Using SCXML for Semantic Sensor Networks

Álvaro Sigüenza[1], José Luis Blanco[1], Jesús Bernat[2] and Luis A. Hernández[1]

[1] ETSI Telecomunicación - Universidad Politécnica de Madrid
28040 Madrid, Spain
{alvaro.siguenza,jlblanco,luis}@gaps.ssr.upm.es

[2] Telefónica R&D, 28043 Madrid, Spain
bernat@tid.es

Abstract. Future Ubiquitous Sensor Networks (USNs) are expected to sense and combine multiple descriptions of user contexts, providing a huge potential for the development of revolutionary context-aware applications bridging the physical and digital worlds. Current Sensor Web initiatives aim to support ubiquitous access to sensor networks, but to make them really useful for context-aware applications there is still a strong need to develop new mechanisms for enriching sensor data with high-level information, which would constitute a step towards Semantic Sensor Networks. In this paper, state-charts technology, described according to the W3C SCXML language, combined with the functionalities arising from Semantic Web technologies, are proposed as an efficient, simple and flexible means to make progress towards these objectives. Working with Sensor Web data, we propose SCXML-based semantic processing to produce high-level information through automatic embedding of semantic annotations, as well as the identification of relevant contexts. Both of these approaches are useful in a wide range of application domains. The capabilities and limitations of an SCXML processing architecture are also discussed in the context of its implementation over an OSGi framework and its integration into an experimental Telco USN Platform based on OGC® Sensor Web Enablement guidelines.

1 Introduction

Many efforts on the Internet are currently focusing on a Content-Centric Future Internet, where users not only consume but also create (*prosumers*) logical data as text, audio and video [1]. Meanwhile, research has been less concerned with providing a global solution to access and sense the physical world. Sensor and Actuator Networks (SANs) represent an inexhaustible resource for real world information that can be used to model particular user or application contexts and situations enabling contextualized interaction [2]. The International Telecommunication Union defines these Ubiquitous Sensor Networks (USNs) as a "technological framework for ambient sensor networks not as simple sets of interconnected networks but as intelligent information infrastructures" [3]. This vision is closely related to Mark Weiser's Ubiquitous Computing [4] as well as to the recent proposal of the "Internet of Things" [5].

Up to now, the challenges for the integration of real world information into the digital world have been mainly addressed in two different research fields: SANs and

Context-aware Services [6]. These two different approaches have led to two clearly related but non-convergent architectures: Sensor Network Frameworks and Context Frameworks. Sensor Frameworks allow services to have universal access to heterogeneous and geographically dispersed sensor networks (a major representative is the Sensor Web), while Context Frameworks enable high-level capabilities related to context; for instance, how to create, update, maintain or process context so that services are able to discover and provide context-aware information.

Semantic Web technologies, as an evolving extension of the current Web, are intended to provide well-defined and machine accessible information, shareable across applications. Based on Ontologies, or formal descriptions of concepts and their relationships, applications may benefit from a shared semantic representation of sensor data [7]. At present, the most widely used ontology languages are W3C recommendations: Resource Description Framework (RDF), Resource Description Framework Schema (RDFS), and Web Ontology Language (OWL), all of which are based on XML.

Using the Semantic Web to progress from Sensor Frameworks to Context Frameworks is not new It has been proposed in the recent Semantic Sensor Web initiative [8], [9]. In Semantic Sensor Web the lack of semantics in the Sensor Web is intended to be overcome by using Semantic Web technologies, so heterogeneous sensor data become homogeneously represented using a unified high-level semantic representation. But just by defining ontologies, which is in itself a complex task, is not enough to take full advantage of sensor data. Two complementary tasks need to be addressed [10]: 1) semantic sensor data processing: mechanisms to derive high-level information from raw sensor data; and 2) semantic sensor data annotation: procedures for embedding semantics into sensor data.

Several approaches have already been proposed to carry out these two tasks automatically. The Semantic Web Architecture for Sensor Networks (SWASN), presented in [10], proposes the use of rules, based on local ontologies, to annotate domain-specific sensor data, while a global sensor ontology is used for querying and inference. Other noteworthy open architectures for managing the semantics of real-world entities in smart environments are being developed in different international projects such as the FP7 SENSEI [2] or Artemis SOFIA [11] EU projects. Our work is different but complementary to these approaches. We focus on exploring the use of general mechanisms provided by state-chart machines for both semantic sensor data processing and annotation.

The motivation to use state-chart machines and the W3C SCXML language can be summarized in the following main points:

- Beyond conventional finite-state machines, which are inappropriate for the behavioral description of complex control systems, state-charts can aggregate multiple inputs or process them individually rather straight-forwardly. As introduced by Harel [12], state-charts can be defined hierarchically, so a state-chart can itself be a state within another state-chart (i.e. state composition), and several state-charts can run in parallel (i.e. simultaneous states) so they can process different sources of information at the same time. They are suitable for modelling multiple cross-functional state diagrams that can transition internally without affecting other state machines in the state-chart. State-charts can also define condition guards for state transitions, access the past state history, use several synchronous (instantaneous broadcast) communication mechanisms, and support the invocation of external ac-

tions or processes. These capabilities (which will be detailed in Section 2) make state-charts a flexible and powerful technology to define a flow of logic rules for the semantic processing of heterogeneous sensor data.

