

# A New Architecture for the Sensor Web: The SWAP Framework

Deshendran Moodley<sup>1</sup> and Ingo Simonis<sup>2</sup>

<sup>1</sup> School of Computer Science, Westville Campus  
University of KwaZulu Natal, Durban, 4001, South Africa  
moodleyd37@ukzn.ac.za

<sup>2</sup> Institute for Geoinformatics, University of Muenster,  
48149 Muenster, Germany  
simonis@uni-muenster.de

<sup>2</sup> ICT for Earth Observation Research Group  
Meraka Institute, Pretoria, South Africa  
isimonis@csir.co.za

**Abstract.** Sensor Web is a revolutionary concept towards achieving a collaborative, coherent, consistent, and consolidated sensor data collection, fusion and distribution system. Sensor Webs can perform as an extensive monitoring and sensing system that provides timely, comprehensive, continuous and multi-mode observations. This new earth-observation system opens up a new avenue to fast assimilation of data from various sensors (both in situ and remote) and to accurate analysis and informed decision makings. So far, most of the sensor networks work in isolated environments rather than being integrated into a higher system of environmental systems. Our vision of the Sensor Web is an open infrastructure that allows end users to automatically access and extract and use appropriate information from multiple sensor sources over the Internet. This paper proposes an architectural framework, the *Sensor Web Agent Platform* (SWAP) that makes use of two of the most promising distributed architectural paradigms i.e. Web Services and Multi Agent Systems. SWAP allows for integrating arbitrary sensors or sensor networks into a loosely coupled higher level environment that facilitates developing and deploying end user applications across multiple application domains. An ontology framework and an abstract hierarchy are proposed for fusing and integrating data and storing and re-using generic information extraction techniques within the system.

## 1 Introduction

Sensor Web is a revolutionary concept towards achieving a collaborative, coherent, consistent, and consolidated sensor data collection, fusion and distribution system. Sensor Webs can perform as an extensive monitoring and sensing system that provides timely, comprehensive, continuous and multi-mode observations. This new

earth-observation system opens up a new avenue to fast assimilation of data from various sensors (both in situ and remote) and to accurate analysis and informed decision makings [14].

Most current sensing systems are designed either for domain specific applications or data collection purposes [15]. While this is significant for military, robotics and other commercial applications that operate in a more closed, tightly controlled environment, it does not help common end users to be able to access and use the vast, ever increasing amount of publicly available sensor data that is currently available. Our vision of the Sensor Web is an infrastructure that allows end users to automatically access and extract and use appropriate information from multiple sensor sources over the Internet (open environment).

There are three key technical challenges to realizing this vision. Firstly to create a publicly accessible open distributed computing infrastructure where heterogeneous sensor resources and complex end-user applications can be deployed, and automatically discovered and accessed. The second is the problem of fusing data from different sensors that have different temporal and spatial resolutions and different data models and formats. By fusing data from different sensors a higher spatial coverage and higher temporal resolution is achieved. The last challenge is to perform context-based information extraction. The technical skill and time required to extract appropriate information from sensor data provides a barrier to a potentially large end-user community who could benefit from this data. Users do not want to deal with the complexity and scale of sensor data, but would prefer a view of the data that only exposes information that can aid them in their application. Depending on their needs (context), user's would be interested in different aspects of the sensor data. Thus the same data could be used for a variety of applications.

In this paper, we introduce, in section 2, the Open Geospatial Consortiums (OGC) Sensor Web Enablement (SWE) initiative, which is an open, standards based, web services approach that provides a starting point for a sustainable development of a global Sensor Web. We analyze the drawbacks of this approach and then in section 3 describe some other approaches. In section 4, we describe an ontological infrastructure for the Sensor Web and present the SWAP framework, a conceptual agent architecture which we believe is the next evolutionary cycle of the Sensor Web.

## 2 OGC Sensor Web Enablement

A key challenge in realizing the Sensor Web is how to automatically access and integrate different types of spatio-temporal data that is observed by sensors or generated using simulation models. The OGC takes an open standards approach based on a members consensus process and provides a framework that specifies standard interfaces to access geographical data and a standard way of encoding and exchanging this data over the Internet. OGC Web Services follow W3C's service oriented, web services framework and support publishing, automatically discovering and automatically accessing geographical resources over the web; leading to Spatial Data Infrastructures (SDI).

## 2.1 SWE Framework

The Sensor Web Enablement (SWE) initiative initiated by the Open Geospatial Consortium (OGC) extends prominent OGC Web services by providing additional services for integrating Web-connected sensors and sensor systems. The SWE architecture was designed to enable the creation of web-accessible sensor assets through common interfaces and encodings [4].

SWE currently defines four Web service specifications and two models and encodings for observations and sensors respectively. The Web services, known as Sensor Observation service for data access, Sensor Planning Service for sensor tasking and feasibility studies, Sensor Alert Service for registering atomic conditions and push based notification, and Web Notification Service as a data transport protocol transformer describe the operational part of the framework. The data models and encodings Observation & Measurement (O&M) and Sensor Model Language (SensorML) are used as data and metadata exchange protocols. SWE attempts to address mostly the first two challenges mentioned above. It provides an infrastructure that follows the publish-find-bind paradigm known from the service oriented architecture [13]. It also allows fusing multiple data models and formats by using a common data model and representation. However, it provides only rudimentary support for the necessary data conversion.

