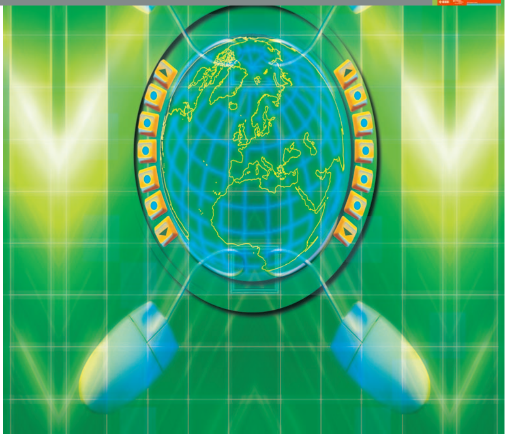
In IT Professional, **8** (**3**):34-39, May-June 2006.



By interweaving and interpreting data streams, AISLE can bring crucial environmental information to a wider audience.

Ioannis N. Athanasiadis

An Intelligent Service Layer Upgrades Environmental Information Management

Service-oriented architectures for software design and implementation are gaining the interest of software engineers and practitioners as a natural way to build distributed, reusable, loosely-coupled systems. Using Web services or software agent implementations, the service-oriented approach can advance software delivery: it allows the design and development of modular, stand-alone software components that can provide services in isolation or collaboratively, by conforming to open interfaces and to strict contracts. This versatility has great potential to improve many types of systems; my colleagues and I have used a service-oriented architecture to address the shortcomings of the typical environmental management information system (EMIS).

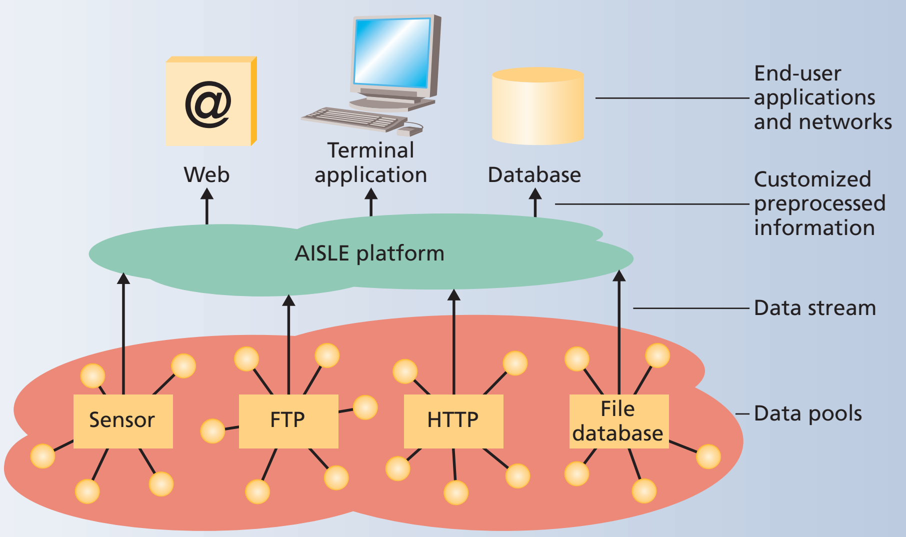
In principle, environmental data belong to the public domain and should be available to everyone interested. However, the availability and dissemination of environmental data differ among countries, and EMISs suffer from what is known as the "environmental information vacuum": In developing countries, the vacuum results from the fact that environmental data are not recorded extensively. Developed countries, on the other hand, have an overflow of environmental information, but quality and availability problems create a vacuum there as well. Whichever the case, environmental data are generally raw and variable in standards, noisy or incomplete, and often hidden in legacy systems, reports, or other nonreusable forms. In such cases, an EMIS must intervene between several data pools—typically in different physical locations and diverse implementations—to fetch relevant environmental information. We developed a service-oriented architecture to meet this challenge: AISLE, our adaptive intelligent service layer for environmental information management, mediates between existing environmental data providers and actual end-user applications that require preprocessed data streams from the sources.