- W3C [13] is developing the State Chart eXtensible Markup Language, SCXML, which corresponds to a generic event-based state-machine execution environment based on Harel state-charts. Although this is still an ongoing working draft, there are implementations available, such as the Java SCXML engine at Commons SCXML [14]. SCXML is already being considered, together with other workflow languages (such as BPEL), for service composition in Service Delivery Platforms (SDPs) within service-oriented architecture (SOA) environments [15]. In these scenarios SCXML is not only demonstrating simple and flexible general-purpose orchestration functionalities, but has also shown very good performance results for composing real-time services [15] [27], and network services in particular [16]. This is an important and promising result for SCXML in terms of scalable capabilities for processing real-time data from different sensor networks.
- Finally, as a general process control mechanism, SCXML is also a major W3C candidate for controlling language in Human-Machine Interactive systems (HMIs). It is being considered for future interactive speech systems, W3C VoiceXML 3.0 [17], as well as for multimodal systems [18]. In this particular field state-charts are used to describe the interaction flow in HMI applications. This, in our view, presents an invaluable opportunity to use the same SCXML technology to design HMI interfaces in context-aware applications, while at the same time providing semantic annotations of context information to the Semantic Sensor Networks. That is, using SCXML, a designer of a context-aware HMI interface could not only define how to handle the user's input/output information, but also how to provide semantic interpretations for the available sensor data, and even share it with other applications. Thus, by using the SCXML processing mechanism the designer becomes a *prosumer* in the Semantic Sensor Network, as he/she can both publish and subscribe semantic sensor data. We will illustrate this in Sub-section 2.2 considering context-aware applications for in-vehicle scenarios.

The rest of the paper is organized as follows: Section 2 explores SCXML's capabilities to semantically annotate sensor data, process it and identify contextual situations; an in-car driving scenario is used as an example. In Section 3 we put these possibilities into practice, defining a high-level architecture where a two-layer SCXML processing strategy is used to generate semantic and contextual information from Sensor Web data. Section 4 presents how the proposed processing architecture has been implemented over an OSGi framework and integrated into a pilot Telco USN Platform based on the guidelines offered by the OGC® Sensor Web Enablement initiative [20]. Finally, conclusions and future work are given in Section 5.

2 Using SCXML for Semantic Annotation and Context Modelling

In this paper we propose the use of state-charts, as implemented in W3C SCXML language [13], to enrich and process low-level data (Sensor Networks) resulting in a

higher semantic representation of data and contextual information (Semantic Sensor Networks). First we present the main modelling and processing capabilities of SCXML; then, through an example, we will describe how these general capabilities can be specifically applied to data for semantic annotation and context modelling.

2.1 Main SCXML functionalities for sensor data processing

SCXML is a general language for control processes, but the addition of several functionalities beyond conventional finite-state machines makes SCXML suitable for integrating different sources of information rather straightforwardly. SCXML allows the definition of state-charts *hierarchically*, so a state-chart can itself be a state within another chart. Depending on the associated semantics, this construction can be interpreted either as a functional aggregation of charts, a compositional or dependency network, or as establishing generalization (i.e. subsumption and/or implication) relations between states. Also, several state-charts can run in *parallel* (i.e. simultaneous states). States that can be active at the same time are said to be functionally disjoint (generally independent), while those sharing a common parent but which run sequentially are, in general, mutually exclusive.

Just as for finite-state machines, in SCXML relations between states are mainly stated as *transitions*, which serve to describe an entire network of dependencies across the state-chart. Transitions can be defined in different ways. The simplest transition in SCXML is direct implication ($p \Rightarrow q$): the transition to the next state q is taken as soon as the first p is entered. Nevertheless, transitions in SCXML are almost always driven by *events*, and sometimes controlled via guard-conditions. In both cases a similar interpretation holds: the direct implication clause is replaced by a conditional implication, which depends on the actual state p, the guard-condition c, the expected event e, or a combination of both (i.e. $p \wedge c \Rightarrow q$, $p \wedge e \Rightarrow q$, $p \wedge e \wedge c \Rightarrow q$). SCXML can also define when events are fired based on the evaluation of a set of conditions a_i (various combinations could be possible), thus introducing an additional relation ($a_1 \wedge a_2 \wedge \ldots \Rightarrow e$) between states. These processing capabilities make SCXML suitable to define deductive analysis over sensor data and events producing high-level interpretations or the identification of particular real-world situations.

The semantic interpretation of raw sensed data is also usually based on decisions depending on past events and facts. Thus, it is necessary to support the logging of past information to support this kind of processing. Additionally, this allows tracing the evolution of states and measures. SCXML supplies a *history* pseudo-state which represents the state configuration in which a certain state was in the last time the machine transitioned from it. A transition with this history pseudo-state as its target is in fact a transition to the set of descendant states, i.e. those that were active the last time this state was left. This allows designers to recover a previous configuration, and, furthermore, to dynamically re-design the flow or analyze unexpected results. SCXML also allows defining probabilities for state transitions, so some statistical modelling is possible for dealing with sensor data uncertainty.

In addition to these processing capabilities, SCXML introduces a `<datamodel>` element to allow a uniform access method to both local and remote data. The `<datamodel>` can encapsulate any number of `<data>` elements, expressed in a XML, defin-

ing variables fetched from an external source or specified in-line. This *data model* can be used both to evaluate guard conditions – when a transition in the machine is required – and to save data when a new state is entered.

Turning back to guard conditions, these are defined as conditional expressions evaluated either when a transition event is triggered or just after the state is entered. The set of operators in these conditional expressions varies, including numeric quantifiers and logical operators, stated in the *data model*. Logical operators including *less than*, *equal to* or *greater than* can be used, which extend the expressive possibilities of propositions.

