


Fig. 2. Core layer general architecture

6 Current status of the project

At the time of the redaction of this paper, the SEMbySEM project is still in its first year. Architectural choices had been done as well as functional and technical specification of most parts of the framework. The MicroConcept standard was drafted and will be checked against the use-cases before release.

The project starts now its development phase. First results and evaluations are expected at the end of this year.

In order to foster SEMbySEM standard and framework, an open-source version of the framework will be released in early 2010. Standards and framework will be available on the official website of the project (http://www.sembysem.org) where additional information will be added progressively.

7 Conclusion and future work

We have presented here the whole idea of the SEMbySEM project aiming at the creation of a semantic infrastructure for service management. The main idea is to use a business-driven standard called MicroConcept to define the semantic model linked to sensors and manageable objects. MicroConcept was designed according to business needs but will be implemented with respect to state-of-the-art standards in order to provide both the expressivity required to model the use-case and the scalability to implement them. Additionally, a production rule engine supports the business logic in order to minimize specific developments.

In upstream of this core system, sensors and manageable objects low-level communications are transformed by the *Façade layer* to feed the semantic model.

In downstream, users can access to the system through a visualisation layer performing queries to the semantic model and supporting actions from users to the system.

This architecture enables a powerful framework able to answer to a large variety of use-cases. The implementation phase is starting and will helps to validate all the architecture presented in this paper.

8 Acknowledgements

This work is carried out by the EUREKA ITEA2 project SEMbySEM partly funded by French, Spanish, Finnish and Turkish governments. Furthermore we want to thank all partners and contributors to this project.

9 References

1. Sheth A., Henson C., Sahoo, S., "Semantic Sensor Web," IEEE Internet Computing, vol. 12, no. 4, pp. 78-83, July/Aug. 2008, doi:10.1109/MIC.2008.87
2. Huang, V. and Javed, M. K. 2008. Semantic Sensor Information Description and Processing. In *Proceedings of the 2008 Second international Conference on Sensor Technologies and Applications - Volume 00* (August 25 - 31, 2008). SENSORCOMM.
3. Imai, M.; Hirota, Y.; Satake, S.; Kawashima, H., "Semantic Sensor Network for Physically Grounded Applications," *Control, Automation, Robotics and Vision, 2006. ICARCV '06. 9th International Conference on* , vol., no., pp.1-6, 5-8 Dec. 2006
4. Smith M.K., Welty C. and McGuinness D.L. OWL web ontology language guide. W3C recommendation, Feb 2004
5. Brickley, D. and Guha, R.V. RDF vocabulary description language 1.0: RDF schema. W3C recommendation, Feb 2004.
6. SWRL: A Semantic Web Rule Language Combining OWL and RuleML, W3C Member Submission, 21 May 2007.

7. Farrell, J., Lausen, H.: Semantic Annotations for WSDL and XML Schema. W3C Recommendation, 2007.
8. Martin, D. et al.: OWL-S: Semantic markup for web services. W3C Member Submission, 2004.
9. OMG PRR (Production Rule Representation), Beta 1, OMG Adopted Specification, November 2007. http://www.omg.org/spec/PRR/1.0/
10. Motik, B., Horrocks, I. and Sattler, U. Bridging the gap between OWL and relational databases, *Conference on World Wide Web*, 2007.
11. Brunner, J-S., Ma, L., Wang, C., Zhang, L., Wolfson, D. C., Pan, Y., and Srinivas, K. 2007. Explorations in the use of semantic web technologies for product information management. *Conference on World Wide Web*, 2007.
12. Boris Motik, Peter F. Patel-Schneider, Bijan Parsia, OWL 2 Web Ontology Language:Structural Specification and Functional-Style Syntax. W3C Working Draft, 02 December 2008, http://www.w3.org/TR/2008/WD-owl2-syntax-20081202/. Latest version available at http://www.w3.org/TR/owl2-syntax/.
13. Grosof, B., Horrocks, I., Voltz, R. and Decker, S., Description Logic Programs: Combining Logic Programs with Description Logic. WWW, 2003.
14. Forgy, C., Rete: A Fast Algorithm for the Many Pattern/Many Object, 1980
15. Meta Object Facility (MOF) 2.0, OMG Document: formal/2006-01-01, http://www.omg.org/cgibin/doc?formal/2006-01-01
16. I. Horrocks, P.F. Patel-Schneider, S. Bechhofer, D. Tsarkov, OWL Rules: A Proposal and Prototype Implementation, J. Web Semantics 3 (2005) 23-40.

Bridging between Sensor Measurements and Symbolic Ontologies through Conceptual Spaces

Stefan Dietze, John Domingue
Knowledge Media Institute,
The Open University,
MK7 6AA, Milton Keynes, UK
{s.dietze, j.b.domingue}@open.ac.uk

Abstract. The increasing availability of sensor data through a variety of sensor-driven devices raises the need to exploit the data observed by sensors with the help of formally specified knowledge representations, such as the ones provided by the Semantic Web. In order to facilitate such a Semantic Sensor Web, the challenge is to bridge between symbolic knowledge representations and the measured data collected by sensors. In particular, one needs to map a given set of arbitrary sensor data to a particular set of symbolic knowledge representations, e.g. ontology instances. This task is particularly challenging due to the potential infinite variety of possible sensor measurements. Conceptual Spaces (CS) provide a means to represent knowledge in geometrical vector spaces in order to enable computation of similarities between knowledge entities by means of distance metrics. We propose an ontology for CS which allows to refine symbolic concepts as CS and to ground instances to so-called prototypical members described by vectors. By computing similarities in terms of spatial distances between a given set of sensor measurements and a finite set of prototypical members, the most similar instance can be identified. In that, we provide a means to bridge between the real-world as observed by sensors and symbolic representations. We also propose an initial implementation utilizing our approach for measurement-based Semantic Web Service discovery.

Keywords: Sensor Data, Conceptual Spaces, Semantic Sensor Web, Vector Spaces.

1 Introduction

Current and next generation wireless communication technologies will encourage widespread use of well-connected *sensor-driven devices* which in fact produce *sensor data* by observing and measuring real-world environments. This has already lead to standardisation efforts aiming at facilitating the so-called *Sensor Web*, such as the ones by the Sensor Web Enablement Working Group[1] of the Open Geospatial Consortium (OGC)[2]. The increasing availability of sensor data raises the need to merge such data with formally specified knowledge representations, such as the ones

[1] http://www.opengeospatial.org/projects/groups/sensorweb
[2] http://www.opengeospatial.org/

provided by *Semantic Web (SW)* standards such as OWL [22] or RDF [23]. However, whereas sensor data usually relies on measurements of perceptual characteristics to describe real-world phenomena, ontological knowledge presentations represent real-world entities through *symbols*. The symbolic approach – i.e. describing symbols by using other symbols, without a grounding in perceptual dimensions of the real world – leads to the so-called *symbol grounding* problem [2] and does not entail meaningfulness, since meaning requires both the definition of a terminology in terms of a logical structure (using symbols) and grounding of symbols to a perceptual level [2][13].

In that, in order to facilitate the vision of the *Semantic Sensor Web (SSW)* [18] the challenge is to bridge between formal symbolic knowledge representations and the measured data collected by sensors by mapping a given set of arbitrary sensor data to a particular set of symbolic representations. This task is particularly challenging due to the potential infinite variety of possible data sets.

Conceptual Spaces (CS) [8] follow a theory of describing knowledge in geometrical vector spaces which are described by so-called quality dimensions to bridge between the perceived and the symbolic world. Representing instances as vectors, i.e. *members*, in a CS provides a means to compute similarities by means of spatial distance metrics. However, several issues still have to be considered when applying CS. For instance, CS as well as sensor data provide no means to represent arbitrary relations between data sets, such as part-of relations.

In order to overcome the issues introduced above, we propose a two-fold knowledge representation approach which extends symbolic knowledge representations through a refinement based on CS. This is achieved based on an ontology which allows to refine symbolic concepts as CS and to ground instances to so-called *prototypical members*, i.e. prototypical vectors, in the CS. The resulting set of CS is formally represented as part of the ontology itself. By computing similarities in terms of spatial distances between a given set of sensor measurements and the finite set of prototypical members, the most similar instance can be identified. In that, our approach provides a means to bridge between the real-world - as measured by sensor data - and symbolic representations.

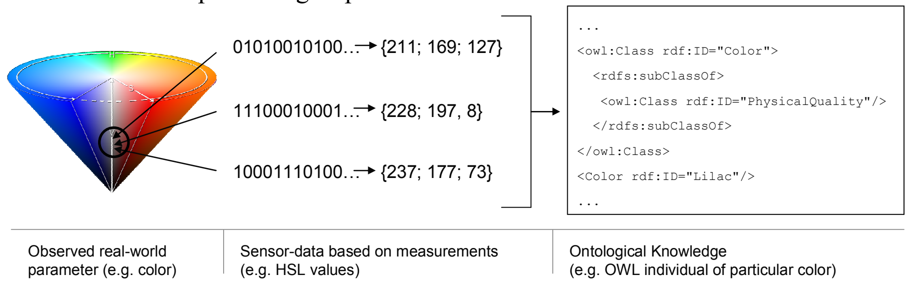
The remainder of the paper is organized as follows: Section 2 introduces the symbol grounding problem in the context of sensor data, while our representational approach based on CS is proposed in Section 3. In Section 4, we introduce an implementation of our approach based on an existing SWS reference model and we introduce first proof-of-concept prototype in Section 5. Finally, we discuss and conclude our work in Section 6.

2 Sensor Data, Symbol Grounding and Spatial Representations

This section motivates our approach by introducing the so-called symbol grounding problem in the context of the SSW and introduces some background knowledge on metric-based spatial knowledge representation.

2.1. Sensor Data and the Symbol Grounding Problem

Sensor data usually consists of measurements which describe observations of phenomena in real-world environments. In order to ensure a certain degree of interoperability between heterogeneous sensor data, recent efforts, such as the OpenGIS Observations and Measurements Encoding Standard (O&M)[3], propose a standardized approach to represent observed measurements based on a common XML schema. However, in order to provide comprehensive applications capable of reasoning in real-time on observed real-world phenomena, i.e. the contextual knowledge produced by sensor-driven devices, one needs to bridge between the measurements provided by sensors and the formally specified knowledge as, for instance, exploited by the Semantic Web [18]. Figure 1 illustrates the desired progression from observed real-world phenomena, e.g. a certain color, to measurements provided by sensors, e.g. measurements of the hue, saturation and lightness (HSL) dimensions, to symbolic knowledge entities such as a particular OWL individual representing a specific color.


01010010100... → {211; 169; 127}
11100010001... → {228; 197, 8}
10001110100... → {237; 177; 73}
Observed real-world parameter (e.g. color)
Sensor-data based on measurements (e.g. HSL values)
Ontological Knowledge (e.g. OWL individual of particular color)

```
...
<owl:Class rdf:ID="Color">
  <rdfs:subClassOf>
   <owl:Class rdf:ID="PhysicalQuality"/>
  </rdfs:subClassOf>
</owl:Class>
<Color rdf:ID="Lilac"/>
...
```

Fig. 1. Envisaged progression from real-world observations to ontological representations through sensor data.

However, whereas sensor data usually relies on measurements of perceptual characteristics to describe real-world phenomena, ontological knowledge presentations represent real-world entities through symbols what leads to a representational gap. Hence, several issues have to be taken into account. The symbolic approach – i.e. describing symbols by using other symbols, without a grounding in the real world or perceptual dimensions what is known as the *symbol grounding* problem [2] – of established SW representation standards, leads to ambiguity issues and does not entail meaningfulness, since meaning requires both the definition of a terminology in terms of a logical structure (using symbols) and grounding of symbols to a perceptual level [2][13]. Moreover, describing the complex notion of any specific real-world entity in all its facets through symbolic representation languages is a costly task and may never reach semantic meaningfulness.

Hence, in order to facilitate the vision of the SSW, the challenge is, to map a given set of sensor observation data to semantic (symbolic) instances which most appropriately represent the observed real-world entity within an ontology. In this

[3] http://www.opengeospatial.org/standards/om

respect, it is particularly obstructive that a potentially infinite amount of real-world phenomena, i.e. measurement data, needs to be mapped to a finite set of knowledge representations, e.g. ontological concepts or instances.

2.2. Exploiting Measurements through spatial Knowledge Representations

Sensor data usually consists of sets of measurements being observed from the surrounding environment. In that, spatially oriented approaches to knowledge representation which exploit metrics to describe knowledge entities naturally appear to be an obvious choice when attempting to formally represent sensor data. *Conceptual Spaces (CS)* [8] follow a theory of describing entities in terms of their quality characteristics similar to natural human cognition in order to bridge between the perceived and the symbolic world. CS foresee the representation of concepts as multidimensional geometrical *Vector Spaces* which are defined through sets of quality dimensions. Instances are supposed to be represented as vectors, i.e. particular points in a CS. For instance, a particular color may be defined as point described by vectors measuring HSL or RGB dimensions. Describing instances as points within vector spaces where each vector follows a specific metric enables the automatic calculation of their semantic similarity by means of distance metrics such as the Euclidean, Taxicab or Manhattan distance [11] or the Minkowsky Metric [19]. Hence, semantic similarity is implicit information carried within a CS representation what is perceived as one of the major contribution of the CS theory. *Soft Ontologies (SO)* [10] follow a similar approach by representing a knowledge domain *D* through a multi-dimensional *ontospace A*, which is described by its so-called *ontodimensions.* An item *I*, i.e. an instance, is represented by scaling each dimension to express its impact, presence or probability in the case of *I*. In that, a SO can be perceived as a CS where dimensions are measured exclusively on a ratio-scale.

However, several issues have to be taken into account. For instance, CS as well as SO do not provide any notion to represent any arbitrary relations [17], such as *part-of* relations which usually are represented within symbolic knowledge models. Moreover, it can be argued, that representing an entire knowledge model through a coherent CS might not be feasible, particularly when attempting to maintain the meaningfulness of the spatial distance as a similarity measure. In this regard, it is even more obstructive that the scope of a dimension is not definable, i.e. a dimension always applies to the entire CS/SO [17].

3 Grounding Ontological Concepts in Conceptual Spaces

We propose the grounding of ontologies in multiple CS in order to bridge between the measurements provided by sensor-driven devices and symbolic representations of the SW.

3.1. Approach: Spatial Groundings for Symbolic Ontologies

We claim that CS represent a particularly promising model when being applied to individual concepts instead of representing an entire ontology in a single CS. By representing instances as so-called *prototypical members* in CS, arbitrary sensor-data can be associated with specific ontology instances in terms of the closest – i.e. the most similar – prototypical member representation.

We propose a two-fold representational approach – combining SW vocabularies with corresponding representations based on CS – to enable similarity-based matchmaking between a given set of sensor data and ontological representations. In that, we consider the representation of a set of *n* concepts *C* of an ontology *O* through a set of *n* Conceptual Spaces *CS*. Instances of concepts are represented as prototypical members in the respective CS. The following Figure 2 depicts this vision:

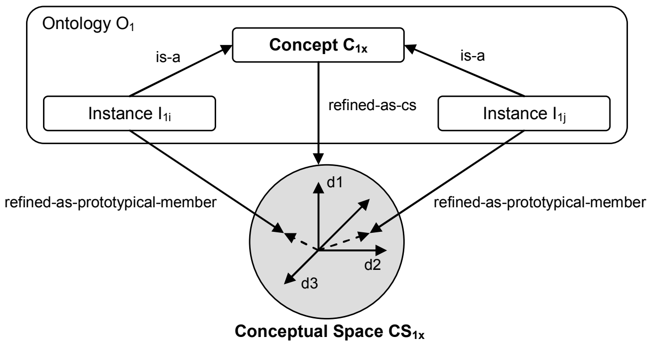
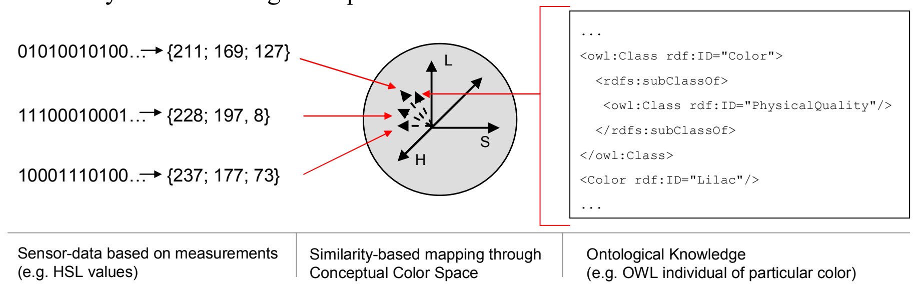


Fig. 2. Representing ontology instances through prototypical members in CS.

While benefiting from implicit similarity information within a CS, our hybrid approach allows overcoming CS-related issues by maintaining the advantages of ontology-based knowledge representations and provides a means to ground knowledge entities to cognitive dimensions based on measurements. To give a rather obvious example, a concept describing the notion of a geospatial location could be grounded to a CS described through quality dimensions such as its longitude and latitude. In previous work [3][4], we provided more comprehensive examples, even for rather qualitative notions, such as particular subjects or learning styles.

Provided our refinement of ontology concepts as CS and of instances as prototypical members, a given set of sensor data which measures the quality dimensions of a particular CS_i represents a vector *v* in CS_i which can be mapped to an appropriate ontology instance *I* in terms of the spatial distance of the prototypical member of *I* and *v*. Figure 3 illustrates the approach based on the color example introduced in Section 2.1. While measurements obtained from sensors are well-suited to be represented as vectors, i.e. members, in a CS, we facilitate similarity-based computation between a given set of sensor data and sets of prototypical members which represent ontological instances. For instance, the example in Figure 3 depicts the utilisation of a CS based on the HSL dimensions to map between color measurements obtained through sensors and prototypical members representing certain color instances. Based on the spatial distance between one measured color vector and different prototypical members, the closest vector, i.e. the most similar one, can be identified. In that, CS provide a means to bridge between observed sensor data and symbolic ontological representations.



01010010100... → {211; 169; 127}
11100010001... → {228; 197, 8}
10001110100... → {237; 177; 73}
L
S
H
Sensor-data based on measurements (e.g. HSL values)
Similarity-based mapping through Conceptual Color Space
Ontological Knowledge (e.g. OWL individual of particular color)

```
...
<owl:Class rdf:ID="Color">
  <rdfs:subClassOf>
   <owl:Class rdf:ID="PhysicalQuality"/>
  </rdfs:subClassOf>
</owl:Class>
<Color rdf:ID="Lilac"/>
...
```

Fig. 3. Similarity-based mapping between distinct sets of sensor-based color measurements and ontological color instances based on a common CS using the HSL dimensions.

3.2. A formal Ontology to represent Conceptual Spaces

In order to be able to refine and represent ontological concepts through CS, we formalised the CS model into an ontology, currently being represented through OCML [12]. Hence, a CS can simply be instantiated in order to represent a particular concept.

Referring to [16][8], we formalise a CS as a vector space defined through quality dimensions d_i of *CS*. Each dimension is associated with a certain metric scale, e.g. ratio, interval or ordinal scale. To reflect the impact of a specific quality dimension on the entire CS, we consider a prominence value *p* for each dimension. Therefore, a CS is defined by

$$CS^n = \{(p_1 d_1, p_2 d_2, ..., p_n d_n) | d_i \in CS, p_i \in \Re\}$$

where *P* is the set of real numbers. However, the usage context, purpose and domain of a particular CS strongly influence the ranking of its quality dimensions. This clearly supports our position of describing distinct CS explicitly for individual concepts. Please note that we do not distinguish between dimensions and domains [8] but enable dimensions to be detailed further in terms of subspaces. Hence, a dimension within one space may be defined through another CS by using further dimensions [16]. In this way, a CS may be composed of several subspaces and consequently, the description granularity can be refined gradually. Dimensions may be correlated. For instance, when describing an apple the quality dimension describing its sugar content may be correlated with the taste dimension. Information about correlation is expressed through axioms related to a specific quality dimension instance.

A particular (prototypical) member *M* – representing a particular instance – in the CS is described through valued dimension vectors v_i:

$$M^n = \{(v_1, v_2, ..., v_n) | v_i \in M\}$$

With respect to [16], we define the semantic similarity between two members of a space as a function of the Euclidean distance between the points representing each of the members. Hence, with respect to [16], given a CS definition *CS* and two members *V* and *U*, defined by vectors $v_0, v_1, ..., v_n$ and $u_1, u_2, ..., u_n$ within *CS*, the distance between *V* and *U* can be calculated as:

$$dist(u,v)=\sqrt{\sum_{i=1}^{n} p_i((\frac{u_i-\overline{u}}{s_u})-(\frac{v_i-\overline{v}}{s_v}))^2}$$

where $\overline{u}$ is the mean of a dataset *U* and s_u is the standard deviation from *U*. The formula above already considers the so-called Z-transformation or standardization [13] which facilitates the standardization of distinct measurement scales utilised by different quality dimensions in order to enable the calculation of distances in a multi-dimensional and multi-metric space. Please note, as mentioned in Section 2.2, different distance metrics could be applied depending on the nature and purpose of the CS.

3.3. Representing Ontologies through Conceptual Spaces

The derivation of an appropriate space CS_i to represent a particular concept C_i of a given ontology *O* is understood a non-trivial task which aims at the creation of a CS instance which most appropriately represents the real-world entity represented by C_i. We particularly foresee a transformation procedure consisting of the following steps:

S1. Representing concept properties pc_{ij} of C_i as dimensions d_{ij} of CS_i.
S2. Assignment of metrics to each quality dimension d_{ij}.
S3. Assignment of prominence values p_{ij} to each quality dimension d_{ij}.
S4. Representing instances I_{ik} of C_i as members in CS_i.

Given the formal ontological representation of the CS model (Section 3.2), we are able to simply instantiate a specific CS by applying a transformation function

$$trans: C_i \Rightarrow CS_i$$

which is aimed at instantiating all elements of a CS, such as dimensions and prominence values *(S1 – S3)*. *S1* aims at representing each concept property pc_{ij} of C_i as a particular dimension instance d_{ij} together with a corresponding prominence p_{ij} of a resulting space CS_i:

$$trans: \{(pc_{i1}, pc_{i2}, ..., pc_{in}) | pc_{ij} \in PC_i\} \Rightarrow \{(p_{i1}d_{i1}, p_{i2}d_{i2}, ..., p_{in}d_{in}) | d_{ij} \in CS_i, p_{ij} \in \Re\}$$

Please note that we particularly distinguish between data type properties and relations. While the latter represent relations between concepts, these are not represented as dimensions since such dimensions would refer to a range of concepts (instances) instead of quantified metrics, as required by *S2*. Therefore, in the case of relations, we propose to maintain the relationships represented within the original ontology *O* without representing these within the resulting CS_i. In that, the complexity of CS_i is reduced to enable the maintainability of the spatial distance as appropriate similarity measure. The assignment of metric scales to dimensions (*S2*) which naturally are described using quantitative measurements, such as size or weight, is rather straightforward. In such cases, interval scale or ratio scale, could be used, whereas otherwise, a nominal scale might be required. *S3* is aimed at assigning a prominence value p_{ij} – chosen from a predefined value range – to each dimension d_{ij}. Since the assignment of prominences to quality dimensions is of major importance for the expressiveness of the similarity measure within a CS, most probably this step requires incremental ex-post re-adjustments until a sufficient definition of a CS is achieved.

