FP7-313161

A holistic, scenario-independent, situation-awareness and guidance system for sustaining the Active Evacuation Route for large crowds

SMART SPACE INFORMATION MODEL: DATA MODELS AND ONTOLOGIES

Deliverable Identifier:	D7.6
Delivery Date:	Dec 30, 2015
Classification:	Restricted
Editor(s):	**Dimitris Drakoulis** (Telesto Technologies)
Document version:	0.9 – 2015 (Draft)
Contract Start Date:	April 1st, 2013
Duration:	48 months
Project coordinator:	EXODUS S.A. (Greece)
Partners:	EXO (GR), IT INNOVATION (UK), ICCS (GR), HKV (NL), TEL (GR), TEK (ES), AIA (GR), VITRO (IT), CDI (UK), INDRA (ES), KUL (BE), DXT (FR), POLITO (IT), STX-FR (FR), TUD (DE), TUC (DE), ASRS (ES), METB (ES), TIM (IT)

Project co-funded by the
European Commission under the
7th Framework Programme

Document Control Page

Title	Description of all the relevant agents for being integrated in the Smart Spaces	
Editors	Dimitris Drakoulis	Telesto Technologies
Contributors	Jose Miguel Landeta	IK4-TEKNIKER
	Jorge Rodriguez	INDRA
	Gianluca Correndo	IT Innovation
	Athanasios Moralis	TELESTO
Peer Reviewers	Name	Partner
	Pedro Garibi	INDRA
	Jorge Berhoza	TEKNIKER
Format	Text - Ms Word	
Language	en-UK	
Work-Package	WP7	
Deliverable number	D.7.6	
Due Date of Delivery	30/12/2015	
Actual Date of Delivery	23/11/2015	
Dissemination Level	Confidential, only for members of the consortium (including the Commission Services)	
Rights	eVACUATE Consortium	
Audience	☒ public ☐ restricted ☐ internal	
Date	23/11/2015	
Revision	none	
Version	DRAFT Version 0.9	
Edited by		
Status	☒ draft ☒ Consortium reviewed ☒ WP leader accepted ☒ Project coordinator accepted	

Table of Contents

List of Tables

1 Introduction

The deliverable is entitled Smart Space Information Model: Data Models and Ontologies. The deliverable reports on the outcomes from the task T7.3 (T7.3 “Raw Data Modeling”) and task T7.4 “Ontologies: Data Processing towards Comprehensive Information Generation”.

1.1 Background

The main objective of the Smart Space in eVACUATE is to provide the means for Situation Awareness and dynamic routing integrating in a virtual environment all the relevant agents in an evacuation scenario. The Deliverable D7.2 (Architecture of Smart Spaces) reported on the definition of the Smart Space architecture and described it in detail, including the functionality of each of its parts and the proposed implementation, as well as the smart space relationships with other parts of the project’s general architecture. D7.2 was the result of the eVACUATE project’s task 7.2 *Agent definition, integration and data flow definition, including sensors, actuators*.

The enabling technology for the Smart Space concept is SOFIA (currently “SOFIA2” for second version, a major redesign of the original system) an interoperability platform. SOFIA’s architecture identifies a central processing part known as the Semantic Information Broker (SIB), which receives the data from the physical devices, and sends data to them. A lighter software functionality known as Knowledge Processor is incorporated in each Agent to provide communication with the SIB, adapted to the technological process.

More specifically, the KPs send semantic information (ontologies) about the situation of the smart space to the SIB. The SIB then processes this information and sends the output, including commands, to the KP’s. The SIB may, on its own initiative, send information to a given KP; however it is more common that a KP subscribes to a specific kind of information and receives any event that fulfils its request.

A critical system such as the product of eVACUATE relies on the quick validation of inputs to provide answers. An automatic control of the inputs, following a predefined set of rules, will be more efficient than a manual verification by human individuals - faster, less error-prone, etc. The Deliverable D7.5 (Rule Set Tool) presented the Rule Set Tool of the eVACUATE Project. The Rule Set Tool is software integrated in the Smart Space that allows the local control of the elements within the Smart Space based on the information collected from it. Its implementation is based on a Complex Event Processing (CEP) compatible with SOFIA2.

The current deliverable D.7.6 reports on the modelling of the data needed to achieve interoperability of the Smart Space with a large set of heterogeneous sensors and sensor systems. Before processing, the data collected may also be annotated with further metadata that can be used to offer opportunities for reasoning about the observed system.

1.2 Purpose and Scope of the Deliverable

In T7.3 the Data Models are defined, aiming to provide a level of abstraction required for interoperability of heterogeneous equipment. Existing standards are presented, as in the case of the SWE of the OGC and specifically the “O&M” (Observations & Measurements) standard which is used to model sensor data and respectively the “SensorML”, an XML variant used to model meta-data about sensors.

The goal of T7.4 was according to the DoW to create the layer in contact with WP8. This is the layer combining semantics & ontology with data processing/fusion at the Smart Space level. The ontology provides definitions for the structure of sensors and observations (accordingly actuators and decisions), leaving the details of the observed domain unspecified. This allows abstract representations of real world entities, which are not observed directly but through their observable qualities. The preprocessed data (formed according to the data models from T7.3) will be formatted in order to generate significant, comprehensive and understandable information for the decision support tool (Ontology mapping). Guidelines from W3C, as in the case of SSN-XG (Semantic Sensor Networks Incubator Group) which has produced a generic ontology to describe sensors, their environment and the measurements they make, will be considered for this task.

1.3 Structure of the Document

The document first (in Section 2) offers insight at domain independent standards such as the Observations & Measurements model of the Open Geospatial Consortium (OGC) for the representation of spatiotemporal measurement data. Then SensorML again of the OGC is described as a data model to describe the sensor related process, i.e. the method followed for getting an observation. In the same section, the mapping of the logical data models used by eVACUATE, already identified and described in the deliverable “D7.4 - Description of all the relevant agents for being integrated in the Smart Spaces”, to the data models and formats specified by the OGC standards is the section 2.3.

Chapter 3 reports on ontology mapping, i.e. results of the task 7.4 of the DoW. In Section 3.1 the ontologies eVACUATE ontology is based on are described. In Section 3.2 Ontologies for semantic reasoning specifically for sensor applications are discussed. We focus on SSN which was incubated by the W3C. Then the ontology proposed by eVACUATE is described.

2 Data Models

2.1 OGC Observations & Measurements

The Observations & Measurements standard defines a domain independent (i.e. has broad scope) model for the representation of spatiotemporal measurement data. It comprises an implementation of this conceptual model as an XML-based GML application schema - ISO defines an application schema as a conceptual schema for data required by one or more applications. Thus, O&M can be seen as a conceptual schema for sensor applications based upon GML. O&M is particularly used for the creation of response documents for the GetObservation operation of the SOS (Sensor Observation Service). Nevertheless, O&M can also be used as a generic means to deal with measurements in a standardized way.

The above has been the case for the first version of O&M and is also the case for the most recent O&M ver. 2.0 (part of the Sensor Web Enablement v2 framework). However, conceptual model and its encoding are now more strictly divided. In fact, the major objective of the development of O&M 2.0 has been to harmonize it with existing foundational ISO models and to bring it in pace with the ISO standardization process. This aim has been achieved, and the conceptual model of O&M 2.0 has reached the status of an ISO final draft international standard while its XML implementation is integrated into the more technical standards landscape of the OGC.

An observation has a relationship to a procedure representing the process which has performed the observation, e.g., a physical sensor or a simulation. The observed property points to a description of the property which observed (e.g. water temperature or salinity). The observation‘s result is not restricted to a certain type in the basic observation model and can be of any type, ranging from a single measurement to an n-dimensional coverage of values. Subtypes of the basic observation then restrict the type of the result. The feature of interest, the computational representation of a real world feature carries the property which is observed. The observation provides a value for this property at a certain time, the phenomenon time. The phenomenon time was formerly called sampling time and was renamed to better reflect that it represents the time when the observation‘s result applies to the observed property (in comparison to when the observation was actually communicated).

In addition to the phenomenon time, an observation contains two other temporal properties: result time and valid time. The mandatory result time property represents the time when the observation‘s result was produced. The valid time is an optional property introduced in O&M 2.0 that defines the time period for which the observation‘s result is usable.

Spatial information for an observation is usually given by a location property of the feature of interest. However, in O&M 2.0, a new spatial profile facilitates the provisioning of an observation‘s sampling geometry—the spatial extent that the result of the observation applies to. This is usually the extent of the observation‘s feature of interest. Without the profile, this information has to be extracted from the feature of

interest, which could involve complex computations of the actual geometry and can also require dealing with previously unknown feature types.

Also newly introduced in O&M 2.0 is the related observation property. This property can be used to express relationships between observations. Removed from the O&M data model is the data type that served as a container for collections of observations. This has been done since containers for multiple observations are now defined by the service specifications (e.g., SOS 2.0) or applications using O&M.

2.2 OGC SensorML

For the description of sensor metadata, the SWE framework defines the Sensor Model Language (SensorML). SensorML 1.0 specifies a model and XML encoding for the description of all kinds of sensor related *processes*. A process can be for example a measurement procedure conducted by a sensor or the post processing of previously gathered data. In SensorML a sensor is defined as a process which is capable of observing a phenomenon and returning an observed value. It allows a detailed description of a process including a listing of its inputs, outputs, parameters, and process methods. Further metadata of a process can be defined including its identification and classification, as well as characteristics such as the temporal availability or its spatial description. SWE services use SensorML as a format for describing their associated sensors.

Thereby, the design of SensorML 1.0 focused primarily on the following functionalities:

- Supporting the discovery of sensors by providing a means for encoding sensor metadata.
- Providing information that can be used for understanding and analyzing data produced by the sensor (e.g., the parameters of the sensor calibration).
- Allowing the description of post processing steps that were performed on sensor data so that it can be reconstructed how a data set has been created.

The work on SensorML 2.0 is currently still in progress and addition of further functionalities is planned. First, a property inheritance mechanism for SensorML will be included. This mechanism aims at reducing the size and redundancy of sensor metadata descriptions by constructing inheritance hierarchies. Second, SensorML shall be extended to enable the precise and well-defined description of a sensor‘s protocol and interface. The vision behind such a detailed description of the sensor protocol is to enable an on-the-fly integration of the sensor with the Sensor Web, by using the protocol definition to transform sensor messages to Sensor Web protocols. The description of the sensor protocol, once designed for a particular sensor type, can then be reused in different scenarios and can be shared among user communities, which facilitates the usage of SWE in general. An extension of SensorML, which allows such a declarative description of the sensor protocol, has been proposed by the Sensor Interface Descriptor (SID) concept which may influence the development of SensorML 2.0.

The Open Geospatial Consortium (OGC) has approved the OGC Sensor Model Language (SensorML) 2.0 Encoding Standard. SensorML 2.0 provides a standard

encoding for describing sensors ("things that measure"), actuators ("things that act"), and processors ("things that calculate").

Because SWE standards are open standards based on open and universally accepted standards for the Internet and web, and for spatial location, they are foundational standards for communicating with sensors, actuators and processors whose location matters. They are a key enabler for the Internet of Things.