The properties and functionalities of SCXML just described allow the use of rules of logic for semantic processing. Accordingly, as long as programming is consistent and structured, it is possible to annotate and process data, as well as to identify particular situations, following the deterministic, deductive flow specified in SCXML. In Figure 1 several state-charts graphs are used to illustrate how input from sensor data (accessed through different interfaces to Sensor Networks) can be processed towards other state-charts for further, higher-level processing. This hierarchical and successive processing and combination of information between states can end in a "stable state" describing or representing a particular context or real-world situation. The resulting information can be regarded as flowing from new "virtual sensors" generated by the Semantic Sensor Networks. It is worth noting that state-charts can also be used to control actuators or actuation processes, but we have not addressed this aspect in this paper.

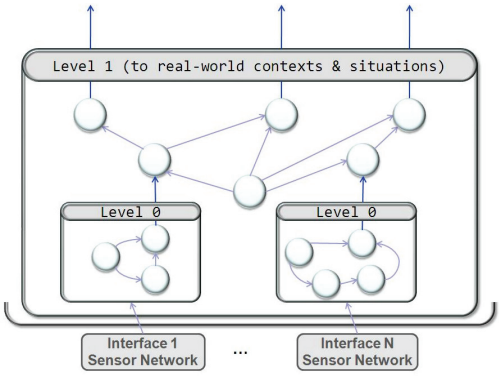

Fig. 1. Deriving context from several sensor sources using SCXML machines

2.2 An example of semantic annotation and context modelling

A context-aware application for an in-car driving scenario will serve as a motivating example to illustrate the possibilities of annotating data and identifying higher semantic situations using SCXML. In the driving context a great variety of heterogeneous data sources are available, some coming from multiple in-vehicle sensors (e.g. vehicle conditions, road conditions, the driver's state of attention and ability, etc.), others provided by "external" *Sensor Webs* (traffic, weather information, etc.). The integration of both types of sources constitutes a true challenge. As discussed in Section 1, the designer of an in-car HMI can take advantage of all those pieces of information

and become a *prosumer* of Semantic Sensor Networks. as he/she can both *consume* traffic information (e.g. suggest a different travel plan to the driver), and at the same time *produce* semantic data by offering the particular car travel plan chosen for external traffic modelling. This kind of information is considered to be vital for future traffic management systems.

The literature identifies three independent domains describing the global context for driver-vehicle interaction: driver, vehicle and environment [19]. In our example, we focus on traffic and weather sensors (provided by external *Sensor Web* resources), the type of road and in-vehicle sensors (local physical sensors).

To annotate the collected sensor data we have followed the Sensor Web Enablement initiative guidelines [20]. SWE is a workgroup of the Open Geospatial Consortium put together to specify interfaces and meta-data encodings with the aim of integrating heterogeneous sensor data so that they could be accessible and controllable through the Web. We have selected the Observations & Measurements language (O&M) [21] which defines a XML schema for describing and encoding real-time observations and measurements of sensor data. The standard also defines a large number of terms needed to describe measurements and the relationships among them. The following snippet represents an annotated measure from a wind speed sensor and a precipitation sensor.

```
<swe:DataRecord>
    <swe:field name="WindSpeed">
        <swe:Quantity definition="urn:ogc:def:property:OGC:WindSpeed">
        <swe:uom code="mph"/>
        <swe:value> 90.0 </swe:value>
        </swe:Quantity>
    </swe:field>
</swe:DataRecord>

<swe:DataRecord>
    <swe:field name="precipitation">
        <swe:Quantity definition="urn:ogc:def:property:OGC:1.0:precipitation">
        <swe:uom code="mm/h"/>
        <swe:value> 40 </swe:value>
    </swe:field>
</swe:DataRecord>
```

In our in-vehicle scenario, when a certain feature changes a sensor, it fires an event towards the context-aware application that is subscribed to it. This event allows the associated sensor data, encoded using O&M, to be processed through the SCXML execution environment to update the state machines representing the driver, vehicle and environment contexts. Each one of these domains can contain other domains (and therefore other state machines) which are paired with the sensors defined in the context. Thus, for instance, the vehicle domain could simply be described by a *velocity* state machine and a *steering wheel angle* state machine, and the environment domain by a *traffic road* state machine and a *weather* state machine. As the XML example below illustrates, this structure for the driver, vehicle and environment contexts can be easily defined using the SCXML <parallel> element. Note that each machine has to be subscribed to events coming from those sensors related to the local context, so only the data associated to relevant events has to be handled.

```
 <parallel id="context">
  <parallel id="driver/>
  <parallel id="vehicle">
    <state id="velocity"/>
    <state id="steeringWheelAngle"/>
  </parallel>
```

```
    <parallel id="environment">
      <state id="trafficRoad"/>
      <parallel id="weatherConditions">
         <state id="speedWind"/>
         <state id="precipitation"/>
      </parallel>
      <state id="roadType"/>
    </parallel>
  </parallel>
```

We now show how a particular state can be defined as another state-chart. Taking, for example, the previous *precipitation* state, the following SCXML code shows how it can be defined by a new state machine using a set of states and transitions driven by a *precipitation.change* event.

```
<state id="precipitation">
  <state id="weakPrecipitation">
    <invoke targettype="semanticAnnotation" src="weakPrecipitation"/>
  </state>
  <state id="moderatePrecipitation">
    <invoke targettype="semanticAnnotation" src="moderatePrecipitation"/>
  </state>
  <state id="heavyPrecipitation">
    <invoke targettype="semanticAnnotation" src="heavyPrecipitation"/>
  </state>
  <state id="torrentialPrecipitation">
    <invoke targettype="semanticAnnotation" src="torrentialPrecipitation"/>
  </state>
  <transition event="precipitation.change" cond="precipitation lt 5" tar-
     get="weakPrecipitation"/>
  <transition event="precipitation.change" cond="precipitation gt 5 and lt 20"
     target="moderatePrecipitation"/>
  <transition event="precipitation.change" cond="precipitation gt 20 and lt 45"
     target="heavyPrecipitation"/>
  <transition event="precipitation.change" cond="precipitation gt 45" tar-
     get="torrentialPrecipitation"/>
 </state>
```