## 2.2 Drawbacks of the SWE Approach

The SWE approach provides an open distributed computing infrastructure where sensor resources can be published, automatically discovered and accessed and thus largely addresses the first challenge. However there are some major gaps when it comes to dynamic data fusion and context based information extraction.

First, there is no explicit ontological structure in the SWE framework. Although all services make use of a common encoding and transport protocol, semantic interoperability is still an issue due to the lack of an explicit common formal conceptual model encoded in the system (there is one in the documentation). The services and encodings focus on providing standard encoding schemas, but are not grounded in any formal ontology. This impacts on dynamically fusing data with different granularities of time, space and measured phenomena, e.g. Suppose that a sensor (mounted on a satellite) produces measurements of blackbody temperature every 15 minutes with a spatial resolution of 3 square kilometers and another in-situ sensor produces measurements of land surface temperature every minute at a resolution of 0.5 square kilometers. As the data has no explicit semantics there is no means to discover that both measurements are related. There is no way of automatically fusing this data unless there was software explicitly configured with the relations and necessary conversions to cater for this. The spatial and temporal conversions are possible, but there are no explicit temporal (e.g. before, after, during) and spatial operators (e.g. within, outside, adjacent) that are often required for dynamic data fusion. This problem is especially apparent when describing the sensors itself. While SWE provides service capability descriptions, allowing the service consumer to discover and access the service, the returned results lack any semantic

markup and cannot be dynamically integrated by the service consumer. The model used by SensorML provides ways of describing sensors but does not provide a semantic description of the sensor nor the phenomenon that it measures. As a result, it is difficult to automatically discover related data and integrate data from sensors that are described in SensorML.

Apart from the lack of semantics the SWE framework mostly addresses data acquisition but neglects filtering and information overload. Data is currently pulled from passive services, rather than being pushed from active services in terms of a publish-subscribe paradigm. Massive observation data from myriads of sensors are exposed to the user by higher-level service providers via predefined interfaces that provide specific views of the data. When users require alternative views, not provided by existing service providers they are forced to interact directly with sensors and determine and implement the necessary processing steps to provide these views. Thus even though services can be dynamically discovered and accessed, their offerings to service consumers are static and not dynamically integratable.

Following on from this, another drawback of the SWE framework is the lack of support for deploying, discovering and accessing Sensor Web applications. Service providers hide complex application logic behind OGC services. Users may be aware of individual instances of OGC services, but it will not be clear on exactly what applications each service, or combination of services supports. As end-user applications are not specifically addressed in the SWE framework, they will be developed on a custom adhoc basis and will often have to be manually upgraded when new services appear. This will eventually result in a duplication of efforts in terms of building new services, when parts of existing service could be re-used as well as duplication in developing end-user applications for the Sensor Web.

Despite these drawbacks, the Open Geospatial SWE framework serves as a starting point for the SWAP framework. It provides an open, standards based architecture that allows sensor and data resources to be published and accessed in a standard way simply by implementing public interface and encoding specifications. This can be done without any invocation on the Sensor Web itself. It allows arbitrary sensor data providers and sensor facilitators to plug-into the Sensor Web.

### 3 Other Approaches

In this section we provide an overview of other Sensor Web approaches. These approaches include four agent based architectures, IrisNet[10] Abacus[2] AIGA[19] and the Biswas and Phoha architecture [3] as well as a GeoSwift[14], a non-agent web services based architecture.

#### 3.1 Non Agent Based Systems

##### *GeoSwift*

In addition to providing a definition for the Sensor Web Liang et. al [14] proposes GeoSWIFT Sensing Services as a distributed geospatial information infrastructure for the Sensor Web. The framework consists of a three layered architecture comprising of

a sensor layer, i.e. the actual sensors, a communication layer, i.e. the physical network or data communications link between the various components and the information layer, that provides interoperability and integration of data from different sensors. The implementation prototype uses a web services approach for service registration and discovery. The architecture advocates use of OGC standards for integrating and exposing sensor data. An extension to the traditional web services approach is the introduction of a sensing server. The sensing server essentially provides the information layer, and is able to integrate and store data in different formats from different sensors and thus abstract sensor specific data formats and communication protocols from the user. New sensors can be integrated into the system by extending the sensing server or deploying a new server.

### 3.2 Agent Based Systems

#### *IrisNet*

The Internet-scale Resource-Intensive Sensor Network Services [10] is a distributed software architecture, which provides high level sensor services to users. It comprises of a two-tier architecture i.e. sensor agents (SA), where each sensor agent provides, pre-processes and reduces raw data from a physical sensor, and organising agents (OA) where each OA provides a sensor service. An OA must collect and analyse data from sensor agents to answer the particular class of queries relating to the service e.g. queries about free parking spots to the parking space finder service. An important characteristic of this architecture is that sensor agents are dynamically programmable. A new service, i.e. an OA can send pre-processing steps, called a senslet, to a sensor agent. The sensor agent can then execute these steps on its raw data stream and return the results. The architecture addresses issues related to designing large scale distributed systems, such as distributed processing, distributed storage and security. Sensor data is represented using XML and queried using the XPATH query language. A key concept that is introduced in this system is that of data reduction by extracting higher level features from raw data from different sensors.