THE AISLE APPROACH

AISLE has two major objectives. The first is to extend the capabilities of existing legacy IT systems residing in environmental monitoring centers and institutions, by targeting typical problems that fetter their quality of service. Common incidents in environmental monitoring systems—such as sensor failure or malfunction, noise, and polarization—often lead to poor data quality and, consequently, poor services. AISLE's goal is to improve service reliability and adaptability to changing environmental conditions by equipping EMISs with intelligent modules for quality assurance and control.

AISLE's second objective is to provide a loosely coupled, extensible infrastructure for supplying improved information services—including data preprocessing and management, information dissemination, and multiparty data distribution.

Figure 1. The AISLE operational environment.



Service-oriented design

Figure 1 shows an abstract view of an AISLE system, situated between environmental data pools and end-user applications. AISLE interweaves multiple data streams produced by (or located within) the environmental data pools and supplies the end-user applications with preprocessed, ready-to-use information. The system's data sources could be sensor devices, remotely accessed documents (via FTP or HTTP), or locally stored databases and files. The end-user applications could include database systems, domain-specific software, or Web-based reporting applications.

Although both the topology and the procedure vary for each installation, we use the generic term *data streams* to describe the data perceived by sensor devices or located in a local or remote data storage medium. These data streams—AISLE's inputs—inherit the critical properties of environmental data, such as low reliability, redundancy, and poor semantics. For most end-user applications, such data streams are usually incomplete, incompatible, or both, and thus cannot be used directly. Typically, some data preprocessing or filtering is necessary.

In such an environment, AISLE must tackle the uncertainty of environmental data while also serving as a proactive communication mechanism connecting data pools with user applications. In fulfilling these roles, AISLE performs several services:

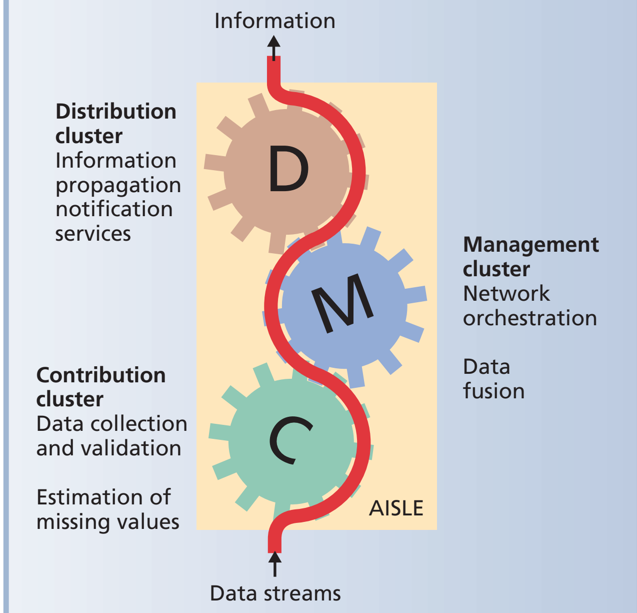
- *Data gathering and validation.* AISLE efficiently captures data streams and validates data pools' content.
- *Substitution or estimation of missing measurements (when needed).* Certain types of applications can't handle missing data properly. In these cases, AISLE uses reasoning abilities to substitute estimates for missing or erroneous measurements.
- *Data management and preprocessing.* AISLE handles data cleaning, normalization, integration, and transformation to custom formats.
- *System extensibility and adaptation.* Because an EMIS can add or remove data pools or end-user applications at runtime, AISLE services are easily configurable to adapt to such changes.
- *Information propagation.* AISLE can handle data streams pushed into the system (as with sensor networks), and it can also pull data streams (such as files or databases) into the system; it subsequently delivers both types to their final destination.

AISLE as a cluster of services

The AISLE architecture arranges the system's services in three distinct yet cooperative clusters that could be implemented as software agents or Web services. The *contribution services* cluster handles data collection and validation and estimates missing information. The *management services* cluster focuses on data management and integration. Finally, the *distribution services* cluster provides the appropriate interfaces to end-user applications. Figure 2 on the next page shows the AISLE mechanism as a synergy of clusters. Incoming raw data streams traverse the three clusters, arranged in a hierarchy. The system's final output is customized information delivered to the end users.