In this simple example we can also see how the state machine, using the *cond* attribute (guard condition), checks the precipitation value to transition to the corresponding state (*weak, moderate, heavy* or *torrential precipitation*), which will be a more precise description of the actual precipitation state (this is equivalent to the subsumption construction described in subsection 2.1). This simple semantic interpretation can be used to enrich raw precipitation measurements by linking semantic information to them. In the SCXML code above this is done through the <invoke> tag that executes an external semantic annotation process each time a particular state is entered. Besides the control flow, using the <invoke> element, SCXML can invoke external processes. In our approach the external process is invoked to annotate the raw O&M sensor data with semantic information using a domain-specific ontology. We use RDFa (or Resource Description Framework –in- attributes) to embed semantic annotations into XML data. RDFa [22] is a W3C Recommendation that allows embedding rich metadata within Web documents. As it was proposed in [28], RDF attributes were added to *O&M* data to represent the desired semantic annotations; we use the RDF *about* attribute to establish a resource reference ontology. This is illustrated in the next example, in which the *windSpeed* and *precipitation* measurements are annotated using a common weather ontology (`weather-ont.daml`) that has to be associated to the environment state-chart.

```
<swe:DataRecord>
  <swe:field name="WindSpeed">
     <swe:Quantity definition="urn:ogc:def:property:OGC:WindSpeed"
       rdf:about="http://.../weather-ont.daml#windSpeed">
     <swe:uom code="mph" rdf:about="http://.../weather-ont.daml#mphSpeed">
     <swe:value> 90.0 </swe:value>
        </swe:Quantity>
   </swe:field>
</swe:DataRecord>
<swe:DataRecord>
  <swe:field name="precipitation">
      <swe:Quantity definition="urn:ogc:def:property:OGC:1.0:precipitation"
      rdf:about="http://.../weather-ont.daml#precipitation>
        <swe:uom code="mm/h" rdf:about:="http://.../uom.owl#mmph/>
        <swe:value> 40 </swe:value>
    </swe:field>
</swe:DataRecord>
```

Once sensor data is semantically enriched, the SCXML *data model* provides a standard way to store it and share it with other state machines. In the *precipitation* example, once the RDF annotation process is completed, an event could be fired to announce its availability so that different state-charts may use it. On top of a first annotation level, other processes may be defined, which, based on higher levels of abstraction, could detect specific situations useful for context-aware applications. In the following example, *precipitation* and *windSpeed* are used to define a higher level domain of *WeatherConditions*.

```
<state id="normalWeatherConditions">
  <transition cond="Data(environment, weather/windSpeed) eq 'strong' and
      Data(environment, weather/precipitation) eq 'heavy'" tar-
      get="adverseWeatherConditions" />
</state>
<state id="adverseWeatherConditions">
  <send event="adverseWeatherConditions.situation"/>
</state>
```

This upper domain is composed of two states representing two different and mutually exclusive situations: *normalWeatherConditions* and *adverseWeatherConditions*. The state machine will remain in the *normalWeatherConditions* state until the transition condition is fulfilled. Just before this, the lowest domains will update the *speedWind* and *precipitation* fields in the data model sending an event to the upper domains signalling a change in one of its variables. Then, when the guard condition is met, a transition to the *adverseWeatherConditions* state occurs, causing this new situation to be annotated in the data model, as well as the firing of an update event (for the sake of simplicity, the invoke tag to the annotation process is not shown in the example).

Finally, we may identify meaningful situations for context-aware applications by integrating information from different domain levels. The following SCXML fragment shows an example where environmental information (*weatherConditions* and *roadType*) together with vehicle (*velocity*) information are used to define a dangerous driving situation and generate the corresponding warning event.

```
<state id="normalDriving">
   <transition cond="Data(environment, weatherConditions) eq 'adverse' and
      Data(environment, roadType) eq 'lane' and Data(vehicle, velocity) eq 'exce-
      sive'" target="dangerousDriving" />
</state>
<state id="dangerousDriving">
  <send event="dangerousDriving.situation"/>
</state>
```

3. SCXML processing: High-Level Architecture

Based on the SCXML capabilities described in the previous Section, Figure 2 represents a two-layer architecture for using SCXML to process sensor data (Sensor Web) and provide semantic annotations and contexts to both Semantic Sensor Networks and Context-Aware applications.

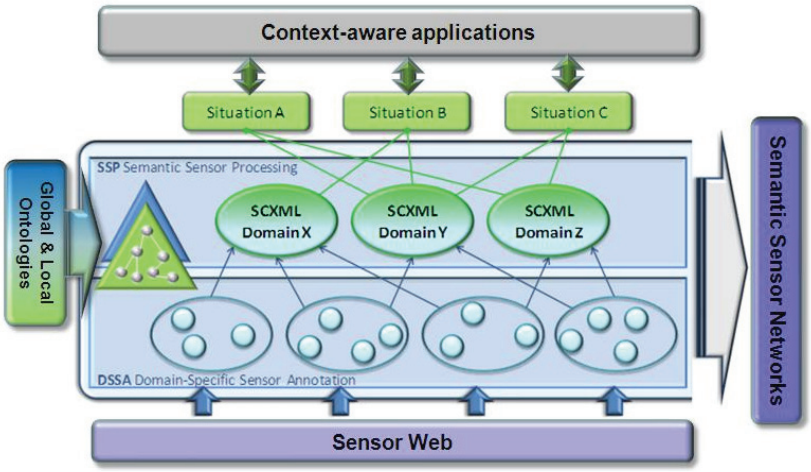

Fig. 2. High-level SCXML processing architecture for semantic sensor annotation and context detection (modelling, identification...)

