Ivo Kenji Koga

“An Event-Based Approach to Process Environmental Data”

“*Um Enfoque Baseado em Eventos para Processar Dados Ambientais*”

CAMPINAS
2013

University of Campinas
Institute of Computing

Universidade Estadual de Campinas
Instituto de Computação

Ivo Kenji Koga

"An Event-Based Approach to Process Environmental Data"

Supervisor:
Orientador(a): **Profa. Dra. Claudia Maria Bauzer Medeiros**

"Um Enfoque Baseado em Eventos para Processar Dados Ambientais"

PhD Thesis presented to the Post Graduate Program of the Institute of Computing of the University of Campinas to obtain a PhD degree in Computer Science.

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Computação da Universidade Estadual de Campinas para obtenção do título de Doutor em Ciência da Computação.

This volume corresponds to the final version of the Thesis defended by Ivo Kenji Koga, under the supervision of Profa. Dra. Claudia Maria Bauzer Medeiros.

Este exemplar corresponde à versão final da Tese defendida por Ivo Kenji Koga, sob orientação de Profa. Dra. Claudia Maria Bauzer Medeiros.

[illegible]

Supervisor's signature / *Assinatura do Orientador(a)*

CAMPINAS
2013

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

K822e

Koga, Ivo Kenji, 1981-
An event-based approach to process environmental data / Ivo Kenji Koga. – Campinas, SP : [s.n.], 2013.

Orientador: Claudia Maria Bauzer Medeiros.
Tese (doutorado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Integração de dados ambientais (Computação). 2. Processamento de eventos (Computação). I. Medeiros, Claudia Maria Bauzer,1954-. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Um enfoque baseado em eventos para processar dados ambientais
Palavras-chave em inglês:
Environment data integration (Computer science)
Event processing (Computer science)
Área de concentração: Ciência da Computação
Titulação: Doutor em Ciência da Computação
Banca examinadora:
Claudia Maria Bauzer Medeiros [Orientador]
Omar Boucelma
Jó Ueyama
André Santanchè
Leandro Aparecido Villas
Data de defesa: 18-09-2013
Programa de Pós-Graduação: Ciência da Computação

TERMO DE APROVAÇÃO

Tese Defendida e Aprovada em 18 de setembro de 2013, pela Banca examinadora composta pelos Professores Doutores:

[illegible]

Prof. Dr. Omar Boucelma
Aix - Marseille University

Jó Ueyama

Prof. Dr. Jó Ueyama
ICMC / USP

André Santanchè.

Prof. Dr. André Santanchè
IC / UNICAMP

[illegible]

Prof. Dr. Leandro Aparecido Villas
IC / UNICAMP

[illegible]

Profª. Drª. Claudia Maria Bauzer Medeiros
IC / UNICAMP

Institute of Computing /*Instituto de Computação*
University of Campinas /*Universidade Estadual de Campinas*

An Event-Based Approach to Process Environmental Data

Ivo Kenji Koga[1]

September 18, 2013

Examiner Board/*Banca Examinadora*:

- Profa. Dra. Claudia Maria Bauzer Medeiros (Supervisor/*Orientador*)
- Prof. Dr. André Santanchè
  Institute of Computing - UNICAMP
- Prof. Dr. Leandro Aparecido Villas
  Institute of Computing - UNICAMP
- Prof. Dr. Omar Boucelma
  Aix-Marseille University
- Prof. Dr. Jó Ueyama
  ICMC-USP

[1]Financial support: CNPq (process 141060/2010-7), FAPESP scholarship (process 2009/52336-7)

Abstract

Environmental sciences studies deal with a large amount of heterogeneous data sources. In each application or scientific study there is a need for a set of tools to manage the data, from the capture, processing to data storage. In particular, in environmental applications there is a need to combine distinct kinds of sensor data from those aboard satellites to those in ground networks. While the first sources are static and usually processed via image management, the second kind is dynamic and processed in streams. Solutions for combining them are usually tailored to a specific problem and geographical region. This work is concerned with solving some of these problems by enabling the integration of heterogeneous data sources whether they are static or streams. The proposed solution uses a pre-processing phase in order to filter the data sources needed for the study, Enterprise Service Bus (ESB) to enable the integration of different data sources and Complex Event Processing (CEP) to process the events that emerge from the integrated environmental data. Events are detected through patterns that are specified by experts and inserted into the CEP engine. Upon detection of an event pattern, events are disseminated, stored in a database or sent to other systems. The main contributions of this work are: (i) a framework to help environmental scientists cope with heterogeneity problems that allows integration of static and stream data sources in a generic way (as opposed to specific solutions in the literature), (ii) treatment of environmental data events, and processing patterns, (iii) application of the aforementioned findings in ecological studies showing how scientists can use our proposal to acquire data of interest from the available data sources.

Resumo

Estudos em ciências ambientais lidam com uma grande quantidade de fontes de dados heterogêneas. Em cada aplicação ou estudo científico é necessário um conjunto de ferramentas para gerenciar os dados desde a coleta, processamento até o armazenamento de dados. Em especial, em aplicações ambientais, existe a necessidade de combinar tipos distintos de dados de sensores desde aqueles à bordo de satélites até os que se encontram em sensores terrestres. Enquanto os primeiros são estáticos e normalmente processado através do gerenciamento de imagens, o segundo é dinâmico e processado em fluxos. Soluções para combiná-los são usualmente adaptadas a um problema e região geográficas específicos. Este trabalho se preocupa em solucionar alguns destes problemas pela possibilidade de integração de fontes de dados heterogêneas sejam elas estáticas ou em fluxos de dados. A solução proposta utiliza uma fase de pré-processamento para requisitar e filtrar as fontes de dados necessárias para o estudo, Enterprise Service Bus (ESB) para possibilitar a integração de diferentes fontes de dados e *Complex Event Processing* (CEP) para o processar os eventos que emergem dos dados ambientais integrados. Eventos são detectados através de padrões que são desenvolvidos e inseridos na *engine* de CEP. Após a detecção de um padrão de evento, eventos são disseminados, armazenados em bases de dados ou enviados para outros sistemas. As principais contribuições deste trabalho são: (i) *framework* para o auxílio aos cientistas ambientais lidarem com problemas de heterogeneidade e que permite a integração de fontes de dados estáticas e em fluxos de forma genérica (em oposição a soluções específicas encontradas na literatura), (ii) tratamento de eventos de dados ambientais, e processamento de padrões, (iii) aplicação dos resultados anteriores em estudos ecológicos mostrando como cientistas podem usar nossa proposta para adquirir dados de interesse a partir das fontes de dados disponíveis.

Acknowledgements

There are many people to be thankful for. Initially, I would like to thank God for my life and all the achievements I had until now.

I would like to thank my family for all the love and support that you have always provided.

I would like to thank my advisor, professor Claudia Bauzer Medeiros. It has been an honor to be your PhD student. I learned a lot in these years of my PhD journey.

I would like to thank professor André Santanchè for his friendship and all helpful advices since the beginning until now in my research career. I will always be grateful for all your help.

I would like to thank professors Omar Branquinho and Felipe Toledo for your contributions to my work.

I would like to thank all my friends: Matheus, Jorge, Joana, Bruno, Ivelize, Eduardo, Alessandra, Daniel Cugler, João Sávio, Lucas, Jacqueline, Celso, Carla, Alan, Fábio, Otavio, Jefferson and all those who are not mentioned. You provide me all the friendship and support I needed in order to achieve the results of this work, thank you very much.