#### *Abacus*

Abacus [2] is a multi-agent system for managing radar data and providing decision support. The abacus architecture is a three layered architecture. The contribution layer contains agents that wrap physical sensors, and provide some pre-processing such as error correction and noise removal. The management and processing layer contains a community of agents with similar processing functionality. Agents are responsible for processing data for a given spatial location or spatial sector. Thus the processing is distributed in a grid type manner. Different constraints or rules can be specified for different sectors and local alarms are triggered according to these. A coordinating agent or abacus agent assembles results from the processing agents to provide a joint data view, and processes local alarms from the processing layer to trigger global alarms. The distribution layer, provides user interfaces for data visualization, and disseminates warnings via email or the web. User interfaces are also available for entering decision rules to generate alarms for the processing and the abacus agents.

*AIGA*

The Agent Based Imagery and GeoSpatial Processing architecture (AIGA) [19] is a multi-agent infrastructure that supports image processing, geospatial data processing as well as text mining. The architecture focuses on performance and scalability issues related to using mobile agents for image processing, geospatial processing and text mining applications, especially the agent mobility and scalable discovery aspects. The architecture differentiates between data agents and processing agents and coordination agent that provide directory services. Agents communicate with each other through a shared communication space. The system provides a user interface for end-users to compose workflows of agents to provide different application functionality. These workflows can be saved in the system and re-used at a later time.

*Biswas and Phoha approach*

The agent based approach by Biswas and Phoha [3] also proposes a three layered logical architecture for developing sensor applications. The data layer consists of sensors that act as the data source and an application layer that fuses and interprets the information. A service layer acts as an integration layer between the physical data layer and the software application layer. Agents are used in the application and service layer and the CCL language is used to command and query sensors. Agents in the service layer integrate data from one or more sensors from the data layer and provide data to application agents in the application layer.

### 3.3 Analysis of Sensor Web architectures

Most of the approaches described above identify the need for information filtering and propose layered architectures to achieve this: in Abacus different agents in the processing layer are responsible for different spatial sectors and detect and report alert conditions to a higher layer for distribution to users; IrisNet uses organizing agents to collect and analyse data from sensor agents to answer specific classes of queries; the Biswas and Phoha approach uses agents in the service layer to integrate data from sensors in the data layer to provide data to application agents in the application layer; and GeoSWIFT has an information layer that provides interoperability and integration of data from different sensors. These approaches all address the first challenge by providing a distributed infrastructure for publishing, discovering and accessing sensor resources and to some extent the challenge of data fusion and aim to expose end users to just the information they need. These approaches are promising for a single or a group of organizations building distributed applications, however it is questionable whether these approaches will scale on the Internet where hundreds of different organizations will be developing and deploying different applications and many more users will need to customize and integrate these applications. As with the SWE approach, the lack of an ontological infrastructure limits the potential for dynamic data fusion and the potential for dynamic re-use of information from higher level services to provide new services. The lack of semantics also restricts the manner in which users can describe what they require from the system and how the system can discover, integrate and extract relevant information that match these requirements.

## 4 The SWAP Framework

The Sensor Web is a large-scale open environment that differs from traditional static closed distributed environments in terms of its users, service offerings and scale and diversity. Multi-agent systems (MAS) are effective in large-scale and open distributed environments [22, 25]. The SWAP framework tries to combine the achievements of the OGC SWE initiative with the flexibility provided by multi-agent based systems (MAS). Below we propose the SWAP abstract framework for building Sensor Web applications over the Internet. The framework is built on a multi-agent infrastructure, which provides underlying distributed computing services such as inter-agent communication and discovery services, and an ontological infrastructure that provides the semantics required for the discovery, reuse and integration of Sensor Web data and processing capacities.

### 4.1 Multi-Agent System Infrastructure

An agent can be defined as a computer program capable of flexible and autonomous action in a dynamic environment, usually in an environment containing other agents. The agents and the infrastructure that supports their interactions are together called a multi-agent system (MAS) [16]. A MAS infrastructure [9,24] for the Sensor Web must be a large-scale [25] and open agent system where tens of thousands of agents must find and interact with each other to utilize various applications. The numbers and availability of agents, as well as the communication languages, ontologies and types of applications may vary over time.

Agents systems require more effort to design and build over traditional systems and there are over 100 MAS toolkits and platforms available (see: <http://www.agentlink.org>) to support building MASes. However, the Foundation for Intelligent and Physical Agents (FIPA) [5] is attempting to create a standard architecture and a standard Agent Communication Language (ACL) and suite of protocols for agent communication, FIPA ACL [7], to facilitate interoperation between different MAS platforms.