Beginning with our lower layer, the Domain-Specific Sensor Annotation layer (DSSA), heterogeneous sources of sensor data (Sensor Web) are processed by domain-specific SCXML machines. In our previous example (Section 2.1) the DSSA included the driver, vehicle and environment state machines. These SCXML machines have to be designed relying on local ontologies, so they can semantically annotate events related to specific sensor data to which they are subscribed. In a second layer, the Semantic Sensor Processing layer (SSP), the functionalities supplied by SCXML are used to process information from different local domains, generating higher levels of semantic information. Going back to the example in Section 2.1, the SSP domain would include the higher semantic level of *weatherConditions*. Also, just as in the lower DSSA layer, the state-char machines in the SSP layer have to be designed on top of specific ontologies, although at this level they must represent more global or wider domains. The SSP layer can provide three main main types of outcome:

1. *Semantically annotated sensors*: As the SSP layer processes sensor data already annotated in the DSSA layer, one possible outcome in the SSP layer can be just to extend the already semantically embedded information in DSSA with higher-level semantics from SSP.
2. *Semantically annotated virtual sensor data*: As SSP can integrate or process semantic sensor data from different local domains at the DSSA level, it can produce new information related to the real-world. This information can be regarded as data from a virtual sensor.

3. *Detection of specific situations*: Driven by the information from the DSSA layer, the particular states of the SCXML machines in the SSP layer can be interpreted as specific higher-level situations derived from local domains. Thus, certain meaningful patterns can be associated with the actual situation.

As shown in Figure 2, semantic annotation in the DSSA and SSP layers (outcomes 1 & 2) can be regarded as a mechanism for exposing sensor data from the Sensor Web to the Semantic Sensor Web. The detection of context and interpretation of the situation, provides valuable high-level information that can be fed to context-aware applications. It is worth pointing out that the annotated semantic sensor data could also be useful for context-aware applications that develop their own strategies based on semantic queries, reasoning or inference engines.

4. Integrating SCXML into OSGi and SWE frameworks

We produced an experimental implementation of our SCXML processing architecture in order to perform an exploratory analysis of its capabilities and limitations. Our implementation ran on an OSGi framework, and was then integrated into a pilot Telco USN Platform based on OGC® Sensor Web Enablement initiative (SWE) guidelines [20]. This implementation was mainly geared towards the design of new In-Vehicle Information Systems (IVIS) under the Spanish research project MARTA (Mobility for Advanced Transport Networks; www.cenitmarta.org), a public-funded project in which several context-aware interactive applications are being designed and implemented for connected-car scenarios.

4.1 OSGi Implementation

The Open Service Gateway Initiative (OSGi) is a consortium of technology to create open specifications for facilitating the interoperability of applications and services over a variety of networked devices in homes, cars and other environments. OSGi is a service platform for the Java programming language that allows applications to be developed from small, reusable and collaborative components called *bundles*.

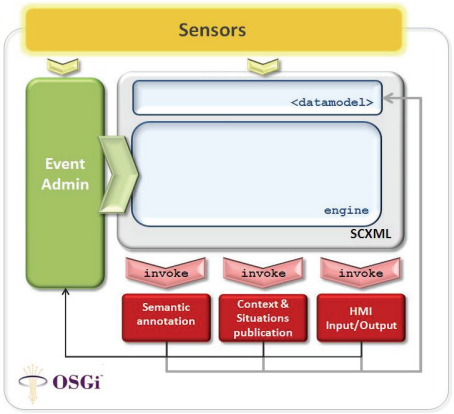

Fig. 3. Implementation of the SCXML processing architecture with an OSGi platform

Our SCXML processing architecture was implemented with the 3.4 release of the OSGi platform [23] (see Figure 3 above). First, a specific *bundle* was implemented that included the SCXML engine provided by the Apache Commons SCXML implementation [14]. Since SCXML is an event-based state-machine execution environment [13], the EventAdmin OSGi Service was used to provide a standard way of dealing with events in the OSGi Environment using the publish/subscribe model. This way the control flow of the different applications that may be running can subscribe to the specific sensors they are interested in. Warping functions have been implemented to code raw sensor data using the OGC® O&M language. To invoke processes from the SCXML control flow we have also implemented several independent *bundles*, the most important of which is the one for semantic annotation. This process is in charge of embedding RDF semantic annotations in the SCXML data model. It can also send messages to the EventAdmin notifying that an annotation process has taken place. As shown in Figure 3, other relevant invoked processes are those that manage the multimodal input/output governing interaction with the user, i.e. the in-vehicle HMI.