With respect to *S4*, one has to represent all instances I_{ki} of a concept C_i as member instances in the created space CS_i:

$$trans : I_{ik} \Rightarrow M_{ik}$$

This is achieved by transforming all instantiated properties pi_{ikl} of I_{ik} as valued vectors in CS_i.

$$trans : \{(pi_{ik1}, pi_{ik2}, ..., pi_{ikn}) | pi_{ikl} \in PI_{ik}\} \Rightarrow \{(v_{ik1}, v_{ik2}, ..., v_{ikn}) | v_{ikl} \in M_{ik}\}$$

Hence, given a particular CS, representing instances as members becomes just a matter of assigning specific measurements to the dimensions of the CS. In order to represent all concepts C_i of a given ontology *O*, the transformation function consisting of the steps *S1-S4* has to be repeated iteratively for all C_i which are element of *O.* The accomplishment of the proposed procedure results in a set of CS instances which each refine a particular concept together with a set of member instances which each refine a particular instance. Please note that applying the procedure proposed here requires additional effort which needs to be further investigated within future work.

4 Implementation - Exploiting Sensor Data for Semantic Web Service Discovery

In previous work [3][4], we applied our two-fold approach to Semantic Web Services (SWS) technology [6] which aims at the automated discovery, orchestration and invocation of Web services based on comprehensive semantic annotations of services. Current results of SWS research are available in terms of reference models such as OWL-S [14], SAWSDL[4] or WSMO [24]. In [3][4], our CS representation was deployed to refine instances which are part of SWS annotations in order to enable interoperability between heterogeneous SWS and SWS requests. In contrast, here we propose the utilization of our CS-based representational approach to facilitate interoperability between observations and measurements provided by sensors and symbolic SWS representations based on extensions which are described in this section.

The representational model described above had been implemented by and aligned to established SWS technologies based on WSMO [24] and the Internet Reasoning Service IRS-III [1]. Further details on the IRS-III Service Ontology and its extension through our CS formalisation can be found in [5]. However, please note that in principle the representational approach described above could be applied to any SWS reference model and is particularly well-suited to support rather light-weight approaches such as SAWSDL or WSMO Lite [21].

[4] http://www.w3.org/2002/ws/sawsdl/spec/

In order to facilitate the representational approach described in Section 3, we aligned the CS Ontology (Section 3.2) with the IRS-III Service Ontology to allow for the refinement of individual concepts – used as part of formal SWS descriptions – as formally expressed CS. In that, instances being used to represent SWS characteristics such as interfaces or capabilities can be refined as vectors.

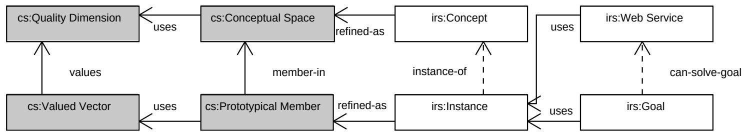


Fig. 4. Core concepts of the CS Ontology aligned to the IRS-III Service Ontology.

Figure 4 depicts the core concepts of CSO and their alignment with the IRS-III Service Ontology. Concepts (instances) as being used by IRS service or goal descriptions are refined as CS (members) within the CSO. In that, following the procedure proposed in Section 3.3, service capabilities are refined in multiple CS. To take into account the representational gap between measurement data as provided by sensors and symbolic SWS goal representations, we introduced a novel way of requesting goal achievements through IRS-III. Instead of simply invoking a goal by providing the goal achievement request SWS_i, including the actual input data, we also foresee the on-the-fly provisioning of underlying assumptions in terms of sets of measurements, i.e. vectors $\{V_1, V_2, ..., V_n\}$, which in fact describe the actual contextual environment of the request.

In order to facilitate automated similarity computation between SWS and SWS requests, we extended the matchmaking capabilities of IRS-III through a set of additional functionalities:

F1. Instantiation of member M_i in CSO for each V_i provided as part of SWS_i
F2. Similarity computation between goal request SWS_i and potentially relevant SWS

Given the ontological refinement of SWS descriptions into CS as introduced in Section 3.2 this new functionality enables to automatically achieve IRS-III goals without being restricted to complete matches between a particular goal achievement request and the available SWS. When attempting to achieve a goal, our new function is provided with the actual goal request SWS_i, named *base,* and the SWS descriptions of all *x* available services that are potentially relevant for the base – i.e. linked through a dedicated mediator:

$$SWS_i \cup \{SWS_1, SWS_2, ..., SWS_x\}$$

Each SWS contains a set of concepts $C=\{c_1..c_m\}$ and instances $I=\{i_1..i_n\}$. We first identify all members $M(SWS_i)$ – in the form of valued vectors $\{v_1..v_n\}$ refining the instance i_l of the base as proposed in Section 3.2. In addition, for each concept *c* within the base the corresponding conceptual space representations $MS=\{MS_1..MS_m\}$ are retrieved. Similarly, for each SWS_j related to the base, prototypical members $M(SWS_j)$ – which refine capabilities of SWS_j and are represented in one of the conceptual spaces $CS_1..CS_m$, – are retrieved:

$$CS \cup M(SWS_i) \cup \{M(SWS_1), M(SWS_2), ..., M(SWS_x)\}$$

Based on the above ontological descriptions, for each member v_l within $M(SWS_i)$, the Euclidean distances to any prototypical member of all $M(SWS_j)$ which is represented in the same space MS_j as v_l are computed. In case one set of prototypical members $M(SWS_j)$ contains several members in the same MS – e.g. SWS_j targets several instances of the same kind – the algorithm just considers the closest distance since the closest match determines the appropriateness for a given goal. For example, if one SWS supports several different locations, just the one which is closest to the one required by SWS_i determines the appropriateness.

Consequently, a set of x sets of distances is computed as follows $Dist(SWS_i)=\{Dist(SWS_i,SWS_1), Dist(SWS_i,SWS_2) .. Dist(SWS_i,SWS_x)\}$ where each $Dist(SWS_i,SWS_j)$ contains a set of distances $\{dist_1..dist_n\}$ where any $dist_i$ represents the distance between one particular member v_i of SWS_i and one member refining one instance of the capabilities of SWS_j. Hence, the overall similarity between the base SWS_i and any SWS_j could be defined as being reciprocal to the mean value of the individual distances between all instances of their respective capability descriptions and hence, is calculated as follows:

$$Sim(SWS_i, SWS_j) = \left(\overline{Dist(SWS_i, SWS_j)}\right)^{-1} = \left(\frac{\sum_{k=1}^{n}(dist_k)}{n}\right)^{-1}$$

Finally, a set of x similarity values – computed as described above – which each indicates the similarity between the base SWS_i and one of the x target SWS is computed:

$$\{Sim(SWS_i, SWS_1), Sim(SWS_i, SWS_2), .., Sim(SWS_i, SWS_x)\}$$

As a result, the most similar SWS_j, i.e. the closest associated SWS, can be selected and invoked. In order to ensure a certain degree of overlap between the actual request and the invoked functionality, we also defined a *threshold similarity value T* which determines the similarity threshold for any potential invocation.

5 Application: Measurement-based SWS discovery of Weather Forecast Web Services

Our measurement-based SWS discovery approach (Section 4) was actualised within an initial proof-of-concept prototype application which mediates between different weather forecast Web services. This example use case illustrates how measurements can be dynamically mapped to symbolic representations, SWS in this case, by means of similarity-computation within CS.

Here, SWS_1, SWS_2 and SWS_3 provide weather forecast information for different locations. Each service has distinct constraints, and thus distinct SWS descriptions. In detail, SWS_1 is able to provide forecasts for France and Spain while SWS_2 and SWS_3 are providing forecasts for the United Kingdom. All services show different Quality of Service (QoS) parameters. Three distinct service ontologies O_1, O_2, and O_3 had been created, each defining the capability of the respective service by using distinct vocabularies. For example, SWS_2 considers concepts representing the notions of location and QoS together with corresponding instances (see also Table 1):

$$\{(country, QoS), (UK, QoS2)\} \subset O_2 \subset SWS_2$$

By applying the representational approach proposed in Section 3, each concept of the involved heterogeneous SWS representations had been refined as a shared CS, while instances - defining the capabilities of available SWS - were defined as prototypical members. For example, a simplified CS (*CS_1: Location Space* in Figure 5) was utilized to refine geographical notions (e.g. country) by using two dimensions indicating the geospatial position of the location:

$$\{(p_1 l_1, p_2 l_2)\} = \{(latitude, longitude)\} = CS_1$$

The two dimensions latitude and longitude are equally ranked, and hence, a prominence value of 1 has been applied to each dimension. Note that each of the depicted concepts and instances, such as *O_2:UK* and *O_3:UK*, are distinct and independent from each other, and thus might show heterogeneities, such as distinct labels, for instance *United Kingdom* and *Great Britain*. In the case of *O_2:UK* and *O_3:UK,* these two instances are refined by two distinct prototypical members: $L_1(SWS_2) = \{(v_1 = 55.378051, v_2 = \text{-}3.435973) | v_i \in CS_1\}$ and $L_1(SWS_3) = \{(v_1 = 55.378048, v_2 = \text{-}3.435963) | v_i \in CS_1\}$. Each member has been defined by different individuals applying similar, but non-equivalent geodata.

In addition, a second space (*CS_2: QoS Space* in Figure 5) has been defined by three dimensions – latency (in ms), throughput (number of Web services), availability (in %): $\{(p_1 r_1, p_2 r_2, p_3 r_3)\} = \{(latency, throughput, availability)\} = CS_2$

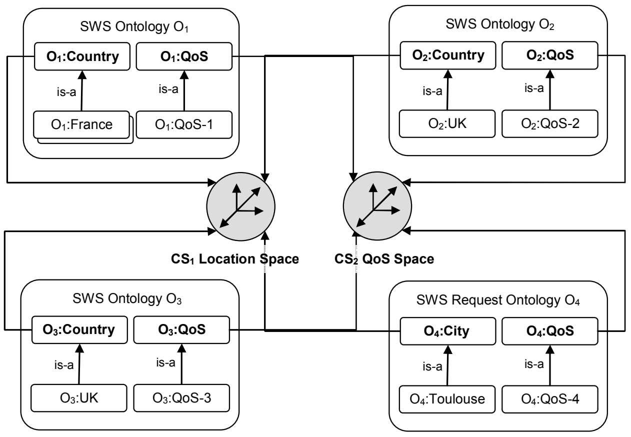


Fig. 5. Grounding assumptions of distinct weather forecast SWS to common CS.

Potential service consumers define a goal (e.g. *SWS_4* in Figure 5) together with the set of input parameters and the underlying assumptions in terms of measurements. After accomplishment of *F.1*, i.e. the dynamic instantiation of members in their corresponding CS to represent the sensor data provided with the actual goal request *SWS_4*, all involved goals and SWS were grounded in the same set of CS as depicted in Figure 5.

In that, assumptions of available SWS had been described independently in terms of simple conjunctions of instances which were individually refined in shared CS as shown in Table 1. As shown in Table 1, the request *SWS_4* assumes a SWS which provides weather forecast for the location UK (*$L_1(SWS_4)$*) and ideal QoS (*$Q_1(SWS_4)$*) demanding zero latency but high throughput and availability.

Table 1. Assumptions of involved SWS and SWS requests described in terms of vectors in *MS_1* and *MS_2*.

	Assumption $Ass_{SWSi} = (L_{1SWSi} \cup L_{2SWSi} \cup .. \cup L_{nSWSi}) \cup (Q_{1SWSi} \cup Q_{2SWSi} \cup .. \cup Q_{mSWSi})$	
	Members L_i in CS_1 (locations)	Members C_j in CS_2 (QoS)
SWS_1	$L_{1(SWS1)}$={(46.227644, 2.213755)} $L_{2(SWS1)}$={(40.463667, -3.74922)}	$Q_{1(SWS1)}$={(155, 2, 91)}
SWS_2	$L_{1(SWS2)}$={(55.378051, -3.435973)}	$Q_{1(SWS2)}$={(15, 50, 98)}
SWS_3	$L_{1(SWS3)}$={(55.378048, -3.435963)}	$Q_{1(SWS3)}$={(78, 5, 95)}
SWS_4	$L_{1(SWS4}$={(55.378048, -3.435963)}	$Q_{1(SWS4)}$={(0,100,100)}

Though no exact SWS matches these criteria, at runtime similarities are calculated between *SWS_4* and the related SWS (*SWS_1*, *SWS_2*, *SWS_3*) through the similarity-based discovery function described in Section 4. This led to the calculation of the following similarity values:

Table 2. Automatically computed similarities between SWS request *SWS_4* and available SWS.

	Similarities
SWS_1	0.010290349
SWS_2	0.038284954
SWS_3	0.016257476

Given these similarities, our introduced goal achievement method automatically selects the most similar SWS (i.e. *SWS_2* in the example above) and triggers its invocation.

6 Discussion and Conclusions

In order to contribute to the vision of the SSW, i.e. the convergence of sensor data and formal knowledge representations as part of the Semantic Web, we proposed a representational model which grounds ontological representations in CS to overcome the symbol grounding problem. The latter is perceived to be as one of the major obstacles towards the SSW. While ontological instances are represented as prototypical members within a CS, arbitrary sensor data which measures the dimensions of the CS can be associated with the most appropriate instance by identifying the most similar, i.e. the closest, prototypical member to the vector which represents the sensor data. Our approach is facilitated through a dedicated CS Ontology which allows to refine any arbitrary concept (instance) as CS (prototypical member). In that, our representational model allows to bridge between sensor measurements and symbolic knowledge representations by means of similarity computation between vectors within CS.

In addition, we implemented our approach by applying it to the field of SWS and utilising it for measurement-based SWS discovery while bridging between symbolic SWS representations and sensor-based measurement data. Therefore, we extended the matchmaking algorithm of an existing SWS Broker, IRS-III, with new capabilities allowing for measurement-based matchmaking based on our two-fold representational model. A first proof-of-concept prototype application utilises our approach to enable measurement-based discovery of weather forecast Web services based on measured parameters such as the geospatial location and the service QoS.

The proposed approach has the potential to further support interoperability between heterogeneous sensor data and symbolic knowledge representations. While our approach supports automatic mapping between ontology instances and sensor-based measurements it still requires a common agreement on shared CS. In addition, incomplete similarities are computable between partially overlapping CS.

However, the authors are aware that our approach requires considerable effort to establish CS-based representations. Future work has to investigate on this effort in order to further evaluate the potential contribution of the proposed approach. Moreover, while overcoming issues introduced in Section 2, further issues remain. For example, whereas defining instances, i.e. vectors, within a given CS appears to be a straightforward process of assigning specific quantitative values to quality dimensions, the definition of the CS itself is not trivial. Nevertheless, distance calculation relies on the fact that resources are described in equivalent geometrical spaces. However, particularly with respect to the latter, traditional ontology and schema matching methods could be applied to align heterogeneous spaces. In addition, we would like to point out that the increasing usage of upper level ontologies, such as DOLCE [9] or SUMO [15], and emergence of common schemas for sensor data such as the OpenGIS Observations and Measurements Encoding Standard, leads to an increased sharing of ontologies at the concept level. As a result, our proposed hybrid representational model becomes increasingly applicable by further contributing to the vision of the SSW.

7 References

[1] Cabral, L., Domingue, J., Galizia, S., Gugliotta, A., Norton, B., Tanasescu, V., Pedrinaci, C.: IRS-III: A Broker for Semantic Web Services based Applications. In proceedings of the 5th International Semantic Web Conference (ISWC 2006), Athens, USA (2006).

[2] Cregan, A. (2007), Symbol Grounding for the Semantic Web. 4th European Semantic Web Conference 2007, Innsbruck, Austria.

[3] Dietze, S., Gugliotta, A., Domingue, J., (2008) Conceptual Situation Spaces for Situation-Driven Processes. 5th European Semantic Web Conference (ESWC), Tenerife, Spain.

[4] Dietze, S., Gugliotta, A., Domingue, J., (2008) Bridging the Gap between Mobile Application Contexts and Semantic Web Resources. Chapter in: Context-Aware Mobile and Ubiquitous Computing for Enhanced Usability: Adaptive Technologies and Applications, Editor: Dragan Stojanovic, Information Science Publishing (IGI Global), November 2008.

[5] Dietze, S., and Domingue, J. (2009) Enriching Service Semantics through Conceptual Vector Spaces, Workshop: Workshop on Ontology, Conceptualization and Epistemology for Information Systems, Software Engineering and Service Science (ONTOSE'09) at The 21st International Conference on Advanced Information Systems (CAiSE'09), Amsterdam, NL

[6] Fensel, D., Lausen, H., Polleres, A., de Bruijn, J., Stollberg, M., Roman, D., Domingue, J. (2006): Enabling Semantic Web Services – The Web service Modelling Ontology, Springer 2006.

[7] Fielding, R.T.: Architectural Styles and the Design of Network-based Software Architectures. PhD thesis, University of California, Irvine (2000)

[8] Gärdenfors, P. (2000), Conceptual Spaces - The Geometry of Thought. MIT Press, 2000.

[9] Gangemi, A., Guarino, N., Masolo, C., Oltramari, A., Schneider, L.(2002), Sweetening Ontologies with DOLCE. In: A. Gómez-Pérez , V. Richard Benjamins (Eds.) Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web: 13th International Conference, EKAW 2002, Siguenza, Spain, October 1-4, 2002.

[10] Kaipainen, M., Normak, P., Niglas, K., Kippar, J., Laanpere, M., Soft Ontologies, spatial Representations and multi-perspective Explorability, Expert Systems, November 2008, Vol. 25, No. 5.

[11] Krause, E. F. (1987). Taxicab Geometry. Dover.

[12] Motta, E. (1998). An Overview of the OCML Modelling Language, the 8th Workshop on Methods and Languages, 1998.

[13] Nosofsky, R. (1992), Similarity, scaling and cognitive process models, Annual Review of Psychology 43, pp. 25- 53, (1992).

[14] OWL-S 1.0 Release. http://www.daml.org/services/owl-s/1.0/.

[15] Pease, A., Niles, I., Li, J.(2002), The suggested upper merged ontology: A large ontology for the semanticweb and its applications. In: AAAI-2002Workshop on Ontologies and the Semantic Web. Working Notes (2002).

[16] Raubal, M. (2004). Formalizing Conceptual Spaces. in: A. Varzi and L. Vieu (Eds.), Formal Ontology in Information Systems, Proceedings of the Third International Conference (FOIS 2004).Frontiers in Artificial Intelligence and Applications 114, pp. 153-164, IOS Press, Amsterdam, NL.

[17] Schwering, A. (2005). Hybrid Model for Semantic Similarity Measurement, in R. Meersman and Z. Tari (Eds.): CoopIS/DOA/ODBASE 2005, LNCS 3761, pp. 1449 – 1465, 2005..

[18] Sheth, A., Henson, C., Sahoo, S., Semantic Sensor Web, IEEE Internet Computing, July/August 2008, p. 78-83.

[19] Spencer, B., Liu, S. Inferring data transformation rules to integrate semantic web services. In S. A. McIlraith, D. Plexousakis, and F. van Harmelen, editors, Int'l Semantic Web Conference, volume 3298 of Lecture Notes in Computer Science, pages 456–470. Springer, 2004.

[20] Suppes, P., D. M. Krantz, et al. (1989). Foundations of Measurement - Geometrical, Threshold, and Probabilistic Representations. San Diego, California, USA, Academic Press, Inc.

[21] Vitvar, T., Kopecký, J., Viskova, J., Fensel, D. WSMO-Lite Annotations for Web Services. ESWC 2008: 674-689.

[22] World Wide Web Consortium, W3C (2004a): Resource Description Framework, W3C Recommendation 10 February 2004, http://www.w3.org/RDF/.

[23] World Wide Web Consortium, W3C (2004b): Web Ontology Language Reference, W3C Recommendation 10 February 2004, http://www.w3.org/TR/owl-ref/.

[24] WSMO Working Group, D2v1.0: Web service Modeling Ontology (WSMO). WSMO Working Draft, (2004). (http://www.wsmo.org/2004/d2/v1.0/).

Towards a Common Event Model for an Integrated Sensor Information System.

Chris Fowler[1] and Behrang Qasemizadeh[2]

[1] ECIT Institute, Queen's University Belfast, Northern Ireland Science Park, Queen's Road, Belfast, UK.
[2] DERI, Unit of Natural Language Processing, National University of Ireland, Galway IDA Business Park, Lower Dangan, Galway, Ireland.

Email: c.fowler@qub.ac.uk, behrang.qasemizadeh@deri.org

Abstract. This paper describes steps towards a common event model for event representation in sensor information systems. The model builds on a representation of events, and introduces the idea of *semantic-role* from linguistics, attaching semantic annotations to the underlying concepts of formal event representation. We describe how semantic-role annotations facilitate linkages across the descriptive layers drawn from sensor data ontologies, allowing us to reason more effectively about *causality* within a sensed environment. An overview of the parent system for which the event model was derived is given to provide application context for the work. This is followed by a detailed description of the model, together with a description of how the model is used within a prototype system that illustrates the model in practice. The paper ends with conclusions and issues for further work.

1 Introduction

The vision of a *semantic reality*, as described by Manfred Hauswirth in 2007 [5], posits a world where sensor technology and the semantic web combine to enable a single unified view that bridges the gap between virtual and physical space. The result would be a machine readable semantic layer, rooted in an ontological domain-description, making possible a machine navigable information-web that mirrors reality. The use of ontologies provides a vocabulary and classification mechanism through which specific domains may be described. These descriptions are encoded in meta-data used to annotate the data gathered from the sensor-web. The addition of a semantic layer enables the possibility to reason about and understand the relationships between the '*things*' described in the ontology-based annotations. The possible benefits and applications for this machine-readable real-time virtual lens on the world are numerous: health-care provision, security and crime prevention, traffic management, wild-life preservation, environmental monitoring, are only a few such examples. Of course, the potential harmful uses are equally present, but we should not let this prevent us from exploring the issues and challenges towards this new technology.

The true semantic reality is some way off. If we consider, however, that the semantic reality envisioned could, if viewed from a different perspective, be described as a collection of intersecting *semantic sensor webs* [10], then we are closer than we think.

Taking Sheth and Henson [10] as a datum, a semantic sensor-web is seen as a network of remote, autonomous sensors, which detect changes (events) in the environment and make these data available as an information source on the Web. These source data may then be used for information-fusion towards a higher-level understanding of the sensed environment; for example, weather tracking, flood monitoring, avalanche prediction, or crime prevention. Each sensor-web may include a number of different modalities. The addition of a semantic layer allows for a richer interpretation of the source data.