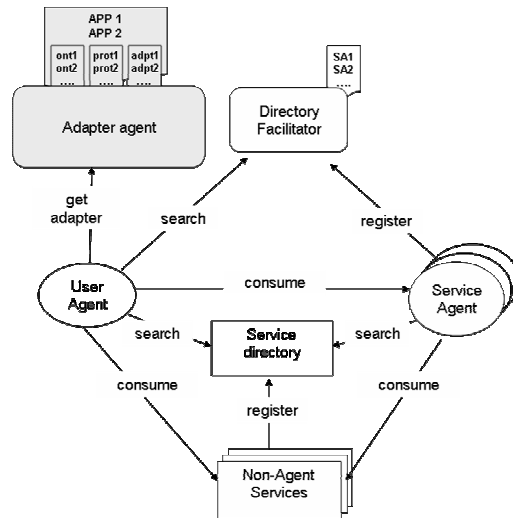
Agent platforms have largely been developed separately from web services but with the ever increasing adoption of web services within the Internet community, it has become important to integrate both approaches. The FIPA agent architecture [6] caters for services and a service directory, but without explicitly defining how these services are to be implemented. Recent efforts, such as the Web Services Agent Gateway (WSAG) [26, 1], have been made to integrate web services within agent platforms.

To support the SWAP architecture, the MAS must contain non-agent based services as well as agent services that together provide end-user applications. In terms of a service oriented architecture, agents can be differentiated into user agents and service agents with support from directory facilitators. The FIPA architecture does not explicitly provide support for application deployment. We are currently working on a MAS infrastructure, MASII [17] that explicitly provides support for end-user application development and deployment. An initial prototype of the system with

some simple examples can be found on the MASII web site<sup>1</sup>. The components of the MASII architecture are illustrated in figure 1 and are described briefly below.

#### *User agent*

This represents an end-user who will consume non-agent services, agent services and applications on the Sensor Web.



**Fig. 1.** Multi-agent system framework (Components added to the FIPA model in grey).

#### *Service agent*

The service agent is a service provider, i.e. provides services for the user agent to use (consume). The service agent differs from a non-agent service in terms of its autonomy, and ability to communicate and reason using the ACL.

#### *Directory facilitator*

Service agents register their capabilities with the directory facilitator so that user agents may discover and use their services.

#### *Adapter agent*

The adapter agent (AA) maintains the ontology infrastructure similar to the FIPA ontology agent as will be described later in section 4.2. Furthermore the AA maintains an application catalogue that contains descriptions of current applications in the MAS. User agents periodically download the application catalogue to be aware of changes in the application offerings in the MAS. The AA forms the central hub of an

<sup>1</sup> <http://www.cs.ukzn.ac.za/deshen/masii>

application. An application may have its own application specific ACL, ontology, reasoning engine and user GUIs. The AA is responsible for packaging and maintaining these in the form of user adapters. It serves these user adapters so that user agents can download these adapters, install them locally and present new applications to end-users.

A possible deployment of the different agents and facilitators for a sample application (*appl*) would be: The *appl* service agent registers with the directory facilitator. The user agent downloads and installs the *appl* application adapter from the adapter agent. The user agent searches the directory facilitator for the details of the *appl* service agent and accesses the *appl* service on the *appl* service agent. All the communication between agents is based on the ontologies on the adapter agent.

#### *Non agent services*

A non-agent service is a resource that is available in the MAS and is accessible by at least one agent. It is important to re-use the ongoing work done by the OGC. By integrating SWE services as non-agent services, we allow standardized components to be integrated in the new framework. This means that existing SWE applications may remain in their common shape but might be used in a more flexible way by agents.

For example, imagine the Web Notification Service. This Web service provides data redirection, i.e. takes in data together with a target destination, target format and transport, transforms the data into this format and redirects the data to the target destination using the specified transport. For example, sending an alert message in the form of an SMS to a mobile phone. These services are not able to understand ACL and do not have any reasoning mechanism.

## 4.2 Ontological Infrastructure

The ontological infrastructure provides the semantics required for the discovery, reuse and integration of Sensor Web data and processing capacities. The Sensor Web will contain agents that represent sensors and serve sensor data, and agents that represent processes that provide simulation models or data processing algorithms. Ontologies [11] will be used to provide explicit descriptions of all services offered by agents in the Sensor Web. This added semantics will now allow agents to reason about sensor data retrieved from sensor agents. This includes reasoning about the phenomenon that is being measured by the data as well as the spatial and temporal aspects of the data. Thus agents will be able to fuse data that measures similar phenomena from different sensor agents, but with different temporal and spatial resolutions, and to be able to extract higher level features from this data by discovering and applying the appropriate algorithms offered by processing agents. In this way an agent has the ability to compose chains of sensor agents, and processing agents and to fuse and extract higher level features from sensor data. These sensor agents and processing agents can be re-used to provide new processing chains which themselves could be used in other processing chains.