I would like to thank Mayumi for all her love and support.

I would like to thank Unicamp Swimming Society Reloaded (USSR), the swim team, where I learned what it means to be a team.

I want to thank all those who were not mentioned, but no less important.

I would like to thank FAPESP (grant 2009/52336-7) for its scholarship.

Finally, I would like to thank Cooxupé, Institute of Computing (IC) - UNICAMP and the support from Brazilian financing agencies CNPq (grant 141060/2010-7), the Microsoft Research FAPESP Virtual Institute (NavScales project), CNPq (MuZOO project), Brazilian Institute for Web Sciences Research (CNPq 557.128/2009-9), PRONEX-FAPESP [2], CAPES (AMIB project), as well as individual grants from CNPq.

[2]Model and Methods in eScience fot the Life and Agricultural Sciences

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Environmental data are growing in importance every day. Governments, companies and citizens are interested in climate changes and how they affect everyday life. New kinds of data sources are emerging constantly, thus providing new opportunities for scientists to develop better models. On the other hand, overcoming data heterogeneity has become a challenge.

In order to accomplish their goals, environmental scientists have to cope with this heterogeneity. Therefore it is necessary to create strategies to retrieve, integrate and process data produced for each specific kind of study.

There is plenty of work that deals with environmental data and that handles the issues of their capture, analysis and storage such as [9, 11, 71, 72]. In acquiring data, such research aims at sampling, data routing, reliable communication and energy-efficient acquisition of data. In analyzing data, the focus is in data exploration, classification, statistics. In storing data, research concentrates in the capacity of data storage and search in data repositories.

An increasing number of environmental studies must cope with static and dynamic data, e.g. [4]. Solutions are geared towards specific problems, and particular data sources. Typically, each study is directed to a given geographic region, having a specific focus (e.g., analyzing drought effects). Once region and focus are defined, researchers specify the data sources of interest and provide an approach that takes advantage of the characteristics of these sources (e.g. in [4], satellite images, soil water and radiation sensors were combined to evaluate drought stress and carbon uptake in a specific region of the Amazon forest). Even when they manage to integrate static and dynamic sources, these approaches are seldom scalable to other regions, or phenomena, or other kinds of data sources. Our approach, instead, is generic. To the best of our knowledge, there is no generic approach on joint integration of stream and static data for environmental sciences.

The goal of this thesis is to help deal with the integration of heterogeneous data via a framework to process environmental data. In our solution, data are integrated using Enterprise Service Bus (ESB) [64], which is an infrastructure focused in integration of systems that communicate via messages. Data are transformed into messages and treated as events. Events are retrieved from the data flowing through the ESB and detected through *event patterns* which are templates that describe events and their contextual descriptions, i.e. all the necessary characterization to detect events.

Events have been researched in computational systems for a long time, from operating systems to real time systems. In particular, we adopted the Complex Event Processing (CEP) [49] paradigm to process events. CEP provides means to deal with rules and events' causal relationships, i.e. it provides means to deal with aggregations of events called complex events. In CEP, an event is an object signifying an activity that a computer can process [49].

Using our framework, environmental scientists can be alerted about given conditions, expressed as patterns. Thus, a scientist can detect combinations of events occurring in a particular order, or if a particular event did not emerge when/where it was expected to occur.

Taking this scenario into account, the main contributions of this thesis are:

- A framework to help environmental scientists cope with heterogeneity problems that allows integration of static and stream data sources in a generic way (as opposed to specific solutions in the literature)
- Treatment of environmental data events, and processing patterns
- Application of the aforementioned findings in ecological studies showing how scientists can use our proposal to acquire data of interest from the available data sources

This thesis is organized as a collection of papers, as follows:

Chapter 2 is the paper Handling and Publishing Wireless Sensor Network Data: a hands-on experiment, published in the Journal of Computational Interdisciplinary Sciences (JCIS) in 2011 [43]. This chapter presents our first steps in solving heterogeneity problems in Wireless Sensor Networks (WSNs). It presents some solutions for managing sensor data using two pillars: Web Services to provide interoperability and components to provide loose coupling and extensibility. We created a framework that enables the development of different types of components for queries and data visualization.

A case study was conducted with a WSN that retrieves data from air humidity, light and temperature, located at the Faculty of Agriculture Engineering (FEAGRI). Data were collected at about two minutes interval, received and published in Web Services and accessed and viewed through a client that supports OSGi components developed by us.

An experiment was carried out using Windows Azure [54] to prove the flexibility of the framework. The integration of different data sources was straightforward because of the architectural solutions adopted. This solution, however, was restricted to sensor data.

Chapter 3 is the paper Managing Environmental Data from Sensor Networks, submitted to the 10th International Conference on Web Information Systems and Technologies (WEBIST 2014) [41]. This chapter presents our first steps in defining environmental event patterns. Our approach uses Enterprise Service Bus (ESB) and Complex Event Processing (CEP). This study treated primarily sensor data flows as events that are reported to users, according to user requests.

Environmental events are sensitive to space and time factors. Thus, as will be seen, the events considered in this thesis take these factors into account and are denoted as a quadruple: <*measured-value*, *nature*, *spatial-variable*, *timestamp*> = <*v*, *n*, *s*, *t*>, where:

- *v* is the value of the variable captured (e.g. 40 [degrees celsius], 80% [humidity]);
- *n* the nature of the variable (e.g. temperature, humidity);
- *s* is the location (coordinates) of the measurement;
- *t* is a time interval [ts, tf][1] for which v was valid at that location (s). If ts = tf this is an instantaneous event (e.g., sensor measurement).

In other words, events here have a temporal and a spatial component. An event pattern is the specification of one or more events concerning the variables being monitored. It is a description of how an event or a set of events of interest looks like [49]. We define two types of patterns: simple and composite. Simple ones can be of type value, spatial, temporal or spatiotemporal, and concern one event; composite patterns comprise a combination of events.

In value patterns, predicates consider the value of an environmental variable measured by a sensor, weather station, etc (for example: temperature, humidity, light, solar radiation, among others). Spatial patterns are based on the spatial properties of events, considering the value of environmental variables given a spatial context. Temporal patterns are the ones that are concerned with variation of events in time, i.e. when time plays a major role [21]. Spatiotemporal patterns are patterns specific to the evaluation of time series of events and spatial trends over time [21]. They combine measured variables, spatial and time properties.

The combination of different sources from stream and non-stream (stored) data to produce correlations is mentioned, but not detailed. Chapter 4 gives more insights on how our framework deals with these kinds of problems.

[1]ts is the start time and tf is the final or end time.

Chapter 4 is the paper Patterns in Environmental Event Processing is a technical report at the Institute of Computing (IC) -UNICAMP [40] and a revised version will be submitted to an International Journal. This chapter presents a review of our framework presented in Chapter 3, considering pre-processing aspects before data enters ESB and an application of the framework in ecological monitoring using events.

Environmental data are by nature much more diverse than those found in enterprise data. That is, environmental data can come in the form of images, text files, tables, sensor data flows, etc., which is not the standard expected format to ESB.

Data can be retrieved in two ways: request or receive; and can be filtered in order to properly enter the framework and be processed by the event processing engine.

This paper presents a real case study combining sensor and satellite data.

Chapter 5 contains conclusions and some directions for future work.

Figure 1.1, transcribed from chapter 4, presents an overview of the framework. This figure is detailed in that chapter.