SensorML 2.0 includes a number of changes to the previous version 1.0.1, which was approved in 2007. SensorML 2.0 includes new or improved features, including:

- Support for using external schemas to describe sensor properties
- Better-defined support for positions and dynamic state (e.g. location, orientation, velocity, and acceleration)
- Better support for inheritance, allowing for more compact descriptions of deployed devices and processes
- Direct access to real-time values and data streams
- Better support for multiplexed data streaming (i.e. streams with disparate message types).

Since SensorML is very generic, potential use cases cover a broad range. However, this fact makes it also necessary to define profiles for SensorML in order to ensure that every SensorML based sensor description contains all metadata for the particular use case. An example for such a profile is the SensorML profile for discovery of sensors.

2.3 eVACUATE Data Model Mapping to OGC standards

This section describes the mapping of eVACUATE logical data models, already identified and described in the deliverable “D7.4 - Description of all the relevant agents for being integrated in the Smart Spaces”, to the data models and formats specified by the OGC standards.

The data models are grouped and represented following three domains:

- “OGC SensorML DM”: this data models are used to identify and discover the element.
- “OGC O&M DM”: this data model represents the actual data of the element.
- “OGC Tasking DM”: this data model is used to task or command the element.

The parameters depicted within quotation marks, i.e. “”, represent literals while parameters in italic represent parameters/variables of the eVACUATE Data Model.

2.3.1 Active Exit Signs

In this agent there are two different types of data models, classified by the functionality they are providing and the domains described in section 2.3. In the first section we have included the data models used for the updating the status of the Active Exit Signs in the system. And, in the second section, we have included the data models used to command these Active Exit Signs.

The following tables show the OGC standard mapping for updating the status data models of the Active Exit Signs.

DM_ExitSigns_STATUS_DATA_T	
OGC O&M DM	**eVACUATE DM Mapping**
Procedure	"tek-aes-" + *identifier*
observedProperty	"STATUS_DATA"
Type	"COUNT OBSERVATION"
Result	*Status*
timeElements[0].timeElement	("GML", *timestamp*)

Table 1: DM_ExitSigns_STATUS_DATA_T mapping to O&M.

DM_ExitSigns_COMMSTATUS_DATA_T	
OGC O&M DM	**eVACUATE DM Mapping**
procedure	"tek-aes-" + *identifier*
observedProperty	"COMMUNICATION_STATUS_DATA "
type	"COUNT OBSERVATION"
Result	*Commstatus*
timeElements[0].timeElement	("GML", *timestamp*)

Table 2: DM_ExitSigns_COMMSTATUS_DATA_T mapping to O&M.

The following tables show the OGC standard mapping for the tasking and commanding data models of the Active Exit Signs.

DM_ExitSigns_STATUS_COMMAND_T	
OGC Tasking DM	**eVACUATE DM Mapping**
procedure	"tek-aes-" + *identifier*
deviceType	"ACTIVE_EXIT_SIGN"
actionObject	*Status*

Table 3: DM_ExitSigns_STATUS_COMMAND_T mapping to Tasking

2.3.2 Chipless RFID printed Tags

In this agent there are two different types of data models, classified by the functionality they are providing and the domains described in section 2.3. In the first section we have included the data models used for the identification of the RFID reader. And, in the second section, we have included the data models used to communicate the counting of the Chipless RFID printed Tags that the reader is detecting.

The following tables show the OGC standard mapping for the identification data models of the RFID readers.

DM_RFID_TAG_IDENTIFICATION_T	
OGC SensorML DM	**eVACUATE DM Mapping**
procedure	"tud-rfid-" + *readerID*
outputs[0].observedProperty	"category"
metadata.inputs	----
metadata.parameters	----
position	*Location*
metadata.identifiers[0]	("VENDOR", "TUD")

Table 4: DM_RFID_TAG_IDENTIFICATION_T mapping to SensorML

The following tables show the OGC standard mapping for updating the status data models of the Chipless RFID printed Tags.

DM_RFID_TAG_DATA_T	
OGC O&M DM	**eVACUATE DM Mapping**
procedure	"tud-rfid-" + *readerID*
observedProperty	"category"
type	"COUNT OBSERVATION"
Result	*codeTAG*
timeElements[0].timeElement	("GML", *timestamp*)

Table 5: DM_RFID_TAG_DATA_T mapping to O&M.

2.3.3 Wireless Sensor Network - Temperature, Humidity and Light sensors

In this agent there are two different types of data models, classified by the functionality they are providing and the domains described in section 2.3. In the first section we have included the data models used for the identification of the sensors. And, in the second section, we have included the data models used to communicate the measurements and the status of the sensors.

The following tables show the OGC standard mapping for the identification data models of the sensors.

DM_WSN_IDENTIFICATION_DATA_T	
OGC SensorML DM	**eVACUATE DM Mapping**
procedure	“tim-wsn-“ + *sensor_id*
outputs[0].observedProperty	“TEMPERATURE"
outputs[1].observedProperty	"HUMIDITY"
outputs[2].observedProperty	"LUMINOSITY"
outputs[3].observedProperty	"COMMUNICATION_STATUS"
metadata.inputs	----
metadata.parameters	----
position	----
metadata.identifiers[0]	(“VENDOR”, “TIM”)

Table 6: DM_WSN_IDENTIFICATION_DATA_T mapping to SensorML

The following tables show the OGC standard mapping for updating the measurements and the status of the sensors.

DM_WSN_DATA_T	
OGC O&M DM	**eVACUATE DM Mapping**
procedure	“tim-wsn-“ + *sensor_id*
observedProperty	"TEMPERATURE", "HUMIDITY", "LUMINOSITY"
type	“MEASUREMENT”
Result	*Value*
timeElements[0].timeElement	(“GML”, *timestamp*)

Table 7: DM_WSN_DATA_T mapping to O&M.

DM_WSN_COMMSTATUS_DATA_T	
OGC O&M DM	**eVACUATE DM Mapping**
procedure	“tim-wsn-“ + *sensor_id*
observedProperty	"COMMUNICATION_STATUS"
type	“CATEGORY”
Result	*communication_status*
timeElements[0].timeElement	(“GML”, *timestamp*)

Table 8: DM_WSN_COMMSTATUS_DATA_T mapping to O&M.

2.3.4 Multimedia Devices

In this agent there are three different types of data models, classified by the functionality they are providing and the domains described in section 2.3. In the first section we have included the data models used for the identification of the Multimedia Devices and the media files available. In the second section we have included the data models used to update the status of the Multimedia Devices in the system. And, finally, we have included the data models used to command these Multimedia Devices.

The following tables show the OGC standard mapping for the identification data models of the Multimedia Devices.

DM_MultimediaDevices_DEVICE_INFO_T	
OGC SensorML DM	**eVACUATE DM Mapping**
Procedure	"exus-ds-" + *identifier*
outputs[0].observedProperty	"File being played"
metadata.inputs	----
metadata.parameters[0]	("Status", *status*)
Position	*Location*
metadata.identifiers[0]	("NickName", *name*)
metadata.identifiers[1]	("Vendor", "EXUS")

Table 9: DM_MultimediaDevices_DEVICE_INFO_T mapping to SensorML

DM_MultimediaDevices_MEDIAFILE_INFO_T	
OGC SensorML DM	**eVACUATE DM Mapping**
Procedure	"exus-mf-" + *file_identifier*
Outputs	----
metadata.inputs	----
metadata.parameters[0]	("Type", *type*)
metadata.parameters[1]	("Status", *status*)
Position	----
metadata.identifiers[0]	("File Name", *name*)
metadata.identifiers[1]	("Vendor", "EXUS")

Table 10: DM_MultimediaDevices_MEDIAFILE_INFO_T mapping to SensorML

The following tables show the OGC standard mapping for the data representation data models of the Multimedia Devices.

DM_MultimediaDevices_MEDIAFILE_UPDATE_T	
OGC O&M DM	**eVACUATE DM Mapping**
Procedure	"exus-ds-" + *identifier*
observedProperty	"File being played"
Type	"COUNT OBSERVATION"
Result	*playing_field*
timeElements[0].timeElement	("GML", *timestamp*)

Table 11: DM_MultimediaDevices_MEDIAFILE_UPDATE_T mapping to O&M

The following tables show the OGC standard mapping for the tasking and commanding data models of the Multimedia Devices.

DM_MultimediaDevices_MEDIAFILE_START_T	
OGC Tasking DM	**eVACUATE DM Mapping**
Procedure	“exus-ds-“ + *identifier*
deviceType	"MULTIMEDIA"
actionObject	(“PLAY”, *field*)

Table 12: DM_MultimediaDevices_MEDIAFILE_START_T mapping to Tasking

DM_MultimediaDevices_MEDIAFILE_STOP_T	
OGC Tasking DM	**eVACUATE DM Mapping**
Procedure	“exus-ds-“ + *identifier*
deviceType	"MULTIMEDIA"
actionObject	“STOP”

Table 13: DM_MultimediaDevices_MEDIAFILE_STOP_T mapping to Tasking

2.3.5 Computer vision

In this agent there is only one type of data model, classified by the functionality they are providing and the domains described in section 2.3.

The following tables show the OGC standard mapping for updating the measurements obtained from the image processing algorithms developed in WP3.

DM_CROWD_BM_VELOCITY_T	
OGC O&M DM	**eVACUATE DM Mapping**
procedure	“iti-cv-“ + *device_id*
observedProperty	“VELOCITY"
type	“ARRAY OBSERVATION”
Result	*[n] x (localX, localY, speed, direction)*
timeElements[0].timeElement	(“GML”, *timestamp*)

Table 14: DM_CROWD_BM_VELOCITY_T mapping to O&M

DM_CROWD_BM_BEHAVIOUR_T	
OGC O&M DM	**eVACUATE DM Mapping**
procedure	“iti-cv-“ + *device_id*
observedProperty	“BEHAVIOUR"
type	“ARRAY OBSERVATION”
Result	*[n] x (behaviour_level, localX, localY, frameX, frameY)*
timeElements[0].timeElement	(“GML”, *timestamp*)

Table 15: DM_CROWD_BM_BEHAVIOUR_T mapping to O&M

DM_CROWD_BM_DENSITY_T	
OGC O&M DM	**eVACUATE DM Mapping**
procedure	"iti-cv-" + *device_id*
observedProperty	"DENSITY"
type	"MEASUREMENT"
Result	*value*
timeElements[0].timeElement	("GML", *timestamp*)

Table 16: DM_CROWD_BM_DENSITY_T mapping to O&M

DM_CROWD_BM_FLOW_PASTLINE_T	
OGC O&M DM	**eVACUATE DM Mapping**
procedure	"iti-cv-" + *device_id*
observedProperty	"FLOW_PASTLINE"
type	"ARRAY OBSERVATION"
Result	*[n] x (line_id, flow)*
timeElements[0].timeElement	("GML", *timestamp*)

Table 17: DM_CROWD_BM_FLOW_PASTLINE_T mapping to O&M

DM_CROWD_BM_OBSTACLES_T	
OGC O&M DM	**eVACUATE DM Mapping**
procedure	"iti-cv-" + *device_id*
observedProperty	"OBSTACLES"
type	"ARRAY OBSERVATION"
Result	*[n] x (obstacle_type, localX, localY, frameX, frameY)*
timeElements[0].timeElement	("GML", *timestamp*)

Table 18: DM_CROWD_BM_OBSTACLES_T mapping to O&M

2.3.6 Application for SmartPhones

In this agent there is only one type of data model, classified by the functionality they are providing and the domains described in section 2.3.