Ontologies must provide support for agents at a conceptual level as well as at a technical level. At the conceptual level, each agent must describe its service offerings, i.e. the data or services that it provides and the spatio-temporal characteristics of this

data, so that it can be viewed in terms of the larger system without the implementation detail. The conceptual description will promote conceptual interoperability between heterogeneous sensor resources and dynamic extraction and integration of higher level features from sensor data. By providing good conceptual descriptions, scenarios for re-use can be inferred. However agents still need to communicate and exchange data. This is supported at the technical level, where specific data structures are specified. The ontology must have support for data structures ranging from a single value at a specific time and space as well as multi-dimensional data e.g. arrays/grids over a specific spatial area taken over a specific time interval or multiple time instants. In this way heterogeneous agents can understand and use data from other agents in its internal processing and know at which stage of its internal processing to

Ontologies unlike other formal modeling languages must be easily understood and managed by end-users as well as be automatically interpreted and used by software agents. This added semantics allows human users with limited technical expertise to easily explore and find relevant sensor data, services and applications that can aid them in their current task. Furthermore, ontologies are somewhat unique in that they provide a run-time model of the system, i.e. if the ontology is changed, the behaviour of the system immediately reflects this change. This dynamism is especially important for the highly dynamic environment of the Sensor Web where individual agents change their offerings and where new agents with new capabilities are added.

#### 4.2.1 Ontology Development

There are many challenges to building, maintaining and using ontologies. An ontology infrastructure must provide support for users with basic computing knowledge to navigate and understand existing ontologies and to extend these to produce new ontologies in the system. Furthermore, these ontologies must be automatically discoverable and readable by all agents in the system.

An ontology infrastructure for the Sensor Web must support the construction, maintenance and accessibility of thousands of ontologies across multiple end-user applications and in many different domains. Ontology development is an ongoing process in the Sensor Web. Guarino's [11] classification of ontologies assists in differentiating ontologies in terms of their extensibility and reusability for specific applications or tasks. The classification differentiates between three levels of ontologies, i.e. top-level or upper ontologies, domain and generic task ontologies, and application ontologies. Developers of Sensor Web applications will typically use domain and domain task ontologies to specify new application ontologies.

An ontology representation language and ontology development tools are required to construct ontologies. Our current efforts are based on using OWL<sup>2</sup> as the web ontology language and the NASA Sweet ontologies<sup>3</sup> as the upper ontology to ground all other ontologies in the system. An alternative ontology, the OntoSensor ontology<sup>4</sup> [21] provides an OWL representation of Sensor ML and the ISO 19115 standard. While SensorML and ISO-19115 describe more generic concepts about sensors and image data, the Sweet ontologies provide more detailed concepts that are useful for

---

<sup>2</sup> <http://www.w3c.org>

<sup>3</sup> <http://sweet.jpl.nasa.gov/ontology>

<sup>4</sup> <http://loki.cae.drexel.edu/~wbs/ontology/iso-19115.htm>

building specialized domain and application ontologies. Domain and domain task ontologies specify terms, relationships between terms and generic tasks and activities that are relevant in a particular domain. Domain concepts are specializations of concepts from the upper ontology. Sensor data must be described in terms of the phenomenon being measured, the time and space over which it is measured and in terms of the larger data set to which this data belongs. The Sweet ontologies provide support for representing time and space and OwlTime [12] is being investigated to strengthen temporal representation and reasoning.

Representation of tasks is a crucial component in the system as they describe actions and processes in the system. Workflow agents specify task chains of tool and modeling agents and automatic or machine assisted workflow composition is an eventual goal of the system. OWL-S<sup>5</sup> is being used for representing processes in the system. OWL-S is an extension of OWL, which provides support for modeling processes and can be integrated with the web services framework. Furthermore there are OWL-S tools available such as the OWL-S Editor<sup>6</sup>.

Together these ontologies provide a framework and a starting point to build new geographical domain and application ontologies. The application ontologies take an application specific view of the world and re-use and extend terms from one or more domain and domain task ontologies to realize the knowledge for a specific application. As each application ontology takes an application specific view of the world, an application ontology generally can not be reused for other applications. Domain ontologies however are application independent and can be re-used for a variety of applications. Guarino's classification promotes re-use of existing domain ontologies and provides a starting point for building new application ontologies.

### 4.3 SWAP Abstract Architecture

The SWAP architecture leverages the MAS infrastructure services and the ontological infrastructure described above to deliver sensor web applications to end-users. The architecture aims to address the three key technical challenges of automatic discovery and utilization of sensor resources, sensor and simulation data fusion and context based information extraction. The Sensor Web Abstract Architecture is based on a three-tier architecture, separating Sensor Layer, Knowledge Layer and Application Layer (figure 2).

The Sensor Layer deals with components working directly on physical or virtual sensors in a tight or loose coupling. The knowledge layer actually realizes the orchestration of complex processing chains based on expert knowledge that was formulated in an ontology beforehand. It contains re-usable simulation and processing components. These components can be dynamically assembled by workflow agents to provide a variety of end-user applications. The application layer contains the user interface that allows human users or other client machines to interact with the system. Further on, it contains resources for data redirection using multiple formats and transport protocols.