Our implementation of AISLE uses software agents. The agent-oriented software-engineering primitives consider agents service providers, capable of creating "open," virtual institutions ("Agent-Oriented Software Engineering: The State of the Art," Paolo Ciancarini and Michael Wooldridge, *Agent-Oriented Software Engineering I,* Lecture Notes in Computer Science vol. 1957, Springer, 2001, pp. 1-28). In AISLE, three distinct generic agent types are the building blocks of the system:

Figure 2. AISLE as a synergy of three cooperative clusters.



- *Contribution agents* (CAs) act as the implicit data receptors.
- *Data management agents* (DMAs) handle data fusion and preprocessing.
- *Distribution agents* (DAs) act as an interface to the end-user applications.

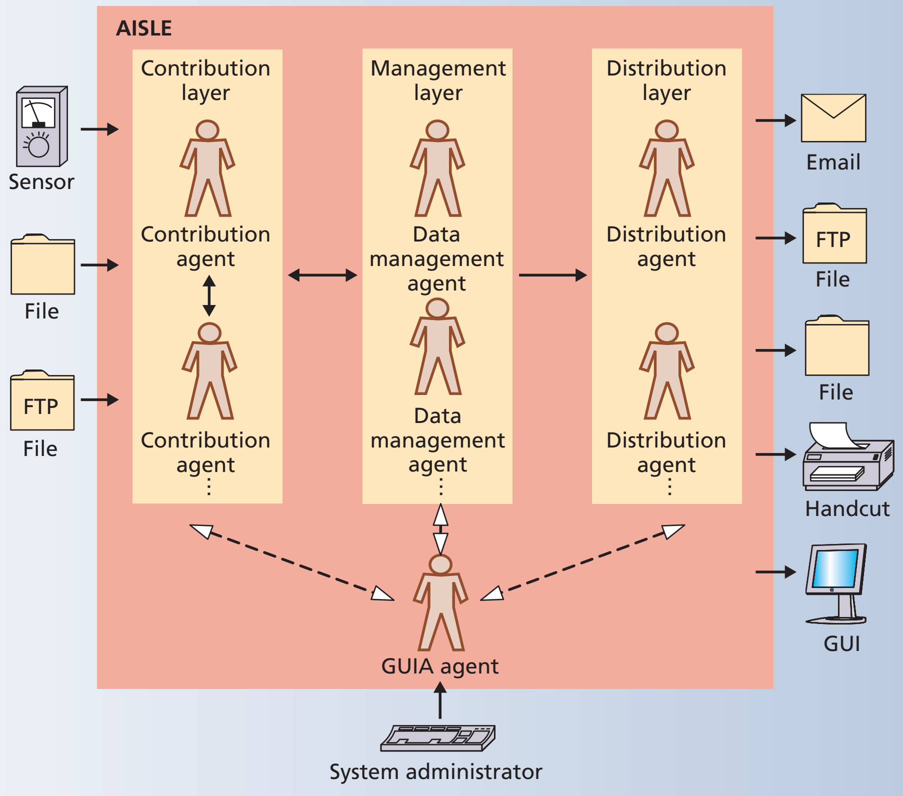
In addition, the system includes a *GUI agent*, which orchestrates all the agents and communicates with the platform administrator. The system uses both inter- and intracluster communication. Intracluster communication lets AISLE agents share perceptions and knowledge within a single cluster, while intercluster communication ensures successful propagation of data between clusters. Figure 3 shows AISLE's multiagent system architecture.

In the contribution layer, CAs originally acquire data in different format types and from diverse sources, such as files on Internet servers, databases, or sensors. Each CA reviews and validates incoming data streams and ultimately delivers preprocessed data to the DMAs. The CAs vary their behavior in accordance with the diverse input sources, following pull or push approaches. Input sources that push data into the system (such as sensors) typically have no local storage, so CAs must monitor them continuously and capture data as they are produced—that is, they must remain online. In other cases, CAs must pull data stored in files or databases, querying these mediums to extract the appropriate data. Having captured data in the original, raw format, CAs must then validate them. Such a service is realized by equipping each CA with a reasoning engine for data validation, which the system administrator could load when instantiating the CA.

In the management layer, DMAs fuse data coming from various CAs into a common data scheme, creating joint presentations (views on data) and performing other preprocessing. Depending on the end user requirements, DMAs prepare presentations, files, or data output streams, which the distribution layer then further disseminates. AISLE administrators have full control of the data manipulation process and can load the DMAs with preprocessing functions.