The following tables show the OGC standard mapping for updating the measurements obtained from the SmartPhones and the events generated.

DM_ParticipatorySensing_EVENT_T	
OGC O&M DM	**eVACUATE DM Mapping**
procedure	"tel-smf-" + *device_token*
observedProperty	"SENSING_EVENT"
type	"CATEGORY"
Result	*participatory_sensing_event*
timeElements[0].timeElement	("GML", *timestamp*)

Table 19: DM_ParticipatorySensing_EVENT_T mapping to O&M

DM_MobileSensor_MEASUREMENTS_T	
OGC O&M DM	**eVACUATE DM Mapping**
procedure	“tel-smf-“ + *device_token*
observedProperty	"STEP_COUNT", “TEMPERATURE”, “HUMIDITY”, “LIGHT”, “PRESSURE”, “USER_STATUS”
type	“MEASUREMENT”
Result	*step_count, ambient_temperature, relative_humidity, light, pressure, mobile_sensor_status*
timeElements[0].timeElement	(“GML”, *timestamp*)

Table 20: DM_MobileSensor_MEASUREMENTS_T mapping to O&M

2.3.7 Information from Social Networks

In this agent there is only one type of data model, classified by the functionality they are providing and the domains described in section 2.3.

The following tables show the OGC standard mapping for the alert events generated by the Social Networks analyzer software.

DM_SocialMedia_ALERT_T	
OGC O&M DM	**eVACUATE DM Mapping**
Procedure	“tel-snwk-“ + *identifier*
observedProperty	"ALERT"
type	“CATEGORY”
Result	*alert*
timeElements[0].timeElement	(“GML”, *timestamp*)

Table 21: DM_SocialMedia_ALERT_T mapping to O&M

2.3.8 TETRA Messaging

In this agent there are two different types of data models, classified by the functionality they are providing and the domains described in section 2.3. In the first section we have included the data models used when receiving a TETRA message from any TETRA device. And, in the second section, we have included the data models used to confirm this message reception or to send a new message to other TETRA device.

The following tables show the OGC standard mapping for the reception of a TETRA message from any TETRA device.

DM_TETRA_RECEIVE_MESSAGE _T	
OGC O&M DM	**eVACUATE DM Mapping**
procedure	“aia-tetra-“ + *identifier*
observedProperty	"Message received"
type	“TEXT OBSERVATION”
Result	*message*
timeElements[0].timeElement	(“GML”, *timestamp*)

Table 22: DM_TETRA_RECEIVE_MESSAGE _T mapping to O&M

The following tables show the OGC standard mapping for confirming the received TETRA message or to send a new message to other TETRA device.

DM_TETRA_RECEIVE_CONFIRMATION _T	
OGC Tasking DM	**eVACUATE DM Mapping**
procedure	“aia-tetra-“ + *identifier*
deviceType	"AIA-TETRA"
actionObject	(*result*, *result_error*)

Table 23: DM_TETRA_RECEIVE_CONFIRMATION _T mapping to Tasking

DM_TETRA_SEND_MESSAGE _T	
OGC Tasking DM	**eVACUATE DM Mapping**
procedure	“aia-tetra-“ + *identifier*
deviceType	"AIA-TETRA"
actionObject	*message*

Table 24: DM_TETRA_SEND_MESSAGE _T mapping to Tasking

2.3.9 Smoke/Fire detection

In this agent there is only one type of data model, classified by the functionality they are providing and the domains described in section 2.3.

The following tables show the OGC standard mapping for the identification and the alarm events generated from the smoke and fire sensors.

DM_FIREDETECTION_IDENTIFICATION_DATA_T	
OGC O&M DM	**eVACUATE DM Mapping**
procedure	“stx-fds-“ + *identifier*
observedProperty	"IDENTIFICATION"
type	“TEXT OBSERVATION”
Result	*call_address*
timeElements[0].timeElement	(“GML”, *timestamp*)

Table 25: DM_FIREDETECTION_IDENTIFICATION_DATA_T mapping to O&M

DM_FIREDETECTION_ALARM_DATA_T	
OGC O&M DM	**eVACUATE DM Mapping**
procedure	“stx-fds-“ + *identifier*
observedProperty	"Fire Alarm"
type	“TEXT OBSERVATION”
Result	----
timeElements[0].timeElement	(“GML”, *timestamp*)

Table 26: DM_FIREDETECTION_ALARM_DATA_T mapping to O&M

2.3.10 Internal wireless DECT & Fixed phones

In this agent there are two different types of data models, classified by the functionality they are providing and the domains described in section 2.3. In the first section we have included the data models used when receiving a status from the wireless and wired phones. And, in the second section, we have included the data models used to confirm the received messages or to send a new message to other wireless and wired phones.

The following tables show the OGC standard mapping for the reception of the status from the wireless and wired phones.

DM_Phone_STATUS_T	
OGC O&M DM	**eVACUATE DM Mapping**
procedure	“stx-dfp-“ + *identifier*
observedProperty	"STATUS"
type	“COUNT OBSERVATION”
Result	*status*
timeElements[0].timeElement	(“GML”, *timestamp*)

Table 27: DM_Phone_STATUS_T mapping to O&M

The following tables show the OGC standard mapping for confirming the received messages or to send a new message to other wireless and wired phones.

DM_DECT_SEND_MESSAGE _T	
OGC Tasking DM	**eVACUATE DM Mapping**
procedure	“stx-dfp -“ + *identifier*
deviceType	"DECT_FIXED_PHONES"
actionObject	*message*

Table 28: DM_DECT_SEND_MESSAGE _T mapping to Tasking

DM_Phone_ACK_T	
OGC Tasking DM	**eVACUATE DM Mapping**
procedure	“stx-dfp -“ + *identifier*
deviceType	"DECT_FIXED_PHONES"
actionObject	*ack*

Table 29: DM_Phone_ACK_T mapping to Tasking

3 Ontology

3.1 Base Ontologies

This section describes the ontologies eVACUATE ontology is based on. The eVACUATE Ontology is an extension of the SSN ontology in which the specific data and device types have been added as well as other eVACUATE specific elements (such as rules). For the implementation, the ontology format defined by Sofia2 has been used. Thus, the eVACUATE ontology follows the logical schema of SSN but is implemented using Sofia2 ontology format and model.

3.1.1 Sofia2 Ontology Model

Sofia2 does not have an ontology in and of itself since it does not relate to a specific application domain, however is a tool where each implementation develops its own ontology model. While Sofia2 offers guidelines to build the ontologies within a solid model, the developers are not restricted to extend a specific base model or class as, indicatively, in JENA.[1] This section instead deals with the use of Sofia2 to create ontologies and ontology models.

Consider first that Sofia2 understands an ontology as "the definition of the set of classes and attributes that will share various applications that interoperate within the SmartSpace."[2] An ontology thus defines a specific entity that may be physical or abstract.

In Sofia2, individual ontologies are defined using JSON.[3] For purposes of modelling ontology, JSON is a format to unambiguously provide the data required for an ontology. Sofia2 defines first a JSON **schema**, a document that specifies the ontology referred to with restrictions - for instance, which fields are optional, which fields accept null value, which fields accept only numeric values, etc. This will be expanded in section 3.2.2 of this same document.

Ontologies are used to store useful information and to broadcast it throughout all the nodes in Sofia2. These nodes are seen as either the client-based Knowledge Processors (KPs) or the core Semantic Information Broker (SIB). The KPs are the generators and consumers of this information, with the SIB performing server-like functions of central processing, notifying a specific KP of changes that are defined as of that KP's interest or getting information from the combination of other factors.[4]

[1] https://jena.apache.org/documentation/ontology/
[2] Ontology Definition in Sofia2: http://sofia2.com/docs/%28EN%29%20SOFIA2-Ontology%20Definition%20in%20Sofia2.pdf
[3] JSON (JavaScript Object Notation) is an open standard, language-independent format that transmits data using objects made of pairs of attribute-value, in a format readable by humans. For more information: http://www.json.org/
[4] Ontology Definition in Sofia2: http://sofia2.com/docs/%28EN%29%20SOFIA2-Ontology%20Definition%20in%20Sofia2.pdf

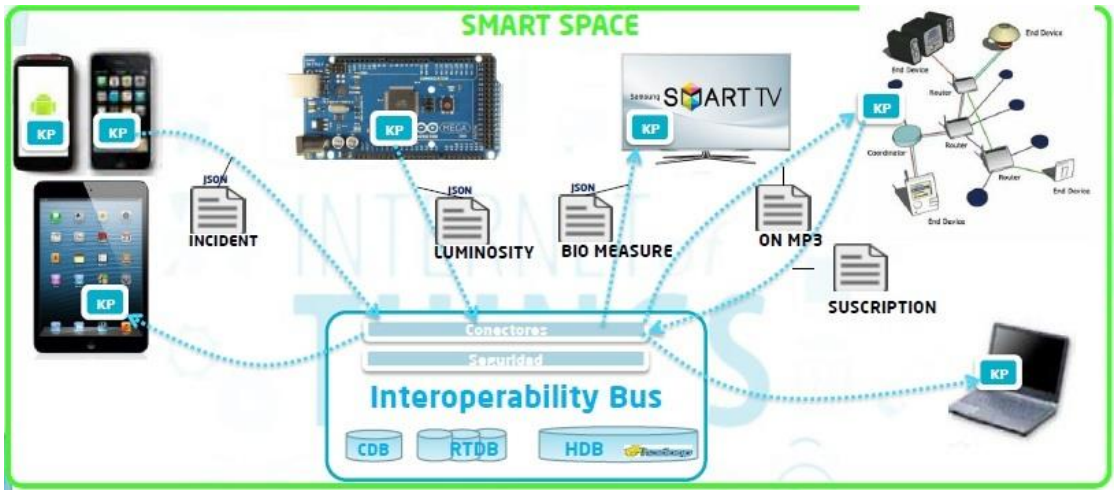

Figure 1: Sofia2

To handle the use of ontologies, Sofia uses **assets**. Assets are physical or virtual elements that can generate and/or consume information from sensors. Sofia2 defines a number of templates for these assets so that their use is easier and can be adapted to many more environments, from easily-seen Feeds to measure the environment to more complex KPI indicators or internal log messages called Audits.[5]

3.1.2 Semantic Sensor Network (SSN) Ontology Model

Semantic Sensor Network Ontology (SSN) is an ontology initially proposed by W3C SSN Incubator Group [SSN XG] and describes sensors, observations and processes. However it does not describe concepts such as time and location, as these needs to be included from other ontologies. It is encoded in Web Ontology Language (OWL). Currently SSN is supported by the W3C Semantic Sensor Networks Community Group[6].