4.2 Integration on a Telco SWE framework

As it was originally designed to support networked devices in homes, cars, etc., our implementation of the SCXML architecture on an OSGi platform is a highly efficient solution for closed or "local" environments, but it is more limited for wide open or "distributed" Sensor Web scenarios. Furthermore, our IVIS application domain is currently being developed in the context of the connected cars of the future. In this scenario, in-car sensor data has to be combined with Sensor Web information (traffic, weather information, etc.). For this purpose our OSGi SCXML semantic processing architecture was integrated into Telefónica's experimental USN Platform based on OGC® Sensor Web Enablement (SWE) initiative guidelines [20]. This allowed us:

- to explore the possibilities of the SCXML architecture on SWE, an initiative that has been widely adopted in many government and industrial applications;
- to evaluate the possibilities of developing IVIS applications for future connected car scenarios accessed through current mobile Telco networks as well as through Next-Generation Networks (NGNs).

4.2.1 Telefónica USN Platform

Telefónica USN [24] is an experimental platform inspired in the Ubiquitous Sensor Networks concept of the ITU-T [2]. It has been designed following a horizontally layered architecture to decouple services and networks, so services can be separately deployed, and independently developed. The platform provides all-IP core communication service capabilities, and is a connectivity substrate for both current fixed and mobile communications (e.g. xDSL, GSM, GPRS, CDMA, WiFi, HSDPA…). As shown in Figure 4, Telefónica USN has two main components: *USN-Gateways* and *USN Service Enabler*.

- *USN-Gateways* are the entry points for different sensor networks to the all-IP core network. Two main situations are addressed: a) sensor networks already connected to the Internet, as Sensor Web infrastructures; and b) non-IP sensor

networks using proprietary technologies (such as ZigBee). In both cases USN-Gateways, as functional components of Telefónica USN, guarantee both unified information (i.e. sensor data) modelling, and communication protocols for heterogeneous Sensor Network infrastructures.

- The *USN Service Enabler* assists different applications and services in gaining access to the information provided by ubiquitous Sensor Networks. Following Service Oriented Architecture (SOA) principles, the USN Enabler was designed as a Sensor Networks service enabler in a Service Delivery Platform (SDP), where IP-based telecommunications can be accessed by third-party service providers with information technology (IT). In particular, the Telefónica USN Enabler was developed according to the common abstraction layer OMA Service Environment (OSE), specified by the Open Mobile Alliance (OMA) [25]. Thus, the USN Enabler relies on existing OSE enablers such as presence, call conferencing, transcoding, billing, etc., the more relevant one being the presence SIMPLE Specification (for mechanisms to publish/subscribe to sensor information) and XML Document Management, also known as XDM (for XML information modelling in Service Enablers).

For information modelling, as in OGC® SWE [20], the USN platform represents sensors using the Sensor Modeling Language (SensorML), while the Observations and Measurements (O&M) standard is used to model the measures (or observations) generated by those resources. The *USN Service Enabler* also implemented several SWE functionalities [24]. The most relevant ones for the work we present here are following (see Figure 4):

- *Sensor Discovery*: The platform provides information about all the registered sensors (described using SensorML), enabling efficient look-up mechanisms.
- *Observation Storage*: A repository where observations or sensor data (using the O&M language) are stored to allow subsequent retrieval or information extraction.
- *Data Processing*: this functionality provides those capabilities needed to implement processing and filtering operations on sensor data.

Also relevant to our work is the development of *Publish/Subscribe* mechanisms (based on both SIP and HTTP) that allow services to subscribe not just to the observations provided by sensors but also to complex conditions involving other sensors and previous observations. We decided to implement both SIP and HTTP protocols in order to allow the development of convergent Telco/IT services in an all-IP environment.

4.2.2 Integration of SCXML semantic processing into Telefónica USN

As discussed in Section 1, one of our motivations for using SCXML was its successful application to service orchestration in Service Delivery Platforms (SDPs) [15] [16] [26] [27], where it has demonstrated very good performance results for composing real-time services [15] [16] [27]. Particularly relevant is the FOKUS Service Broker [27] that, also based on OSGi, provides orchestration of real-time services with SCXML and the execution of heterogeneous Telco services under real-time constraints.

Our SCXML OSGi implementation was integrated into the Telefónica USN Platform in a similar fashion. As depicted in Figure 4, the SCXML semantic processing

was integrated as a natural extension of the OGC® Data Processing entity. Several implementation issues were addressed:

- First, the EventAdmin OSGi Service of the SCXML architecture had to be bridged to the USN Publish/Subscribe functionality, so that when a sensor notifies an observation to the USN Enabler, the corresponding event is communicated to the DSSA layer, thus driving the SCXML semantic processing (as described in Section 3). In this adaptation mechanism also the USN sensor data, modelled using O&M language (and possibly from the Observation Storage Entity), was stored into the proper SCXML data model.

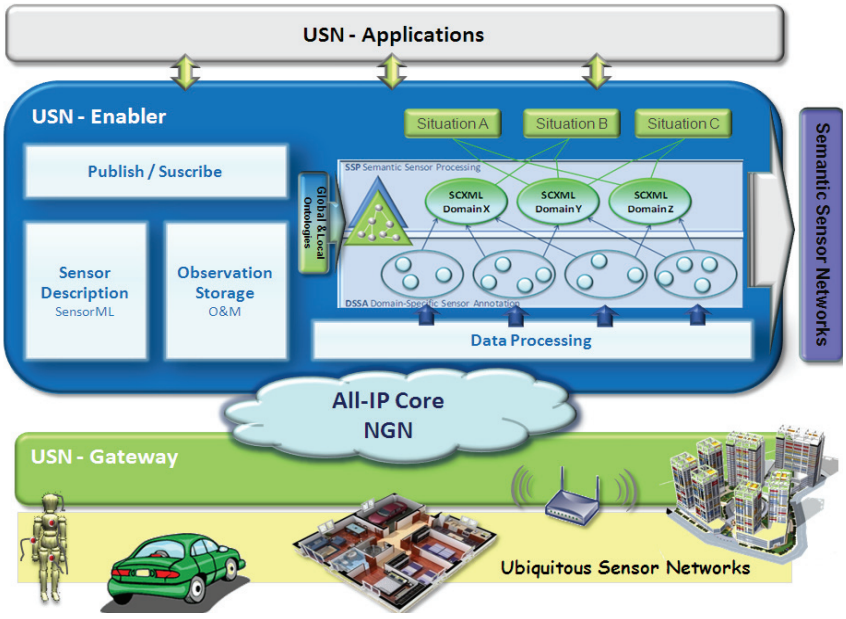

Fig. 4. Integrating SCXML semantic processing architecture into Telefónica's USN Platform