In the distribution layer, DAs deliver the processed

Figure 3. Multiagent system architecture.



information to their final destination—the end-user applications. DAs can use several methods for diffusing information: Web messages, e-mail, or handouts using SMPT, FTP, or HTTP protocols. Administrators can configure e-mail distribution lists and customize the diffusion services.

At start up, only the GUI agent is active. Then the GUI agent launches the system's user interface, fetches the system configuration parameters, launches all CAs, DMAs, and DAs at runtime, and, ultimately, starts the system process. The administrator can fully customize agent communication channels and more precisely define agents' functionality parameters—decision structures, data processing functions, and distribution customizations.

Development

We developed AISLE using JADE, the Java Agent Development Environment, which supports seamless agent-based development in Java ("JADE—A White Paper," Fabio Bellifemine et al., *EXP in Search of Innovation*, Sept. 2003, pp. 6-19). An agent system implemented with JADE can be distributed across several machines, and administrators can even change the configuration at runtime by moving agents from one machine to another, as and when required.

In addition, we used Protégé-2000 to develop an ontology for describing all AISLE agent concepts and the predicates used to create the communication contracts for the services.

AISLE's implementation conforms to the FIPA specifications for agent development and communication (*ACL Message Structure Specification,* document no. SC00061 G, Foundation of Physical Intelligent Agents, 2002). Figure 4 presents a typical message communicated by AISLE agents for exchanging environmental information, structured using FIPA's Agent Communication Language. Figure 5 depicts the system's GUI, through which the user can extend the provided services by adding new agents, customize service provision, or terminate services.

Although we've used software agents, we could realize the same architecture through Web services. The major difference is that instead of using SOAP and WSDL, our JADE implementation relies on the FIPA-HTTP protocol and the data facilitator agent. We chose agents because they are generally more proactive, they can move from machine to machine, and they can take the initiative when necessary or have internal beliefs about their environment, as in the case of CAs that operate in a pull fashion.

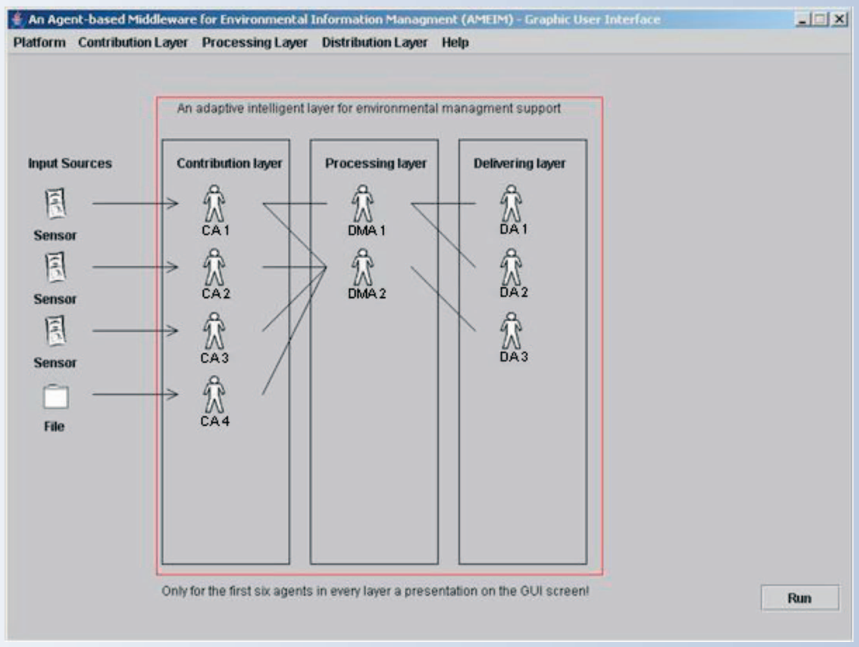
TEST CASE: AIR-QUALITY ASSESSMENT DEMONSTRATION

We evaluated the AISLE architecture by applying it to an ambient air-quality monitoring and assessment system. The typical installation of such a system involves a sensor network measuring air-quality data, including pollutant distributions and meteoro- logical conditions. Usually such