The Integrated Sensor Information System project (ISIS [1]) maybe viewed as an example of a semantic sensor-web. Its application is crime prevention on public transport. Its fundamental structure is a distributed sensor-web, with both remote and central semantic-based analytics capability, designed to fuse sensor data towards an understanding of the environment under surveillance from a security and crime prevention perspective - so called *situation awareness*. ISIS is designed to: assert threat levels on public transport using embedded sensor-array nodes positioned on-board buses as they traverse the transport network; inform key decision makers of changes in threat-level via a control room interface; and manage its own network. It is an example of applying semantic senor technology in a real-world domain.

A key aspect of ISIS is the use of multi-modal sensors (video cameras, microphones and radio-frequency sensors). Because we are using different modes of sensor, each type will '*speak*' a different language. To make sense of this, we must unify the differences towards a *common language*, in order that the system as a whole may be mutually understood. This requirement is not unique to ISIS and the common approach is to use ontologies to markup identifiable objects and events as they occur in the data. (We have defined a set of ontologies for this purpose). In addition to marking-up objects and events, we also want to be able to reason about the causality of events and their relationship to the objects involved. We have achieved this by introducing the notion of *semantic-role* from linguistics.

The introduction of semantic-role allows us to define intra-ontology-relations as a common platform for event modelling and causality inferencing. The semantic-role-relations provide us with the logical linkages we need between the different elements of the data-model (see figure 1). They allow for a formal interpretation of the different relationships between the informative elements defined in different classes of ontology. Consider this example, "a *man approaches* the *chair*": in this case we would assign the semantic-role *agent* and *goal* to the man and chair respectively. (The vocabulary for describing the *goal* and *agent* are taken from the domain ontologies, as are the *event* descriptions). This structure (*agent-event-goal*) may, at a given time, and within the rules of the ontology governing the event, be evaluated using inference rules, and said to be a true/false statement. The concepts within each class of ontology describing a specific object-, or event-class, may now be related by assigning the semantic-role relation to the participant objects of an event. This allows us to reason about events and search for specific events involving specific objects identities. The semantic-role relations between different object and event ontologies in our model, therefore, link the objects in a scene to the events in a scene across different sensor-modalities. In this way they act as a semantic-bridge, linking the knowledge-base of the domain. Reasoning-agents are then able to navigate the information-space via formally defined semantic relationships. The links between the descriptive layers that focus on the objects and properties, and the layers describing the events within a scene, can be now be used to determine *who did what.* This is not sufficient for our purpose; we also need to know the *when.*

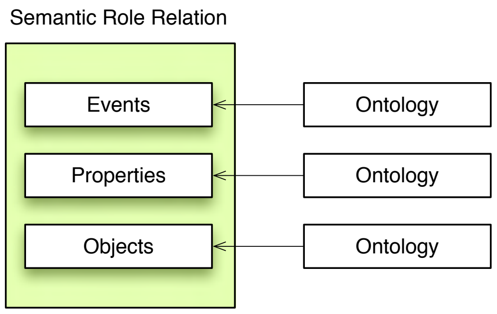


Fig. 1: Semantic Role Relations.

Events do not happen without *time.* We must therefore include time in our modelling process. Events happen over time intervals, therefore, synchronised data streams are needed to determine unique object-event-relations as viewed from different sensors.

In ISIS, sensors, together with their associated analytics, identify objects in the sensed data from multiple sources. These data need to be unified at a particular time instance across these multiple sources. For example, a noise heard by an audio sensor may be matched in time with a video data of the same scene that shows a person dropping a plate; we may then say that these two sensors sensed the same event in time; and say: who dropped the plate! Another key addition to time is the notion of *space.* An array of sensors, fixed to survey a bounded environment, are, by their physical location, co-located in space. If their sensor-view is aligned in both time and space, then we are able to infer that they are 'watching' the same scene.

Additional contextual information can also be used to unify a scene. Consider this example: if a distance relationship is known between two finger-print readers, and at two different time-instances the same finger-print scan is read at each, we can infer that in the time-interval between readings the person (or at least their finger) moved from the location of one reader to the location of the other. This example begins to show the rich set of information that comes together to create understanding, and shows the links between objects, events, sensor activity, physical space, time, and causality.

To begin the process of building a machine readable semantic-skin over the sensor data captured by the ISIS system, we must define the fundamental elements of our data-model. It must be capable of capturing the objects, properties, events, time and space, as well as the semantic-role-relationship between the different ontology conceptualizations. To do this we adopt the same principals outlined by Westermann and Jain [11] for a *Common Event Model*.

This paper describes our interpretation of Westermann's model towards a common event model for ISIS. Our model may be seen as the rich descriptive *skin* that wraps different layers of abstraction within multiple sensor data sources. At its core are the unique objects in the scene, captured at each time instant, with additional layers that describe atomic events, low-level events, higher-level events and ultimately domain-specific behaviors, each occurring over increasing time intervals. By linking ontologies across each layer using the notion of a semantic-role-relation, our model allows for greater understanding of causality within the sensor data towards an integrated sensor information system. Our work uses the Video Event Representation Language (VERL) and the Video Event Markup Language (VEML) presented by Fraccois et al [4] and the theory of *Causality* from Hobbs [6] as its base.

This paper is structured as follows: section 2 presents an overview of the ISIS system that places our work in context. Section 3 gives a full description of the Common Event Model for ISIS, together with database representation, example event annotations. A prototype system developed to test and explore issues with the work is described in Section 4, with details of how our model integrates the elements of the system. Conclusions and further work are presented in section 5.

2 ISIS System Overview

In this section we present a high-level view of the ISIS system. Although the application for our system is crime prevention, we believe that by making a separation between the sensor hardware infrastructure and the language layers through which the system represents and interprets the sensed environment, ISIS can be applied to many different domains.

Figure 2 shows a high-level view of the ISIS system. Four key elements exist:

1. *A remote sensor-array node*. In our application this is located on-board a bus traversing the transport network. Its main function is to sense the scene and detect in the data the profile and mix of passengers on-board, and infer any domain specific behaviors relating to security and crime detection. This real-time risk inferencing contributes to an on-board risk level. Once the risk

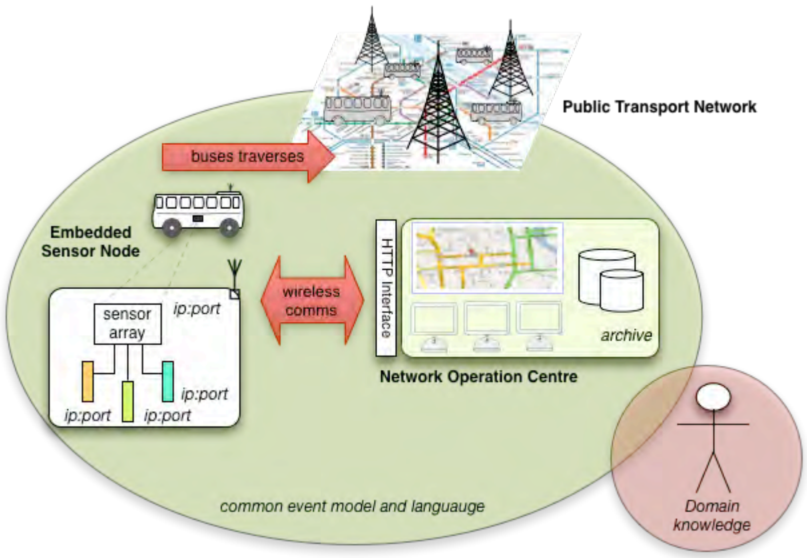


Fig. 2: Separation of Concerns in the ISIS Infrastructure

level rises above a certain threshold, an alert message is communicated to a human-operator in a network control centre.

2. *A wireless communications infrastructure*. Vehicles traversing the transport system communicate via message exchange with a control room. This is done over a wireless network.
3. *Network Control Centre*. The control centre is manned by human operators - domain experts. It has two main functions: a) to provide real-time visualisation of the current state of the sensor-array network, allowing operators to respond to alert messages as they are triggered, and b) archive and retrieval capabilities for storing data and gathering evidence in the event of a crime. The human operator is an intimate part of the system. Their domain knowledge is a bridge between the events in the context scene - detected, annotated and stored in the archive - and the world-view of witnesses, or other interested parties. As such, the vocabularies used should reflect the domain under scrutiny (in our case, crime and security). By using ontologies we are able to semantically map queries from an operator to queries over the data archive.
4. *Common Event Model and Language*. Key to unifying the ISIS infrastructure is a common event model, capable of capturing the physical environment being surveyed in terms of time and space as well as the objects and relationships within the scene. The ontological language used to describe each scene must be shared across all participants in the system - human and process. Providing this common language platform has proved a key enabler

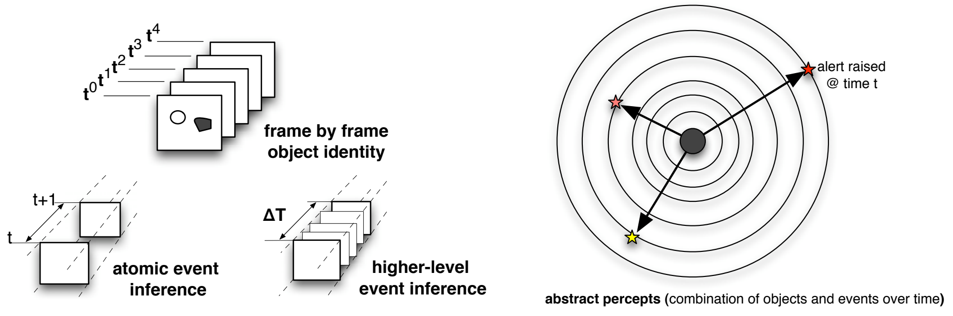
for human interaction with the system when retrieving specific events from the sensor data archive. It also provides the necessary separation of data-model from hardware infrastructure. As new domains are added to the event model ontologies and mapped on to rules within the system model, so new behaviors and events may be monitored.

We now go on to discuss the breakdown of sensor data towards the fundamental constructs of the common event model proposed.

2.1 Perceiving the Physical Environment

In the words of Marvin Gaye "*The World is just a great big Onion*" [2]. This is a view we take when perceiving the world via the ISIS sensor-web towards the goal of triggering an alert of a specific security/crime event on-board a bus in our network.

Figure 3 shows two views on the data. Figure 3a shows how the data stream is broken down into individual frames (in reality these may be key-frames rather than every frame). At a time instant we detect within the frame the identifiable objects and their properties. The objects are assigned a unique identifier. To determine events, we must examine the difference between frames over a time interval. At the lowest level - atomic-event - this is done with consecutive frames. For higher-level events a great time interval is used, as are more frames. Taking this approach to determining events, we can see that over time, layers within the data appear that correspond to the activity within the scene. This is the approach we use to trigger an alert within the system. By considering domain specific behaviors as a collection of related events, we are able to determine at what point a set of events may be perceived as a '*looked-for*' behavior. At this point the system will raise an alert. This is illustrated in figure 3b.


t0 t1 t2 t3 t4
frame by frame object identity
t+1
t
atomic event inference
ΔT
higher-level event inference
alert raised @ time t
abstract percepts (combination of objects and events over time)

(a) Frame analysis of Sensor Data (b) High-level Event Triggers

Fig. 3: Abstract layers and frame instances within sensor data.

To achieve this requirement we need a formal language to describe the sampling of the real-world discussed above, and to represent this in a conceptual model. This language and process of representation is now discussed.

3 Towards a Common Event Model

Figure 4 shows the proposed model. The model has three elementary data types, namely: *property*, *object* (entity) and *event*. Data elements hold values that correspond to the vocabulary introduced by the ontology/ies for that data element. Furthermore, each data element may relate to another data element through a semantic/thematic role. A Time Ontology supports the temporal aspect of the model such as the temporal granularity, i.e. how often the model is refreshed by inputs from sensory devices, as well as temporal metrics.

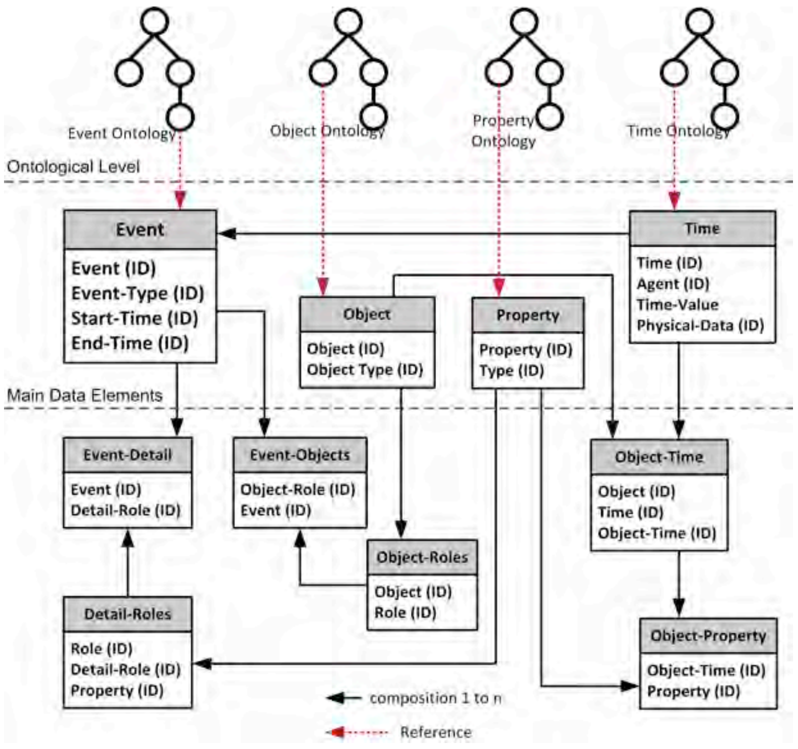


Fig. 4: Common Event Model for the Integrated Sensor Information Systems

The two most important views of the data scheme are Event and Object. The Event is a constituent for representing actions e.g. approaching, coming near or nearer, in space or time. The Object refers to *things,* or entities, that we identify in a domain of interest, for example, in an office surveillance model, objects may include persons, and stationary items such as computers or desks. The Property refers to the qualities of objects, or is used to describe an event through a semantic role. For example, location can be a quality assigned to Objects for a specific time, or it can be a factual datum that completes the meaning of an action like “approaching a location”. In a domain of interest, there might be more than one Property; in this case, each Property will be described by an individual ontology of that Property.

In the proposed model, each instant output of a sensor is uniquely tagged by the vocabulary provided by the Object and Property ontology, and accompanied by a temporal tag. The temporal tag uniquely identifies the source of information i.e. a sensor device,and its modality; moreover, each temporal tag has a pointer to real data sampled by a sensor. As an example, a temporal tag for a surveillance camera identifies one camera in a multiple camera network. Moreover, the temporal tag provides a pointer to the video frame that has been captured, at that time instant, and by that camera - a pointer can be a URL of a jpeg image file.

As the model provides a common vocabulary for annotating the output of sensors, it is possible to check the output of sensors against each other by defined relations within the ontologies. A checking procedure can then be employed - whether for assigning a confidence measure, and/or the discovery of anomalies - allowing the checking-rules for data consistency to be written for concepts introduced by ontologies, rather than for each individual sensor. This ability separates the language of description and inference from the sensor hardware infrastructure.

As mentioned earlier, another distinct feature of the proposed model is the use of semantic-role [7] in its structure. As figure 4 (Event Objects and Event Details) shows, Object and Property are related to Event through a composition of semantic-role labeled entities. The introduction of semantic-role into the model plays two major roles: firstly, it holds a relation between concepts which are defined in two different ontologies, e.g. between concepts in Object Ontology, and Event Ontology, forming an intra-ontology relationship between the distinct concepts, and second, semantic-role labels provide linguistics knowledge about how to interpret and map factual data to/from natural language utterances.

To explain the importance of semantic role, we continue with an example. The Video Event Representation Language (VERL) [4] is a formal language for video content modeling. VERL is formed on first order logic to describe an ontology of events; individuals may then define their own event ontology in a domain of interest and exploit VERL to describe that event in an ontology. In the VERL framework, each video instance is accompanied by a Video Event Markup Language (VEML) tag [3] - VEML describes the content of a video stream according to its companion VERL. In this matter, our work has benefited from the underlying logic behind the VERL framework and relevant event detection procedures. In addition our proposed approach takes advantage of ontologies in the supported domain's background knowledge, and it uses the definitions of events and their semantics in the event ontology to go one step further, by introducing semantic-roles into the model proposed by a formal language like VERL.

A VEML annotation for the sample "approach" event is shown below (example 1). The approach event has a certain meaning encoded in *rules*, conveyed by the VERL ontology. The definition of the approach event holds two arguments (argNum1 and argNum2) each with a corresponding value. In addition, other details such as the start frame and end frame for a specific instance of approach event in a specific video stream, as well as a name for the event. This complete VEML annotation refers to a specific event instance.

Example 1. VEML Approach Event
```
<event type="APPROACH" id="EVENT1">
    <begin unit="ms">136</begin>
    <end unit="ms">147</end>
    <property name="name" value="Person1-approaches-DOOR1"/>
    <argument argNum="1" value="Person1"/>
    <argument argNum="2" value="DOOR1"/>
</event>
```

The VEML representation of the approach event above implies the statement "*Person1 approaches Door*1" in a human observer's mind and is encoded in the definition of "approach" event in the VERL rule ontology. To enable machines to have such an interpretation from the above video annotation however, we need a formal description, which tells a machine how to interpret/translate the VEML annotation to/from natural language. (We say natural language here as this refers to the expressiveness of the proposed model - this is emphasised by Westermann and Jain [11]) - this expressiveness requirement can be achieved by the help of semantic-role.

If we introduce the first argument of an approach event as the *agent* of the event and the second argument as the *goal* of the event, then we are able to map an utterance like the above statement into/from its companion VEML representation. The following shows our suggested XML representation for the first and second arguments of VEML representation (example 2):

Example 2. XML Representation introducing Semantic Role
```
<event type="APPROACH" id="EVENT1" begin=''T03'' end=''T07''>
    <argument semantic_role="agent" value="Person1"/>
    <argument semantic_role="goal" value="DOOR1"/>
</event>
```

Because VEML is a formal language it is possible to write unambiguous ontological mappings from the VEML representation into the proposed model, where we know the semantic role of each argument. In effect, the above XML representation will be encoded through a set of facts organized around the elements of the data model. To give more insight, the next section describes the architecture of a prototype system that uses the event model described above, to integrate the elements of a doorway surveillance system.

4 Prototype System

The proposed data model has been employed in a prototype system for a doorway surveillance system (see figure 5). The system automatically captures video from multiple sources and annotates the video, identifying people as well as their gender property as they walk and enter into a controlled environment.

The system comprises three main components: a sensor based analysis component (shown as camera sensors and their companion Image Analyzers (IA)), a Data Manager (DM), and an Event Detection (ED) component. The system components are implemented as autonomous agents communicating through TCP/IP connections.

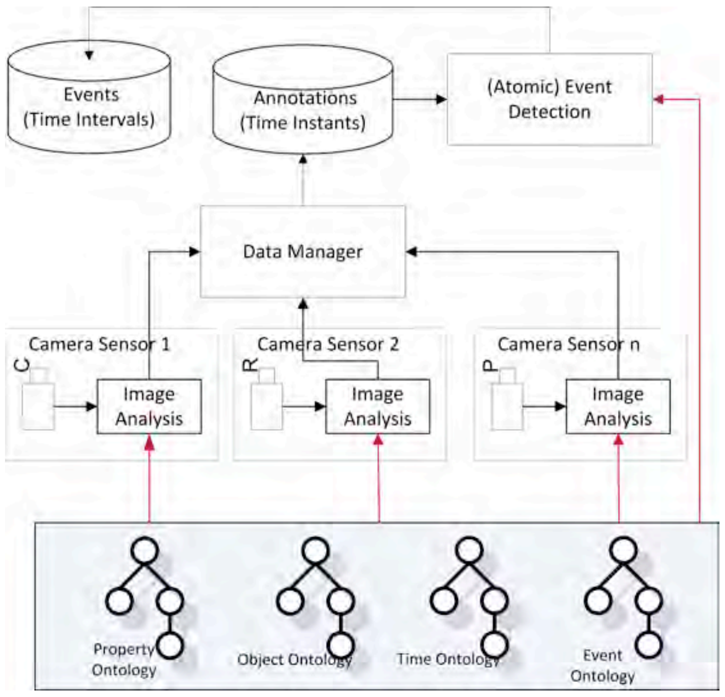


Fig. 5: Block Diagram of the Sensor-based Prototype System.

The Camera Sensors are annotating observations using vocabularies provided by the time, property, and object ontologies and writing the annotations to a sub-part of the data model. The Data Manager checks data aggregation and assigns confidence measures to annotations. The Event Detection process mines for events in the annotated observations and writes these to another sub-part of the model. The Image Analysers identify people and their location, as well as their gender, and assign them a unique ID. This is done by mapping extracted features using Principal Component Analysis [8,9] to high level concepts described in the ontologies, for example the type of object.

Figure 6 shows how the proposed model integrates the physical aspects of the ISIS sensor network. Referring back to figure 3b it is possible to see how the model integration illustrated in figure 6 produces layers within the data, where each layer is rooted in the information-base described by the pool of ontologies that make up the domain. At the core (1st layer) the atomic events are captured as time invariants. These represent the lowest level of detail infered by the system. Each subsequent layer represents a skin of new, infered knowledge, whose pool of knowledge is drawn from that held by the the previous $n-1$ layers.

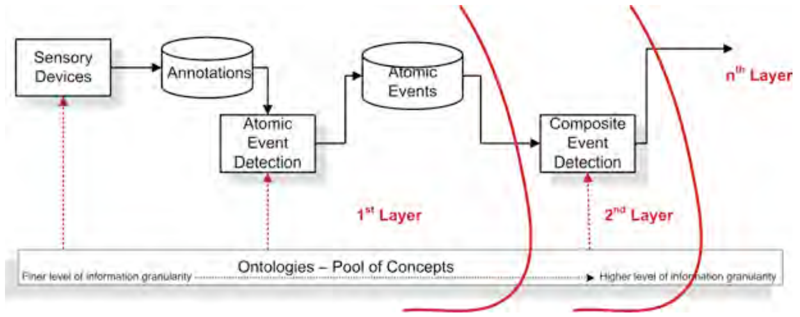

Sensory Devices
Annotations
Atomic Event Detection
Atomic Events
Composite Event Detection
nth Layer
1st Layer
2nd Layer
Ontologies – Pool of Concepts
Finer level of information granularity
Higher level of information granularity

Fig. 6: A layered view on system inferences. At the core of the system, sensors are annotating time invariants.

The Event Detection procedure, as it is described above, may be repeated for several turns. Figure 6 shows this layered view of system inferences. The Atomic Event Detection procedure detects the most granular events. These are then used by the system as higher level abstract definitions for inferencing events at the next level of granularity; this may also be viewed from a temporal granularity perspective. Such a setup for event detection may be helpful when employing different communication technologies for data exchange at the physical network layer, as each communication may refer to specific abstractions captured within the data.