- The semantic annotation process was also extended so that when the representation update was completed, the enriched sensor data (RDFa annotated), stored in the SCXML data model, were moved back to the Observation Storage entity. In that way annotated data can be shared with third party applications through the USN Publish/Subscribe entity. Consequently, not only sensor data but also semantic links between data and formal (and often universal) descriptions based on ontologies are published. Thus it can be said that the USN Platform becomes a Semantic Sensor Network Platform.
- A final implementation issue is related to the publishing of high-level real-world contexts or situations obtained as outcomes of the SCXML processing in the SSP layer (as described in Section 3). A new OSGi communication *bundle* was implemented so these situations or contexts could be published as "virtual sensors" to interested applications using the USN Publish/Subscribe entity. This new publishing function was implemented as an external process that could be invoked inside SCXML states.

To illustrate the integration protocol between the SCXML processing architecture and the USN Platform, Figure 5 presents a call flow where a sensor event is notified to the SCXML framework (already subscribed to this event) and both semantic annotation and context identification are performed and notified to a given application.

The flow in Figure 5 starts when the SCXML processing component receives a sensor event notification from the USN Publish/Subscribe entity. The DSSA layer enriches the received O&M sensor data by embedding semantic annotations and stores them into the Observation Storage entity. Then, through Publish/Subscribe, the availability of this new information is notified to the interested application (already subscribed to it). Finally, changes in the lower DSSA layer induce the SSP layer state-chart machines to move so that they are able to detect relevant contexts. Once again, the Observation Storage entity is updated and, through USN Publish/Subscribe, the new context information is notified to the same context-aware application.

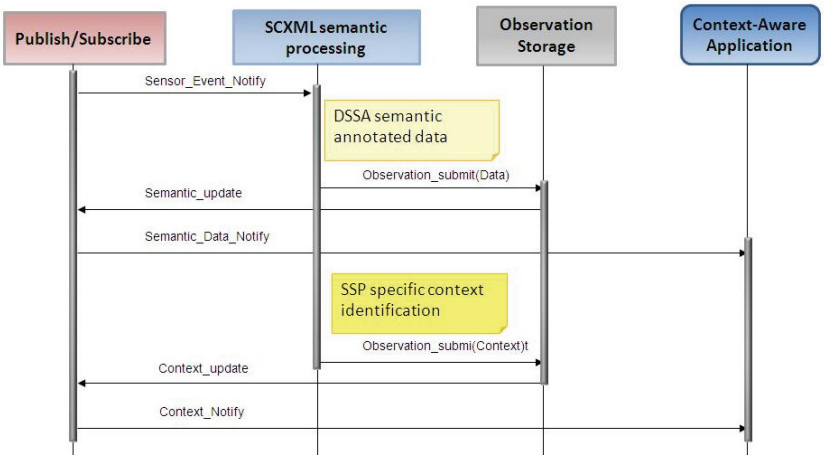

Fig. 5. Call flow example for semantic annotation and context notification

5. Conclusions and Future Work

In this paper we have considered the use of SCXML as a possible semantic processing component contributing to bridge the gap between Sensor Web and Semantic Sensor Networks. SCXML capabilities have been reviewed to explore its possibilities for processing sensor data and providing high-level semantics for context-aware applications. A two-layer SCXML processing architecture has been presented. In the first layer (DSSA) raw sensor data is semantically annotated using domain-specific SCXML machines relying on local ontologies, while in the second layer (SSP) information from different local domains are combined to generate higher levels of semantic information. The implementation of the proposed SCXML architecture over an OSGi framework has also been presented, as well as its integration into a Telco USN Platform based on OGC® Sensor Web Enablement initiative (SWE) guidelines. This has allowed us to explore and present some experimental experiences on the capabilities and limitations of our SCXML semantic processing architecture acting on wide Sensor Web environments.

Based on these analyses and experimental results, future work will address the main limitations we have found with our practical implementations:

- Even though SCXML provides high flexibility and rapid development for moderate sensor data processing, combining multiple sources of information becomes a tough task when the number of sources and the amount of information is potentially very big. This limitation to both managing and designing is particularly important since no powerful graphic development environment is yet available.
- The semantic annotation process needs to be improved. Its efficiency could be improved by exploring the latest SCXML draft on using JSON (ECMAScript) for binding expressions for access to the data model. New mechanisms should also be developed to assist in the mapping between ontologies and SCXML machines or particular states. The modelling power of SCXML could be greatly improved if it were possible to associate metadata on states and events.
- Specific improvements are needed to avoid redundancies in the two main mapping processes: between the OSGi EventAdmin Service and the USN Publish/Subscribe entity, and between the SCXML data model and the Observation Storage entity.
- Finally, besides the good performance of the SCXML execution environment, scalability and latency demand further analysis and research efforts. Latency, mainly when processing distributed sensor data from the USN platform (Sensor Web), has been shown to impact the event-driven SCXML processing very negatively. Therefore, specific time-synchronization mechanisms should be sought. Execution times could be reduced by applying SCXML script compilation and by efficient caching of these scripts. Also compact binary representations for the SCXML data model should be considered. Scalability is therefore extremely important for this kind of infrastructures, as well as the granting access to Semantic Sensor Networks both to retrieve and share information.