In Figure 1.1, data flows from the bottom to the top. Arrows indicate data flow. First, the framework absorbs data by pushing or pulling data in (1). Filtering (2) is required before data enter the ESB, since non relevant data could flood unnecessarily the framework. After this pre-processing phase, data are encapsulated into messages by channel adapters (3) and some of them are chosen to be detected by event patterns (4). These event patterns are written by domain experts that can provide means to define important patterns out of the integrated data. These event patterns are created and implemented in Event Processing Agents (EPAs) (5), which evaluates events and trigger a new event if there is a mach between pattern and events in their observation window. This new triggered event is encapsulated in a new ESB message (6) and sent to the interested users (7). In order to notify users or store events (8), the output can occur in two ways: notification or request/reply. Events are sent to subscribed users to receive notification or sent for storage in databases, file servers, etc. Subsequently, other users can make a request for these stored data. This request goes through the ESB and recovers the repository where the data lies.

This architecture is generic. The main effort required to adopt it lie in two situations: (a) adapters and (b) specifying patterns. For every new kind of data source, a specific adapter needs to be developed. However, once it is created, any application that needs that kind of data can just reuse that adapter. The second kind of effort (b) depends on experts to indicate the kind of event of interest, so that patterns can be created.

Based on this figure, chapter contents are the following. Chapter 2 concentrates on mechanisms for acquiring and processing data from sensor networks (i.e., the data capture phase, before entering the framework). Chapter 3 concerns the specification of patterns to be processed via CEP – in part (5) of Figure 1.1. Chapter 4 follows the data cycle

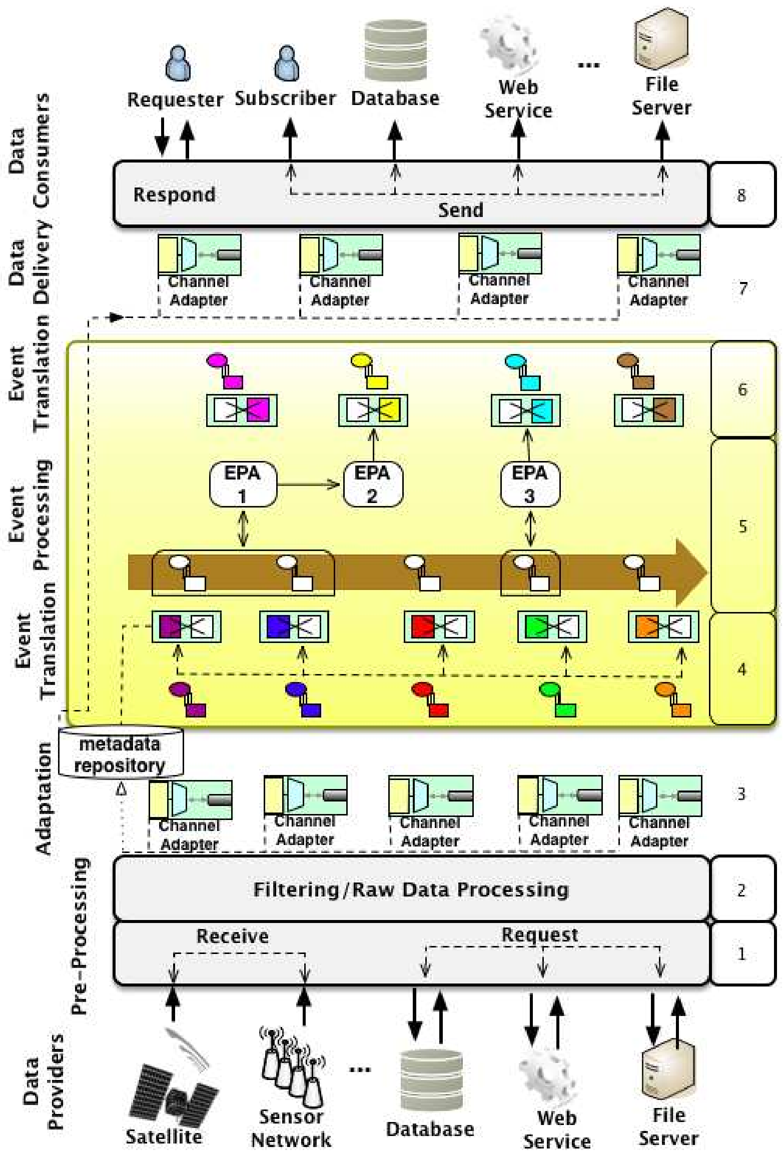


Figure 1.1: Architecture of the framework.

from entry to output, showing screen copies of the implementation.

The solutions adopted evolved as the research was conducted. The first results, in chapter 2, are restricted to getting data from a homogeneous sensor network, and publishing these data on the Web. Also, this was an experiment on components to customize user interfaces. Chapters 3 and 4 concern our framework, which is concerned with handling heterogeneous sources. Chapter 3 is focused in dealing with sensor network data, while chapter 4 generalizes the work to any kind of environmental data sources. Thus, chapter 3 does not use preprocessing features of the figure.

This thesis gave origin to the following publications:

- I. Koga, C. B. Medeiros, and O. Branquinho. Handling and publishing wireless sensor network data: a hands-on experiment. In Proceedings IV eScience Workshop - XXX Brazilian Computer Society Conference. SBC, July 2010 [42].

- I. Koga, C. B. Medeiros, and O. Branquinho. Handling and publishing wireless sensor network data: a hands-on experiment. Journal of Computational Interdisciplinary Sciences, 2(1):13–22, March, April 2011 [43] – extended version of the previous publication.

- I. Koga and C. B. Medeiros. Integrating and processing events from heterogeneous data sources. In Proceedings VI eScience Workshop - XXXII Brazilian Computer Society Conference, July 2012 [39].

Chapter 2

Handling and Publishing Wireless Sensor Network Data: a hands-on experiment

2.1 Introduction

Wireless Sensor Networks (WSN) [2] are a special kind of ad hoc network, composed of a huge amount of small nodes with low processing capacity, limited power source, high mobility and higher probability of failures than other kinds of networks due to communication, power, and/or node failures. Nodes potentially have different types and functionalities and monitor a wide scale of physical and environmental variables (e.g. temperature, humidity).

WSNs allow the acquisition of data in difficult conditions, for a wide range of spatial and temporal resolutions and scales. The sensors can be intimately connected with the observed phenomena, being kept active during a long time, being deployed everywhere – under the sea, underground or in space. Ubiquitous and pervasive, they can be also implanted in our bodies (and thus generate data for eHealth studies) or our home (for ambient applications).

This possibility of monitoring many phenomena, in various temporal and spatial scales, produces a large volume of heterogeneous data. Heterogeneity and volume of data, combined with heterogeneity in user requirements, pose many problems. The storage, retrieval and visualization of data in this kind of setting is a challenge which is associated with the first Grand Challenge in computer science defined by the Brazilian Computer Society – Management of large distributed multimedia data volumes [53].

The goal of this work is to contribute towards solving one of the many facets of this grand challenge, by proposing a practical way of storing and publishing sensor data, making possible the extraction of information by different types of users. Each kind of user profile can determine their special needs, defining what they want from the available data allowing the extraction of relevant information.

This work is related with ongoing research on the management of sensor data in eScience – in particular, for biodiversity and environmental studies. We present our proposal by means of a case study of management of environmental data in an application related to agriculture.