5 Conclusions and Further Work

This paper introduces a scheme for content modeling of temporal media in an integrated sensor network. The aim of the work is to move a step closer towards a common event model for integrated media data as described by [11]. To do so, an ontology-supported data model that connects data elements using semantic-role-relations was introduced. Our aim was to show that by introducing the notion of semantic-role from linguistics, we are able to better represent semantic content of sensor-data captured within our sensor-web. The use of ontologies aids the checking of data aggregation and consistency towards a unified view of the world under surveillance, independent of the physical sensor devices. Introducing semantic roles in an event modeling framework provides a means for systematic mapping of the outcome of semantically labeled natural language constituents, into elements of a data model and vice versa. Moreover, semantic-role-relations can be used for managing intra-ontology semantic relations, i.e. semantic relations between concepts that are defined in different ontologies. We showed how this model may be used to integrate the elements of an integrated sensor information system, representing infered domain knowledge as layered skins with increasing information granularity.

The current system is implemented in Prolog with ontologies implemented in first order logic. Converting the ontologies to a standard ontology language such as Ontology Web Language is considered for immediate future. Although temporal reasoning and representing temporal inference rules remains untouched, this also forms a part of our future work. In addition, the approach proposed raises an issue regarding the trade-off between the real-time inferencing of events and the storage of events as higher level abstractions used in higher level reasoning. For further experimental study and investigation therefore, is the balance between the granularity of the stored events and those infered in real-time.

6 Acknowledgement

The authors would like to thank Dr Jiali Shen of ECIT, Queens University Belfast, for his work towards object identification and gender profiling in video data. The work presented in this paper is supported by the Integrated Sensor Information System (ISIS) Project, funded by the Engineering and Physical Science Research Council, reference number EP/E028640/1.

References

1. ISIS - An Integrated Sensor Information System for Crime Prevention.
2. N. Ashoford, V. Simpson (performed Marvin Gaye, and Tammi Terrel). The onion song. In *Easy*, number TS294. Tamla, 1969.
3. Bolles B. and Nevatia R. ARDA Event Taxonomy Challenge Project, Final Report, 2004.
4. Alexandre R.J. Francois, Ram Nevatia, Jerry Hobbs, and Robert C. Bolles. Verl: An ontology framework for representing and annotating video events. *IEEE MultiMedia*, 12(4):76–86, 2005.
5. Manfred Hauswirth. Semantic Reality - Connecting the Real and the Virtual World. In *SemGrail Workshop 2007*. Microsoft, Seattle, 2007.

6. Jerry R. Hobbs. Causality. In *Fifth Symposium on Logical Formalizations of Commonsense Reasoning*, Common Sense. New York University, New York, New York, 2001.
7. R. Jackendoff. *Semantic Structures*. MIT Press, Cambridge, MA, 1990.
8. Ratika Pradhan, Zangpo Gyalsten Bhutia, M. Nasipuri, and Mohan P. Pradhan. Gradient and Principal Component Analysis Based Texture Recognition System: A Comparative Study. In *ITNG '08: Proceedings of the Fifth International Conference on Information Technology: New Generations*, pages 1222–1223, Washington, DC, USA, 2008. IEEE Computer Society.
9. Sameena Shah, S H Srinivasan, and Subhajit Sanyal. Fast object detection using local feature-based SVMs. In *MDM '07: Proceedings of the 8th international workshop on Multimedia data mining*, pages 1–5, New York, NY, USA, 2007. ACM.
10. A. Sheth, C. Henson, and S. S. Sahoo. Semantic sensor web. *Internet Computing, IEEE*, 12(4):78–83, 2008.
11. Utz Westermann and Ramesh Jain. Toward a common event model for multimedia applications. *IEEE MultiMedia*, 14(1):19–29, 2007.

Views from the coalface: chemo-sensors, sensor networks and the semantic sensor web

Jer Hayes[*], Edel O'Connor[1], John Cleary[1], Harry Kolar[2], Robert McCarthy[*], Richard Tynan[3], Gregory M.P. O'Hare[3], Alan Smeaton[1], Noel E. O'Connor[1], Dermot Diamond[1]

[1] CLARITY: Centre for Sensor Web Technologies, Dublin City University, Glasnevin, Dublin 9, Ireland
[2]IBM Systems & Technology Group, East Fishkill, New York
[*]IBM Innovative Environmental Solutions, Mulhuddart, Ireland
[3]CLARITY: Centre for Sensor Web Technologies, University College Dublin, Belfield, Dublin 4, Ireland
{hayesjer@ie.ibm.com; edel.oconnor@computing.dcu.ie; john.cleary@dcu.ie; kolar@us.ibm.com; rjmccarthy@ie.ibm.com;richard.tynan@ucd.ie;gregory.ohare@ucd.ie; alan.smeaton@dcu.ie; oconnorn@eeng.dcu.ie; dermot.diamond@dcu.ie}

Abstract. Currently millions of sensors are being deployed in sensor networks across the world. These networks generate vast quantities of heterogeneous data across various levels of spatial and temporal granularity. Sensors range from single-point in situ sensors to remote satellite sensors which can cover the globe. The semantic sensor web in principle should allow for the unification of the web with the real-word. In this position paper, we discuss the major challenges to this unification from the perspective of sensor developers (especially chemo-sensors) and integrating sensors data in real-world deployments. These challenges include: (1) identifying the quality of the data; (2) heterogeneity of data sources and data transport methods; (3) integrating data streams from different sources and modalities (esp. contextual information), and (4) pushing intelligence to the sensor level.

Keywords: Environmental sensor networks, chemo-sensors, metadata standards, sensor intelligence

1 Introduction

The semantic sensor web offers the unique opportunity to unify the real and virtual world [1]. The notion of unifying the real world with the virtual world has been described before as *internet-scale control*, a concept that which originated with IBM researchers Ron Ambrosio and Alex Morrow [2]. The notion of *internet-scale sensing* and how it relates to chemo-sensors has also been examined [3], and this suggestion made that the realization of large-scale sensor networks should be based on internet enabled sensors that allow for external browsing of the sensor's status, provide command and control, and facilitate feedback of information to individuals and other devices (see Fig. 1). Such a view suggests pushing analytics down to the sensor level. The *internet-scale sensing* concept is very similar to current proposals for the Semantic Sensor Web (SSW) where such a system will need to be automatically deployed, automatically configured and have tailored delivery of information for a variety of users [1].

At its simplest a SSW is one where sensor data is annotated with semantic metadata to increase interoperability as well as to provide contextual information essential for situational knowledge [4]. According to the 'Sensor Model Language (SensorML) for In-situ and Remote Sensors' discussion paper a sensor is "an entity capable of observing a phenomenon and returning an observed value. A sensor can be an instrument or a living organism (e.g. a person)…", p12, [5]. Clearly, manual sampling still takes place in many parts of the world (where it may be a legal requirement) and this results in under-sampling. Under-sampling can be overcome either by using remote sample collection devices (auto-samplers) or by using sensor networks to continuously measure over long time periods. However, we should note that the SSW will have to deal with data from manual sampling (perhaps via a lab-based data management system) as well as near real-time data streams. This is important as "events" may be detected via SSW but confirmation may require autosamplers to grab samples from the same source at the same time, and flag that they are available; an operator has to go to the sample, and collect for a more sophisticated lab-based analysis that generate data acceptable in a court.

The variety of existing sensor networks is extensive but the core consistent with that of SSW in that it envisages a world in which the status of the real world is monitored by large numbers of distributed sensors, forming a sensor 'mesh' that continuously feeds data into integration hubs, where it is aggregated, correlations identified, information extracted, and feedback loops used to take appropriate action [6]. Sensor networks provide a web of interconnectivity that provides the multiple sources of information that will underpin more accurate decision making. Decision-making in the SSW will be absolutely essential. Decision making is a complex and demanding process which is often constrained in a number of possibly conflicting dimensions including quality, responsiveness and cost. We suggest that analytics in the SSW will happen across a continuum from one edge of the SSW to the other, from the sensors right through to the end user. By this we mean data stream analytics can occur on the sensor itself (see Section 4) and from any point in SSW to the end user who may apply his or her own analytics to a variety of data streams from the SSW.

Sensor networks are composed of sensor nodes which are the smallest component of a sensor network that has integrated sensing and communication capabilities and these can be wired (as in some coastal observatories) or wireless (as in wireless sensor networks). The sensor node has basic networking capabilities through communications with a base-station and sometimes other nodes. The simplest sensor nodes will have a microcontroller to perform basic processing operations but sensor nodes can also be highly complex, e.g. remote sensing instruments on satellite platforms, and which perform complex processing operations locally.

At present the culture in SSW/wireless sensor network (WSN) research is very heavily biased towards transducers like thermistors which is understandable as they exhibit almost ideal behaviour - low cost, long-life, very low-power, small form factor, high accuracy and precision, rugged, reliable, etc. This bias colours the expectations of SSW/WSN researchers in that they expect all sensors to conform to this ideal. The physical sensor bias, in short, is the notion that *all* sensors act like thermistors and thus are held to be reliable or at least reliable enough that questions of data quality are relatively straightforward, and the sensors are simple to use and require little maintenance. In reality this is invariably not the case for a variety of reasons including, leaching of active components from sensing membranes (see [7]), physical damage, lack of selectivity, non-linear performance, baseline drift and biofouling (particularly in the marine environment). And as such the data stream generated by sensors (especially environmental sensors) is prone to data quality (trust) issues [8]. Physical sensors which are encapsulated can also be affected by extreme changes in temperature. The SSW system itself must be capable of examining the streams of data being imported, and both observing environmental events as they take place (e.g. pollution event) or observing erratic behaviour from a particular sensor and flagging it as unreliable and requiring attention. An end user who accesses data from the SSW should be given information on the quality of the data from a sensor or set of sensors and ultimately be given enough information to ascertain whether they wish to trust the data or not. But how is the analytics to take place ? Should there be a standard? The solution to this may lie in current research on the interaction between the rule and the ontology layers of the Semantic Web.

In dealing with raw data streams we can ask – what does this data stream mean ? Generally speaking data streams are not self identifying and we require outside information, metadata, to understand the stream. The main driver for the use of metadata has been data sharing. Scientists generate large amounts of data and often we wish to share this data with other researchers. This "data sharing" is made easier when the data formats are the same or at least interoperable. However, it is often the case that "data sharing" is difficult due to competing standards and a general lack of metadata. In the marine area there has been much work on establishing metadata standards. This work has been driven by practical needs as often because researchers are interested in phenomena that cover large areas where several groups or institutes are gathering data. To get a full picture of a particular phenomenon, e.g. an algal bloom along a coast line, a researcher may need to augment the in-situ sensor data streams with information from a variety of other sources (e.g. satellite information). Therefore in considering SSW applications it is worth examining current standards to see if they can fit into a SSW.

Another goal of *internet-scale sensing* and the SSW is to allow for plug'n'play sensors (or alternatively deploy-and-forget sensors). It is worth noting how difficult this in the real world. Consider the example of an ocean observatory such as the one being developed by the Monterey Bay Aquatic Research Institute (MBARI). MBARI highlight that the network for the ocean observatory will use a wide variety of communication links: optical fibre, microwave, packet radio, satellite and acoustic [9]. This results in a diversity of throughput, latency and intermittence throughout the network. These are factors which will also affect the SSW in the real-world, namely –

can the SSW handle a diversity of throughput, latency and intermittence ? There has been work done on data transport protocols that guarantee delivery (e.g. [IBM's MQTT) but where a sensor is continuously sending data it may be the case that dropping several packets is not considered a large problem. This is major challenge (and perhaps the most obvious) for the SSW – given the heterogeneity of data sources and data transport methods how can they all fit neatly into the SSW ?

Therefore, from the perspective of sensor developers (especially chemo-sensors) and with respect to integrating sensors data in real-world deployments there are four main challenges to the unification of the real and virtual world:

1. The heterogeneity of data sources and data transport methods that all must neatly fit into the SSW.
2. The quality of the data must be described and understood.
3. Data streams from different sources and modalities (esp. contextual information) which vary in across many dimensions, including spatial, temporal, granularity of data, must be integrated.
4. The SSW must be capable of supporting analytics (e.g. decision making) across the SSW nodes.

Some of these challenges are inter-related e.g. a solution to integrating data streams from different sources and modalities must respond to challenge (1).

2 Integrating data from heterogeneous sensors and modalities: a marine example

The question of what constitutes a sensor must not be constrained when considering the SSW. For example in the context of environmental monitoring, in-situ wireless sensor networks (WSN's) substantially reduce the need for costly labour-intensive on-site sampling and data collection. However WSN's pose some distinct disadvantages and we are often required to consider alternative or complementary sensing modalities. In order to identify and highlight some of the issues for the SSW in relation to integrating data from heterogeneous sensors, we will examine the scenario of marine monitoring. However many of the issues outlined here can equally be applied to other application scenarios.

Marine monitoring includes the observation of various aspects of the marine environment. It ranges from the detection of pollution and the development of harmful algal blooms to the monitoring of coastal features and coastal erosion. A recent analysis of developments in mote-based wireless sensor networks with respect to environmental monitoring [6] suggests that there are still many limitations with the current capability of these platforms for sustainable environmental sensing. As previously mentioned, in-situ sensors which are in direct contact with the environment are subject to the problem of bio-fouling and require regular maintenance. This can result in unreliable and noisy data or gaps in the sensing data. Furthermore, the more advanced chemo-sensors are still quite expensive to produce, and at present require regular maintenance (from days to weeks). Therefore only a limited number of these may be deployed in the environment and are subject to regular maintenance. Due to the expense and logistical difficulties associated with in-situ networks and some of the problems outlined above, it is currently not possible to monitor a wide area over long periods of time with current mote-based wireless sensor networks. Also, in-situ sensor networks may not be suited to certain types of applications. For example Alexander and Holman [10] used an alternative sensing mechanism (video cameras) to quantify near-shore morphology of a coastal location since the turbulent nature of the surf zone often makes it difficult to maintain in-situ instrumentation.

2.1 Satellite-based remote sensing

Due to the limitations outlined above, alternative sensing modalities are often considered for the purposes of marine monitoring. Sophisticated satellite sensors are very effective for monitoring many parameters such as sea surface temperature, sea surface height, ocean currents, turbidity, and chlorophyll pigment concentration (which subsequently can be used to determine the amount of algal growth in the water). A number of these sensors are orbiting the earth on various satellite platforms. These sensors have differing spatial resolutions and operate on satellites with varying orbits and orbit cycle times (which subsequently affect geo-spatial comparison and temporal resolution).

Some of these sensors only operate in the solar reflective spectral range; hence they only gather useful data on cloud-free days during periods of daylight (i.e. when illumination conditions are suitable). For example, MERIS (Medium Resolution Imaging Spectrometer) onboard ESA's Envisat platform is a programmable, medium-spectral resolution, imaging spectrometer, which operates in this range. Its primary purpose is to measure the colour of the ocean and subsequently derive estimates of the concentration of parameters such as chlorophyll and suspended sediments. It has a spatial resolution of 1200m over the ocean and 300m over land and coastal zones and it completes coverage of the Earth in 3 days [11].

Other sensors are not subject to these restrictions and can provide data during periods of darkness and cloud-cover. An example of such a sensor is a radar altimeter which transmits microwave frequency pulses to the sea surface and receives the reflected echoes [12]. This type of remote sensor is often used for measuring sea surface height. The Ocean Surface Topography Mission (OSTM)/Jason-2 is an international satellite mission that was launched in June 2008 to extend the continuous climate record of sea surface height measurements. The main instrument onboard Jason-2 is an altimeter that measures the distance from the satellite to the ocean surface. It repeats its ground track every ten days, covering 95 percent of the world's ice-free oceans [13].

Many of these satellite sensor streams also arrive in various formats. Furthermore data is not overlaid on one common grid which affects geospatial comparison. There exists a number of European and international projects aimed at improving the interoperability of satellite sensor data; an example of such a project is the Global High resolution Sea Surface Temperature pilot project (GHRSST-PP) which was initiated by GODAE – Global Ocean Data Assimilation Experiment. GODAE identified that numerical ocean forecasting models require a near real-time supply of SST data, sampled often enough to resolve the diurnal cycle, along with an accuracy better than 0.2K and a spatial resolution better than 10 km which is only possible by combining the best capabilities of different types of sensors.

In 2002 GODAE initiated GHRSST-PP [14] and the data products from this programme satellite provide SST observations from various satellite sensors in a common format (netCDF) together with a measure of uncertainty for each observation. This means that all satellite SST data are presented in a common format and the user doesn't have to re-code for the ingestion of different satellite data. The ancillary data provided allows the user to filter data based on the criteria outlined to their specific application. A combined analysis of all available SST data is carried out enabling the benefits of using in situ, microwave satellite SST and infra-red satellite SST in synergy. Diagnostic datasets are also produced for a number of sites around the globe. This is where all available data for a number of areas are gathered and subsequently resampled onto a common grid to assist intercomparison and characterisation of the various input data streams [15].

2.2 Alternatives to Satellite-based remote sensing

Another alternative sensing modality is that of optical airborne remote sensing. Its major benefit as opposed to satellite remote sensing is that the user can define its operational and deployment characteristics. It generally can provide much higher spatial resolution data and be deployed when atmospheric (i.e. cloud free), environmental, and solar conditions are acceptable to study specific phenomenon [16]. This can also be coordinated with in-situ sampling for algorithm validation or development. In coastal aquatic environments, many processes occur over space and time scales that cannot be adequately monitored using satellite remote sensing systems. The use of airborne remote sensing offers unique capabilities that enable specific coastal events to be studied. Coastal video systems have also been identified as effective tools for coastal monitoring and can be used to monitor coastal erosion, sea conditions, etc. A prime example of this is a major European research project entitled CoastView [17]. This is an alternative to the more expensive satellite and airborne remote sensing data which can also provide data over long periods of time at high spatial and temporal scales which is suitable for monitoring inland and coastal marine locations. Web cams and CCTV cameras are cheap and easily deployed. In fact, there are an abundance of web-cams available on the World Wide Web that can be used for monitoring purposes.

The singular use of video and images from cameras can act as a powerful sensing tool but even more so when used in combination with other sensing modalities. Davidson et al. [17] point out that despite the potential to improve monitoring of coastal zones with coastal video systems, that there are many coastal management issues that may only be addressed adequately through the integration of additional data sources and expert knowledge alongside the image data. For example, O'Connor et al have investigated the use of multiple sensing modalities in a river location [18] using video feeds with data streams on pH, temperature, turbidity, conductivity and depth. It can provide some context around what is being sensed by monitoring various parameters. The conditions surrounding certain events may subsequently be deduced e.g. what are the environmental conditions surrounding an algal bloom event, what were the prior conditions to increased phosphate detected in the water. This can subsequently be used to automatically control the sampling frequency of in-situ sensors. Multiple sensing signals can also be used to deduce the quality of data and provide this information to the user.

2.3 Problems facing SSW from the coalface of the marine enviroment

The SSW should be aiming at providing similar functionality to GHRSST-PP on a much larger scale. In effect the SSW must be able to provide an awareness of the capabilities, limitations and differences of the sensors and associated data streams. This is necessary in order to select appropriate data streams, from the diverse array currently available, to meet the needs of specific applications. Other problems outlined in this section were that: (a) the phenomena sensed is broad and this requires a broad suite of sensors / instruments which all have to described / classified within the SSW; (b) In the real world we will have unreliable and noisy data or gaps in the sensing data and the SSW must account for this; (c) data is often in different formats (which currently results in users having to recode) and these must neatly fit into the SSW; (d) SSW should allow reasoning over heterogeneous multimodal sensor data and push intelligence to the sensor level i.e. if condition (x), condition(y) and condition(z) are met, start sampling more frequently or alert the responsible authorities that samples need to be taken immediately, (e) the SSW should accommodate interoperability of data streams and be able to deduce that the sensing signal in question is faulty or offline and subsequently provide the alternative.

All of these problems relate to the four challenges outlined in section 1. Challenge (1) which relates to the heterogeneity of data sources covers problems (a), (c), and (e) and so appears to be the largest challenge to SSW. While challenge (2) which relates to data quality is highlighted by problem (b), and (d) relates to challenge (4) where analytics occur across the SSW.

3 Precursors & building blocks of the Semantic Sensor Web

Considering the maxim that it is best not to reinvent the wheel we will outline a number of attempts at describing data and sensors. These attempts can be viewed as the precursors to the SSW or as possible components in future architectures. One movement in sensor research is towards making sensors web-resident thus making it possible to remotely discover, access, and use real-time data taken directly from the sensors. These three activities, discovery, access and use are fundamental to working with sensors. We will briefly describe three standards of data description and sensor description (1) the CDI XML schema [19], (2) MarineXML [20] / CSML and (3) SensorML [21]. SensorML falls into the category of sensor description rather than just data description. We should stress that these three standards are just three among a larger community of standards, e.g. KeelyBricks [22], MBARI [23] , MIML [24], ESML [25] , and OBIS [26]. The common data index (CDI) is designed to be used as an index to the individual datasets held by sea-search partners and as such could be described as a description of data sets [19]. This metadata about data sets has been implemented in an XML format known as the CDI XML schema. . The CDI XML format has adopted the ISO19115 metadata standard which is an ISO standard that defines the schemas for describing geographic information and services [19]. The CDI is supposed to provide enough information to answer the following questions: (1) Where? - What is the geographical location of the captured data? (2) When? - When the observation began and when did it end? What was the sampling interval? (3) What? - What was measured? (4) How? - What instruments were used? What platforms were involved? (5) Who? - Who is the originator of the data? (6) Where to find data? - Which partner holds the data? Is there web access? Are there restrictions?

MarineXML gives a common framework for the data and its structure in terms of a catalogue of feature types but does so by largely using the Climate Science Markup Language (CSML) to tag data. In fact, it is probably more correct to say that MarineXML is a framework for allow interoperability of marine data. The framework requires the implementation of a common vocabulary for measurement systems by use of parameter dictionaries for storing agreed definitions of phenomena and the units used to measure them. This is the same solution that the CDI uses for instruments and the respective measurements. However, MarineXML adapts the unit and phenomena dictionary definitions inherent in the Climate Science Markup Language (CSML) in applying a GML encoding of CFStandardNames for referencing phenomena dictionaries and UDUnits for unit definitions. Essentially, phenomena (things that can be sensed/measured) are measured in terms of units. GML has a dictionary of phenomena and associated units (of measurement). Whereas CDI schema XML is based on ISO19115 and MarineXML is based on ISO 19136 both these ISO standards fall into the general ISO 19100 category of geographic standards. Thus we use the term "MarineXML/CSML" to refer to the data standard rather than the whole data interoperability framework of MarineXML.

As with the CDI XML schema there is the conception that a common framework and grammar for expressing the data and its structure is needed and that this also necessitates a common vocabulary of measurement systems and feature types. But abstractly all marine data in general should have the following attributes: (1) Position: all data will have associated positional information; (2) Time; (3) Units; (4) Tolerances: accuracy, precision, resolution; (5) Source; (6) Agent: what person / organisation carried out the data recording? (7) Method: method by which the data was obtained, (8) Promoter: entity that initiates data collection, e.g. a government agency; (9) Original purpose: what was the original purpose of the data collection? (10) Restrictions: are there copyright restrictions etc? (11) Errors; (12) Quality control; (13) Form; (14) Format; (15) Metadata.