SSN is part of the Semantic Sensor Web (SSW) research effort. SSW is the outcome of the combination of Sensor Networks and Semantic Web [BELT01]. SSW's main goal it to enrich the sensor's information with spatial, temporal and thematic metadata. Research on SSW according to [SHET08] examines the role of semantic annotation, ontologies and reasoning for improving the sensor web, with operations such as discovery and integration of sensors. This completes the vision of OGC about sensors, by adding the reasoning that Semantic Web offers.

As has already been mentioned, OGC SWE offers data models and sensor description standards, such as SensorML, O&M, that describe the sensors, their capabilities and the measurements they produce. These standards allow for the registration of sensors, understanding their processes, acquiring observations and the inter-exchange of data according to standardized xml tags.

[5] Ontology Governance in Sofia2: http://sofia2.com/docs/SOFIA2-Gobierno%20Ontologias.pdf (in Spanish)
[6] https://www.w3.org/community/ssn-cg/

On the other hand, SWE offers neither any semantic interoperability nor any reasoning that could lead to the development of advanced applications. By reasoning we mean deriving facts that are not expressed in ontology or the data (called knowledge base) explicitly.

Ontologies and related technologies are significant for the interoperability and integration in M2M systems, as they permit reasoning and classification functionality, which is lacking from OGC standards. The Semantic Sensor Network allows for sensor and acquired sensor data to be:

- Organized
- Installed
- Managed
- Queried
- M2M understood
- Controlled

under high level specifications. Due to the event-driven nature of the sensors and their temporal and spatial correlation, services that combine sensor data should take into account additional reasoning that is relative to the sensor physical restrictions such as power availability, restricted memory, variable conditions and lack of persistent connectivity. Ontologies can incorporate these restrictions.

3.2 Ontologies in sensors

Ontologies are necessary along with the knowledge base in semantic reasoning. The analytical definition of sensor models presents a research challenge. Many research groups have been working on the subject such as SWEET (Semantic Web for Earth and Environment Terminology[7]) that focuses on the modeling of environmental features and their observations. Their work is based on the O&M (see [PROB06], [NEUH09]) and specifically the ontology for the sensors descriptions is affected by SensorML [RUSS05].

In the same context, there has been research in specific domains on Sensor Web, like the project Marine Metadata Interoperability[8] that focuses on marine data [BERM06].

3.3 W3C Semantic Sensor Network Incubator Group

SSN is currently supported by the W3C Semantic Sensor Networks Community Group[9]. Community groups are run by the community but are hosted by W3C. Currently[10] the group has not published any report. The group continues the work of the W3C Semantic Sensor Web Incubator Group [SSNXG]. The analysis will be based on the final report [LEFO11] of the Incubator Group.

[7] http://sweet.ipl.nasa.gov/ontology
[8] https://marinemetadata.org
[9] https://www.w3.org/community/ssn-cg/
[10] 13 November 2015

The main objectives of the group were a) **the development of ontologies for describing sensors**, and b) the extension of the Sensor Model Language (SensorML), one of the four SWE languages, to **support semantic annotations**.

The first objective was met by providing a framework for describing sensors that allows classification and reasoning on capabilities and measurements of the sensors. Following W3C recommendation, OWL 2 DL is the selected language for ontology specification. These ontologies reflect the OGC standards, given that can encode sensor descriptions, mapping between the ontologies and OGC models. More specifically, it is based on:

- Observations and Measurements: provides standard models and XML Schema for encoding observations and measurements from a sensor, both archived and real-time.

The second objective, of semantic annotation of sensor descriptions and services that support sensor data exchange and sensor network management, serves a similar purpose to that offered by the semantic annotation of Web services. It is made possible by:

- SensorML: provides 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.
- Sensor Observation Service: GetCapabilities: The Sensor Observation Service includes three core operations: GetObservation, DescribeSensor, and GetCapabilities. The GetObservation operation provides an interface to query over observation data and returns an O&M document. The DescribeSensor operation provides an interface to query for the description of a sensor and returns a SensorML document. The GetCapabilities operation provides an interface to query for the description of a Sensor Observation Service. GetCapabilities allows clients to retrieve service metadata about a specific service instance and returns a GetCapabilites response document.

The Group has identified three directions for future work have been identified:

- standardise the SSN ontology to use it in a Linked Sensor Data context
- standardise the SSN ontology to bridge the Internet of Things and the Internet of Services
- foster the adoption of the SSN ontology in the OGC community

3.4 SSN Ontology Structure

SSN Ontology is a formal OWL DL ontology, based on OGC O&M utilizes conceptual modules to cover key sensor concepts. It uses DOLCE[11] (Descriptive Ontology for Linguistic and Cognitive Engineering) Ultra Light as an upper ontology. An overview of

[11] WonderWeb Project IST Programme of the Commission of the European Communities as project number IST-2001-33052

the SSN classes and properties organized as modules and their relationships is depicted in Figure 2.

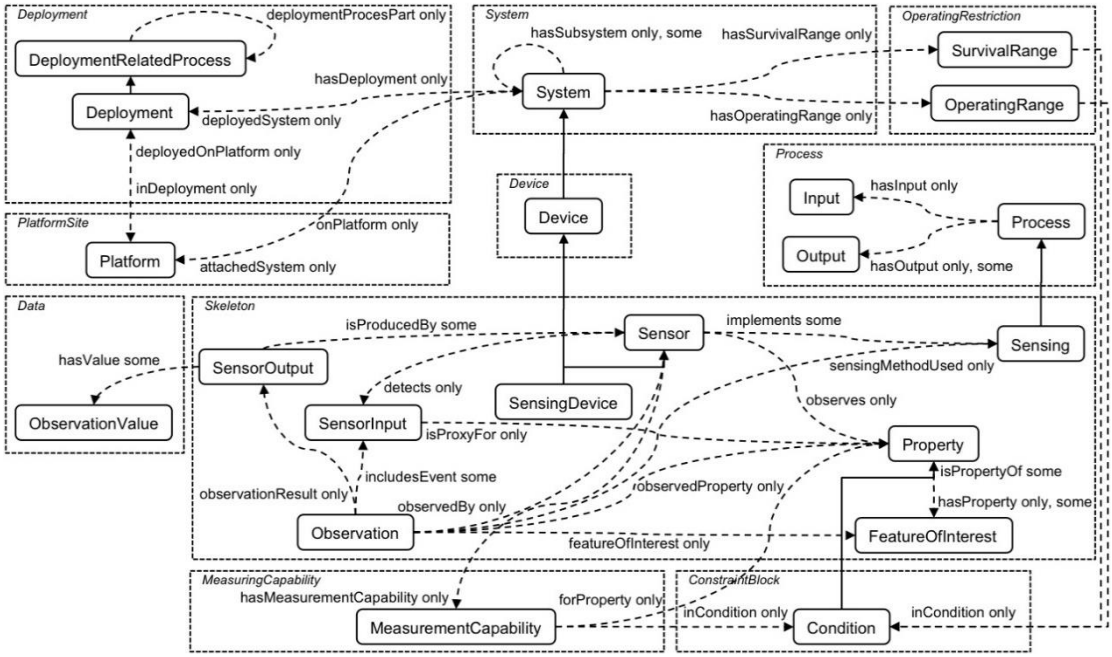

Figure 2: SSN Ontology classes and properties [LEFO11]

Basic classes that the ontology covers are:

- **SensingDevice** that inherits from Device and Sensor classes. A sensing device as a sensor, observes properties that are part of a FeatureOfInterest and produces SensorOutput by detecting SensorInput. It also implements a sensing that is described as a Process that has specific Input and Output. As a Device, a SensingDevice inherits the System class attributes and relations. As such, it is attached to Platform system and has specific OperatingRestrictions.
- **Process** describes the Input data and Output results of a Sensing Device.
- **Observation** is the result of SensorOutput, can include some SensorInput, is associated with a specific FeatureOfInterest, is observed a specific SensorDevice and follows a Sensing Process.

The Ontology modules that SSN has defined are:

- Skeleton module
- System module
- Process module
- Measuring module
- Measuring capability module
- Observation module
- Deployment module
- Platform Site module
- Operation Restriction module
- Energy module
- Device module

The development of the SSN follows a two-iteration approach:

- Phase 1: development of the ontology modules and examples,
- Phase 2: alignment to the DOLCE Ultra Lite (DUL) [upper ontology.

DOLCE Ultra Lite[12] is the foundational ontology and SSN ontology must align with it to facilitate reuse and interoperability. As an upper ontology, it describes the general concepts that are the same across all knowledge domains.

The alignment of the SSN ontology with the upper DOLCE Ultra Lite ontology is presented in Figure 3.

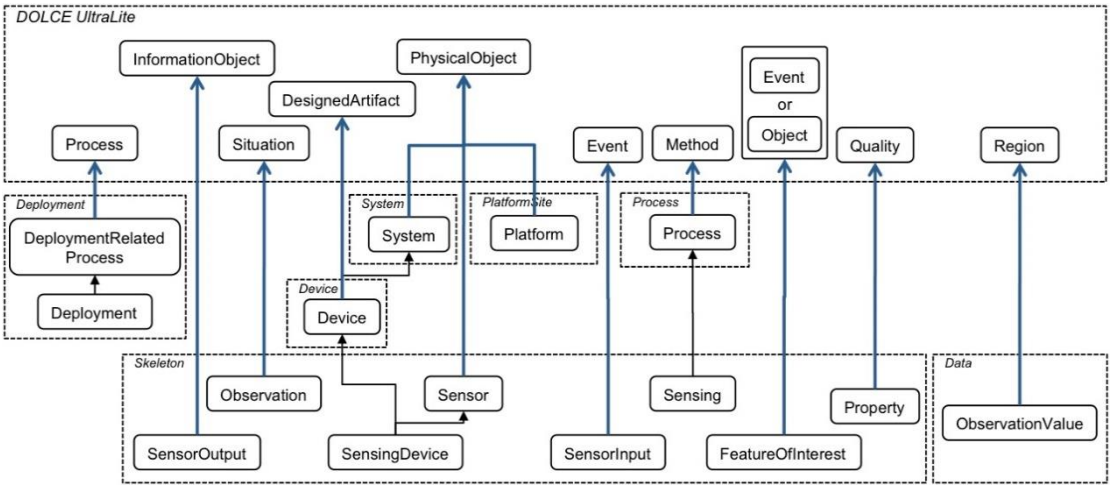

Figure 3: Alignment of DOLCE with SSN [LEFO11]

Regarding the modules the Skeleton module is based on the design pattern Stimulus –Sensor-Observation (SSO) and it assimilates almost all the main sensor ontologies. It aims at all kind of sensor or observation based ontologies and vocabularies for the Semantic Sensor Web and especially Linked Data. The pattern is developed following the principle of minimal ontological commitments to make it reusable for a variety of application areas. It is not aligned to any other top-level ontology and introduces a minimal set of classes and relations centered around the notions of stimuli, sensor, and observations.

The skeleton defines stimuli as the (only) link to the physical environment. Stimuli act as triggers for sensors and has as an immediate effect the production of measurements. A stimulus may only be usable as proxy for a specific region of an observed property.