---

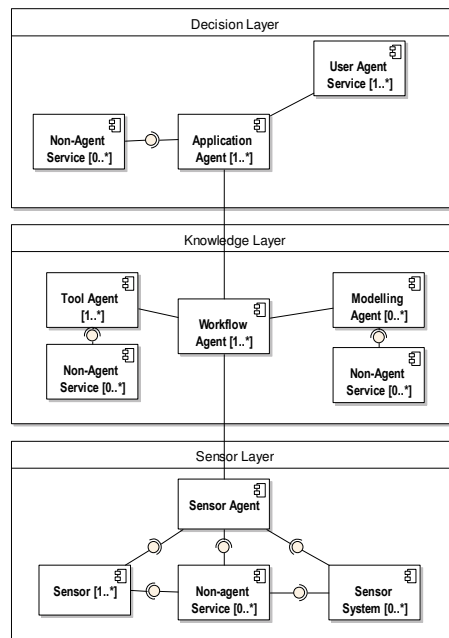
<sup>5</sup> <http://www.daml.org/services/owl-s>

<sup>6</sup> <http://owlseditor.semwebcentral.org>

### 4.3.1 Sensor Layer

Sensor Agents will access physical or virtual sensors directly or make use of intermediary services like those specified by the Open Geospatial Consortium, e.g. Sensor Observation Service, Sensor Planning Service, Sensor Alert Service, Web Coverage Service, etc.

Sensor Agents represent either data access services or are used to control sensors or actuators or combine both functionalities. Messages sent and received by Sensor Agents follow a specific structure that is described in ontologies. It is important to note that the conceptual description must be independent of the technical-implementation motivated realization. By formally specifying the structure and content of messages, agents are able to automatically generate and interpret messages. Furthermore, the message specification provides an automated mapping from conceptual intent to the actual interactions required with other agents to fulfill this intent.



**Fig. 2.** The SWAP framework is based on a three-tier architecture

A message consists of a header that contains information about the message type (e.g. QUERY or INFORM\_RESULT similar to the FIPA message design [7]), the language used to encode the message content payload and the relevant ontology used to structure and semantically ground the message content; the message content section that separates metadata that describes the observational data, from the observational data itself. Basically it represents a container that provides information about the kind of phenomenon measured, the unit of measurement and where and when the

phenomenon was observed. It is an open research question to which extent this separation can be realized properly, or in other words: Is it possible to make a clear distinction between metadata and data at this point. The data part of the message content section has to provide sufficient information to process the content automatically, which means that data types and encodings have to be designed properly.

#### **4.3.2 Knowledge Layer**

The core components of the knowledge layer are Workflow Agents, Tool Agents and Modeling agents. Workflow agents capture and store expert knowledge in the system. Based on a workflow description language, such as OWL-S which is briefly described in section 4.2.1, the workflow agent combines a set of tool agents and modeling agents to solve a specific problem. All agents in the knowledge layer are sensor independent. Based on a common understanding of exchange formats and general concepts specified in multiple ontologies, the workflow agent receives (raw) data from the sensor agents and invokes tool and modeling agents to run predefined processing steps and simulations on this data. The processed data will be passed back to the workflow agent in order to be passed on to the next service or forwarded to the application agent in case the workflow has reached its final step.

Tool agents usually represent well defined deterministic processing steps that will always produce a meaningful result if valid data is inputted, whereas modeling agents may be non-deterministic and may never complete. Whereas tool agents run predefined processes that usually do not require further data other than the one provided or referenced by the workflow agent, modeling agents provide complex processing capacities that might take some time to get accomplished or even fail in producing a proper outcome. Modeling agents may request further data other than the one provided by the workflow agent and usually receive additional processing instructions on request. Further on, modeling agents will be able to provide scenario based information (what if-scenarios) and provide multiple, concurrent outcome on a single request.

#### **4.3.3 Application Layer**

The application layer exposes Sensor Web applications to end-users. It has to provide user interfaces that allow humans as well as machines to interact with the system. It consists of two types of agents, called User Agents and Application Agents. The application agent aims at integrating the output of one or more workflow agents to realise specific applications. The goal of the user agent is to provide a custom view of selected Sensor Web applications to each user. Every Sensor Web user will typically have their own user agent. Users can select the combination of applications that serve their purpose and will be able to integrate and create a custom view of these applications via their user agent. The user agent must also alert the user to new applications and capabilities in the system and must be easily re-configured to reflect the ever changing requirements of the user.

#### 4.3.4 Non-Agent Resources

It has to be noted that not all functionalities need be provided by agents. Non-agent resources such as Web services can be incorporated into the system. Non-agent technologies will be applied whenever existing technologies are available that can serve the required purpose. This applies for example for data redirecting components that forward incoming messages to the final recipient by making use of a different protocol than the incoming protocol. The outgoing message might even be encoding in another ontology defined format.

#### 4.3.5 Ontologies to Support Agents

The Sensor Web must cater for a variety of ontologies developed by different user communities. An ontology repository is required to allow agents to automatically discover and use available ontologies at run-time. Users must also be able to discover and modify appropriate ontologies and also to extend these to construct new ones. FIPA proposes a specialised ontology agent for discovering, maintaining and translating between ontologies [8] in a MAS. The FIPA ontology agent uses the Open Knowledge Base Connectivity (OKBC) as the ontology representation language. There are two partial implementations of the ontology agent [20, 23]. Obitko and Snasel have constructed an ontology agent using OWL and RDF instead of OKBC as the ontology representation language.