SensorML is an XML based description of the process or processes of measurement that a sensor or sensor systems performs. Processes are entities that take one or more inputs and through the application of well-defined methods using specific parameters, results in one or more outputs. In addition there is a large amount of metadata related to the sensor with respect to system location, capabilities, characteristics, contacts, time constraints, legal constraints, security constraints amongst others.

3.1 The problem of granularity and metadata standards

One problem which reoccurs in the creation of SSW systems is - what metadata is to be used and how specific should it be? However, a second problem arises when organizations use different types of metadata and this problem is a question of semantics in the broadest sense – what do the metadata terms actually mean? It may be the case that different groups may represent the same data in different ways or use codes that have different levels of granularity. The SSW system has to be able to deal with all these problems. In Table 1 are listed two parameters which list the GF3 codes. In this example the code "DRYT" refers to "Dry bulb temperature". In Table 2 we list a number of BODC codes related to "air temperature" using a "dry bulb thermometer". As can be seen the BODC codes list the instrument used, in this case a "dry bulb thermometer", as well as what is being measured. The BODC also appears to have more entries for "air temperature" using a "dry bulb thermometer" than GF3. The BODC in this case is more detailed and thus has a higher level of granularity and different levels pose a problem which data with different tags have to be integrated. Does "DRYT" map onto all three BODC codes or is it just equivalent to one of the BODC codes? This type of question must be answered by developers of the SSW system but also more importantly is must be answered by the community of users. The problem of granularity is a general problem for ontology-builders and those who wish to map ontologies. This problem also falls under challenge (2), the heterogeneity of data sources, when the SSW uses descriptions to aid in classifying data sources the choice of appropriate metadata standards is fundamental.

4.1 Chemo-sensors & the semantic sensor web

Chemo/bio-sensor Networks employ emerging molecular sensing technologies in order to monitor specific targets in the environment, and in some cases develop linked proxies for predictive use. The Adaptive Sensors Group (based in Dublin City University) have developed a ground-based sensing device, in this case, an autonomous phosphate analyzer [27]. This is a field-deployable system for long-term monitoring of phosphate levels in natural waters was developed incorporating sampling, pumping, reagent and waste storage, optical detection, and wireless communication in a robust and portable device. The analyzer is more complex in design than common sensors such as thermistors and passive infrared sensors as it uses "wet chemistry" to analyse phosphate which involves pumps, valves and fluid handling, the use of reagents and storage of waste.

The phosphate monitoring analyser is designed to operate autonomously in long-term field deployments. Figure 2 depicts trial results from a waste water treatment plant obtained during a 30+ day trial. A trial of this length is a major achievement as many chemo-sensors operate over days rather than weeks to months. Comparable data were obtained by the plant’s monitoring system suggesting that the wireless phosphate analyser can produce reliable data and is sufficiently robust to be operated in a completely autonomous manner for at least seven weeks.

4.1 Analytics: pushing intelligence to the edge of the semantic sensor web

SmartBay is a program of national infrastructure investment with the aim of enabling the development of next generation advanced coastal and marine monitoring and management technologies [28]. The Marine Institute (Ireland) and IBM are engaged in a multiyear collaboration to develop and provide advanced capabilities for global water management solutions. This collaboration is multifaceted but two major outcomes have been the development of an advanced embedded sensor platforms and the development SmartBay information portal. The advanced embedded sensor platforms are based on hardware that is ultra low power and embedded software builds IBM technology (e.g. J9 JVM, Lotus Expeditor components, MQ Microbroker). This platform has been developed to push intelligence down to the sensor level where real-time decision making can take place.

The autonomous phosphate analyzer [27] was used as a testbed for the advanced embedded sensor platforms. The core idea here is to push intelligence onto the sensor. The new system has the following capabilities: (1) Self monitoring - the system is able to monitor its consumables and change sample rates in response to outside events; (2) Scalability – the command and control can be updated remotely and multiple units can be updated simultaneously; (3) Verifiable data transmission – MQTT is used to ensure data is delivered.

The concentration of phosphate in the treatment plant is affected by the local ambient weather conditions and so it is desirable to vary the sampling rate as local weather conditions change. Rainfall can result in increases in phosphate levels in water bodies due to increased run off from agricultural/forestry land where manure/fertilizers are used. Heavy rainfall or storm events can also lead to large increases in the flow in a river which can also increase phosphate levels. The inclusion of satellite meteorological data, which is usually available directly on the web or can be acquired from the local meteorological institute, can give a more complete picture of the reasons behind the changes of the pollutants measured. This is exemplified by the causal link between deterioration in water treatment effectiveness and the waste volume throughput. A major increase in water volume input for example due to heavy rainfall in the local catchment, may overwhelm a plant’s capacity and lead to a deterioration in the treated water quality. For a chemo-sensor such as the phosphate system on board analytics may identify events based on local changes in phosphate level but data from contextual sources are required to provide the full picture. The seamlessly joining of plug-n’-play sensors into the SSW requires the system to be able to handle (1) the sensor and (2) contextual information. Can the SSW provide a sensor with contextual information that can allow for predictive modeling?

Recent developments in wireless sensor node technologies have resulted in devices with increased CPU, memory and transmission capabilities. Such developments have lead to the possibility of deploying goal based reasoners onto the leaf nodes of the network to engage in real time, in-situ and intelligent decision making. Given the remote operation, potential latency in message transmission and data volume, such abilities may be crucial to the successful operation of the sensing system. Each entity resident on a node is termed an agent and there may be multiple agents on a single node. An example of one such system is AgentFactory Micro Edition (AFME) [29]. AFME has been successfully deployed on a wide range of devices with varying capabilities. For example, it has been used on SunSPOTs to provide adaptive sensing capabilities. AFME has also been deployed to the SmartBay Phosphate Monitoring system. Using AFME provides a common programming model for the wide range of sensor devices that may possibly be deployed to compliment the core system. As mentioned previously, it also provides in network decision making so for example, decisions based on trade-offs between system accuracy and power consumption can be taken without human intervention in the field. The degree of cleaning of the device will impact power consumption, as well as sample quality. In some cases it may be vital to have a very precise reading when, for example, no other sensors are within the locality. However, when numerous other sensors are also participating, then minor inaccuracies may be tolerated as they can be averaged out by using a combination of all sensor readings. Further standard energy saving decisions such as adaptive transmission and sampling frequencies can also be taken by the agents.

In addition to network based decisions, the agents can also provide some analytic mechanism to signify important trends in the data. For instance, if a phosphate level is breached as in Figure 2, the agent might decide to notify a local or government authority. Such thresholding and event detection can be disseminated to the agents in a similar way to the policy level considerations such as prioritizing power consumption discussed previously. The thresholds may be automatically adjusted on a daily or even hourly basis depending on the cumulative levels detected over a given period of time. A code snippet from AFME which would classify three high phosphate events in the Figure 2 but more importantly a series of actions can occur from this ongoing event detection is given below:

```
newThreshold(?t) > setThreshold(?t)
threshold(?x), reading(?y) > checkReading(?x, ?y);
thresholdBreached(?amount) > informUserAgent(?amount);
severeThresholdBreached(?amount), strictPolicy() > informPlantAgent(?amount);
```

Further code from AFME could also detect sensor drift, diagnose operational issues and identify further user-defined events and integrate outside data sources.

However, Figure 2 also highlights the challenges to the SSW. The reference sensor (in red) does not identify the first event that is flagged by the prototype phosphate sensor. Which raises the question - is this event real or is it a false positive? Many sensors will need access to contextual information and have sophisticated on-board intelligence to assist in the process of deciding whether detected events are true or false. The SSW should allow sensors to discover, access, and process relevant contextual information - even sophisticated instruments such as the prototype phosphate sensor can benefit from contextual information that improves the quality of event detection. The contextual information should also be quality tagged (e.g. via metadata) to identify whether it should be used or not by other nodes in the SSW and this quality checking may further require access to other sets of related contextual information; which highlights the challenge of quantifying data quality. It will also be the case that false negatives can occur (what events have been missed?) and so as the numbers of devices scale up, the complexity of decision-making also scales up. However, in both cases (false positives and false negatives), the quality of event detection, and dependent decision making, can be improved. In the case of false positives, the confidence in a positive decision is enhanced through, for example for the phosphate sensor, correlation of sampling rate with rainfall level; i.e. water quality decreases when there is a heavy rainfall event in the local catchment; therefore increase sampling rate to get more independent measurements for cross-validation. On the other hand, if an event is predicted from contextual information but not detected (possible false negative), the instrument could be instructed to check the data using more sophisticated algorithms to see if there is any evidence of an event.

4.2 Problems facing SSW from the coalface of analytics and chemo-sensors

To summarise our discussion of analytics and chemo-sensors, it is clear that more sophisticated decision making tools are needed to ensure that the incidence of false positives and false negatives is minimized. If this is not done then the usefulness of the aggregated information will be unacceptably compromised, and WSN effectively useless. In short, decision-making tools are required to if we are to achieve workable, functioning internet-scale sensing. This problem falls under challenge (4) where analytics may occur across the whole SSW and challenge (3) where contextual information will have to accessed from different data streams (and different data sources and modalities).

5 Conclusions

Currently millions of sensors are being deployed in sensor networks across the world. These networks generate vast quantities of heterogeneous data across various levels of spatial and temporal granularity. The semantic sensor web will handle sensor data ranging from networks to single-point in-situ sensing to remote sensing which can cover the globe. This will result in the unification of the web with the real-word. In this position paper, we discussed the major challengers to this unification from the perspective of sensor developers (especially chemo-sensors) and integrating sensors data in real-world deployments. These challenges are:

1. The heterogeneity of data sources and data transport methods that all must neatly fit into the SSW.
2. Identifying the quality of the data.
3. Integrating data streams from different sources and modalities (esp. contextual information).
4. Analytics (e.g. decision making) may occur across the SSW.

These challenges were discussed in relation to current metadata standards, integrating data sources in the marine environment and in relation to a chemical analyzer. These challenges cannot be dealt with separately as we have seen in the marine environment that the heterogeneity of data sources makes integrating data streams from different sources and modalities extremely difficult, and makes analytics based on contextual information problematic. The identification data quality will also rely on contextual information that is difficult to automatically process given the heterogeneity of data sources. Thus heterogeneity of data sources (and data transport methods) is the core challenge but the other challenges must be dealt with for the SSW to offer a fully scaleable, integrated solution to environmental monitoring.

Acknowledgements
We gratefully acknowledge the financial support from Science Foundation Ireland under the CLARITY CSET award (07/CE/I1147), the Marine Institute for the Beaufort Marine Research Award in Environmental Sensing (BEAU-SENS-10) and Enterprise Ireland Technology Development Award (No: TD/08/111).

References

1. Manfred Hauswirth and Stefan Decker, "Semantic Reality - Connecting the Real and the Virtual World," Microsoft SemGrail Workshop, Redmond, Washington, June 21-22, 2007.
2. R. Ambrosio, “Internet-Scale Data Acquisition and Control Systems — Programming Paradigm Challenges”, Paper presented at the conference, Creating An Expanded DER Industry, November 28–30, Loews L’Enfant Plaza Hotel, Washington, DC. (2001).
3. D. Diamond, “Internet-scale sensing”, Anal Chem., 15, 278A-286A (2004)
4. Amit Sheth, Cory Henson, and Satya Sahoo, "Semantic Sensor Web," IEEE Internet Computing, July/August 2008, p.78-83.
5. Sensor Model Language (SensorML) for In-situ and Remote Sensors portal.opengeospatial.org/files/?artifact_id=11516
6. D. Diamond, S. Coyle, S. Scarmagnani, and J. Hayes, “Wireless Sensor Networks and Chemo-/Biosensing”, Chem. Rev., 108, 2, 2008, pp. 652-679
7. Sonia Ramirez-Garcia and Dermot Diamond. Internet-scale Sensing: Are Biomimetic Approaches the Answer?, Journal of Intelligent Material Systems and Structures, 18 (2) (2007) 159-164.
8. G. M. P. O’Hare, D. Diamond, K. T. Lau, J. Hayes, C. Muldoon, M. J. O’Grady, R. Tynan, G. Rancourt, H. R. Kolar and R. J. McCarthy, IBM Journal of Research and Development (2009), submitted for publication.
9. O’Reilly, T.C., et al., 2001: “Smart Network” infrastructure forthe MBARI Ocean Observing System, Proceedings of theOceans 2001 MTS/IEEE Conf., Honolulu, Hawaii, November5-8, 2001.
10. P. Alexander and R. Holman. Quantitative analysis of nearshore morphological variability based on video imaging. Marine Geology, 208(1):101{111, 2004.
11. Christopher W. Brown, Laurence N. Connor, John L. Lillibridge, Nicholas R. Nalli and Richard V. Legeckis. Remote Sensing of Coastal Aquatic Environments, Chapter 2, An introduction to satellite sensors, observations and techniques, 21-49. Springer, 2007.
12. Ocean Surface Topography Mission/Jason-2, http://www.nasa.gov/mission_pages/ostm/overview/index.html

13. MERIS Product Handbook, http://envisat.esa.int/handbooks/meris/
14. GODAE – Global Ocean Data Assimilation Experiment, http://www.godae.org/
15. GHRSST – Group for High Resolution Sea Surface Temperature, http://www.ghrsst-pp.org/index.htm
16. J. S. Myers and R. L. Miller. Remote Sensing of Coastal Aquatic Environments, Chapter 3, Optical Airborne Remote Sensing, 51-66. Springer, 2007.
17. M. Davidson, M. V. Koningsveld, A. de Kruif, J. Rawson, R. Holman, A. Lamberti, R. Medina, A. Kroon, and S. Aarninkhof. The coastview project: Developing video-derived coastal state indicators in support of coastal zone management. Coastal Engineering, 54(6-7):463-475, 2007.
18. E. O'Connor, A. F. Smeaton, N. E. O'Connor, and D. Diamond. Integrating multiple sensor modalities for environmental monitoring of marine locations. In SenSys '08: Proceedings of the 6th ACM conference on Embedded network sensor systems, pages 405{406, New York, NY, USA, 2008. ACM.
19. Common Data Index (CDI) - Metadata Format and full description of XML schema -version 2.04. http://www.sea-search.net/cdi_documentation/
20. MarineXML, http://www.iode.org/marinexml/
21. Tutorial 1: Using SensorML to describe a Complete Weather Station (2006). http://vast.uah.edu/SensorML/tutorial/SensorML%20Tutorial%201%20-%20Weather %20Station%20System.pdf
22. Keeley, R, Isenor A, Linguanti, J (2003), XML Bricks., http://ioc.unesco.org/marinexml/contents.php?id=19
23. Monterey Bay Aquarium Research Institute, http://www.mbari.org/ssds/ReferenceDocuments/MOOSMetadataSchema.xsd
24. Marine Information Mark-up Language, http://www.rdc.uscg.gov/iws/pubs/miml.pdf
25. Earth Science Mark-Up Language, http://esml.itsc.uah.edu/index.jsp
26. Ocean Biogeographic Information System, http://iobis.org//obis/obis.xsd
27. C.M. McGraw, S.E. Stitzel, J. Cleary, C. Slater and D. Diamond. Autonomous microfluidic system for phosphate detection, , Talanta 71 (2007) 1180–1185.
28. SmartBay, http://www.marine.ie/home/services/operational/SmartBay/
29. Muldoon, C., O'Hare, G.M.P., O'Grady, M., Tynan, R., Agent Migration and Communication in WSNs, 1st International Workshop on Sensor Networks and Ambient Intelligence, December 1-4, 2008, Dunedin, New Zealand.

Table 1. GF3 codes for a number of parameters captured by M3A buoys.

Parameter	Unit	GF3 codes
Air Temperature	Celsius	DRYT
Wind Speed	m/sec	WSPD
…	…	…

Table 2. BODC codes for 'Air temperature' using a dry bulb thermometer.

Parameter	Description	BODC codes
Air Temperature	"AirTemp":Temperature of the atmosphere by dry bulb thermometer	CDTBSS01
Air Temperature	"AirTemp":Temperature of the atmosphere by dry bulb thermometer	CDTASS02
Air Temperature	"AirTemp":Temperature of the atmosphere by dry bulb thermometer	CDTASS03

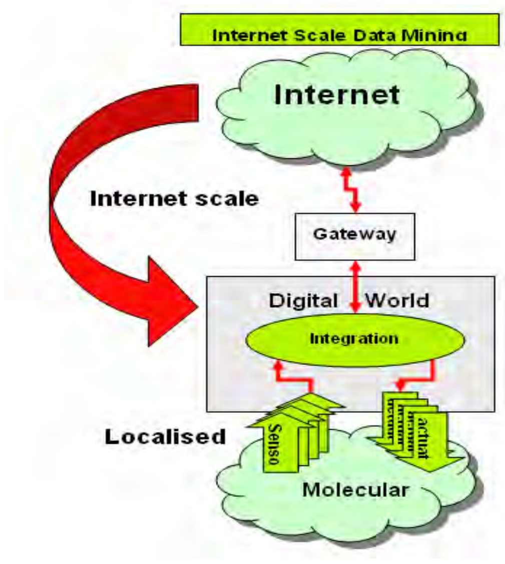


Fig. 1. Establish the chain. All analytical measurements must be linked to realize the concept of Internet-scale sensing. Localized control of important parameters is maintained, but the information is shared with external users via the Internet.

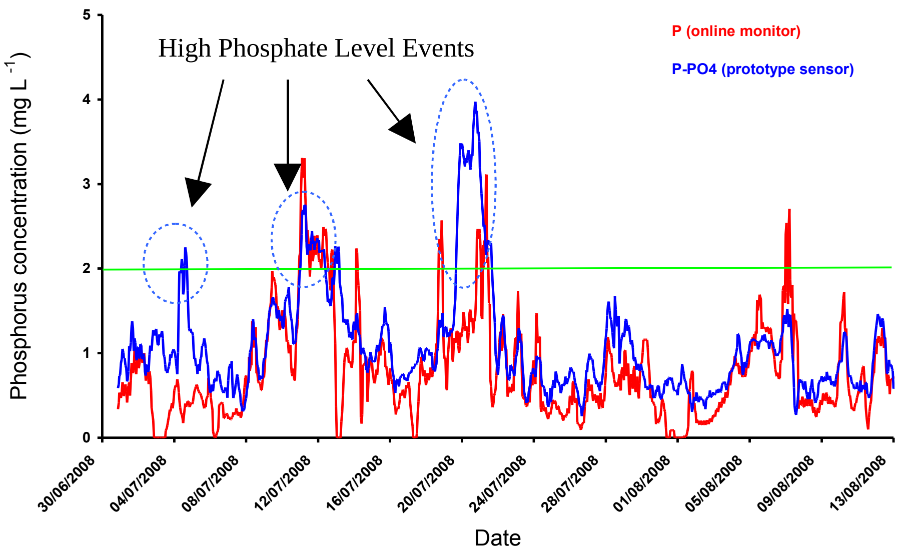


Fig. 2. Chemo-sensor in action: trial results. The phosphate levels of a waste water treatment plant obtained with a prototype analyzer during a 30+ day trial are shown. Comparable data were obtained by the plant's monitoring system for reference values (on-line monitor). The AgentFactory Micro Edition operating on the phosphate analyzer detects three high phosphate events based on the prototype analyzer data using the indicated threshold, which can set off a chain of remedial action. However, the first event is possibly a false positive as the reference system does not indicate high levels. Furthermore, high levels of phosphate are indicated towards the end of the trial by the reference monitor, but not by the prototype system, which is potentially a false negative.

An Ontological Representation of Time Series Observations on the Semantic Sensor Web

Cory A. Henson[1], Holger Neuhaus[2], Amit P. Sheth[1], Krishnaprasad Thirunarayan[1], and Rajkumar Buyya[3]

[1] Kno.e.sis Center, Department of Computer Science and Engineering
Wright State University, Dayton, OH 45435, USA
{cory, amit}@knoesis.org

[2] CSIRO Tasmanian ICT Centre
GPO Box 1538, Hobart, TAS, 7001, Australia
holger.neuhaus@csiro.au

[3] GRIDS Lab, Department of Computer Science and Engineering
University of Melbourne, Australia
raj@csse.unimelb.edu.au

Abstract. Time series observations are a common method of collecting sensor data. The Open Geospatial Consortium (OGC) Sensor Web Enablement (SWE) provides a standard representation for time series observations within the Observations and Measurements language, and therefore is in heavy use on the Sensor Web. By providing a common model, Observations and Measurements (O&M) facilitates syntax-level integration, but lacks the ability to facilitate semantic-level integration. This inability can cause problems with interoperability between disparate sensor networks that may have subtle variations in their sensing methods. An ontological representation of time series observations could provide a more expressive model and resolve problems of semantic-level interoperability of sensor networks on the Semantic Sensor Web. In this paper, such an ontology model is proposed, as well as a real-world use-case from sensor networks currently measuring rainfall in the South Esk river catchment in the North East of Tasmania, Australia.

Keywords: Observations and Measurements, Ontology, Semantic Sensor Web, Sensor Web Enablement, Time Series Observations

1 Introduction

Sensors are quickly becoming ubiquitous and can be found in a vast range of environments. Therefore, not surprisingly, there are multitudes of ways that sensors generate and represent observation data. Such differences may include the data formats, units of measurement, spatiotemporal resolution, domain of application, quality of observation, and the characteristics of the data over time, e.g. frequency, percentage of data loss, when data loss occurs, etc. All of these factors affect the integration of data from different sensors measuring phenomena.

This is equally true in the water resource management domain. In the Tasmanian South Esk river catchment, several sensor systems of different types are deployed for measuring rainfall. These sensors provide a data rich environment for continuous flow forecasting using Data Driven Modeling (DDM). When integrating data from different sources or mapping data to sensor (or measurement) information models, the semantics of the data need to be well understood. It is also important to register the semantics of shared data elements so that consumers of the data (any system designer, domain experts, and end users) can precisely determine the exact meaning of data occurring at interfaces between components of the information models. Of all the possible types of sensor data models, we focus on time series.

A time series is a sequence of observations which are ordered in time. A time series observation model is a common method of representing sensor data with a linear temporal order. As such, time series observations are utilized in a wide variety of fields such as statistics and signal processing for advanced analysis and forecasting. Many sensing systems on the Sensor Web use data collection methods that naturally lend themselves to representation as time series observations. Accordingly, the OGC Sensor Web Enablement (SWE) [1] provides a standard representation for time series observations within the Observations and Measurements (O&M) language [2]. O&M is an XML-based model for representing sensor observations on the Web. By providing a common model, O&M facilitates syntax-level integration, but lacks the ability to facilitate semantic-level integration. In this paper, we intend to show how time series observations can be modeled in an ontology that can (in future work) be used to overcome problems of integration and querying. One integration problem results from the fact that while different sensor networks may represent sensor observation data using a common model, they may use various sensing methods that are not explicitly represented. One query problem results from the necessity to know a-priori the sensing method used to generate a dataset (which, again, is not explicitly represented) in order to correctly interpret a query result. Both can be overcome through a semantic description of time series observations.