The publication of sensor data, on the Web, involves issues that go beyond the nature of the data being collected and are intimately related with problems of the Web itself – such as data heterogeneity, privacy, volume of data and user requirements. Hence, additionally, we investigate this proposal under the light of the perspective of Web Science [6].

The main scientific contributions are the following:

- framework that combines distinct technological solutions;
- interoperability to support sensor data publication;
- discussion of sensor networks in the context of Web Science;
- validation of the framework for a real case study, emphasizing extensibility and flexibility.

The rest of this paper is organized as follows. Section 2 presents a brief overview of WSN data management. Section 3 presents our approach. Section 4 discusses our case study and section 5 presents ongoing work.

2.2 Related Work

This paper concers the handling and publication of large volumes of sensor network data. There is a wide range of open problems in this domain. This paper is concerned with the issues of flexibility in data publication on the Web and interoperability across networks. Thus, we concentrate on discussion of work on publication of data in eScience, and some Web Science issues.

2.2.1 Data publication – interoperability and flexibility issues

Different types of systems are being proposed and deployed to support scientists' work in many research areas handling heterogeneous data sources, including sensor data. Biodiversity systems are an example of this type of system to support the work of biologists. Examples are studies in ecology or environmental monitoring. On closed environments, sensor networks are being used in scientific studies concerning health (e.g. patient monitoring) or chemistry (experiment monitoring) – see [31].

In all these contexts, there are countless initiatives concerning WSN data management, that range from network configuration and energy management to data processing and publication [56, 32, 83, 60]. This paper is concerned with solutions that support the latter stage – i.e., once data are collected, how to provide flexible mechanisms that will forward data to be processed and published, hiding low-level details. Our choice of related work reflects this, concentrating on architectures for sensor data management and publication.

Architectures to support flexibility

Many solutions have been proposed to overcome the problems of heterogeneity and interoperability of sensor data management. Chu et. al. [12] created an architecture called NICTA[1] Open Sensor Web Architecture (NOSA) which combines a Service Oriented Architecture and WSNs using the services specified in the Sensor Web Enablement [58] from the OpenGIS Consortium. Figure 2.1, reproduced from [12], shows NOSA and its components. There are 4 layers: Sensor Fabric, Application Services, Application Development and Applications. The first layer deals with sensors and their emulation/simulation, the second is composed by services that support network management, the third provides the APIs, tools and configuration and the last has the applications that use the sensor data.

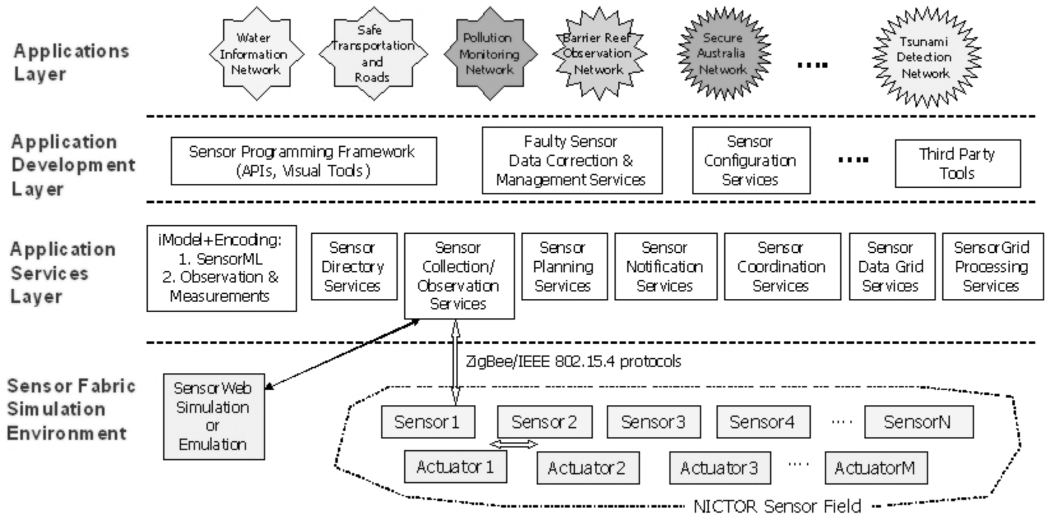

Applications Layer
Water Information Network
Safe Transportation and Roads
Pollution Monitoring Network
Barrier Reef Observation Network
Secure Australia Network
....
Tsunami Detection Network
Application Development Layer
Sensor Programming Framework (APIs, Visual Tools)
Faulty Sensor Data Correction & Management Services
Sensor Configuration Services
....
Third Party Tools
Application Services Layer
iModel+Encoding: 1. SensorML 2. Observation & Measurements
Sensor Directory Services
Sensor Collection/ Observation Services
Sensor Planning Services
Sensor Notification Services
Sensor Coordination Services
Sensor Data Grid Services
SensorGrid Processing Services
ZigBee/IEEE 802.15.4 protocols
Sensor Fabric Simulation Environment
SensorWeb Simulation or Emulation
Sensor1
Sensor2
Sensor3
Sensor4
....
SensorN
Actuator1
Actuator2
Actuator3
....
ActuatorM
NICTOR Sensor Field

Figure 2.1: NICTA Open Sensor Web Architecture.

[1]Australia's Information and Communications Technology (ICT) Centre of Excellence

In NOSA, sensor data are always processed by entities external to the sensor network. This can be an advantage in scenarios where the deployment and maintenance of the sensors are easy. These scenarios consider that sensors will only sense and send data, without any processing, which consumes more power (most expensive activity in face of power consumption) [2]. However, in specific scenarios, it could be more appropriate to use WSN pre-processing capacity before actually sending data. Another disadvantage is that applications cannot reuse code.

Pastorello Jr [62] also followed a multiple layer approach, but from another perspective. He dealt with the problem of production and management of WSN data through a framework that uses software components called Digital Content Components [69] and scientific workflows to provide management facilities and easy access to the sensor data. Unlike NOSA, this work does not consider the OpenGIS Consortium standards for WSNs. However, it has some advantages such as flexibility, letting open the possibilities for development of new components for access and management of sensors, regardless of the sensors' implementation and technology.

Global Sensor Networks (GSN) [1] is a platform developed in Java that provides an infrastructure for the integration of technologies of heterogeneous sensor networks using a set of abstractions and XML. GSN has the advantage of facilitating the WSNs deployment when it hides its implementation details. On the other hand, it also hides platform specific parameters that might render each deployment more flexible.

Handling heterogeneity for specific applications

While the previous section concerned generic architetural solutions, other kinds of sensor-related research propose solutions that are tailored to specific applications, or application domains. This requires, for instance, designing special purpose databases, or developing special software.

For instance, the GeoCENS project [48] focus on capturing data from local scale sensor networks, deployed and operated by individual scientists. These kinds of data are more likely to remain underutilized and eventually lost. The project deals with challenges like heterogeneity (there is much more heterogeneity in these deployments), protecting data ownership (researchers spend time and funding on data collection, so they want to protect their property), motivation (there should be incentives to motivate people to publish their data), and provide an intuitive and coherent user interface. For instance, they consider digital watermarking to protect sensor data and have created a 3D web interface to allow users to manipulate sensor data in a more intuitive way.

The Southeast Alaska MOnitoring Network for Science, Technology, Education and Research (SEAMONSTER) [30] was developed and deployed in Alaska to study glaciated watersheds. It stores all the data measured in a PostGIS database. The SensorWeb Enablement (SWE) protocols from Open Geospatial Consortium (OGC) [65] were used in SEAMONSTER in order to provide interoperability. Geoserver [59] was used to deliver dynamically generated geospatial output in their web portal. KML (Keyhole Markup Language) was adopted for interoperability. KML is an XML language focused on geographic visualization, including annotation of maps and images. The project developed a portal to provide temperature, humidity, precipitation and voltage data using openLayers and accessing Bing from Microsoft to give the node location.