#### 4.3.6 A prototype application for wildfire detection

The Advanced Fire Information System (AFIS) is the first near real-time satellite-based fire monitoring system in Africa. A prototype implementation of SWAP is currently being developed for AFIS [18] as shown in figure 3 below. A sensor agent will access the blackbody temperature from the SERVERI sensor and expose this to a workflow agent configured to detect hotspots. The workflow agent is responsible for tasking a modeling agent to predict the current background temperature, sending both the predicted background temperature and observed temperature to a tool agent that checks for hotspots, and communicates detected hotspots to the AFIS application agent. The modeling agent will retrieve historical blackbody temperature needed for building a time series prediction model from the MSG sensor agent. The AFIS application agent uses hotspots retrieved from the workflow agent to issue fire alerts using a WNS. The AFIS application server can be accessed either via an AFIS client installed on a user's computer, or via a web interface for configuring fire alerts. Extensions to the system include integrating blackbody temperature from the modis sensor agent to increase temporal and spatial resolution and incorporating a fire spread modeling agent that uses wind and air temperature obtained from a weather agent to predict fire spread.

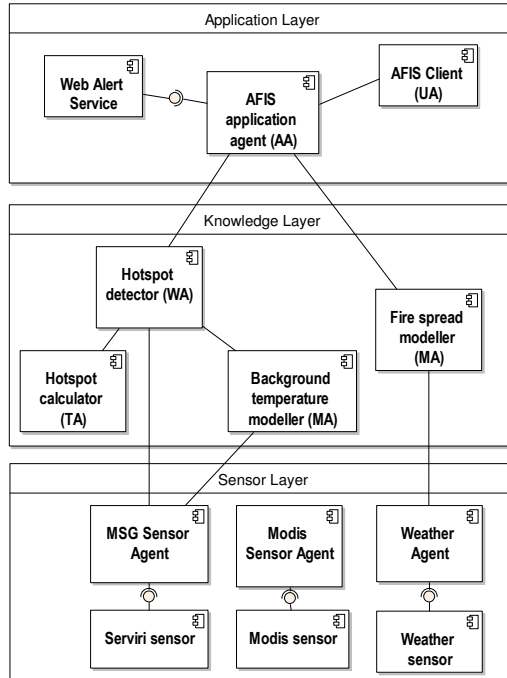


Fig. 3. The design of a prototype application for wildfire detection using SWAP

## 5 Discussion and Conclusions

This paper describes a new high level architecture for the Sensor Web to address developing and deploying sensor web applications over the Internet. Focusing on open standards based interoperable platforms; we used the OGC Sensor Web Enablement framework as a starting point and evolved it to a multi-agent based system, called the SWAP framework. The new framework will provide far more flexibility and pro-activeness than the concurrent system that primarily concentrates on pure data provision and data processing capacities without integrating them into workflows or even applications. The architecture proposes a MAS environment where heterogeneous sensor resources and complex end-user applications can be deployed, and automatically discovered and accessed. Sensor agents expose their data offerings using a common representation of space, time and phenomena. In this way workflow agents in the knowledge layer are able to fuse data from different sensors that have different temporal and spatial resolutions and different data models and formats. The knowledge and application layer fuse and interpret sensor data and presents this to the user. Expert knowledge for sensor data understanding and knowledge extraction is encoded and stored on workflow, tool and modeling agents in the knowledge layer.

Application agents interpret and present high-level features to the user in the form of end-user applications. New applications can be easily composed and deployed by re-using components in the knowledge layer.

In section four we put emphasis on the ontology framework that becomes crucial to the new model we proposed, as the performance of agent based systems in terms of interoperability, scalability, and usability very much depends on agent communication.

Though many open issues remain, first prototypical tests that implement a multi-agent based system based on the MASII platform show promising results. Therein, we use the open source implementations of OGC SWE services products provided by 52°North (<http://www.52north.org>) as non-agent based resources and integrate them into the sensor web in the context of a wildfire detection and spread prediction system. The results of this ongoing work will be presented at a later stage.

### Acknowledgements

This research is sponsored by the Meraka Institute (Council for Scientific and Industrial Research, South Africa) and by the National Research Foundation, South Africa. The authors would also like to thank Mr. Andrew Terhorst for his support in writing this paper and the Sensor Web Alliance (<http://www.sensorweb-alliance.org>) for providing an inspiring platform.