In order to make our discussion more clear, we will use descriptions of the sensor systems monitored by the CSIRO Tasmanian ICT Centre as a running example. As of this writing, there are twenty rain gauge sensor systems in Tasmania monitored by the Australian Commonwealth Scientific and Industrial Research Organization (CSIRO). The sensing systems at CSIRO adhere to the OGC-SWE standards and publish observation data in O&M. In particular, the rain gauge sensors publish rainfall observation data with the *om:TimeSeriesObservation* model (the *om* namespace is used to represent concepts in O&M). These rain gauge sensors collect rainfall in a bucket (or cup) and, when filled, the bucket tips and empties its contents. Because the system is aware of how much rainfall is required to fill the bucket, the rainfall level can be accurately recorded by monitoring when the bucket tipping events occur. Figure 1 shows an illustration of a rain gauge sensor [3].

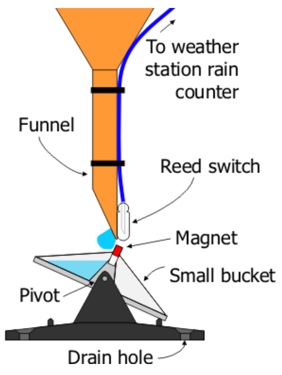


Figure 1. Illustration of a rain gauge sensor [3].

The remainder of the paper is organized as follows. Section 2 presents background material on the sensor network in the Tasmanian South Esk river catchment, the Sensor Web Enablement, and Semantic Web. Several different types of time series observations are introduced in Section 3. In Section 4, an ontological representation of time series observations is discussed. Finally, conclusions and future work are detailed in section 5.

2 Background

Scientists have long understood the importance of quality time series observations for conducting research and analyzing data. This is also true for the sensor network project in the South Esk river catchment in Tasmania. The models for time series observations, as described in this paper, are reliant on two sets of standardizations, (1) the Semantic Web languages defined by the World Wide Web Consortium (W3C), and (2) the Observations and Measurements (O&M) language defined by the Open Geospatial Consortium (OGC) Sensor Web Enablement (SWE). This combination is typical of applications on the Semantic Sensor Web [4][5].

2.1 Sensor Network in the Tasmanian South Esk River Catchment

Drought is a common problem that has been plaguing Australia for many years. The state of Tasmania is especially affected, with drought conditions worsened in 2008 and many areas reporting no significant rainfall for three years [6]. Consequently, water has become an exceptionally scarce resource. The inefficient management of water resources is exacerbated by a deficiency of quality information about Australia's water conditions. To overcome this problem, CSIRO has developed the 'Water for a Healthy Country' Flagship [7], a national research program addressing sustainable management of Australia's water resources. As part of this program, the CSIRO Tasmanian ICT Centre aims at establishing a technology platform to provide water information systems delivering dynamic, timely reporting and forecasting of water resources. This will be achieved through four key research areas that will [7]:

1. *Enable water information interoperability* through standards development, web service integration, semantic web, model interoperability.
2. *Improve the usability and availability of water data* through development in wireless and wired sensor networks, improved telemetry integration, novel hydrologic measurement techniques, data analysis and data assimilation methods.
3. *Develop next generation modeling and forecasting tools* through interoperable, modular computer models, advanced computing algorithms and powerful scenario planning tools.
4. *Develop improved reporting and visualization tools* through new interoperable and modular tools, products and technologies for operating, reporting and accounting of water resources at multiple scales.

The CSIRO Tasmanian ICT Centre is building a test bed system that attempts to incorporate sensors, models and data from multiple organizations operating within the South Esk Catchment [8]. The South Esk Catchment covers an area of approximately 3350 square kilometers and experiences widely varying climatic conditions with rainfall ranging from 500 mm in the low lying areas to 1500 mm in the highlands. Consequently, there is a high spatial variability in runoff yield [9]. Runoff yield is the quantity of water that travels over the land surface, through the soil, and groundwater, and is discharged into surface streams (i.e. the amount of water that leaves the catchment). There is an opportunity to improve water planning and management through continuous monitoring and forecasting of river flow. The project will explore how environmental sensors, hydrological models and decision support tools can be combined in a pluggable hydrological sensor web for continuous flow forecasting. A pluggable hydrological sensor web would have the ability to integrate any sensor into the web-based system without explicit re-configuration.

2.2 Sensor Web Enablement

The Open Geospatial Consortium established the Sensor Web Enablement as a suite of specifications related to sensors, sensor data models, and sensor Web services that will enable sensors to be accessible and controllable via the Web [1]. The following list describes the languages and service interface specifications of the SWE:

- *Observations & Measurements (O&M)* - Standard models and XML Schema for encoding observations and measurements from a sensor, both archived and real-time.
- *Sensor Model Language (SensorML)* - Standard models and XML Schema for describing sensors systems and processes; provides information needed for discovery of sensors, location of sensor observations, processing of low-level sensor observations, and listing of taskable properties.
- *Transducer Model Language (TransducerML)* - Standard models and XML Schema for describing transducers and supporting real-time streaming of data to and from sensor systems.

- *Sensor Observations Service (SOS)* - Standard web service interface for requesting, filtering, and retrieving observations and sensor system information. This is the intermediary between a client and an observation repository or near real-time sensor channel.
- *Sensor Planning Service (SPS)* - Standard web service interface for requesting user-driven acquisitions and observations. This is the intermediary between a client and a sensor collection management environment.
- *Sensor Alert Service (SAS)* - Standard web service interface for publishing and subscribing to alerts from sensors.
- *Web Notification Services (WNS)* - Standard web service interface for asynchronous delivery of messages or alerts from SAS and SPS web services and other elements of service workflows [1].

2.3 Semantic Web

The Semantic Web, as described by the W3C Semantic Web Activity, is an evolving extension of the World Wide Web in which the semantics, or meaning, of information on the Web is formally defined [10]. Formal definitions are captured in ontologies, making it possible for machines to interpret and relate data content more effectively. In this project, we use the Web Ontology Language (OWL) [11] to encode ontologies and the general purpose rule engine for the Jena Semantic Web Framework to encode rules [12].

2.4 Observations and Measurements Ontology

As mentioned in the introduction, time series observations are often encoded in O&M. Several attempts have been made in creating an ontological representation of O&M. Probst [13] performs an ontological analysis of the core O&M terms. Through this analysis, an OWL encoding of O&M is aligned with the DOLCE [14] foundational ontology. In a more recent attempt [5], the authors generate an OWL-DL encoding of O&M, called O&M-OWL, in order to reason over sensor data and infer complex features. The ontological representation of time series observations discussed in this paper uses O&M-OWL. The relationships discussed in Section 4.1 were originally described in [5] (with the exception of *om-owl:memberOf* and without the detailed RDF/XML serialization provided here). In order to avoid confusion, from this point forward we will refer to O&M in OWL as O&M-OWL and prefix concepts with the namespace *om-owl*, and refer to O&M in XML as O&M-XML and prefix concepts with the namespace *om-xml*.

3 Types of Time Series Observations

There are various ways to monitor, collect, and represent sensor data with time series observations. At the CSIRO Tasmanian ICT Centre, there are four distinct methods of monitoring rain gauge sensors, which can be divided along two dimensions: (1) cumulative vs. non-cumulative and (2) interval-based vs. event-based.

- *Cumulative* systems continually increment the observation result value as the monitoring progresses through time.
- *Non-cumulative* systems are not incremental and thus provide an independent value for each observation result.

- *Interval-based* systems generate observation result values at discrete points within a specified interval of time.
- *Event-based* systems generate observation result values only when a defined event occurs.

Interval-based/Non-cumulative systems generate independent observation result values at fixed time points. Each observation result value represents the amount of rainfall measured since the end of the previous interval. Figure 2 shows an example with fixed time points every thirty minutes from 1:00 AM to 3:00 AM. The vertical lines represent the fixed intervals and the dots represent observation result values that have measured rainfall. Each bucket tip event represents 0.2 mm of measured rainfall. So, from this example, we can see that between 1:00 AM and 1:30 AM, one bucket tip event occurred. No such events occurred between 1:30 AM and 2:00 AM. Two events occurred between 2:00 AM and 2:30 AM, and one between 2:30 AM and 3:00 AM.

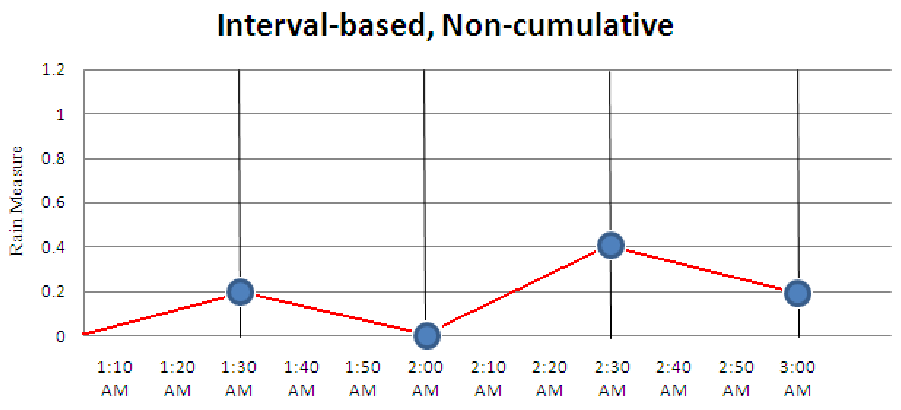


Figure 2. Interval-based, non-cumulative time series observation graph

Interval-based/Cumulative systems generate incremental observations result values at fixed time points. Each observation result value represents the cumulative amount of rainfall measured since the start of the process. Figure 3 shows an example with fixed time points every thirty minutes from 1:00 AM to 3:00 AM. The vertical lines represent the fixed intervals and the dots represent the incremental addition of observation result values measuring rainfall. So, from this example, we can see that between 1:00 AM and 1:30 AM, one bucket tip event occurred. Between 1:00 AM and 2:00 AM, still only one bucket tip occurred. Three events occurred between 1:00 AM and 2:30 AM, and four between 1:00 AM and 3:00 AM.

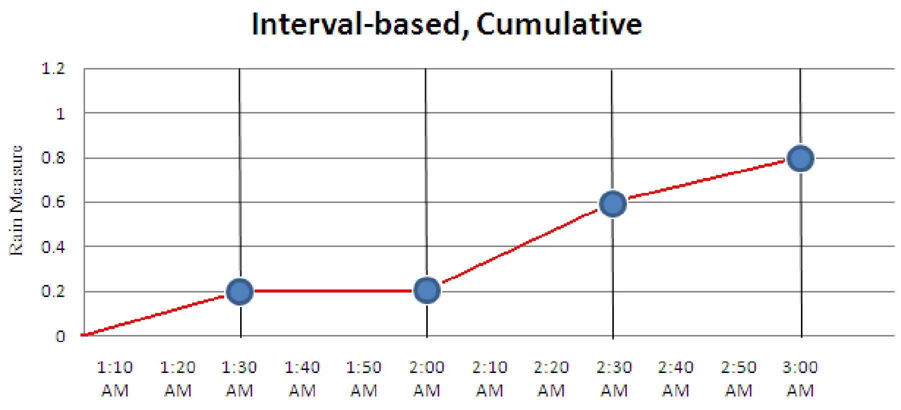


Figure 3. Interval-based, cumulative time series observation graph

Event-based/Non-cumulative systems generate independent observations result values whenever a defined event occurs. Each observation result value represents the amount of rainfall measured since the previous event. Figure 4 shows an example with a total time interval from 1:00 AM to 3:00 AM. The vertical lines represent bucket tip events and the dots represent observation result values that have measured rainfall. So, from this example, we can see that at 1:20 AM the first bucket tip event occurred, the second at 2:10 AM, the third at 2:20 AM, and the fourth at 2:50 AM.

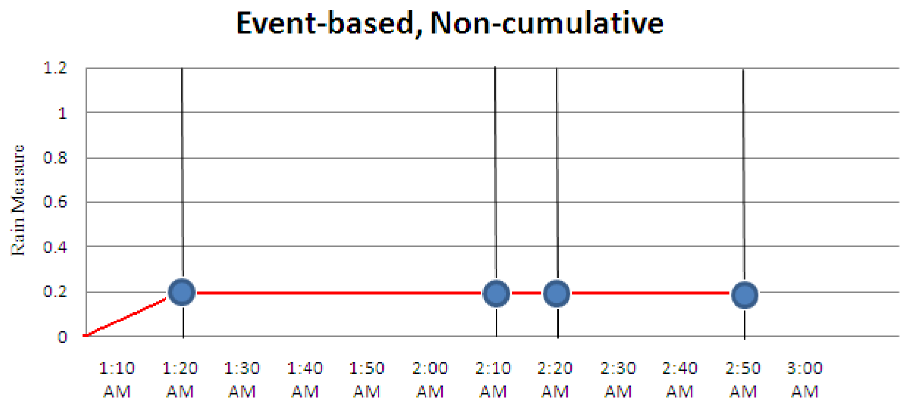


Figure 4. Event-based, non-cumulative time series observation graph

Event-based/Cumulative systems generate incremental observation result values whenever a defined event occurs. Each observation result value represents the cumulative amount of rainfall measured since the start of the process. Figure 5 shows an example with a total time interval from 1:00 AM to 3:00 AM. The vertical lines represent bucket tip events and the dots represent the incremental addition of observation result values measuring rainfall. So, from this example, we can see that at 1:20 AM the first bucket tip event occurred, the second at 2:10 AM, the third at 2:20 AM, and the fourth at 2:50 AM.

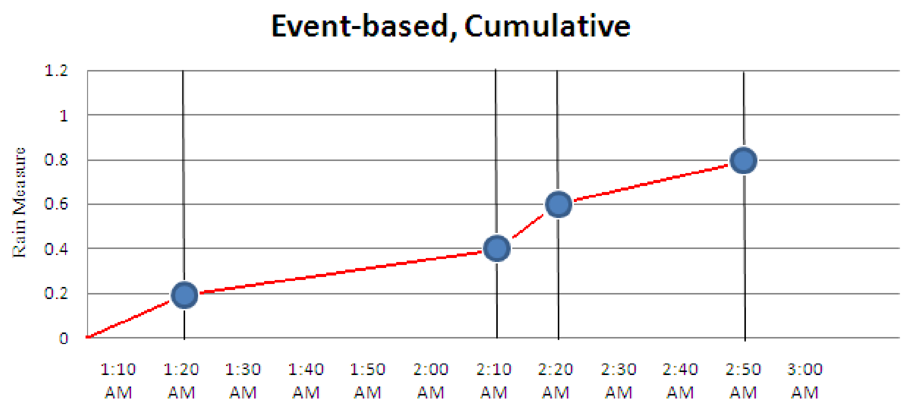


Figure 5. Event-based, cumulative time series observation graph

The authors admit there could be additional methods and categories; however, we hope these will be adequate and are sufficiently general for the current discussion. Table 1 shows how many rain gauge systems monitored by the CSIRO Tasmanian ICT Centre have the properties described above.

Table 1. Number of rain gauge systems with the selected properties.

	Non-cumulative	Cumulative
Interval-based	7	7
Event-based	3	3

4 Representation of Time Series Observations

A time series observation is a specialized observation collection. More specifically, if the member observations of an observation collection have the same feature of interest, the same observed property, and different sampling times, this set of observations may be represented as a time series observation whose sampling time is the period encompassing all the member times [2]. An example would include a rain gauge sensor that measures rain levels at discrete time intervals. In order to create an ontological representation of time series observations, there are three significant classes to be discussed: a class describing a basic observation (*om-owl:Observation*), a class describing an observation collection (*om-owl:ObservationCollection*), and a class describing a time series observation (*om-owl:TimeSeriesObservation*). Each of these classes defines properties. However, in comparison to an XML-based specification, such as O&M-XML, an ontology specification, such as O&M-OWL, enables the explicit representation of typing constraints on properties in terms of domain and range. This is exposed through the RDF/XML code below.

4.1 Observation Class (*om-owl:Observation*)

An observation is an act of observing a property or phenomenon, with the goal of producing an estimate of the value of the property [2]. O&M-OWL provides the following relationships for observations (with RDF/XML encoding):

- *om-owl:featureOfInterest* is a "representation of the observation target, being the real-world object regarding which the observation is made [2]." Example includes a coverage feature, such as the South Esk Catchment in Tasmania, Australia.

```
<owl:ObjectProperty rdf:about="#featureOfInterest">
       <rdfs:domain rdf:resource="#Observation"/>
       <rdfs:range rdf:resource="#Feature"/>
       <owl:inverseOf rdf:resource="#propertyValueProvider"/>
</owl:ObjectProperty>
```

- *om-owl:observedProperty* "identifies or describes the phenomenon for which the observation result provides an estimate of its value. It must be a property associated with the type of the feature of interest [2]." Example includes a rainfall property.

```
<owl:FunctionalProperty rdf:about="#observedProperty">
       <rdf:type rdf:resource=
              "http://www.w3.org/2002/07/owl#ObjectProperty"/>
       <rdfs:domain rdf:resource="#Observation"/>
       <rdfs:range rdf:resource="#PropertyType"/>
</owl:FunctionalProperty>
```

- *om-owl:samplingTime* is the "time that the result applies to the feature-of-interest [2]," or, in other words, it is the time when the phenomenon was measured in the real-world. Example includes a single instant sampling time at 5:00 am on Jan. 26, 2009.

```
<owl:FunctionalProperty rdf:about="#samplingTime">
       <rdf:type rdf:resource=
              "http://www.w3.org/2002/07/owl#ObjectProperty"/>
       <rdfs:domain rdf:resource="#Observation"/>
       <rdfs:range rdf:resource="#Time"/>
</owl:FunctionalProperty>
```

- *om-owl:observationLocation* is the location of an observation event; usually associated with the location of the sensor when an observation occurred (i.e., *om:samplingTime*). Example includes a single point observation location with latitude, longitude, and elevation coordinates.

```
<owl:FunctionalProperty rdf:ID="observationLocation">
       <rdf:type rdf:resource=
              "http://www.w3.org/2002/07/owl#ObjectProperty"/>
       <rdfs:domain rdf:resource="#Observation"/>
       <rdfs:range rdf:resource="#Location"/>
</owl:FunctionalProperty>
```

- *om-owl:result* is an "estimate of the value of some property generated by a known procedure [2]." Example includes a rain-level measurement result of 5.2 mm.

```
<owl:FunctionalProperty rdf:about="#result">
       <rdf:type rdf:resource=
              "http://www.w3.org/2002/07/owl#ObjectProperty"/>
       <rdfs:domain rdf:resource="#Observation"/>
       <rdfs:range rdf:resource="#ResultData"/>
</owl:FunctionalProperty>
```

- *om-owl:procedure* is a "description of a process used to generate the result. It must be suitable for the observed property [2]." Note that in this schema a sensor is defined as a type of process, along with other methods, algorithms, instruments, or systems of these. Example includes a rain gauge sensor as the procedure.

```
<owl:FunctionalProperty rdf:ID="procedure">
       <rdf:type rdf:resource=
              "http://www.w3.org/2002/07/owl#ObjectProperty"/>
       <rdfs:domain rdf:resource="#Observation"/>
       <rdfs:range rdf:resource="#Process"/>
       <owl:inverseOf rdf:resource="generatedObservation"/>
</owl:FunctionalProperty>
```

- *om-owl:memberOf* is a relation to a set of observations, or observation collection. Example includes a rainfall observation that is a member of a time series observation collection.

```
<owl:TransitiveProperty rdf:ID="memberOf">
       <rdf:type rdf:resource=
              "http://www.w3.org/2002/07/owl#ObjectProperty"/>
       <rdfs:domain rdf:resource="#Observation"/>
       <rdfs:range rdf:resource="#ObservationCollection"/>
       <owl:inverseOf rdf:about="#member"/>
</owl:TransitiveProperty>
```

4.2 Observation Collection Class (*om-owl:ObservationCollection*)

An observation collection is composed of a set of member observations [2]. O&M-OWL provides the following relationship for observation collections (with RDF/XML encoding):

- *om-owl:member* is a relation from an observation collection to a constituent observation (inverse of *om-owl:memberOf*). Example includes time series observation collection that has rainfall observations as members.

```
<owl:TransitiveProperty rdf:about="#member">
       <rdf:type rdf:resource=
              "http://www.w3.org/2002/07/owl#ObjectProperty"/>
       <rdfs:domain rdf:resource="#ObservationCollection"/>
       <rdfs:range rdf:resource="#Observation"/>
       <owl:inverseOf rdf:resource="#memberOf"/>
</owl:TransitiveProperty>
```

4.3 Time Series Observation Class (*om-owl:TimeSeriesObservation*)

In addition to being a specialized type of observation collection, a time series observation is also considered a type of observation. Therefore, *om-owl:TimeSeriesObservation* inherits properties from both *om-owl:Observation* and *om-owl:ObservationCollection* described above. While *om-owl:TimeSeriesObservation* is a sub-class of *om-owl:Observation*, it does not normally make use of the *om-owl:result* relationship. (It is conceivable that this property could be useful when modeling cumulative observation result values, however, this is not used in the current model for reasons to be detailed below.) On the other hand, *om-owl:samplingTime* is a very important property for *om-owl:TimeSeriesObservation*, whose sampling time is the period encompassing all the member times [2]. Remember that the sampling time of event-based systems is based on when an event occurred and the sampling time of interval-based systems is based on fixed-time points. In order to make this distinction explicit, we have created two sub-classes of *om-owl:TimeSeriesObservation*, including *om-owl:EventBasedTimeSeriesObservation* and *om-owl:IntervalBasedTimeSeriesObservation*, and two sub-properties of *om-owl:samplingTime*, including *om-owl:eventBasedSamplingTime* and *om-owl:intervalBasedSamplingTime*.

```
<owl:ObjectProperty rdf:about="#samplingTime">
      <rdfs:domain rdf:resource="#Observation"/>
      <rdfs:range rdf:resource="#Time"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="eventBasedSamplingTime">
      <rdfs:subPropertyOf rdf:resource="#samplingTime"/>
      <rdfs:domain rdf:resource="#EventBasedTimeSeriesObservation"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="intervalBasedSamplingTime">
      <rdfs:subPropertyOf rdf:resource="#samplingTime"/>
      <rdfs:domain rdf:resource="#IntervalBasedTimeSeriesObservation"/>
</owl:ObjectProperty>
```

From the O&M specification, we know that a time series observation is a specialization of an observation collection with the restriction that all member observations must share the same feature of interest and the same observed properties

[2]. Such constraints are difficult to represent in an XML encoding. In O&M-XML, these constraints are simply implied through the wording of the specification with the intention that implementations will faithfully adhere to the intended definition. While difficult for XML representations, such constraints may be naturally represented in an OWL-DL ontology using the OWL property restrictions. In the code below, we show an observation sub-class, *csiro:SouthEskCatchmentRainGuageObservation*, which contains the restriction that all instantiated observations of this type have an observed property *csiro:rainfall* and a feature of interest *csiro:SouthEskCatchment* through the *owl:hasValue* restriction. (The *csiro* namespace is used in an extension of O&M-OWL with concepts targeted toward the CSIRO Tasmanian ICT Centre's sensing systems).

```
<owl:Class rdf:about=
    "http://www.csiro.au#SouthEskCatchmentRainGuageObservation">
    <rdfs:subClassOf rdf:resource="#Observation"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#observedProperty"/>
            <owl:hasValue rdf:resource="http://www.csiro.au#rainfall"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#featureOfInterest"/>
            <owl:hasValue rdf:resource=
                "http://www.csiro.au#SouthEskCatchment"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

In addition, a time series observation sub-class, *csiro:SouthEskCatchmentRainGuageTimeSeriesObservation*, contains the restriction that all instantiations of this type of time series observation have all member observations of type *csiro:SouthEskCatchmentRainGuageObservation* through the *owl:allValuesFrom* restriction.