In order to define SSO in the context of SSN, the main defined classes are: *ssn:Observation, ssn:Sensor, ssn:Stimulus, ssn:Property and ssn:FeatureOfInterest.* The Skeleton module aligned with DOLCE ontology is presented in Figure 4.

[12] http://www.w3.org/2005/Incubator/ssn/wiki/DUL_ssn#Section_DUL

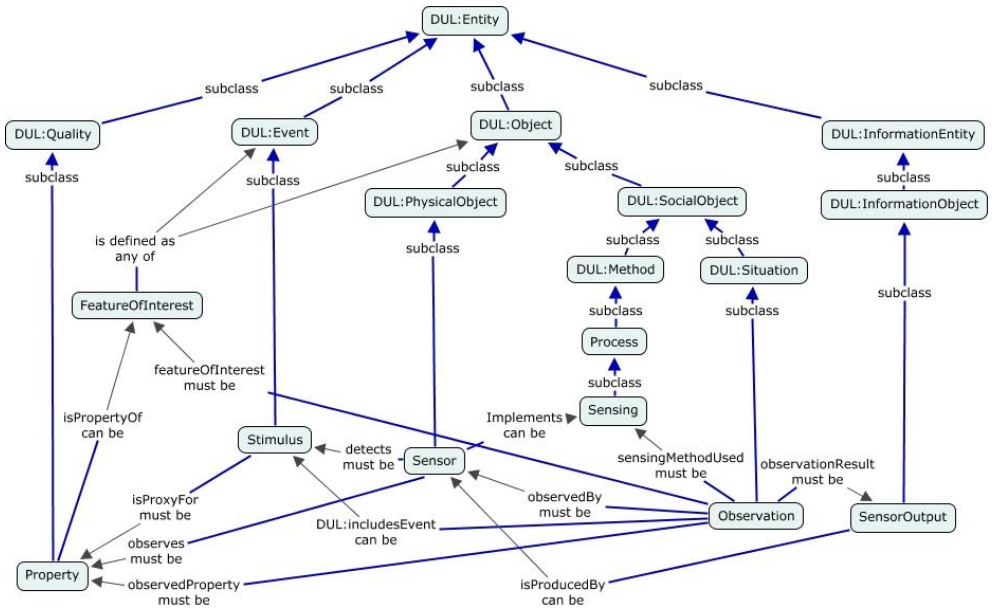

Figure 4: Skeleton module aligned with DOLCE ontology [LEFO11]

Finally, the SSN-XG has recognized 4 Use Cases for SSN ontology:

1. **Data Discovery and Linking**: Search for observations following specific criteria (spatial, time window) and relate the results with external data sources.
2. **Sensor Device Selection and Discovery**: Search for devices that match specific combination of criteria, such as device type, geographical region, observed phenomenon and availability, operator name etc.
3. **Provenance and Diagnostic**: support for additional device information that improves the acquired sensor data process and evaluation.
4. **Tasking, Programming**: Programming of the device based on the description and the provided terms of use.

The Use Cases and the required ontology (Figure 2) modules are presented in Figure 5.

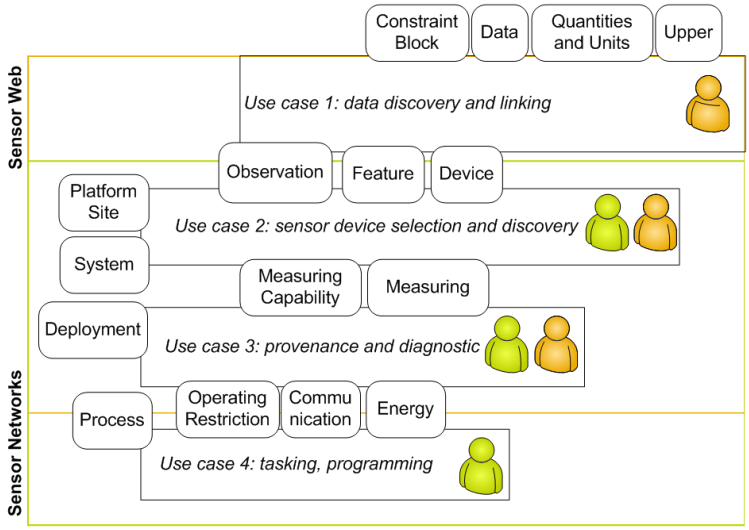

Figure 5: Use Cases and Ontology Modules [LEFO11]

This figure explains the area of concern of each use case and explains the need modules in each use case. For example, users developing Semantic Sensor Web Applications (actors represented in orange) should be more specifically interested in the modules that are linked to “data discovery and linking” and “provenance and diagnostics” use cases, where users developing Semantic Sensor Web applications (actors represented in green) need the modules connected to Device Discovery and Selection and “tasking, programming”.

3.5 eVACUATE Ontology

This section describes the logical schema of the eVACUATE ontology as well as its implementation following Sofia2 ontology model.

3.5.1 Logical Schema

Sofia2’s overall logical architecture is known and document in other sources such as the Sofia2 online documentation[13] and the Sofia2 blog for instance in this entry about IoT architecture gateways.[14]

[13] Sofia2 client Architecture: Reference KP http://sofia2.com/docs/SOFIA2-Desarrollo%20de%20un%20cliente%20Sofia2%20siguiendo%20Arquitectura%20KPModelo.pdf

[14] http://about.sofia2.com/2015/08/20/gateways-en-una-arquitectura-iot/ (in Spanish)

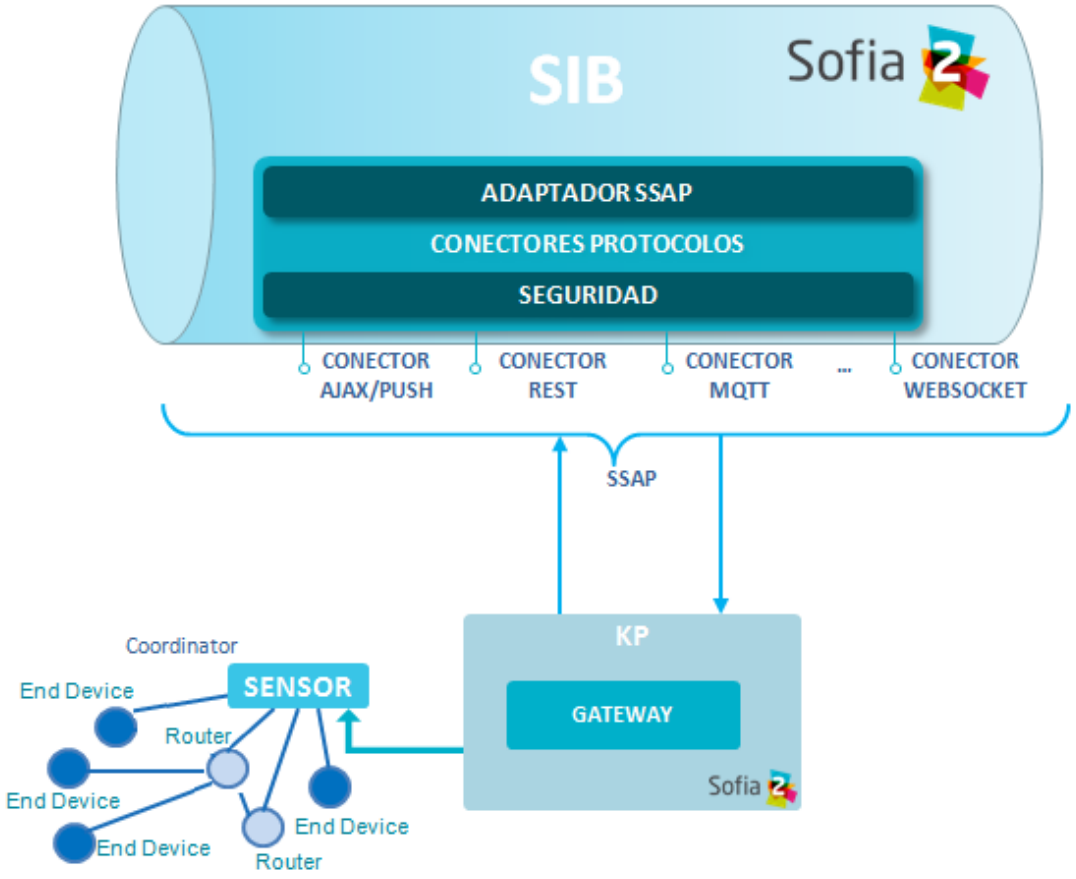

Figure 6: Sofia2 general architecture

Adapted to the specific needs of the eVACUATE implementation and at the same time restricting the explanation to the Sofia2 environment: A sensor identifies the data it wants to send and prepares it following a template. The sensor uses a specific interface so that this information reaches locally-running software known as Agent. All the sensors of a given kind in an area are centralized in a same Agent. An Agent can be used for more abstract concepts than sensors.

The Agent then adapts the information to ontologies in JSON format following a predefined JSON schema, unless this is already done before, and then uses the specific software KP to send this information to Sofia2's SIB, in the practice using SSAP (albeit this is beyond the logical aspect of the implementation). The KP runs locally within the Agent unlike the SIB, which is distributed (commonly in the Cloud).

The SIB performs its processing, storing information in a real-time database to make the best use of it, and accessing the rule engine to identify potential combinations of inputs that may unleash a certain protocol. Those protocols are also ontologies.

A certain kind of information consumer, represented to the SIB as a specific KP, may be interested in ("subscribed to") a certain kind of ontologies, either in any situation or with restrictions. For example, the consumer "Security Booth Computer" may want to

be notified periodically on the temperature of a given room (subscription); or whenever there is a change of more than 10 degrees in the temperature of a given room in less than 2 minutes (subscription with restriction); or only if there is a combination of factors such as sudden change of temperature in a room where there are at least two people (an ontology that appears only if the rule engine detects all of this and generates that information).

If required, the SIB will then send ontologies to the KP of a given consumer. The agent will forward the ontology to that consumer.

The SIB will also ensure that the data is stored in the historical data base, which is different from the real-time database both logically and physically. The HDB is accessed, for instance, by forensic analysts wanting to study past activities. Due to the sheer amount of data recovered, Big Data analysis is an interesting option to make good use of this. This kind of analysis can be done without the use of KP.

Similarly, the definition of rules for the rule engine can be made through the console. However, as the rules do use ontologies as their input, the rule administrator must have access to the ontologies to use them in the definition of rules. Logically, the rules are not directly stored in the same databases as the ontologies.