### References

1. Agentcities Web services Working Group. Integrating Web services into Agentcities Technical Recommendation, <http://www.agentcities.org/rec/00006/>.
2. Athanasiadis, I. N. A methodology for developing agent-based systems in environmental informatics applications, Doctoral Dissertation, Electrical and Computer Engineering Dept, Aristotle University of Thessaloniki, 2005.
3. Biswas, P.K., Phoha S., A middleware-driven architecture for information dissemination in distributed sensor networks. In Proceedings of the Intelligent Sensors, Sensor Networks and Information Processing Conference 14-17 Dec. 2004, IEEE, 2004, pp. 605- 610.
4. Botts, M. et al.:OpenGIS® Sensor Web Enablement Architecture Document. Open Geospatial Consortium, OGC 06-021, 2006
5. FIPA, Foundation for Intelligent Physical Agents (FIPA), Foundation for Intelligent and Physical Agents, <http://www.fipa.org>, 2004.
6. FIPA01, FIPA Abstract Architecture Specification. FIPA00001, Foundation for Intelligent Physical Agents, <http://www.fipa.org/specs/fipa00001/index.html>, 2002.
7. FIPA37, FIPA Communicative Act Library Specification. FIPA00037, Foundation for Intelligent Physical Agents, <http://www.fipa.org/specs/fipa00037>, 2002.
8. FIPA86, FIPA Ontology Agent. FIPA00086, Foundation for Intelligent Physical Agents, <http://www.fipa.org/specs/fipa00086>, 2000.
9. Gasser L, MAS Infrastructure, Definitions, Needs, and Prospects, in Thomas Wagner and Omer Rana, editors, Infrastructure for Agents, Multi-Agent Systems, and Scalable Multi-Agent Systems, Springer-Verlag, 2001.

10. Gibbons, Phillip B., Brad Karp, Yan Ke, Suman Nath, Srinivasan Seshan, IrisNet: An Architecture for a Worldwide Sensor Web, IEEE Pervasive Computing, vol. 02, no. 4, pp. 22-33, Oct-Dec, 2003.
11. Guarino, N. Formal Ontology and Information Systems. In N. Guarino, editor, Proceedings of the 1st International Conference on Formal Ontologies in Information Systems, FOIS'98, Trento, Italy, pages 3-- 15. IOS Press, June 1998.
12. Hobbs, J.R., Pan, F. 2004. An Ontology of Time for the Semantic Web. *ACM Transactions on Asian Language Processing (TALIP): Special issue on Temporal Information Processing*, Vol. 3, No. 1, March 2004, pp. 66-85.
13. IBM: Web Services architecture overview. The next stage of evolution for e-business. developerWorks, IBM, 2000
14. Liang, S. H. L., Croitoru, A., Tao, C. V., A distributed geospatial infrastructure for Sensor Web. *Computers & Geosciences* 31(2): 221-231, 2005.
15. Liu J, Zhao F, Towards Semantic Services for Sensor-Rich Information Systems, The 2nd IEEE/CreateNet International Workshop on Broadband Advanced Sensor Networks (Basenets 2005), Boston, MA, Oct. 3, 2005.
16. Luck, M., McBurney, P., Shehory, O., Willmott, S., Agent technology roadmap: A roadmap for agent based computing [Online] Available WWW: <http://www.agentlink.org/roadmap/index.html>, 2005. (Accessed 28 May 2006)
17. Moodley D., Kinyua J.D.M. Towards a multi-agent infrastructure for distributed Internet applications, 8th Annual Conference on WWW Applications, Bloemfontein, South Africa, 5-6 September, [Online] Available WWW: <http://www.zaw3.co.za>, 2006.
18. Moodley D., Terhorst A., Simonis I., McFerren G., van den Bergh F., Using the sensor web to detect and monitor the spread of wild fires Second International Symposium on Geo-information for Disaster Management (Gi4DM) September 25-26, Pre-Conference Symposium to ISPRS TC-IV and ISRS Symposium on "Geospatial Databases for Sustainable Development" September 27-30, at Goa, India, 2006
19. Nolan, J., An Agent-based Architecture for Distributed Imagery and Geospatial Computing. PhD dissertation, George Mason University, 2003.
20. Obitko, M. and V. Snasel: Ontology Repository in Multi-Agent System. Proceedings of IASTED, Austria, 2004
21. Russomanno, D. J., C. Kothari, et al., Building a Sensor Ontology: A Practical Approach Leveraging ISO and OGC Models. The 2005 International Conference on Artificial Intelligence, Las Vegas, Nevada, 2005.
22. Singh MP, Huhns MN. Service Oriented Computing. John Wiley & Sons, Chichester, England, 2005. pp. 1-17, 341-358
23. Suguri, H., E. Kodama, M. Miyazaki, H. Nunokawa, S. Noguchi, Implementation of FIPA Ontology Service, Workshop on Ontologies in Agent Systems, 5th International Conference on Autonomous Agents, 2001
24. Sycara K. "Multi-agent Infrastructure, Agent Discovery, Middle Agents for Web Services and Interoperation", in Luck M. et al. (eds), Multi-Agent Systems and Applications, LNICS vol. 2086, Springer-Verlag, Berlin, 2001, pp. 17-49.
25. Wijngaards N. J. E., Overeinder B. J., Van Steen M. and Brazier F. M. T. "Supporting Internet-Scale Multi-Agent Systems, Elsevier Science, Dec 2001.
26. Whitestein Information Technology Group AG. Web services Agent Integration Project, <http://wsai.sourceforge.net/index.html>.