```
<owl:Class rdf:about=
"http://www.csiro.au#SouthEskCatchmentRainGuageTimeSeriesObservation”>
    <rdfs:subClassOf rdf:resource=”#TimeSeriesObservation”/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#member"/>
            <owl:allValuesFrom rdf:resource=
        "http://www.csiro.au#SouthEskCatchmentRainGuageObservation"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

Through this combination of OWL property restrictions, we are able to more faithfully and explicitly represent the concept of *om:TimeSeriesObservation*.

4.4 Generating Time Series Observation Instances

Given the heavy use of O&M-XML on the Sensor Web, it seems reasonable that translating O&M-XML documents into O&M-OWL instances could be a popular means of populating the ontology knowledge base. When generating instances of *om-owl:TimeSeriesObservation*, several property values can be directly translated from corresponding O&M-XML documents, including: *om-owl:featureOfInterest*, *om-owl:observedProperty*, *om-owl:samplingTime*, *om-owl:observationLocation*, and *om-owl:procedure*. The individual *om-owl:Observation* instances share many property values with the *om-owl:TimeSeriesObservation* instance of which they are related through the *om-owl:memberOf* relation. Several of these shared properties can be directly propagated given that a *om-owl:member* relation holds from an instance of *om-owl:TimeSeriesObservation*. The relations that may be propagated include *om-owl:featureOfInterest*, *om-owl:observedProperty*, *om-owl:observationLocation*, and *om-owl:procedure*. This translation of property values can be encoded in a set of rules. As an example, the rule for propagating *om-owl:featureOfInterest* follows (in Jena rule engine syntax [9]):

```
[PropagateFeatureOfInterestRule:
       (?tso  rdf:type  om-owl:TimeSeriesObservation)
       (?tso  om-owl:member  ?obs)
       (?tso  om-owl:featureOfInterest  ?foi)
→(?obs  om-owl:featureOfInterest  ?foi)]
```

The other translatable property values have similar rules which we omit for the sake of brevity. The two remaining relations of *om-owl:Observation* to be instantiated include *om-owl:samplingTime* and *om-owl:result*. Sampling time for instances of *om-owl:Observation* can be directly translated from *om-xml:samplingTime* of the *om-xml:TimeSeriesObservation*. The instantiation of *om-owl:result* relation is more involved since the cumulative observation result values are dependent on previous observations, and we want to generate an independent representation for all observations. In order to accomplish this, we simply convert the cumulative result values into non-cumulative result values. Unlike the conversion of *om-owl:samplingTime*, *om-owl:result* can be translated without loss of expressiveness since the cumulative result can always be recalculated. Therefore, there is no need to create sub-classes of *om-owl:ResultData* nor sub-properties of *om-owl:result* in order to explicitly represent the cumulative/non-cumulative distinction. The conversion of cumulative result values into non-cumulative result values is a straightforward process of subtracting from each observation the result values of those observations that were generated at a previous time point (either at a fixed time point, or when an event occurred).

5 Conclusion and Future Work

The Semantic Sensor Web aims to integrate Semantic Web technologies with sensing systems in order to provide more expressive representation, enhanced analysis, and improved access and discovery of sensor data on the Web. In this paper, we present an ontological representation of time series observations that could add much value to time series sensor data on the Semantic Sensor Web.

In the future we hope to utilize this ontology to provide advanced query and manipulation of time series observations. Previously, queries of time series observations could only return data formatted in the same manner in which it was collected. We believe that by leveraging an ontological representation of time series observations, we may allow for automatic conversion of event-based time series observation to interval-based time series observation, and vice-versa. For example, a user could query against an event-based system and receive an interval-based time series observation as a result. At the CSIRO Tasmanian ICT Centre, a practical use of this representation would be to enable the automated conversion of such observations for input into forecast models, which may, for example, require a time series observation with daily frequency of a given phenomenon which is only available as an hourly measurement. The required conversion methods could be encoded in the time series ontology. In addition, a set of operations on time series observations, such as union, concatenation, and intersection, would be useful for advanced integration. And finally, since time is such an obviously important component of time series observations, we intend on integrating this ontology with OWL-Time [15], a W3C recommended ontology based on temporal calculus that provides descriptions of temporal concepts such as *instant* and *interval*, and the relations between them.

We believe that an ontological representation of time series observations is an important addition to the Semantic Sensor Web, and the practical use of this representation at the CSIRO Tasmanian ICT Centre provides a much needed experimental platform for future investigation into the integration of Semantic Web technologies with sensing systems.

Acknowledgments. This research was supported in part by The Dayton Area Graduate Studies Institute (DAGSI), AFRL/DAGSI Research Topic SN08-8: "Architectures for Secure Semantic Sensor Networks for Multi-Layered Sensing." We also thank Andrew Pratt and the sensor network project team at the CSIRO Tasmanian ICT Centre for their contribution towards a better understanding of the use-case scenario involving rain gauge sensors. The Tasmanian ICT Centre is jointly funded by the Australian Government through the Intelligent Island Program and CSIRO. The Intelligent Island Program is administered by the Tasmanian Department of Economic Development and Tourism.

6 References

1. Botts, M., et al.: OGC Sensor Web Enablement: Overview and High Level Architecture (OGC 07-165). Open Geospatial Consortium white paper, 28 Dec. 2007.
2. Observations and Measurements (O&M), http://www.opengeospatial.org/standards/om
3. Rain Gauge, http://www.weatherhut.com/site/1298901/LearningCenter/RainGauge.html
4. Sheth, A., Henson, C., and Sahoo, S.: Semantic Sensor Web. IEEE Internet Computing, July/August 2008, p. 78-83.

5. Henson, C., Pschorr, J., Sheth, A., and Thirunarayan, K.: SemSOS: Semantic Sensor Observation Service. International Symposium on Collaborative Technologies and Systems (CTS2009), Workshop on Sensor Web Enablement (SWE2009), Baltimore, Maryland, 2009.
6. Drought in Australia, http://en.wikipedia.org/wiki/Drought_in_Australia
7. Water for a Healthy Country Flagship, http://www.csiro.au/org/WfHC.html
8. Guru, S.M., Taylor, P., Neuhaus, H., Shu, Y., Smith, D., Terhorst, A.: Hydrological Sensor Web for the South Esk Catchment in the Tasmanian state of Australia. 4th IEEE International Conference on e-Science, 7-12 December 2008, Indianapolis, Indiana, USA.
9. D. P. I. W. Water Assessment Branch: Surface Water Hydrology of the South Esk River Catchment. Technical Report. Tech. Rep. WA 07/02, 2007.
10. W3C Semantic Web Activity, http://www.w3.org/2001/sw/
11. Web Ontology Language (OWL), http://www.w3.org/TR/owl-ref/
12. Jena Semantic Web Framework, http://jena.sourceforge.net/
13. Probst, F.: An Ontological Analysis of Observations and Measurements. 4th. International Conference on Geographic Information Science (GIScience), Munster, Germany, 2006.
14. Masolo, C., et al.: WonderWeb Deliverable D18, Ontology Library (final). http://www.loa-cnr.it/Papers/D18.pdf, 2003
15. Time Ontology in OWL (OWL-Time), http://www.w3.org/TR/owl-time/

Flexible Resource Assignment in Sensor Networks: A Hybrid Reasoning Approach

Geeth de Mel[1], Murat Sensoy[1], Wamberto Vasconcelos[1], and Alun Preece[2]

[1] Department of Computing Science, University of Aberdeen,
Aberdeen AB24 3UE, Scotland, United Kingdom
{g.demel,m.sensoy,w.w.vasconcelos}@abdn.ac.uk

[2] Cardiff School of Computer Science, Cardiff University,
Queen's Buildings, 5 The Parade, Roath,
Cardiff CF24 3AA, United Kingdom
A.D.Preece@cs.cardiff.ac.uk

Abstract. Today, sensing resources[3] are the most valuable assets of critical tasks (e.g., border monitoring). Although, there are various types of assets available, each with different capabilities, only a subset of these assets is useful for a specific task. This is due to the varying information needs of tasks. This gives rise to assigning useful assets to tasks such that the assets fully cover the information requirements of the individual tasks. The importance of this is amplified in the intelligence, surveillance, and reconnaissance (ISR) domain, especially in a coalition context. This is due to a variety of reasons such as the dynamic nature of the environment, scarcity of assets, high demand placed on available assets, sharing of assets among coalition parties, and so on. A significant amount of research been done by different communities to efficiently assign assets to tasks and deliver information to the end user. However, there is little work done to infer sound alternative means to satisfy the information requirements of tasks so that the satisfiable tasks are increased. In this paper, we propose a hybrid reasoning approach (viz., a combination of rule-based and ontology-based reasoning) based on current Semantic Web[4] technologies to infer assets types that are necessary and sufficient to satisfy the requirements of tasks in a flexible manner.

Key words: Sensors, Platforms, Resource Assignment, Semantic Web, Rules, Hybrid Reasoning

1 Introduction

A sensor network [1] is a collection of heterogeneous sensing resources[3], composed of sensors and platforms. Sensors capture phenomena whereas platforms provide the durability, mobility, communication capabilities, and so, on to the mounted sensor(s). Advances in technology have made the deployment of sensor networks

[3] A sensing resource (henceforth referred to as an "asset") is a platform which contains one or more sensors.

[4] http://www.w3.org/2001/sw/

a robust and viable solution to reliably monitor and obtain timely, continuous, and comprehensive observations about dynamic situations [17, 19]. Therefore, for many critical tasks like border monitoring or surveillance, selection of sensing assets for tasks play a key role in their success or failure. This leads to the problem of assigning proper assets to tasks such that the assigned assets cover the information needs of the individual tasks.

Effective and efficient assignment of assets to such tasks is an important but computationally hard problem in sensor networks domain. The difficulty of this problem is amplified in the intelligence, surveillance, and reconnaissance (ISR) domain, and especially in a coalition context, where the assets belonging to different parties are shared to archive tasks. This is due to a variety of reasons. First, the environments in which these resources are deployed could rapidly change (i.e., new high-priority tasks emerge, assets become unreliable, weather conditions change, and so on) yielding new information requirements or assets requirements. Second, the demand placed on available assets typically exceeds the inventory [14] resulting in complex assignment choices. Last but not the least, the inability to obtain a bird's-eye view of the available assets to tasks makes it impossible to perform assignments in an informed manner. All these reasons imply the necessity to infer sound alternative means to satisfy the information requirements of tasks so that the different capabilities provided by assets can be used to cover the information requirements of tasks properly, thus increasing the number of satisfiable tasks.

Many communities have investigated the assignment problem and proposed different mechanisms that could be applied to solve it. Some of these approaches rely on having a *human in the loop* to decide which assets are appropriate to satisfy the requirements of tasks [4] whereas other approaches have tried to automate the assignment process [5, 13, 22]. However, these automated approaches are highly constrained in terms of their assumptions. For example, the work discussed in [5] assumes an unlimited inventory of assets, whereas [13] assumes assets to be of the same type (i.e., any assets could provide some utility to a task). This is not the case in general and especially in the environments highlighted above. Assets are heterogeneous (different capabilities, operational conditions etc.) by nature and only suitable for particular tasks.

Most of the current approaches have ignored important qualitative attributes such as the capability provided by assets, prevailing weather conditions, etc. These attributes play a major role in deciding which assets could be deployed to achieve the information needs of tasks. Moreover, important many-to-many relationships between assets and tasks (i.e., a task could be accomplished in several different ways; an asset could be used to achieve several different kinds of tasks) are not considered. We argue that considering these relationships allows agile management of information providing assets by enabling reasoning about different capabilities of assets and requirements of tasks.

In this paper, we propose knowledge-rich models and mechanisms based on Semantic Web[4] technologies to address the issues highlighted above. We propose a rule-based system to infer multiple capabilities that could be used to satisfy the information requirements of tasks. We then discuss an ontology-based reasoning framework to identify suitable asset types that meet those identified capabilities, thus increasing the flexibility of the assignment. We present tools that are built around these models to assist the decision makers in the assignment process in order to identify suitable asset types for tasks. The proposed system not only recommends asset types in an agile manner but also guarantees the soundness of the solution inference process.

The rest of this document is structured as follows. In section 2, we survey related research done in sensor networks and other domains that have inspired our work. Section 3 introduces a rule-based system that enables inference of different capabilities to satisfy the goals of tasks and gives some example outputs from the rule system. In section 4, we highlight an ontology-based matchmaking framework to infer sound solutions to the assignment problem based on the capabilities provided by the assets and the requirements advertised by the tasks. A case study, applying out approach is illustrated in section 5 and we conclude in section 6, also providing future directions for this work.

2 Related Work

As stated previously, different communities have proposed a variety of approaches to solve the problem of assigning assets to tasks. These approaches can be grouped and summarized as follows:

Algorithmic Approaches. Many approaches have proposed a utility-based solution with heuristics-based enhancements. For example, in [5] Byers and Nasser propose a framework to solve the assignment problem based on energy conservation to maximize the utility of a sensor network while keeping the cost of the assignment per task under a pre-defined budget. Johnson *et al.* propose an energy-aware approach to select assets for tasks in both static and dynamic environments [13] for competing tasks. One major drawback in these approaches is the fact that all assets are assumed to be of the same type. We argue that this is not the general case. Assets are heterogeneous (different capabilities, operational conditions etc.) by nature and only suitable for particular tasks.

In [22], Tatton proposes an approach to optimize the assignment of assets to task based on probability of target detection. In [8], Doll has further extended the *sensor allocation model* by introducing notion of probability of line of sight and field-of-view to the model in order to better estimate the asset performance. The drawback of these approaches is the assumption that there exists a classification that pre-identifies assets being suitable for some particular tasks in order to perform the assignment.

Semantic-based Approaches. In [23], Whitehouse *et al.* propose a framework based on semantics to allow users to perform declarative queries over a sensor network (i.e., rather than querying raw data, users query whether a vehicle is a car or a truck). A major drawback in this approach is the fact that all the desired inference units must be declared for a sensor network before users can start using the system. This is difficult, if not impossible, for a heterogeneous sensor network deployed in a dynamic situation. Also the declarative language described in the work is not standardised (i.e., the language is described using Prolog [6] predicates) which hinders the extensibility of the system.

Recent research has considered standardised descriptive schema representations (e.g., XML [21], RDFS[5], OWL [7]) to assist in assets-to-tasks assignment [4]. The keystone of this approach is to have standardised schemas to describe assets, asset properties, and requirements. There is already a significant amount of work done in this area, for example XML-based approaches such as the OpenGeospatial Consortium (OGC)[6] suite of Sensor Web Enablement (SWE) [4] specifications to ontologies such as OntoSensor [10], the Marine Platforms Ontology [3], etc.

Lack of semantics in SensorML [18] (i.e., descriptions of assets and their capabilities are in plain text) makes it difficult to be used in automated capability inference mechanisms. *OntoSensor* [10] was created to assist in semantic data fusion. Therefore, a great deal of emphasis has been put on modelling the data from assets, but not their functional aspects. Hence, it cannot also be used as it is in capability inferences.

The proposed approach builds upon the existing standards and mechanisms for knowledge representation and reasoning in order to enable semantic-aware assignment of assets to tasks. In the next section we propose a knowledge-based rule system to address the issue of inferring different capabilities that can satisfy the same information requirements of tasks.

3 Agile Inference of Capabilities: A Rule-based Approach

In an environment where there are many-to-many relationships between tasks and assets, it is prudent to allow tasks' requirements to be specified in manner that is independent of specific capabilities of assets. For example, in a surveillance task rather than asking for *infrared capability* one could specify the information requirement for *detecting vehicles*. Let us assume that, according to the available inventory, detecting a vehicle could be done with infrared, radar, or acoustic capabilities, thus, yielding multiple degrees of freedom in (re)assignment of assets to tasks.

We propose a rule-based system to address this issue. The proposed system allows users to describe what they want to achieve (e.g., detect vehicles, identify a particular building, etc.) and use the rule-based system to infer the different capabilities that could be used to achieve tasks. In order to infer the required capabilities to achieve tasks, tasks must be formalised with respect to the capabilities that are required to achieve them. There are many knowledge corpora that provide adequate information about the different capabilities required to achieve the same task. In the next subsections, we discuss one of these knowledge corpora and show how we have formalised it so that different capabilities can be inferred to satisfy the same task.

[5] http://www.w3.org/TR/rdf-schema/
[6] http://www.opengeospatial.org

3.1 National Image Interpretability Rating Scale (NIIRS)

NIIRS[7] is an approach embraced by intelligence and civilian communities to express the information potential of different image types [12]. NIIRS is defined for visible, infrared, radar, and multispectral imagery, providing a 10-level scale with each level containing several interpretation tasks or criteria. Within each spectrum higher NIIRS levels inherit the criteria of their subordinates. For example, with a NIIRS-3-rated image one can satisfy criteria set out by NIIRS 1 and 2.

The criteria indicate the expressivity of an image in terms of the amount of information that could be extracted from it at the given scale. For example, with visible NIIRS 4, identification of individual tracks is possible whereas with an image of visible NIIRS 6, identification of a vehicle is made possible (i.e., the make/model of the vehicle can be identified). Additionally a task can be achieved using different spectra with different NIIRS values. For example, detecting a large aircraft could be done with infrared and visible imagery using ratings 2 and 3 respectively. The image classification criteria could be broadly categorized as detect[8], distinguish[9], and identify[10]

In section 3.3 we show how we formalised the NIIRS knowledge corpus. In order to formalise NIIRS, first we need to come up with a classification of the elements in the environment. In the next section, we discuss a possible representation for this classification using an OWL-DL ontology.

3.2 Detectable Ontology

Let us first introduce the notion of *detectable*: detectable are the objects (e.g., vehicles, building, people and so on) of interest. For example let us take the task *detect large buildings (e.g., hospitals, factories).* In this case detectables can be classified as buildings. Let us take another example task *detect individual vehicles in a row at a known motor pool.* Vehicles are the detectables in this example.

We have created a detectables ontology to represent these concepts. Figure 1 shows a fragment of the taxonomies we have developed. We have used these concepts in the formalization of the criteria described in NIIRS, as we explain in Section 3.3. As Figure 1 shows, detectable concepts are broadly categorized into *Area*, *Component*, *Equipment*, *LinesOfTranspotation*, *Platform*, *Sensor*, and *Structure*. We have classified other detectable concepts as subclasses of these main concepts. A *Car* which is a subconcept of *WheeledVehicle* is a *GroundPlatform* (i.e., Figure 1(b)). *SiteConfiguration* represents a collection of buildings whereas *SiteComponent* refers to an individual building such as *Pier*, *Hanger*

[7] http://www.fas.org/irp/imint/niirs.htm

[8] Ability to find or discover the presence of an item of interest, based on its general shape, contextual information, etc.

[9] Ability to determine that two detected objects are of different types or classes based on one or more distinguishing features

[10] Ability to name an object by type or class, based primarily on its configuration and detailed components

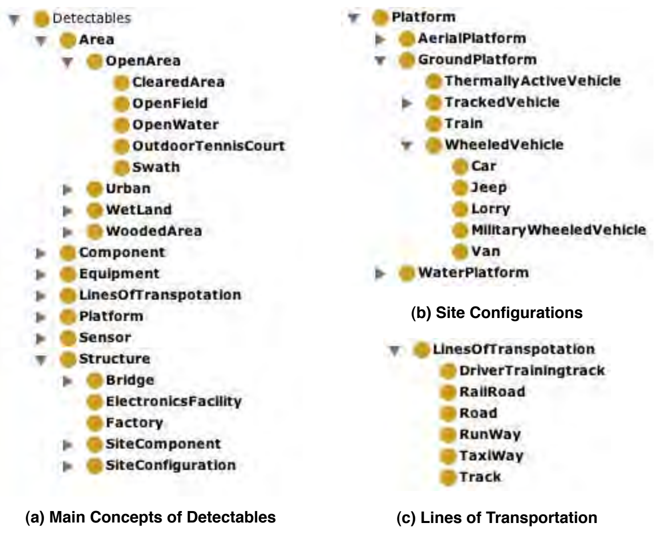


Fig. 1. Taxonomy of "Detectables"

or part of a building such as *BoilerHall*. *Factory* is not classified under either of them since it could be a single building structure or a multiple building configuration. Also we have introduced an object property *hasFeature* to describe the distinctive features of *Detectables*. For example, piers and hangars are both detectable concepts but they also are parts of a port, which makes them features of the port. Furthermore, this classification helps us formally define the concepts *detectable*, *distinguishable*, and *identifiable*:

1. **Detectable**: If the concept of interest has any sub-concept then it is detectable (e.g., *WheeledVehicle*).
2. **Distinguishable**: If a set of concepts are detectable, then they are also distinguishable. For example, if we detect a *Jeep* and a *Car*, then we can distinguish between them based on their shape.
3. **Identifiable**: If the concept of interest has no sub-concepts, then it is identifiable. For example, one can say a *SAAB 9-3 sedan* is identifiable.

3.3 Formalisation of Interpretation Tasks

We define a criterion as a 6-element tuple $FIT(T, W, F, C, I, V)$, where T represents the type of the *interpretation task* to perform (e.g., detect, distinguish, identify, and so on); I is the type of capability/intelligence (e.g., imagery spectra in NIIRS) that could be used to perform the interpretation task; $W = \{w_1, w_2, \ldots, w_i\}$ is a set of detectables (e.g., {*port, hospital*}) that can be observed using the capability/intelligence I; $F = \{f_1, f_2, \ldots, f_j\}$ is a set of features (e.g., {*pier, warehouse, loading bay, ambulance*}) describing W; C represents the context of the detectables; V is a numeric value that represents the quality of the intelligence (e.g., the rating of an imagery source in NIIRS). Below we provide examples of this formalism based on NIIRS criteria.