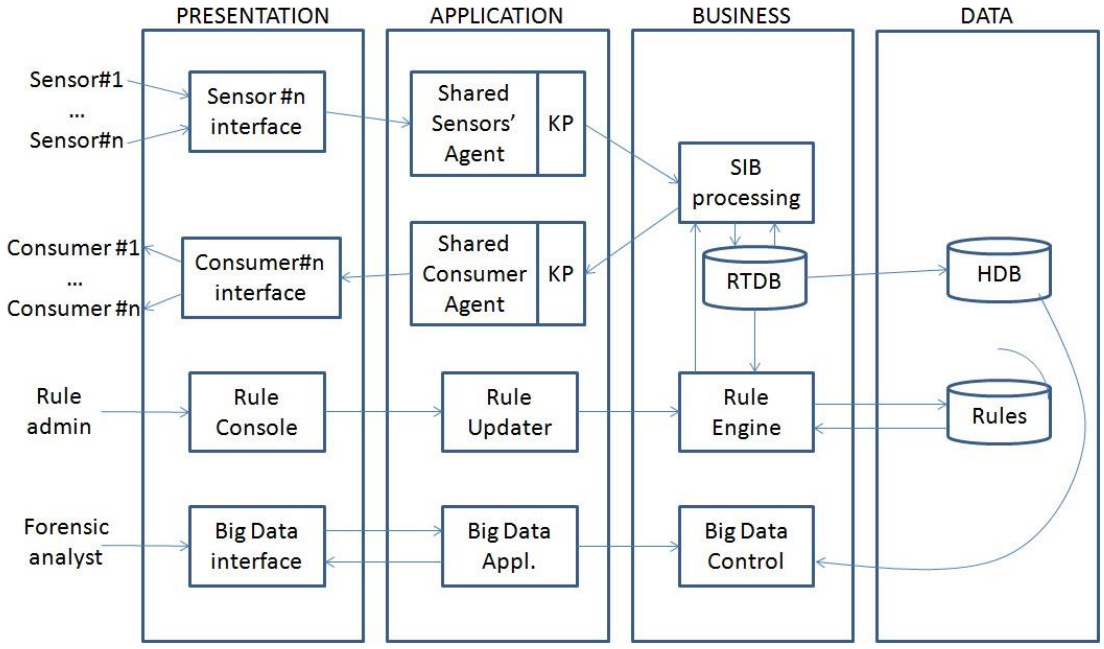

Figure 7: Sofia2 logical architecture for eVACUATE

3.5.2 Sofia2 Ontology Model Mapping

The definition of ontologies is not automatic in Sofia2. The user in charge of the sensor must follow a web-based interface to implement the details:

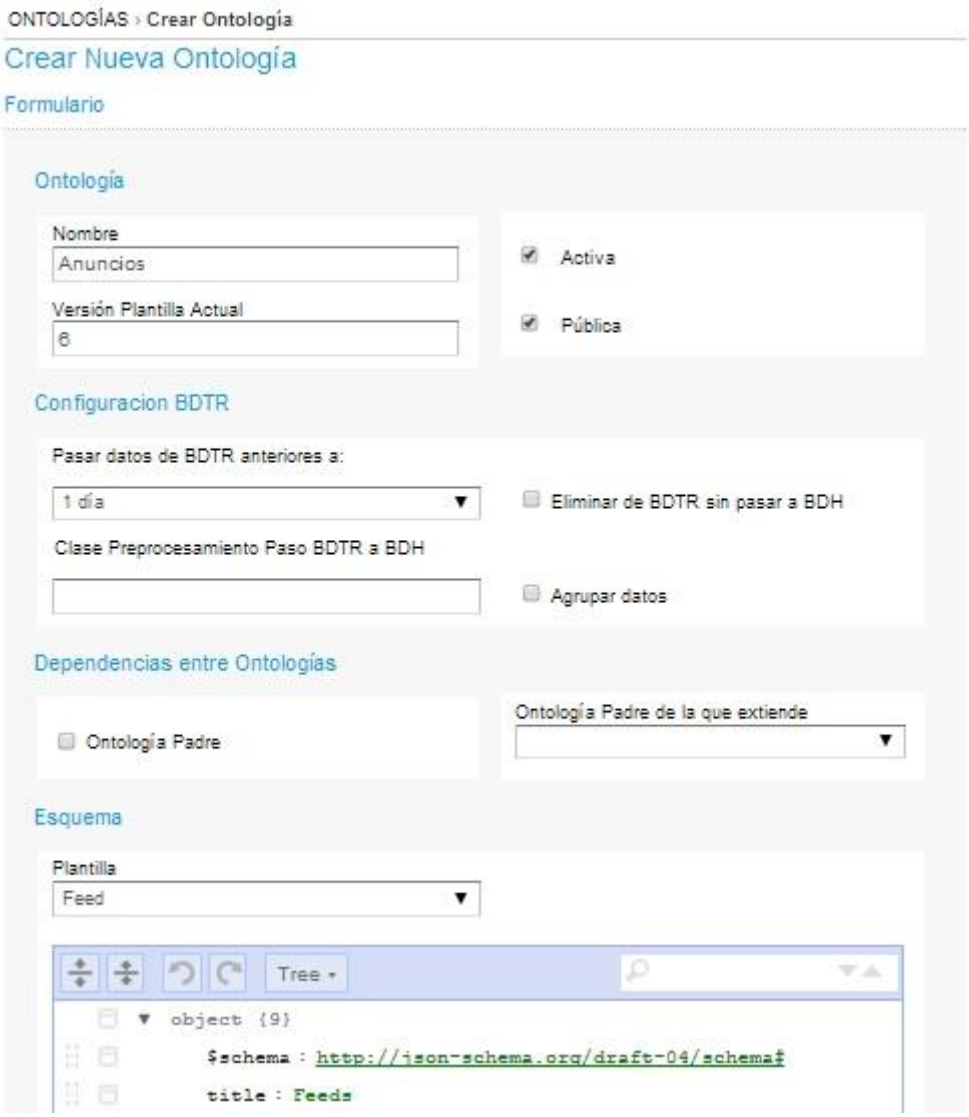

Figure 8: Ontology creation interface

The user must then define the JSON schema, adapted to their specific needs. Sofia2 offers a guide with ideas on how to do this.[15]

This is an example of a JSON schema, with all the possible data that this sensor can use:

[15] How to develop with Sofia2: http://sofia2.com/docs/SOFIA2-Como%20desarrollar%20sobre%20la%20Plataforma%20SOFIA2.pdf , section 6.2

```
SensorAnuncio.json
{
        "$schema": "http://json-schema.org/draft-04/schema#",
        "title": "Anuncios Schema",
        "type": "object",
        "required": [
                "SensorAnuncio"
        ],
        "properties": {
                "_id": {
                        "type": "object",
                        "$ref": "#/titulo"
                },
                "SensorAnuncio": {
                        "type": "string",
                        "$ref": "#/datos"
                }
        },
        "additionalProperties": false,
        "titulo": {
                "title": "id",
                "description": "Titulo insertado del Anuncio",
                "type": "object",
                "properties": {
                        "$oid": {
                                "type": "string"
                        }
                },
                "additionalProperties": false
        },
        "datos": {
                "title": "datos",
                "description": "Info contenido",
                "type": "object",
                "required": [
                        "titulo",
                        "timestamp",
                        "contenido"
                ],
                "properties": {
                        "titulo": {
                                "type": "string"
                        },
                        "timestamp": {
                                "type": "object",
                                "required": [
                                        "$date"
                                ],
                                "properties": {
                                        "$date": {
                                                "type": "string",
                                                "format": "date-time"
                                        }
                                },
                                "additionalProperties": false
                        },
                        "contenido": {
                                "type": "string"
                        }
                },
                "additionalProperties": false
        }
}
```

Figure 9: Example of JSON schema

Once the schema is defined, an instance of ontology will fill in the gaps with real data, which may be much shorter than the schema because, unlike the later, the instance can leave out any optional section:

```
SensorAnuncio-instance.json
{
        "SensorAnuncio": {
                "titulo":"Coca-Cola",
                "timestamp": {
                        "$date": "2014-02-24T13:32:03.176Z"
                },
                "contenido":"Lanza su nuevo envase de botella en formato de 350ml"
}
```

Figure 10: Example of JSON ontology instance

Consider on the other hand a different access of the process. The physical sensor has some real data in any format available. That data must be transformed into an ontology before sending it to Sofia2. This is done by the **ontologizer**. The ontologizer is a software that maps the sensor's raw data to adapt them to the JSON schema and thus generate an ontology. Being mostly a parsing program, easy to implement and consuming few resources, it is mostly done by the developers in charge of a given sensor. Currently the automation of the creation of ontologizers is not offered nor is considered a priority.

From a logical point of view, the ontologizer stands between the agent and the KP in the sending process. However, the ontologizer is a functionality that could exist in any previous point - for instance, if the output from a sensor is already adapted to send Sofia2-compatible ontologies.

The mapping in the other way round, from ontologies to real data, is particularly easy because JSON is even human-readable. It is not necessarily automatic but Sofia2 offers a number of interfaces that ease this translation.

3.5.2.1 Accessing ontologies without a KP

There is another point to be taken into account. Users access ontologies mostly using KPs. There are some cases where a user can reach an ontology without a KP - as long as it is a public ontology or that user is authorized to access that ontology.

Consider the rules. The rules manage the internal processing of one or more ontologies to get a less evident output that may -or may not- unleash an action. This can be done using Sofia2's web interface. In that case, the rule administrator creating (or editing) a rule can access the ontology using a web application. That nonetheless does not necessarily map the ontology's field into data.

The forensic analysis of the historical ontologies, using Big Data techniques or not, does not follow the traditional access to the ontologies either. However, the database still stores the ontologies and the mapping in itself is responsibility of the database analyst.

3.6 Mapping of eVACUATE Data Model to eVACUATE Ontology

In the following we report on the case-by-case mapping between Data Models and Ontologies in eVACUATE.

3.6.1 Dynamic Exit Signs

In the following tables the ontologies developed for the agent "Dynamic Exit signs" are presented.

Ontology of the data sent from the Agent to SOFIA:

eVACUATE Exit Sign Agent Schema:

Ontology	AgentExitSign		
Field name	**Field type**	**Description**	**Required**
identifier	string	Embedded Id of exit sign device	*
timestamp	string		
type	enum[UNKNOWN, WALL, FLOOR, DOOR]		
status	enum[OFF, RIGHT, LEFT, UP, DOWN, CLOSED, BLINK]		
commstatus	string		

Table 30: eVACUATE Exit Sign Agent Schema

The data models mapped in this ontology are:

- "DM_EXITSIGNS_IDENTIFICATION_DATA_T"
- "DM_ExitSigns_STATUS_DATA_T"
- "DM_ExitSigns_COMMSTATUS_DATA_T"

And the relationship between the data models and the ontology is indicated in the following table:

Ontology Mapping	AgentExitSign
DM_EXITSIGNS_IDENTIFICATION_DATA_T	
sensor_id	identifier
etype	type
timestamp	timestamp
DM_ExitSigns_STATUS_DATA_T	
sensor_id	identifier
timestamp	timestamp
estatus	status
DM_ExitSigns_COMMSTATUS_DATA_T	
sensor_id	identifier
timestamp	timestamp
ecommstatus	commstatus

Table 31: Relationship between the data model and the ontology

Ontology of the commands sent from SOFIA to the Agent:

eVACUATE Exit Sign command Agent Schema:

Ontology	AgentExitSign_cmd		
Field name	**Field type**	**Description**	**Required**
identifier	string	Embedded Id of exit sign device	*
status	enum[OFF, RIGHT, LEFT, UP, DOWN, CLOSED, BLINK]		*

Table 32: eVACUATE Exit Sign command Agent Schema

The data models mapped in this ontology are:

- "DM_ExitSigns_STATUS_COMMAND_T"

And the relationship between the data models and the ontology is indicated in the following table:

Ontology Mapping	AgentExitSign
DM_ExitSigns_STATUS_COMMAND_T	
sensor_id	identifier
estatus	status

Table 33: Relationship between the data model and the ontology

3.6.2 Chipless RFID printed Tags

In the following tables the ontologies developed for the agent “Chipless RFID printed Tags” are presented.