6 Acknowledgements

The activities described in this paper were funded by the Spanish Ministry of Science and Technology as part of the TEC2009-14719-C02-02 project.

References

1. P.Daras et. al: Why do we need a Content-Centric Future Internet? Proposals Towards Content-Centric Internet Architectures, European Commission, Networked Media Unit, Information Society and Media, May (2009).
2. Presser, M., Gluhak, A., Krco, S., Hoeller, J. and Herault, l.: SENSEI - Integrating the Physical with the Digital World of the Network of the Future, 17th ICT Mobile Summit, Stockholm, Sweden, 10-12 June (2008)
3. ITU TSTAG: A preliminary study on the Ubiquitous Sensor Networks. TSAG-C 22-E. February (2007)
4. Weiser, M.: The computer for the 21st century, Scientific American, vol. 265, no. 3, pp. 66-75, September (1991).
5. I. C. Union: The internet of things, 2005, iTU Internet Reports 2005, ExecutiveSummary

6. Raz, D., Juhola, A. T., Serrat-Fernandez, J. and Galis A.: Fast and Efficient Context-Aware Services, Wiley Series in Communications Technology, (2006)
7. Berners-Lee, T., Lendler, J. and Lassila, O.: The Semantic Web, Scientific American, vol. 284, no. 5 (2001)
8. Knoesis Center, Semantic Sensor Web Project, http://wiki.knoesis.org/index.php/SSW
9. W3C Semantic Sensor Network Incubator Group, http://www.w3.org/2005/Incubator/ssn/
10. Huang, V. and Stefanov, S.: Semantic Sensor Information Description and Processing, Proceedings of the Second International Conference on Sensor Technologies and Applications, pp. 456-461 (2008)
11. SOFIA project – Smart Objects For Intelligent Applications, funded through the European Artemis program, http://www.sofia-project.eu/
12. Harel, D: StateCharts: A visual Formalism for Complex Systems. In: Science of Computer Programming 8, North-Holland. (1987).
13. W3C: State Chart XML (SCXML): State Machine Notation for Control Abstraction, http://www.w3.org/TR/scxml/, W3C Working Draft 13 May (2010)
14. Apache Commons, Commons SCXML, http://commons.apache.org/scxml/
15. Magedanz, T., Blotny, A., Blum, N., Lange, L.: Agile Dienstkomposition für einen konvergierenden Web-Telco-Markt, Fraunhofer FOKUS, http://www.fokus.fraunhofer.de/en/ngni/files/Agile_Dienstekomposition_fuer_einen_konvergierenden_Web-Telco Markt.pdf , 12 May (2010)
16. Moro, S., Bouat, S., Odini M-P, O'Connell, J.: Service Composition with Real Time Services, (Hewlett-Packard), unpublished, http://www.docstoc.com/docs/27168951/Service-Composition-with-Real-Time-Services
17. W3C: Voice Extensible Markup Language (VoiceXML) 3.0, W3C Working Draft 17 June 2010 http://www.w3.org/TR/2010/WD-voicexml30-20100617/
18. W3C: Multimodal Interaction Activity, http://www.w3.org/2002/mmi/
19. Amditis, A., Kussmann, H., Polynchronopoulos, A., Engström, J., and Andreone, L.: System Architecture for integrated adaptive HMI solutions. In: Intelligent Vehicles Symposium. Tokyo, Japan (2006)
20. OGC® Open Geospatial Consortium, Inc. : Sensor Web Enablement (SWE), http://www.opengeospatial.org/ogc/markets-technologies/swe
21. OGC® Open Geospatial Consortium, Inc.: Observations and Measurements (O&M), http://www.opengeospatial.org/standards/om
22. W3C: RDFa Primer. Bridging the Human and Data Webs. W3C Working Group Note 14 October 2008http://www.w3.org/TR/xhtml-rdfa-primer/
23. OSGi Alliance, OSGi Service Platform Release 3, The Netherlands, March 2003, http://www.osgi.org/Download/File?url=/download/r3/r3.book.pdf
24. Bernat, J., Marín, S., González. A., Sorribas, R., Villarrubia, L., Campoy, L., Hernández, L. Ubiquitous Sensor Networks in IMS: an Ambient Intelligence Telco Platform. ICT Mobile Summit, 2008. 10-12 June, Stockholm.
25. OMA Service Environment Archive, http://www.openmobilealliance.org/technical/release_program/ose_archive.aspx.
26. Rodríguez Escola, J. and Rionda Rodríguez, A.: SCXML Flow Engine, Implementation Report, MyMobileWeb Project, Fundación CTIC October 2008 http://mymobileweb.morfeo-project.org/reports/scxml/
27. Blum, N. and Schreiner, F.: Smart Bit Pipes: Open APIs and their Role in Emerging SOA SDPs for Converging Networks, Fraunhofer Inst. FOKUS, IMS 2009 Workshop (2009). http://www.fokus.fraunhofer.de/en/fokus_testbeds/open_soa_telco_playground/_docs/SOA_telco_tutorial_2010.pdf
28. A. Sheth, C. Henson, and S. Sahoo.: Semantic sensor web. IEEE Internet Computing, 12(4) (2008).