Figure 4. A typical agent message communicated by a contribution agent to a data management agent.

```
(inform
   :sender (agent-identifier :CA1)
   :receiver (set (agent-identifier :DMA1))
   :content
      "((SendCADataToDMAMessage
       (CADataToDMA
         :TypeOfData Sensor
         :Data   <value>28.0</value>
                 <validationTag>VALID</validationTag>
                 <level>Low</level>
                 <variability>increasing</variability>")))
   :language fipa-sl
   :ontology AISLE
```

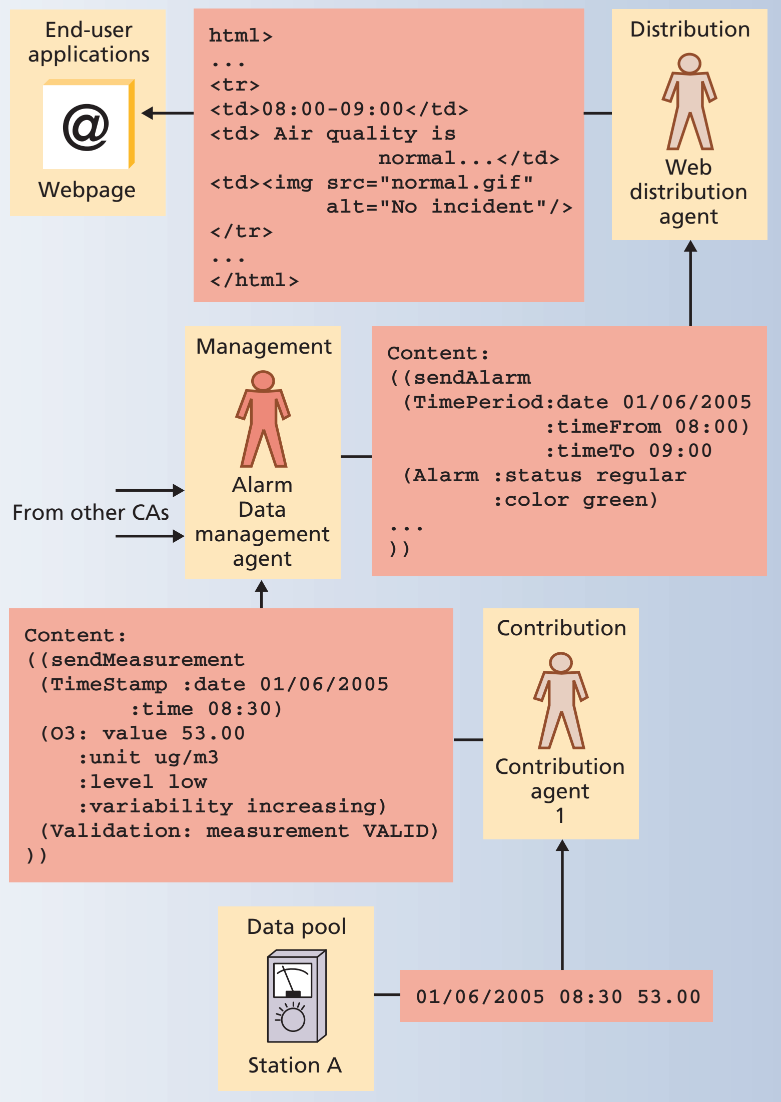
Figure 5. AISLE's graphical user interface.


An Agent-based Middleware for Environmental Information Managment (AMEIM) - Graphic User Interface

An adaptive intelligent layer for environmental managment support

Only for the first six agents in every layer a presentation on the GUI screen!

Figure 6. How information flows through the system (partial view).



```
html>
...
<tr>
<td>08:00-09:00</td>
<td> Air quality is
               normal...</td>
<td><img src="normal.gif"
          alt="No incident"/>
</tr>
...
</html>
```

```
Content:
((sendAlarm
 (TimePeriod:date 01/06/2005
             :timeFrom 08:00)
             :timeTo 09:00
 (Alarm :status regular
        :color green)
...
))
```

```
Content:
((sendMeasurement
 (TimeStamp :date 01/06/2005
        :time 08:30)
 (O3: value 53.00
    :unit ug/m3
    :level low
    :variability increasing)
 (Validation: measurement VALID)
))
```

```
01/06/2005 08:30 53.00
```

systems post their measurements into a database, and the actual end users—human experts—postprocess the sensed measurements for offline study and analysis of the atmospheric phenomena.