The Life Under Your Feet project was developed and deployed for soil monitoring at an urban forest in Baltimore [73]. It measures and saves soil moisture and temperature in situ. Their key requirements for soil ecology sensor systems include fidelity, accuracy, precision, sampling frequency, fusion with external sources, experiment duration and deployment size. Their solution, employed at a micro underground scale, is now being ported to a very different environment – monitoring conditions in Brazil's rainforest.

Interoperability and publication

The problems faced by all these proposals analyzed in this section range from a micro perspective (a large amount of sensors in a single network) to a macro one (between WSNs and between them and the Web). Pastorello Jr proposed components and workflows to deal with the heterogeneity problem. GSN was proposed as an infrastructure to overcome some deployment problems using XML and abstractions implemented in Java. NOSA encapsulates the operations in a software layer that uses the Sensor Web Enablement standards and grid computing to provide a middleware that provides services that overrides sensors' implementation complexity.

On the application side, GeoCENS dealt with the absence of local scale sensor network data. SEAMONSTER deployed a system that provides some measurements and publishes it on a simple interface using openLayers and Bing, geared towards specific user needs. Life Under Your Feet provided soil measurements and its database considers requirements to provide quality in monitoring sensor networks for scientific data for underground soil conditions.

In these and other efforts, the idea is to provide several layers of isolation between the sensor networks and the users. Then, one can customize and develop each layer and concentrate on the solution of a few problems at a time. As will be seen in section 3, our proposal combines features from some of the reviewed papers.

2.2.2 Publishing sensor data on the Web – a few Web Science concerns

The discussion on Section 2.1 concerned interoperability and publication issues, for eScience needs. However, most of those projects are concerned with the Web environment, which is increasingly becoming a prime environment for eScience research.

In this context, which also touches our work, we should also look at associated issues, in Web Science.

The term Web Science was first introduced by Berners-Lee in [6]. It has since given origin to large international research efforts, including The Web Science Trust [79]. In Brazil, the theme motivated a National Institute of Science and Technology (INCT) in Web Science – the Brazilian Institute for Web Science Research [24].

Formally, research in Web Science is concerned with the Web as the primary object of interest. Thus, rather than considering the Web as a medium for collaboration, communication and socializing, it studies the Web itself. In our work, this means among others concentrating in two issues:

- the effects of data publication on the Web and its impact on the long tail of data e.g., see [48];
- the use of Web Services as a basis for interoperability.

Long tail concerns are becoming increasingly popular among eScientists. The idea is to access data collected by thousands of individual researchers, but which are difficult to find. Publication of these data on the Web makes sure they become public, but does not ensure their accessibility, nor their visibility (both of which Web Science concerns [6]).

Data publication via Web Services increases accessibility, since service interfaces must follow specific standards. However, these same standards sometimes hamper particular needs. For instance, services do not allow updates. Also, since they have been conceived to enhance interoperability, they may hide information that would be useful to an end-user – e.g., sampling frequency or data quality provided by a sensor gateway. Thus one must consider extending services, to provide more flexibility.

Visibility is even more complicated. Since, as shown in section 3, we use components to display sensor data coupled to services, additional semantics must be conceived for publication. This is subject of future work.

2.3 Proposed Solution

We are concerned with interoperability and publication flexibility, as two facets of the first Grand Challenge. Our solution is based on two aspects:

- Web services – to provide interoperability between applications, WSNs, data servers and user applications;
- Components – to support reuse and loose coupling, and to allow multiple user-friendly visualizations of sensor data.

Figure 2.2 gives a high level view of our proposal. It has three main components (or layers): WSNs (on the left), data servers (on the right), and user applications. Data communication among components is supported by Web Services. Specific functionalities are implemented by software components. Each WSN is assumed to connect to an access point. Each access point runs a Data Load service that sends (pushes) the raw data to a central data server.

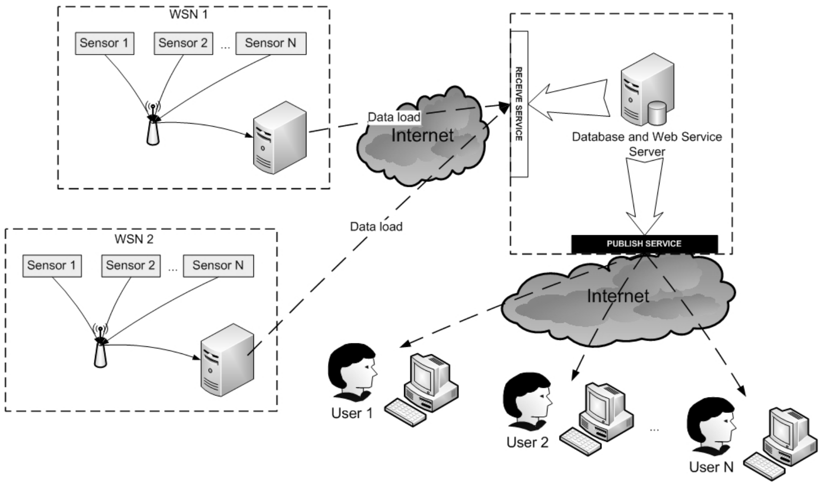


Figure 2.2: Architecture of the solution.

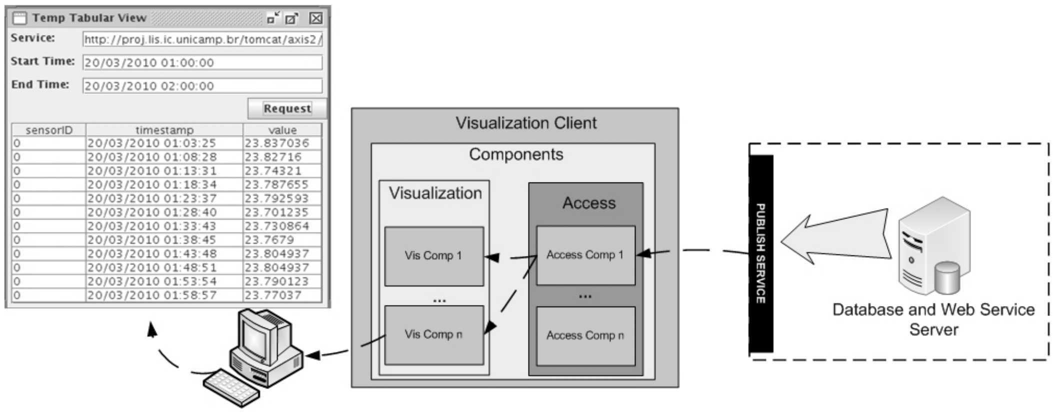
The data server (right side of the figure) implements two Web Services: a Receive service and a Publication service. The Receive service formats the data received from each Data Load service into standard tuples, and stores them in a database. The Publication service publishes basic methods that execute SQL queries on the database.

User requests are treated as follows: Distinct query parameters and visualization requests are implemented as components that invoke the Publication service. Storage and visualization thus follow two independent pipelines. In the “push” pipeline, Data Load pushes data into the Receive service. In the “pull” pipeline, software components request data from the Publication service. Hence, different applications can build their own components and this provide user-tailored visualizations.