With an image rated *Visible NIIRS 1*, one can *detect* a medium-sized port facility and/or distinguish between taxi-ways and runways at a large airfield [12]. So, from this criterion, we can derive if there is a port facility in the image then one can detect it. Also according to the *Radar NIIRS 1*, one can *detect* a port facility based on its features (i.e., presence of piers and warehouses). Example 1 and Example 2 shortly describe how tasks could be presented in our formalism to exploit $features$ and *context* of criteria while inferring different capability.

Example 1 The task of detecting a port can be formalised as *FIT(detect, {Port}, {}, {}, image(Visible), 1)*. In this case, a reasoner can infer detection of a port can be achieved by using *Visible NIIRS 1*. However, in many cases, using explicit features of ports (e.g., piers and warehouses), we can detect objects more accurately. Therefore, the representation *FIT(detect, {Port}, {Pier,Warehouse}, {}, image(Radar), 1)* allows a reasoner to use some explicit features of a port while detecting it.

Example 2 Some tasks are highly sensitive to the context. For example, distinguishing between a taxiway and a runway using imagery intelligence can only be achieved if the context of the task enables clear images to be taken. If the context is *airfield*, which means that the task will be executed over an airfield, it is possible to distinguish between a taxiway and a runway. This can be represented as *FIT(distinguish, {Taxiway,Runway}, {}, {AirField}, image(Visible), 1)*. Similarly, to detect individual vehicles in a row at a known motor pool using radar intelligence, we have *FIT(detect, {Vehicle}, {}, {Motor-Pool}, image(Radar), 4)*.

We believe the proposed FIT formalism can be used to formalise knowledge from other intelligence domains too. For example, Guo *et al.* [11] propose an approach to detect and distinguish vehicles based on their *acoustic signatures*. Therefore, detect and distinguish tasks in our framework can also be formalised using *acoustic signatures* instead of NIIRS. In this case, if an acoustic signature of value 5 enables us to detecting a vehicle, we should formalise our statement as *FIT(detect,{Vehicle},{},{},5,Acoustic)*.

An extensive knowledge base has been created using the representation above by formalizing the NIIRS corpus. In the next section, we present a set of rules that are implemented to draw conclusions from this knowledge base to find diverse but feasible set of capabilities to perform a task. This makes the assignment of assets to tasks more flexible and agile; we reason about multiple ways in which assets can satisfy the requirements of a task.

3.4 Rules to Derive Capabilities

In this section, we present a set of rules to make inferences from the created knowledge base using the FIT formalism. These rules derive minimal, but necessary and sufficient capabilities needed to achieve a particular task. For example, let X be a set of objects that need to be observed. Detecting an element $x_i \in X$ is defined using the rules below.

$$\begin{aligned}
detect(x_j, i_j, v_j) &\leftarrow distinguish(x_j, i_j, v_j) && (1)\\
distinguish(x_j, i_j, v_j) &\leftarrow identify(x_j, i_j, v_j) && (2)\\
detect(x_j, i_j, v_j) &\leftarrow FIT(detect, w, f, c, i_j, v_j) \ \wedge \ x_j \in w && (3)\\
identify(x_j, i_j, v_j) &\leftarrow FIT(identify, w, f, c, i_j, v_j) \ \wedge \ x_j \in w && (4)\\
distinguish(x_j, i_j, v_j) &\leftarrow FIT(distinguish, w, f, c, i_j, v_j) \ \wedge \ x_j \in w && (5)
\end{aligned}$$

These rules can be interpreted as follows. Rule 1 states that the object of interest x_i can be detected using intelligence i_j and the quality of intelligence v_j (corresponds to ratings in $NIIRS$ terminology) if it can be distinguished using i_j and v_j. Similarly, Rule 2 states that x_i can be distinguished using i_j and v_j if it can be identified using i_j and v_j. Rules 3, 5 and 4 state that you can detect, identify or distinguish an object x_i if you can find a related FIT statement in which x_i is a member of the set w of detectables declared in the statement.

3.5 Example Results

We have developed a proof-of-concept prototype using CIAO Prolog[11] to show how these rules draw conclusions from the knowledge base. For this purpose, we first query the system for required capabilities of the tasks *detect*, *distinguish*, and *identify*. Then, in this section we summarize the inferred capabilities. For example, a query to *detect a large airplane* returns the following result set.

```
?- detect(largeAirliner,Results).
Results = [(image(infrared),2),(image(radar),2),(image(visible,3))]
```

The inferred solution recommends three capabilities that could be used to perform the task using one of *visible*, *infrared*, or *radar* imagery with a minimum $NIIRS$ of 3, 2, and 2 respectively. However, *detection of a small airplane* can only be achieved using an infrared imagery with a minimum $NIIRS$ of 3.

```
?- detect(smallAirliner,Results).
Results = [(image(infrared),3)]
```

Therefore, according to the definitions of the interpretation tasks, *distinguishing between a large plane and a small plane* could only be done using infrared image with a minimum $NIIRS$ of 3. This is because, infrared $NIIRS$ 3 is the smallest common denominator in the above two queries to detect a large plane and a small plane. Below is the result of the query that confirms the expected result.

```
?- distinguish([largeAirliner,smallAirliner],Results).
Results = [(image(infrared),3)]
```

[11] http://clip.dia.fi.upm.es/Software/Ciao/

4 Capability-Requirement Matching

In [9, 16], we proposed the *Sensor Assignment to Missions* (SAM)[12] framework to improve asset-to-task assignments based on current Semantic Web technologies together with semantic matchmaking [15]. The core of the approach is a set of interlinking ontologies to describe scenarios (i.e., missions, operations, tasks), assets (i.e., sensors and platforms), capabilities of the assets, and the requirements of the tasks. These ontologies are represented in OWL DL [7].

This approach was inspired by the Missions and Means Framework (MMF) [20]. MMF was developed by the US Army Research Laboratory to provide means for specifying a military mission in order to evaluate the utility of alternative means (i.e., assets) to accomplish the goals. Based on MMF we have defined an architecture to infer the types of assets that are fit for the purpose (i.e., can meet the information requirements of the task). We use semantic reasoning and a matchmaking mechanism to derive these asset types. Figure 2 depicts the architecture of the system.

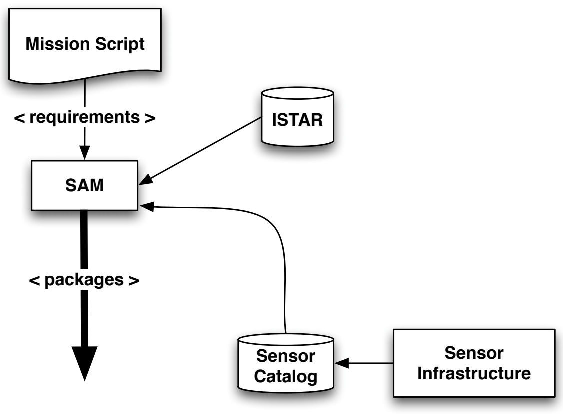


Fig. 2. SAM architecture

The architecture is composed of two main components, SAM the reasoner and the sensor infrastructure, and some data sources (viz., ISTAR ontology, and sensor catalogue). The *ISTAR*[13] ontology represents the domain knowledge of intelligence, surveillance, target acquisition, and reconnaissance aspects (e.g., types of intelligence). Figure 3 depicts the main concepts of the ISTAR ontology. The left-hand side decomposes a mission into a collection of tasks with specific information requirements (e.g., surveillance) and the right-hand side represents capabilities provided by assets (e.g., target detection provided by an UAV) as a

[12] http://www.csd.abdn.ac.uk/research/ita/sam
[13] http://www.csd.abdn.ac.uk/research/ita/sam/downloads/ontology/ISTAR.owl

composition of the functions provided by sensors and platforms. Requirements of tasks are broadly categorized into two sections: intelligence (i.e., kinds of intelligence disciplines such as imagery intelligence) and operational (i.e., desired capabilities of a task such as constant surveillance) requirements.

The *sensor catalogue* contains the attributes of assets (i.e., location, energy, current status, and so on.). These assets are particular instances of the asset types described in sensor and platform ontologies of the ISTAR ontology. The attributes of assets are retrieved from a *sensor infrastructure* [2].

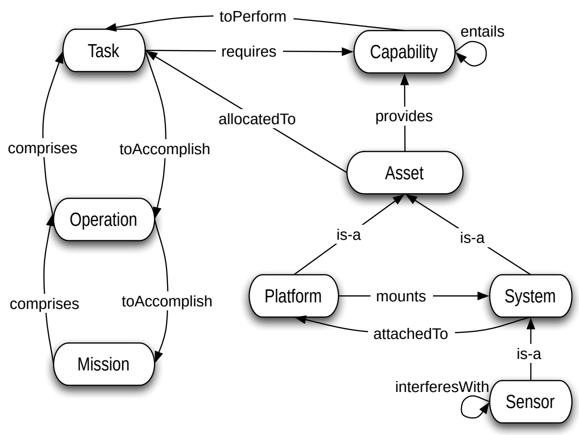


Fig. 3. Main concepts and relations in the ISTAR ontology. Reproduced from [9]

The reasoner checks the requirements of a given task and suggests asset types that are feasible and logically sound for the task. These solutions are logically sound due to the logical properties of OWL-DL [7] and the inference mechanisms used. We use Pellet[14] as a DL reasoner for inferences. Some solutions recommended by the reasoner are collection of asset types. This is because a task may not be satisfied with only one asset. For example, to achieve the goals of the task, visual and audio information are needed but there is no single asset to provide both. *SAM* uses a set-covering algorithm to compute this. Since a solution may contain more than one asset type, we refer to a solution collectively as an asset package. Furthermore, using subsumption[15] relationships, the reasoner finds all the plausible assets types for a particular task. We believe these solutions can be used in many useful ways, such as to analyse the feasibility of a mission with respect to an assets inventory, to assist in planning and re-planning stages of the mission, and so on.

[14] http://clarkparsia.com/pellet/

[15] A concept A subsumes a concept B if the definitions of A and B logically imply that members of B must also be members of A.

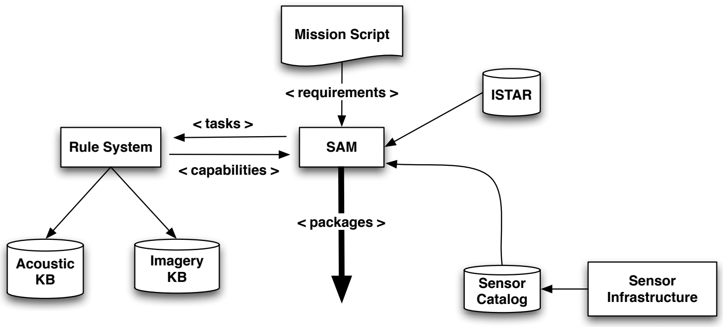


Fig. 4. SAM architecture with integrated rule system

We have extended the SAM architecture by incorporating the rule system discussed in Section 3 as shown in Figure 4. With the resulting integrated system, users can specify their information needs at higher level. That is, they do not have to express every capability requirement of a task explicitly; instead they simply let the rule system infer multiple capabilities in which the task could be accomplished. These inferences allow the system to compute many different asset types that could be used to satisfy the requirements of a given task.

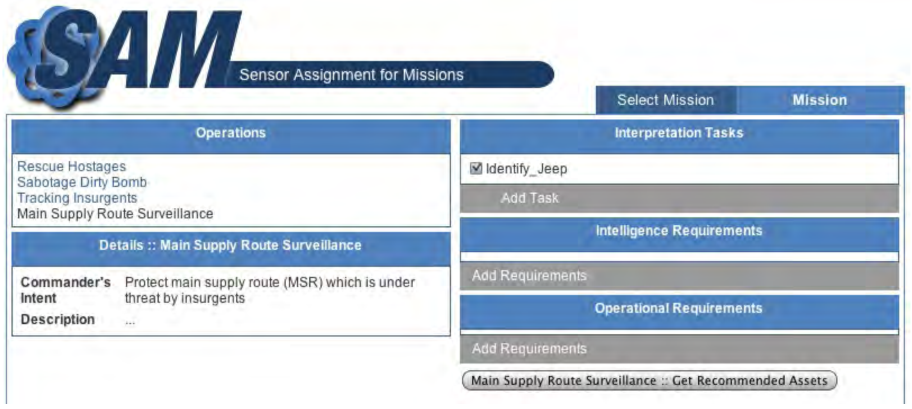
5 A Case Study

In this section, we introduce an example scenario and demonstrate how the system proposed in Section 4 computes feasible asset types for tasks in a realistic situation. Let us suppose a mission where an international peacekeeping force has to maintain a safe corridor between two countries. In order to perform this mission, many operations need to be carried out. Let one of those operations be "Perimeter Surveillance", which could be broken down into a set of tasks. Some possible tasks for the operation are:

1. **Detect human activity in the region.** This task is a part of the operation because a suspicious gathering near or in the region of the safe corridor may imply a critical breach in perimeter.
2. **Detect vehicle movement.** This may imply the movement of troops or militia.
3. **Identify vehicle of particular type.** For example armoured vehicles might imply an imminent treat.

Let us consider the task *identify vehicle*. A high-level requirement of identifying a vehicle task could be *identifying jeeps*. The *SAM* tool discussed in section 4 allows users to specify their requirements in this manner (e.g., *detect vehicles, identify jeeps,* etc.) as shown in figure 5.

When the *SAM* tool receives such requirements, they are automatically passed onto the rule system discussed in section 3.4. Within the rule system, the



Rescue Hostages
Sabotage Dirty Bomb
Tracking Insurgents
Main Supply Route Surveillance

☑ Identify_Jeep

Details :: Main Supply Route Surveillance

Commander's Intent	Protect main supply route (MSR) which is under threat by insurgents
Description	...

Fig. 5. SAM tool

appropriate rule is executed (e.g., *detect* rule is fired for detect activities whereas for identifying activities, *identify* rule is fired.). The rule traverses through the knowledge-bases (KBs) known to the rule system, and infer minimum capability ratings required to satisfy requirements. These KBs are created with respect to the formalism described in section 3.3. In order to satisfy the requirement *identify jeeps*, the rule system derives **{VisibleNIIRSRating6, RadarNIIRSRating6, ACSignature7}** as the required ratings.

This result set represents the fact that, in order to identify a jeep, one needs assets that could either provide visual, radar, or acoustic capability at a particular rating or above. These results are handed back to the *SAM* tool as shown in figure 4. *SAM* tool then passes these results to the ontology-based reasoner to identify the potential assets types that satisfy these capability requirements. It is important to note that these capability ratings are provided by an asset: sensors provide the capabilities such as radar, acoustic whereas platforms provide the capabilities such as altitude, range capabilities required to compute a particular rating. We represent this using the following logical formula.

$$\begin{aligned}\textbf{Asset([P,S]):providesCapabilityRating([C,R])} \leftarrow\ & \textbf{Platform(P):canProvideRating([C,R])} \wedge \\ & \textbf{Platform(P):carriesSensor(S)} \wedge \\ & \textbf{Sensor(S):providesCapability(C)}\end{aligned}$$

Therefore, in order to infer suitable asset types the reasoner first has to identify the suitable platform and sensor types based on the above capability ratings. We have created an ontology to represent these ratings and rating types concepts. The figure 6 depicts the NIIRS [12] imagery types and NIIRS imagery rating concepts of this ontology. We have imported this ontology into our ISTAR[13] ontology and associated these concepts with sensors and platforms types. At the reasoner level, we then use Pellet[14] to identify platform types that could provide a particular rating or above (using subsumption relationships among the rat-

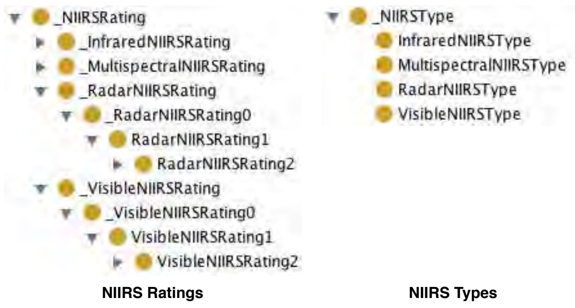


Fig. 6. NIIRS imagery types and ratings

ings) and sensor types that could be used to satisfy the capabilities for a specific rating. The reasoner then uses a set covering algorithm to compute all possible asset types that could be used to satisfy the task requirements. For example, to *identify a jeep* following asset types are recommended.

Asset Type	**Explanation**
iRobotPackbot **with** AcousticArray	Provides an **acoustic** signature of **value 9**
Raven **with** DaylightTV	Provides a **visual** rating of **value 7**
Reaper **with** DaylightTV	Provides a **visual** rating of **value 6**
Reaper **with** SAR	Provides a **radar** rating of **value 6**
GlobalHawk **with** EOCamera	Provides a **visual** rating of **value 6**
GlobalHawk **with** SAR	Provides a **radar** rating of **value 6**
HarrierGR9 **with** EOCamera	Provides a **visual** rating of **value 7**
NimrodMR2 **with** EOCamera	Provides a **visual** rating of **value 6**

Table 1. Assets capable of identifying a jeep

6 Conclusions and Future Work

In this paper, for the assets-to-tasks assignment problem, we have proposed an approach motivated by the importance of the litheness to the assignment problem. We have combined an ontology-based and rule-based reasoning mechanisms to achieve this. We have proposed a formalism to represent tasks. A well known knowledge corpus is formalised to create a knowledge-base, based on this formalism. A set of rules has been implemented to draw conclusions from this knowledge-base and we have validated the flexibility of this inference process by examples and a case study. In this architecture, the rule-based system is used to infer the information providing capabilities whilst an ontology-based reasoner is used to produce sound asset types that are necessary and sufficient to meet the information requirements of the tasks.

We have demonstrated the usefulness of the proposed approach by means of an example scenario. Our experiments imply that the research is promising even though it is currently in its early stages. Hence, we plan to investigate the following issues as a future work. First, we want to generalize the task representation so that a number of other domains could be represented using the same formalism. Second, the current version of the rule-based reasoning depends on the rule engines such as Prolog and Jess[16]. This is partly due to the existing limitations of the rule languages and tools catered for Semantic Web (e.g., SWRL does not support negation). We are currently investigating other rule representations that enable us to formalise rules using first order logics constructs.

Acknowledgments

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

References

1. I. F. Akyildiz, Y. S. W. Su, and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, 40:102–114, 2002.
2. F. Bergamaschi, D. Conway-Jones, C. Gibson, and A. Stanford-Clark. A distributed test framework for the validation of experimental algorithms using real and simulated sensors. In *Proceedings of the 1st International Technology Alliance Conference*, 2007.
3. L. Bermudez, J. Graybeal, and R. Arko. A marine platforms ontology: Experiences and lessons. In *Proceedings of the ISWC 2006 Workshop on Semantic Sensor Networks*, Athens GA, USA, 2006.
4. M. Botts, A. Robin, J. Davidson, and I. Simonis. OpenGIS© sensor web enablement architecture document. Technical report, Open Geospatial Consortium Inc, 2006.
5. J. Byers and G. Nasser. Utility-based decision-making in wireless sensor networks. In *MobiHoc '00: Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing*, pages 143–144, Piscataway, NJ, USA, 2000. IEEE Press.
6. W. F. Clocksin and C. S. Mellish. *Programming in Prolog: Using the ISO Standard.* Springer, 5 edition, 2003.
7. M. Dean and G. Schreiber. OWL web ontology language reference. W3C recommendation, W3C, February 2004.
8. T. M. Doll. Optimal sensor allocation for a discrete event combat simulation. Technical report, Naval Postgraduate School, Monterey, CA 93943-5000, June 2004. Thesis: MSc in Operations Research.

[16] http://www.jessrules.com/

9. M. Gomez, A. Preece, M. P. Johnson, G. de Mel, W. Vasconcelos, C. Gibson, A. Bar-Noy, K. Borowiecki, T. L. Porta, D. Pizzocaro, H. Rowaihy, G. Pearson, and T. Pham. An ontology-centric approach to sensor-mission assignment. In *EKAW '08: Proceedings of the 16th international conference on Knowledge Engineering*, pages 347–363, Berlin, Heidelberg, 2008. Springer-Verlag.
10. C. Goodwin and D. Russomanno. An ontology-based sensor network prototype environment. In *Proceedings of the 5th International Conference on Information Processing in Sensor Networks (IPSN 2006)*, Nashville TN, USA, 2006.
11. B. Guo, M. S. Nixon, and T. R. Damarla. Acoustic information fusion for ground vehicle classification. In *The 11th International Conference of Information Fusion*, Cologne, Germany, July 2008.
12. J. M. Irvine. *National Imagery Interpretability Rating Scales (NIIRS)*, pages 1442–1456. Marcel Dekker, Oct. 2003.
13. M. P. Johnson, H. Rowaihy, D. Pizzocaroz, A. Bar-Noy, S. Chalmers, T. L. Porta, and A. Preece. Frugal sensor assignment. In *4th IEEE International Conference on Distributed Computing in Sensor Systems*, June 2008.
14. Joint publication, "Joint Publication 2-01: Joint and National Intelligence Support to Military Operations", 2004.
15. M. Paolucci, T. Kawamura, T. R. Payne, and K. P. Sycara. Semantic matching of web services capabilities. In *ISWC '02: Proceedings of the First International Semantic Web Conference on The Semantic Web*, pages 333–347, London, UK, 2002. Springer-Verlag.
16. A. Preece, M. Gomez, G. de Mel, W. Vasconcelos, D. Sleeman, S. Colley, G. Pearson, T. Pham, and T. L. Porta. Matching Sensors to Missions using a Knowledge-Based Approach. In *Proceedings of SPIE Defense Transformation and Net-Centric Systems 2008*, to appear, mar 2008.
17. D. Roberts, G. Lock, and D. C. Verma. Holistan: A futuristic scenario for international coalition operations. In *Proceedings of Knowledge Systems for Coalition Operations (KSCO 2007)*, 2007.
18. A. Robin, S. Havens, S. Cox, J. Ricker, R. Lake, and H. Niedzwiadek. OpenGIS© sensor model language (SensorML) implementation specification. Technical report, Open Geospatial Consortium Inc, 2006.
19. J. Scholtz, J. Young, J. L. Drury, and H. A.Yanco. Evaluation of human-robot interaction awareness in search and rescue. Technical report, The MITRE Corporation, Bedford, MA, USA, 2004.
20. J. H. Sheehan, P. H. Deitz, B. E. Bray, B. A. Harris, and A. B. H. Wong. The military missions and means framework. In *Proceedings of the Interservice/Industry Training and Simulation and Education Conference*, pages 655–663, 2003.
21. C. M. Sperberg-McQueen, E. Maler, J. Paoli, F. Yergeau, and T. Bray. Extensible markup language (XML) 1.0 (third edition). first edition of a recommendation, W3C, Feb. 2004. http://www.w3.org/TR/2004/REC-xml-20040204.
22. S. J. Tutton. Optimizing the allocation of sensor assets for the unit of action. Technical report, Naval Postgraduate School, Monterey, CA 93943-5000, June 2003. Thesis: MSc in Operations Research.
23. K. Whitehouse, F. Zhao, and J. Liu. Semantic streams: A framework for composable semantic interpretation of sensor data. In *Wireless Sensor Networks*, pages 5–20. Springer, 2006.