However, identifying environmental incidents at the time they occur could yield more immediate benefits. For example, distributing air-quality indicators in urban areas to asthma patients could improve their quality of life by enabling them to take prevention measures when appropriate. In this respect, this technology lets us broaden the goals of air-quality monitoring and assessment systems. Trustworthy information should be disseminated on time to distinct stakeholders that have dissimilar needs and require different levels of access to information.

To demonstrate how this could work, we implemented an air-quality assessment system whose data sources are a network of three stations that monitor several meteorological and air-pollutant attributes. (The test-case system is similar to one described in a previous work, "An Agent-Based Intelligent Environmental Monitoring System," Ioannis Athanasiadis and Pericles Mitkas, *Management of Environmental Quality,* Mar. 2004, pp. 238-249.) The test case system employs AISLE to review the original data, calculate air-quality indexes based on ozone concentrations, and update a local database. (Ozone is a secondary pollutant widely used as an overall indicator of air quality.)

To ensure data flow into the system, we set up three CAs on top of the data pools residing at each meteorological station. We extracted data-driven models and used them to implement engines for validating incoming data and estimating missing measurements. The system employs two DMAs: one integrates data tuples for the database, and the second implements an air-quality indicator alert service, based on ozone indexes, using a deterministic reasoning module. This second DMA's reasoning module incorporates thresholds for ambient ozone concentration, defining several alarm types. Finally, the system includes two DAs in the distribution cluster. One acts as an interface to the database (DB-DA), while the other posts data measurements and ozone alerts to the final users over the Internet (Web-DA).

We tested this system on real air-quality data recorded every quarter hour at three monitoring stations for a three-year period. AISLE managed a massive flow of data and ultimately provided on-time, reliable information to its end users, over the Internet.

Figure 6, showing the voyage of a single ozone measurement to the final user, gives an example of the way the AISLE mechanism integrates and enriches data coming from the sensors, while providing efficient information services to its final users. Suppose that sensors at Station A measure an ozone concentration value. Contribution agent 1 captures and processes it accordingly: First, CA1's data-validation reasoning engine examines it for quality, and, based on the result, replaces missing or erroneous values. CA1 enriches the captured measurement with additional qualitative information such as the validation and variability tags shown in Figure 6, and then forwards the data to the DMA agents. The ozone alarm DMA receives messages from all CAs containing validated ozone measurements. Having gathered all the required information, it uses its ozone-alert reasoning engine to decide on an ozone status, ultimately assessing air quality. Figure 6 shows the outcome in this particular case—current status of ozone concentration, regular; alarm state, green. Finally, the ozone-alert DMA communicates to the Web-DA, which updates the content of the published Web page. Current lab experiments have shown very fast response times for alarm delivery.

Our work on AISLE provides a proof of concept of how it is possible to provide useful public services based on environmental data residing in legacy IT systems or sensor networks. The architecture uses agent-oriented software engineering with service-oriented primitives to deliver on-time, enriched information to its end users without human intervention. Because all agents are loosely coupled, the present AISLE architecture—and the implemented system—are fully extensible. We focused particularly on making the services provided by the agents self-sufficient. The design considered exit strategies and extreme event handling for both the individual agents and the communication channels among them. New services can be added to all three clusters without changing the existing services. Our future research will focus on scaling up and extending the current implementation.

The AISLE approach paves the way for environmental institutes to incorporate dissemination services. It also shows the potential of service-oriented architectures in operational decision making, data fusion, and dissemination of environmental information. ■

Acknowledgments

Thanks go to Andreas Solsbach, of Technical University of Clausthal, and Pericles Mitkas, of Aristotle University of Thessaloniki, for their cooperation and assistance.

Ioannis N. Athanasiadis *is a researcher at Istituto Dalle Molle di Studi sull'Intelligenza Artificiale in Lugano, Switzerland. Contact him at ioannis@idsia.ch; www.athanasiadis.info.*