This solution has the following merits. First, it takes advantage of Web Services to provide access interoperability. Second, since it is based on components, it is extensible

– e.g., components can also be developed at the WSN access point side to preprocess the data before it is sent to the server – e.g., providing fusion facilities. Alternatively, as in our case, services can be developed at the server side to integrate and customize data according to distinct application requirements. Also, different servers can be installed to support, for instance, distinct needs or to integrate data from different networks.

The use of software components makes possible the development of user-specific components to access and visualize WSN data. This is shown in Figure 2.3 where we have distinct components for accessing and visualizing data, separating this in a Model-View-Controller pattern [45]. Here, one access component can be used by many visualization components and also one visualization component can use many access components.



Service: http://proj.lis.ic.unicamp.br/tomcat/axis2/

Start Time: 20/03/2010 01:00:00

End Time: 20/03/2010 02:00:00

sensorID	timestamp	value
0	20/03/2010 01:03:25	23.837036
0	20/03/2010 01:08:28	23.82716
0	20/03/2010 01:13:31	23.74321
0	20/03/2010 01:18:34	23.787655
0	20/03/2010 01:23:37	23.792593
0	20/03/2010 01:28:40	23.701235
0	20/03/2010 01:33:43	23.730864
0	20/03/2010 01:38:45	23.7679
0	20/03/2010 01:43:48	23.804937
0	20/03/2010 01:48:51	23.804937
0	20/03/2010 01:53:54	23.790123
0	20/03/2010 01:58:57	23.77037

Figure 2.3: Components to request published data and visualize it.

In order to provide a first prototype for visualization of sensor data, we used the FLAVOR framework [44]. FLAVOR was developed to support flexible design and construction of software components to visualize measurements of network traffic. We point out that such measurements can be treated as time series – and thus FLAVOR was used to visualize our sensor measurements. As new requirements to access and visualize WSN data appear, FLAVOR may need to be progressively extended.

Our solution combines aspects from NOSA and Pastorello's work (see section 2). From the latter, it adopts the philosophy of components to encapsulate functionality and increase modularity. From NOSA, it uses aspects of publication using Web Services, thereby increasing interoperability. Moreover, we treat the heterogeneity problem at the storage level, standardizing the format to store sensor data. Thus, the role of components is to provide distinct visualization formats, including simultaneous views of multiple sensors.

Our infrastructure uses OSGi [74] as the software component standard. The OSGi defines a standard and component oriented environment that provides a Java framework which supports the deployment of extensible components called bundles.

OSGi provides standard primitives that allow applications to be constructed through small, reusable and collaborative components. It manages the installation and update of bundles in a dynamic and scalable way. It also provides resources for one to take advantage of the dynamic load of code, that is, it allows the dynamic deployment of components in the environment without the need of restarting the entire application. [52]

Using OSGi, our environment can be updated at execution time (without the need of restarting). For instance we can add new functionalities at the WSN client, preprocessing the data before sending it to the data server. This new functionality can be added seamlessly by just installing a new component at the WSN client, without the need of stopping and restarting the application and without the need of interrupting the data flow from the WSN to the data server.

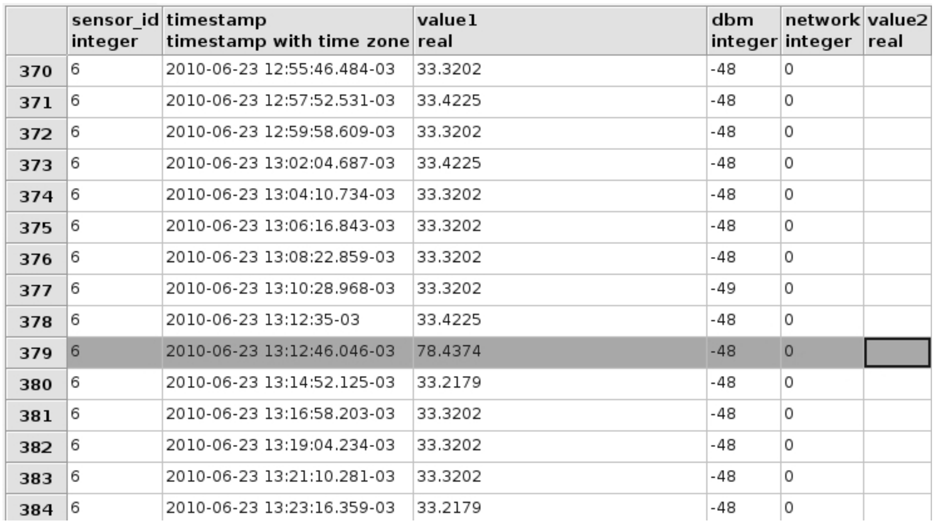
Our solution also provides flexibility for multiple kinds of queries – e.g., involving aggregation, interpolations or transformations. There are two ways of doing that: adding new methods at the Publication service or implementing new components at the user application level. The former is done by adding new methods to the service that will map to the PostgreSQL queries. The latter can be implemented as a chain of components that implement such functions on top of basic invocations of the Publication service (i.e., our solution differs from others, since instead of changing the service we add components). In such a case, a request for an aggregation over a period of time for "n" temperature measurements can be translated into an execution of a sequence of components – the first component will request from the Publication service data and the second will compute the aggregation.

2.4 Case Study: a hands-on experiment

Our case study concerns managing data from a WSN deployed at the Faculty of Agriculture Engineering (FEAGRI) at UNICAMP. Sensor data are collected at an access point installed at FEAGRI, to be processed at the Laboratory of Information Systems (LIS), at the Institute of Computing. For this experiment, we were concerned with basically three issues – sensors heterogeneity, data publication and processing. To create a test case of heterogeneous sensors, we collect data from sensors sensing air humidity, light and temperature. Even though we had a small amount of sensors, we had to deal with 3 types of measurement, each with different frequency of data acquisition and units.

All the sensors send data to a local base station which is connected to a computer (the access point) that has a web service (the Data Load service). This service, developed by us, sends the measured sensor data to a server located at LIS. This WS server (Receive Service) at LIS then stores the data in an appropriate way in a PostgreSQL database. In our specific implementation, data are collected at about 2 minute intervals. Finally, data are published through another WS (Publication Service).

Since our concern in this first stage was in interoperability and publication flexibility, our data records are very simple and store a minimum of information. Figure 2.4 shows a short list of records in the raw sensor data database. This table has 6 attributes: record id, sensor id, value, timestamp (when measure was taken), and network id. The last attribute, named value2, indicates whether this sensor is capturing more than one value. This particular table is a snapshot of a humidity measurement in june 23rd, 2010. Record 379 shows an outlier at Universal Time 13:12:46 where it measured 78.4, while all other measurements for the same time period had values between 33.2 and 33.4. The publication of these data must take this into consideration – here, this was probably due to some sensor malfunction. For temperature measurements, our extension to FLAVOR consisted