Ontology of the data sent from the Agent to SOFIA:

eVACUATE RFID-Producer Agent Schema:

Ontology	AgentRFID_producer		
Field name	**Field type**	**Description**	**Required**
readerID	string	Embedded Id of the entry RFID	*
location	string		
codeTAG	integer		
timestamp	string		

Table 34: eVACUATE RFID-Producer Agent Schema

The data models mapped in this ontology are:

- “DM_RFID_TAG_DATA_T”

And the relationship between the data models and the ontology is indicated in the following table:

Ontology Mapping	AgentRFID
DM_RFID_TAG_DATA_T	
ReaderID	readerID
CodeTAG	codeTAG
TimeStamp	timestamp

Table 35: Relationship between the data model and the ontology

3.6.3 Wireless Sensor Network - Temperature, Humidity and Light sensors

In the following tables the ontologies developed for the agent “Wireless Sensor Network - Temperature, Humidity and Light sensors” are presented.

Ontology of the data sent from the Agent to SOFIA:

eVACUATE ENV_WSN_TUL-Producer Agent Schema:

Ontology	AgentENV_WSN_TUL_producer		
Field name	**Field type**	**Description**	**Required**
sensor_id	string	Embedded Id of the entry ENV_WSN_TUL	*
timestamp	string		
Type	enum["UNKNOWN", "TEMPERATURE", "HUMIDITY", "LUMINOSITY"]		

Value	number
communication_status	enum["UNKNOWN", "CONNECTED", "DISCONNECTED"]

Table 36: eVACUATE ENV_WSN_TUL-Producer Agent Schema

The data models mapped in this ontology are:

- "DM_WSN_IDENTIFICATION_DATA_T"
- "DM_WSN_DATA_T"
- "DM_WSN_COMMSTATUS_DATA_T"

And the relationship between the data models and the ontology is indicated in the following table:

Ontology Mapping	AgentENV_WSN
DM_WSN_IDENTIFICATION_DATA_T	
sensor_id	sensor_id
etype	type
timestamp	timestamp
DM_WSN_DATA_T	
sensor_id	sensor_id
etype	type
timestamp	timestamp
Value	value
DM_WSN_COMMSTATUS_DATA_T	
sensor_id	sensor_id
timestamp	timestamp
ecommstatus	communication_status

Table 37: Relationship between the data model and the ontology

3.6.4 Multimedia Devices

In the following tables the ontologies developed for the agent "Multimedia Devices" are presented.

Ontology of the data sent from the Agent to SOFIA:

eVACUATE Multimedia Devices Agent Schema:

Ontology	AgentMultimediaDevices		
Field name	**Field type**	**Description**	**Required**
Identifier	string	Embedded Id of Multimedia device	*
file_identifier	string		
Name	string		
Location	string		

Type	string
status	string
playing_field	string
timestamp	string

Table 38: eVACUATE Multimedia Devices Agent Schema

The data models mapped in this ontology are:

- "DM_MultimediaDevices_DEVICE_INFO_T"
- "DM_MultimediaDevices_MEDIAFILE_INFO_T"
- "DM_MultimediaDevices_MEDIAFILE_UPDATE_T"

And the relationship between the data models and the ontology is indicated in the following table:

Ontology Mapping	AgentMultimediaDevices
DM_MultimediaDevices_DEVICE_INFO_T	
sign_id	identifier
timestamp	timestamp
name	name
location	location
status	status
DM_MultimediaDevices_MEDIAFILE_INFO_T	
file_id	file_identifier
timestamp	timestamp
name	name
type	type
status	status
DM_MultimediaDevices_MEDIAFILE_UPDATE_T	
sign_id	identifier
timestamp	timestamp
playing_field	playing_field

Table 39: Relationship between the data model and the ontology

Ontology of the commands sent from SOFIA to the Agent:

eVACUATE Multimedia Devices command Agent Schema:

Ontology	AgentMultimediaDevices_cmd		
Field name	**Field type**	**Description**	**Required**
command	enum["START", STOP"]	Multimedia device command	*
identifier	string		*
Field	string		

Table 40: eVACUATE Multimedia Devices command Agent Schema

The data models mapped in this ontology are:

- "DM_MultimediaDevices_MEDIAFILE_START_T"
- "DM_MultimediaDevices_MEDIAFILE_STOP_T"

And the relationship between the data models and the ontology is indicated in the following table:

Ontology Mapping	AgentMultimediaDevices
DM_MultimediaDevices_MEDIAFILE_START_T	
sign_id	identifier
Field	field
DM_MultimediaDevices_MEDIAFILE_STOP_T	
sign_id	identifier

Table 41: Relationship between the data model and the ontology

3.6.5 Computer vision

In the following tables the ontologies developed for the agent "Computer vision" are presented.

Ontologies of the data sent from the Agent to SOFIA:

eVACUATE CROWD_BM-Producer Agent Velocity Schema:

Ontology	AgentCROWD_BM_velocity		
Field name	**Field type**	**Description**	**Required**
device_id	string	Embedded Id of the entry CROWD_BM	*
timestamp	string		*
velocities	array [*localX*, *localY*, *speed*, *direction*]	Array element of velocities for CROWD_BM Velocity	*
localX	number		
localY	number		
speed	number		
direction	integer		

Table 42: eVACUATE CROWD_BM-Producer Agent Velocity Schema

The data models mapped in this ontology are:

- "DM_CROWD_BM_VELOCITY_T"

And the relationship between the data models and the ontology is indicated in the following table:

Ontology Mapping	AgentCROWD_BM
DM_CROWD_BM_VELOCITY_T	
DeviceID	device_id
TimeStamp	timestamp

Velocities	velocities
LocalX	localX
LocalY	localY
Speed	speed
Direction	direction

Table 43: Relationship between the data model and the ontology

eVACUATE CROWD_BM-Producer Agent Behaviour Schema:

Ontology	AgentCROWD_BM_behaviour		
Field name	**Field type**	**Description**	**Required**
device_id	string	Embedded Id of the entry CROWD_BM	*
timestamp	string		*
behaviour_levels	array [*behaviour_level*, *localX*, *localY*, *speed*, *direction*]	Array element of behaviour_levels for CROWD_BM Behaviour	*
behaviour_level	string		
localX	number		
localY	number		
frameX	integer		
frameY	integer		

Table 44: eVACUATE CROWD_BM-Producer Agent Behaviour Schema

The data models mapped in this ontology are:

- "DM_CROWD_BM_BEHAVIOUR_T"

And the relationship between the data models and the ontology is indicated in the following table:

Ontology Mapping	AgentCROWD_BM
DM_CROWD_BM_BEHAVIOUR_T	
DeviceID	device_id
TimeStamp	timestamp
BehaviourLevels	behaviour_levels
BehaviourLevel	behaviour_level
LocalX	localX
LocalY	localY
FrameX	frameX
FrameY	frameY

Table 45: Relationship between the data model and the ontology

eVACUATE CROWD_BM-Producer Agent Density Schema:

Ontology	AgentCROWD_BM_density		
Field name	**Field type**	**Description**	**Required**
device_id	string	AgentCROWD_BM_density	*
timestamp	string		*
value	integer		*

Table 46: eVACUATE CROWD_BM-Producer Agent Density Schema

The datamodels mapped in this ontology are:

- "DM_CROWD_BM_DENSITY_T"

And the relationship between the datamodels and the ontology is indicated in the following table:

Ontology Mapping	AgentCROWD_BM
DM_CROWD_BM_DENSITY_T	
DeviceID	device_id
TimeStamp	timestamp
Value	value

Table 47: Relationship between the data model and the ontology

eVACUATE CROWD_BM-Producer Agent Flow PastLine Schema:

Ontology	AgentCROWD_BM_flow_pastline		
Field name	**Field type**	**Description**	**Required**
device_id	string	AgentCROWD_BM_density	*
timestamp	string		*
flows_pastline	array [*line_id*, *flow*]	Array element of flows_pastline for CROWD_BM Flow PastLine	*
line_id	string		
flow	number		

Table 48: eVACUATE CROWD_BM-Producer Agent Flow PastLine Schema

The data models mapped in this ontology are:

- "DM_CROWD_BM_FLOW_PASTLINE_T"

And the relationship between the data models and the ontology is indicated in the following table:

Ontology Mapping	AgentCROWD_BM
DM_CROWD_BM_FLOW_PASTLINE_T	
DeviceID	device_id

TimeStamp	timestamp
FlowPastLine	flows_pastline
LineID	line_id
Flow	flow

Table 49: Relationship between the data model and the ontology

eVACUATE CROWD_BM-Producer Agent Obstacle Schema:

Ontology	AgentCROWD_BM_obstacle		
Field name	**Field type**	**Description**	**Required**
device_id	string	AgentCROWD_BM_density	*
timestamp	string		*
obstacles	array [*obstacle_type*, *localX*, *localY*, *frameX*, *frameY*]	Array element of obstacles for CROWD_BM Obstacle	*
obstacle_type	string		
localX	number		
localY	number		
frameX	integer		
frameY	integer		

Table 50: eVACUATE CROWD_BM-Producer Agent Obstacle Schema

The data models mapped in this ontology are:

- "DM_CROWD_BM_OBSTACLES_T"

And the relationship between the data models and the ontology is indicated in the following table:

Ontology Mapping	AgentCROWD_BM
DM_CROWD_BM_OBSTACLES_T	
DeviceID	device_id
TimeStamp	timestamp
Obstacles	obstacles
ObstacleType	obstacle_type
LocalX	localX
LocalY	localY
FrameX	frameX
FrameY	frameY

Table 51: Relationship between the data model and the ontology

3.6.6 Application for Smartphones

In the following tables are represented the ontologies developed for the agent "Application for Smartphones".

Ontologies of the data sent from the Agent to SOFIA:

eVACUATE AppMobile-Environment-Producer Agent Schema:

Ontology	AgentAppMobile_producer		
Field name	**Field type**	**Description**	**Required**
device_token	string	Embedded Id of the entry AppMobile Environment	*
location_id	string		*
timestamp	string		*
step_count	number		
ambient_temperature	number		
pressure	number		
light	integer		
relative_humidity	integer		
participatory_sensing_event	enum ["UNKNOWN", "HIJACKING", "LOOTING", "MAN_OVERBOARD", "RIOTING", "FIRE", "FLOOD", "SMOKE", "DISTURBANCE"]		
mobile_sensor_status	enum ["UNKNOWN", "FALLEN", "RUNNING"]		

Table 52: eVACUATE AppMobile-Environment-Producer Agent Schema

The data models mapped in this ontology are:

- "DM_ParticipatorySensing_EVENT_T"
- "DM_MobileSensor_MEASUREMENTS_T"

And the relationship between the data models and the ontology is indicated in the following table:

Ontology Mapping	AgentAppMobile
DM_ParticipatorySensing_EVENT_T	
device_token	device_token
location_id	location_id
timestamp	timestamp
event	participatory_sensing_event
DM_MobileSensor_MEASUREMENTS_T	

device_token	device_token
location_id	location_id
timestamp	timestamp
step_count	step_count
ambient_temperature	ambient_temperature
relative_humidity	relative_humidity
light	light
pressure	pressure
user_status	mobile_sensor_status

Table 53: Relationship between the data model and the ontology

3.6.7 Information from Social Networks

In the following tables are represented the ontologies developed for the agent "Information from Social Networks".

Ontologies of the data sent from the Agent to SOFIA:

eVACUATE Social Media data Agent Schema:

Ontology	AgentSocialMedia		
Field name	**Field type**	**Description**	**Required**
identifier	string	Embedded Id of Social Media source	*
timestamp	string		*
alert	enum ["GENERAL_OVER_THRESHOLD", "FIRE", "CROWDING", "EXPLOSION", "FALLEN_PERSON", "TRANSPORTATIONS_HALTED"]		*

Table 54: eVACUATE Social Media data Agent Schema

The data models mapped in this ontology are:

- "DM_SocialMedia_ALERT_T"

And the relationship between the data models and the ontology is indicated in the following table:

Ontology Mapping	AgentSocialMedia
DM_SocialMedia_ALERT_T	
location_id	identifier
alert	alert
timestamp	timestamp

Table 55: Relationship between the data model and the ontology

3.6.8 TETRA Messaging

In the following tables the ontologies developed for the agent “TETRA Messaging” are presented.

Ontology of the data sent from the Agent to SOFIA:

eVACUATE TETRA data Agent Schema:

Ontology	AgentAIATETRA		
Field name	**Field type**	**Description**	**Required**
identifier	string	Embedded Id of AIA TETRA device	*
timestamp	string		*
message	string		
result	enum["SUCCESSFUL_TRANSMISSION", "TRANSMISSION_FAILED"]		
result_error	enum["MT_FAILURE", ”OPERATION_NOT_SUPPORT", “NO_PROCESSING", “NETWORK_TIMEOUT", "INVALID_PDU_MODE", "INVALID_TEXT_MODE_PARAMETER", "SYNTAX_ERROR", "DATA_WITHOUT_REQUEST", "DATA_TIMEOUT", "REGISTRATION_TIMEOUT_EXPIRY", "DEREGISTRATION_TIMEOUT_EXPIRY", "OTHER_APPLICATION_REGISTERED", "REGISTRATION_TABLE_FULL", "REGISTRATION_DISABLED", "DEREGISTRATION_ILLEGAL", "DEREGISTRATION_NOT_POSSIBLE"]		

Table 56: eVACUATE TETRA data Agent Schema

The data models mapped in this ontology are:

- “DM_TETRA_RECEIVE_MESSAGE _T”
- “DM_TETRA_RECEIVE_CONFIRMATION _T”

And the relationship between the data models and the ontology is indicated in the following table:

Ontology Mapping	AgentTETRA
DM_TETRA_RECEIVE_MESSAGE _T	
sender_id	identifier
timestamp	timestamp
message	message

DM_TETRA_RECEIVE_CONFIRMATION _T	
sender_id	identifier
timestamp	timestamp
resultAIATetra	result
errorAIATetra	result_error

Table 57: Relationship between the data model and the ontology

Ontology of the commands sent from SOFIA to the Agent:

eVACUATE TETRA cmd Agent Schema:

Ontology	AgentAIATETRA_cmd		
Field name	**Field type**	**Description**	**Required**
identifier	string	Embedded Id of AIA TETRA command device	*
timestamp	string		*
message	string		*

Table 58: eVACUATE TETRA cmd Agent Schema

The data models mapped in this ontology are:

- "DM_TETRA_SEND_MESSAGE _T"

And the relationship between the data models and the ontology is indicated in the following table:

Ontology Mapping	AgentTETRA
DM_TETRA_SEND_MESSAGE _T	
receiver_id	identifier
timestamp	timestamp
message	message

Table 59: Relationship between the data model and the ontology

3.6.9 Smoke/Fire detection

In the following tables the ontologies developed for the agent "Smoke/Fire detection" are presented.

Ontologies of the data sent from the Agent to SOFIA:

eVACUATE fire Detection Agent Schema:

Ontology	AgentFireDetection		
Field name	**Field type**	**Description**	**Required**

identifier	string	Embedded Id of Fire Detection device	*
timestamp	string		*
call_address	string		

Table 60: eVACUATE fire Detection Agent Schema

The data models mapped in this ontology are:

- "DM_FIREDETECTION_IDENTIFICATION_DATA_T"
- "DM_FIREDETECTION_ALARM_DATA_T"

And the relationship between the data models and the ontology is indicated in the following table:

Ontology Mapping	AgentFireDetection
DM_FIREDETECTION_IDENTIFICATION_DATA_T	
fire_detector_id	identifier
call_address	call_address
timestamp	timestamp
DM_FIREDETECTION_ALARM_DATA_T	
fire_detector_id	identifier
timestamp	timestamp

Table 61: Relationship between the data model and the ontology

3.6.10 Internal wireless DECT & Fixed phones

In the following tables the ontologies developed for the agent "Internal wireless DECT & Fixed phones" are presented.

Ontology of the data sent from the Agent to SOFIA:

eVACUATE DECT Agent Schema:

Ontology	AgentDECT		
Field name	**Field type**	**Description**	**Required**
identifier	string	Embedded Id of DECT device	*
sequence_count	integer		
timestamp	string		*
ack	enum["NOT_READ", "READ"]		
status	enum["UNKNOWN", "CONNECTED", "DISCONNECTED"]		

Table 62: eVACUATE DECT Agent Schema

The data models mapped in this ontology are:

- "DM_Phone_ACK_T"
- "DM_Phone_STATUS_T"

And the relationship between the data models and the ontology is indicated in the following table:

Ontology Mapping	AgentDECT
DM_Phone_ACK_T	
call_address	identifier
sequence_cnt	sequence_count
ack	ack
timestamp	timestamp
DM_Phone_STATUS_T	
call_address	identifier
sequence_cnt	sequence_count
status	status
timestamp	timestamp

Table 63: Relationship between the data model and the ontology

Ontology of the commands sent from SOFIA to the Agent:

eVACUATE DECT Agent_cmd Schema:

Ontology	AgentDECT_cmd		
Field name	**Field type**	**Description**	**Required**
identifier	string	Embedded Id of DECT command device	*
sequence_count	integer		
timestamp	string		*
message	string		*

Table 64: eVACUATE DECT Agent_cmd Schema

The data models mapped in this ontology are:

- "DM_DECT_SEND_MESSAGE _T"

And the relationship between the data models and the ontology is indicated in the following table:

Ontology Mapping	AgentDECT
DM_DECT_SEND_MESSAGE _T	
call_address	identifier
sequence_cnt	sequence_count

Message	message
Timestamp	timestamp

Table 65: Relationship between the data model and the ontology

4 Conclusion

The deliverable reported on the actual implementation of activities relative to Data Models and Ontologies (tasks T7.3 and T7.4, respectively).

The Data Models were defined, to provide a level of abstraction required for the interoperability of heterogeneous equipment. Existing standards are presented, as in the case of the SWE of the OGC and specifically the “O&M” (Observations & Measurements) standard which is used to model sensor data and respectively the “SensorML”, an XML variant used to model meta-data about sensors.

With regard to the Ontology, a layer that interfaces with WP8 was implemented, that combines semantics & ontology with data processing/fusion at the Smart Space level. The ontology provides definitions for the structure of sensors and observations (accordingly actuators and decisions), leaving the details of the observed domain unspecified.

Specific implementations are offered for all components of the Smart Space, namely the Dynamic Exit Signs, the Chip less RFID printed Tags, the Temperature, Humidity and Light sensors, that are part of the Wireless Sensor Network, the Multimedia Devices, Computer vision, the Application for Smartphones, the Information from Social Networks, the TETRA Messaging, the Smoke/Fire detection, and finally the Wireless DECT & Fixed phones.

4.1 Future Work

In this draft version the ontologies presented are still ad-hoc, they were created to enable the functionality of the Agent Framework. A unifying, “global” ontology description which will cover all the independent ontologies developed so far, will be presented in the final version, which comes two months after this version.

The final version of the deliverable will also include the ontology already produced for communication internally with the WP3 modules, i.e. the description of the features extracted from video analysis (Density, Velocity, Collectiveness, Target direction, etc.) and the resulting recognized unusual behaviors by using semantic web standards (RDF).

5 References

[SSNXG] Semantic Sensor Network Incubator Group (SSN-XG), http://www.w3.org/2005/Incubator/ssn/ssnx/ssn

[SHET08] Sheth, A.; Henson, C.; Sahoo, S. Semantic Sensor Web. IEEE Internet Comput. 2008, 12, 78–83.

[BELT01] Berners-Lee, Tim; James Hendler και Ora Lassila (May 17, 2001). "The Semantic Web", Scientific American Magazine

[PROB06] Probst, F. Ontological Analysis of Observations and Measurements. In Proceedings of Geographic Information Science, 4th International Conference, GIScience 2006, Münster, Germany, September 2006; Raubal, M., Miller, H.J., Frank, A.U., Goodchild, M.F., Eds.; Lecture Notes in Computer Science; Springer: Berlin, Germany, 2006, Volume 4197, pp. 304–320.

[NEUH09] Neuhaus, H.; Compton, M. The Semantic Sensor Network Ontology: A Generic Language to Describe Sensor Assets. In Proceedings of 12th AGILE International Conference on Geographic Information Science, Workshop on Challenges in Geospatial Data Harmonisation, Hannover, Germany, June 2009; Available online: http:// plone.itc.nl/agile_old/Conference/2009-hannover/ shortpaper.htm

[RUSS05] Russomanno, D.; Kothari, C.; Thomas, O. Building a Sensor Ontology: A Practical Approach Leveraging ISO and OGC Models. In Proceedings of The 2005 International Conference on Artificial Intelligence (IC-AI 2005), Las Vegas, NV, USA, June 2005; CSREA Press: Las Vegas, NV, USA, 2005; pp. 637–643.

[BERM06] Bermudez, L.; Graybeal, J.; Arko, R. A Marine Platforms Ontology: Experiences and Lessons. Workshop on Semantic Sensor Networks (SSN 2006). In Proceedings of conjunction with 5th International Semantic Web Conference (ISWC 2006), Athens, GA, USA, November 2006; Available online: http://www.ict.csiro.au/ssn06/ (accessed on 15 September 2015).

[LEFO11] L. Lefort et al., eds., Semantic Sensor Network XG Final Report, W3C Incubator Group Report, June 2011; Available online: www. w3.org/2005/Incubator/ssn/XGR-ssn-20110628.