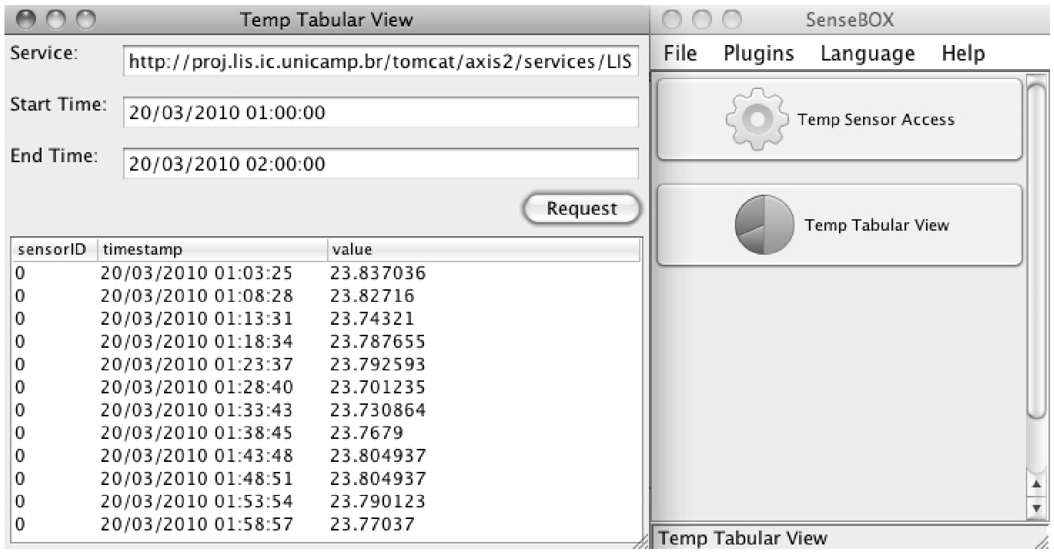


	sensor_id integer	timestamp timestamp with time zone	value1 real	dbm integer	network integer	value2 real
370	6	2010-06-23 12:55:46.484-03	33.3202	-48	0	
371	6	2010-06-23 12:57:52.531-03	33.4225	-48	0	
372	6	2010-06-23 12:59:58.609-03	33.3202	-48	0	
373	6	2010-06-23 13:02:04.687-03	33.4225	-48	0	
374	6	2010-06-23 13:04:10.734-03	33.3202	-48	0	
375	6	2010-06-23 13:06:16.843-03	33.3202	-48	0	
376	6	2010-06-23 13:08:22.859-03	33.3202	-48	0	
377	6	2010-06-23 13:10:28.968-03	33.3202	-49	0	
378	6	2010-06-23 13:12:35-03	33.4225	-48	0	
379	6	2010-06-23 13:12:46.046-03	78.4374	-48	0	
380	6	2010-06-23 13:14:52.125-03	33.2179	-48	0	
381	6	2010-06-23 13:16:58.203-03	33.3202	-48	0	
382	6	2010-06-23 13:19:04.234-03	33.3202	-48	0	
383	6	2010-06-23 13:21:10.281-03	33.3202	-48	0	
384	6	2010-06-23 13:23:16.359-03	33.2179	-48	0	

Figure 2.4: Measurement of humidity from sensor 6 on june 23rd, 2010.

in creating two components: TempSensorAccess and TemperatureSensorTabularView, respectively the access and the visualization components for accessing the temperature sensor data available. Figure 2.5 shows an example of visualization of temperature data, using a table format. An alternative means of visualizing the sensor data appears in Figure 2.6. This was developed using a distinct software, but using the same underlying data stored in PostgreSQL. Such flexibility in handling data is only possible because of our architectural choices.

There were several difficulties in deploying the first sensors, ranging from engineering problems to defining a storage format for data. For instance, the setting up of the communications infrastructure and the calibration of sensors took more than one year.



Service: http://proj.lis.ic.unicamp.br/tomcat/axis2/services/LIS

Start Time: 20/03/2010 01:00:00

End Time: 20/03/2010 02:00:00

sensorID	timestamp	value
0	20/03/2010 01:03:25	23.837036
0	20/03/2010 01:08:28	23.82716
0	20/03/2010 01:13:31	23.74321
0	20/03/2010 01:18:34	23.787655
0	20/03/2010 01:23:37	23.792593
0	20/03/2010 01:28:40	23.701235
0	20/03/2010 01:33:43	23.730864
0	20/03/2010 01:38:45	23.7679
0	20/03/2010 01:43:48	23.804937
0	20/03/2010 01:48:51	23.804937
0	20/03/2010 01:53:54	23.790123
0	20/03/2010 01:58:57	23.77037

Figure 2.5: Application visualizing Temperature sensor data in a data table.

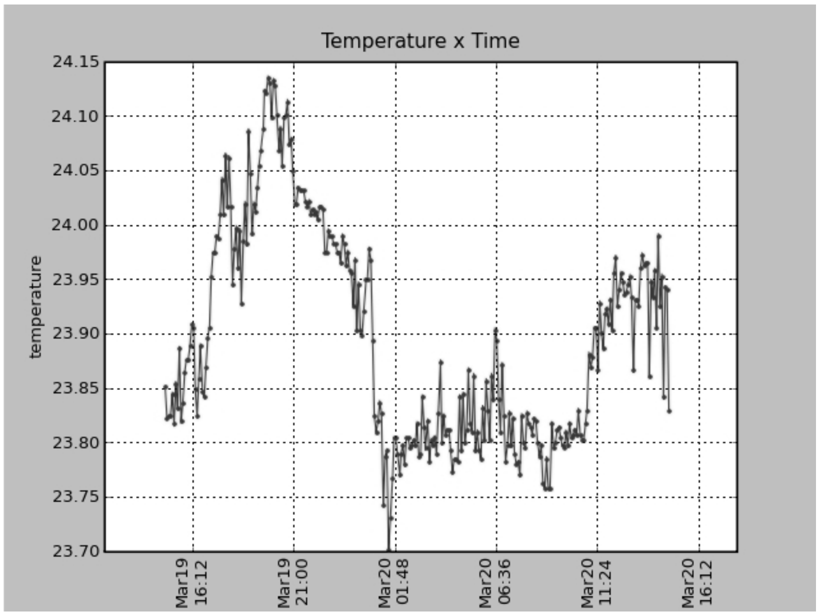


Figure 2.6: Another view of some of the data presented in Figure 2.5.

Web Services was one of the many solutions discussed for communication between layers. Again, service specification and implementation was time consuming. However, once the services were running and the network was deployed, the extension of access and visualization alternatives is proving to be relatively straightforward, because of the architectural solution adopted.

2.5 Publishing data in the cloud

To attest the flexibility of our approach, we conducted another experiment using cloud computing [81] – in our case, Microsoft Windows Azure [54]. This was accomplished in a very straightforward way: we just had to modify our Receive service to store the WSNs data into the local dabatase and into the cloud. In more detail, this, was performed as follows:

1. the data that came from the WSNs continued to flow to the local database in the same predetermined frequency (i.e., about every 2 minutes);

2. the data that came from the WSNs into the cloud were stored according to the real collection frequency of the WSNs (i.e., one or more data points per minute), without the need to reduce sampling frequency to save storage space

The applications querying the data from the data server infrastructure continued to access the Publication service, getting the data of interest without any need of code modification. At the same time, we developed another application, to query the cloud database and display the data. Again, this just required a simple access to the cloud service to get data from there.

Figures 2.7 and 2.8 show charts of temperature and humidity, respectively, from the cloud (a) and from the local database (b). The underlying databases are different (PostGreSQL in our server, SQL Server in the cloud) and the data volume is different (one or more values per minute in the cloud, one value per 2 minutes in our local server). However, at the visualization/application end, the graphs are presented at 5 minute average intervals, thereby showing the same curves. The final visualization is nevertheless different because two distinct rendering algorithms were used, one for each application.

Extending the problem to run in another platform was simple and straightforward, thanks to our infrastructure choices. We just had to update the Receive Service we provide for the clients to store data and create another application to access the cloud, for the same basic visualization. We point out that our goal was not to compare the performance of the two experiments (with and without the cloud). Rather, the idea here was to check the extensibility and flexibility of our solution.

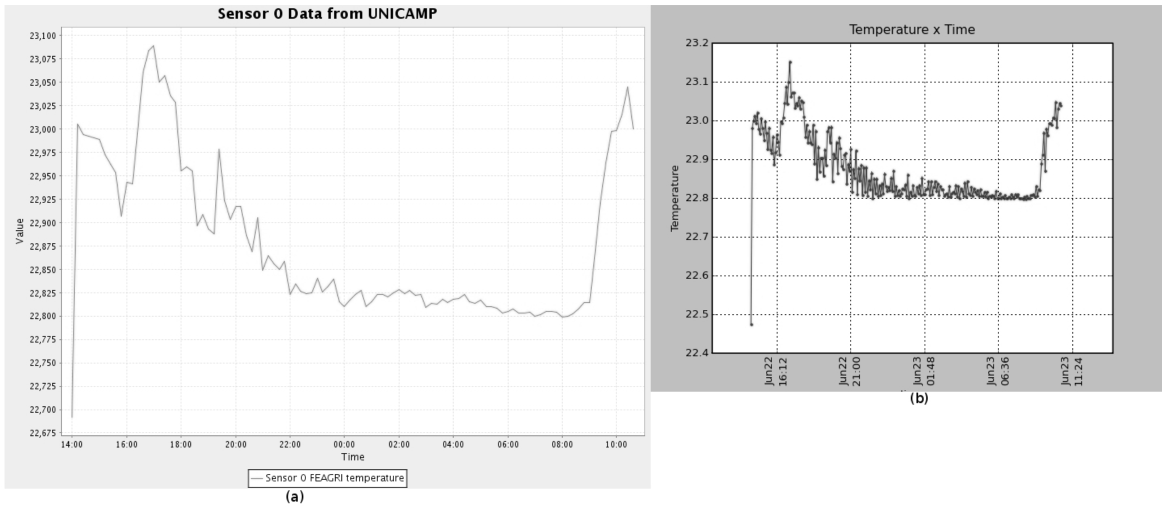


Figure 2.7: Temperature data in the cloud (a), and in the local database (b).

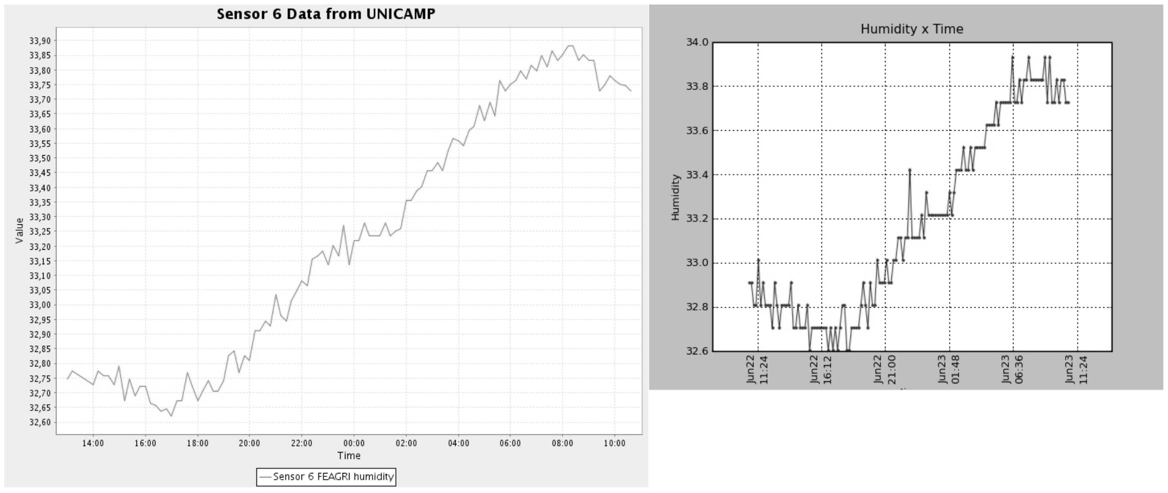


Figure 2.8: Humidity data in the cloud (a), and in the local database (b).

2.6 Conclusions and ongoing work

This paper presented our proposal to process and visualize sensor network data. Our approach is based on combining Web Services (to provide interoperability among sensor networks, data servers and user applications) and components (to provide flexibility in data preprocessing and visualization). Since OSGi was used as the component standard, we can dynamically update an application at execution time without the need to restart the entire application. New functionalities (components) can be added or removed without breaking the data flow from the WSN to the data server or restart the user visualization application. As a consequence, the features that are not involved in the update will not be affected and the new ones can be instantaneously run.

There are many directions for continuing this work, that range from solving issues such as detecting faulty sensors to providing users with a wide range of visualization and filtering options. Another direction involves extending the sensor data database with additional information – e.g. on quality – such as adding attributes discussed by [73].

At the same time, we need to concern ourselves with the Web Science issue of visibility. It is not enough to publish data on the Web: indeed, means must be found to ensure that these data are found and correctly interpreted. One possibility is to register the Publication Services, in which the registration is enhanced with enough semantic information – e.g., with ontological annotations. This kind of solution is part of our ongoing research.

Chapter 3

Managing Environmental Data from Sensor Networks

3.1 Introduction

There are several challenges in managing environmental data. Researchers have to deal with different types of data from many sources and captured with a wide range of spatial and temporal resolutions. To yield good research results, scientists have to overcome many issues such as heterogeneity and large volume of data.

Sensor networks are growing in importance, as data providers enable means to measure different types of environmental variables. However, they present several levels of heterogeneity, i.e. they can be produced by different manufacturers, having different sensor models that can make measurements at different time intervals.

Research in sensor networks treats problems under different perspectives. At the sensor layer, research concentrates in acquiring and processing data with the lowest energy consumption. At the network layer, main concerns involve routing of data through the best route considering node errors, low-level signal, etc. At the application layer, in which we are concerned, challenges involve means to acquire, integrate and assess data.

Our approach to manage sensor-based environmental data integrates these data sources using two mechanisms. The first is Enterprise Service Bus (ESB), a distributed infrastructure that uses messages and open standards to provide integration of systems [10]. The second consists in treating data provided as events, processed using Complex Event Processing (CEP) [18, 50]. CEP offers means to deal with rules and events' relationships, providing ways to create multi-layered architectures of events and their multiple sensor-based data.

This paper concentrates in proposing and exemplifying distinct event patterns to analyze geospatial environmental sensor data. The rest of the paper is organized as follows.

Section 3.2 provides some basic definitions we use throughout the paper. Section 3.3 mentions work that intersects ours. Section 3.4 presents some background of our framework for environmental data management detailed in [39]. Section 3.5 details the event patterns covered in the framework. Section 3.6 presents a short example involving environmental monitoring and pattern detection and finally, section 3.7 concludes the paper.

3.2 Basic Concepts

This section provides some of the concepts we use throughout the paper in order to clarify and standardize definitions.

An ESB is an infrastructure focused in integration of systems that communicate via messages. It provides routing, invocation, mediation and other capabilities to facilitate communication between systems. Therefore if a new system is added to this infrastructure it has only to provide an adapter that fits its message format to couple with other systems that are already coupled with the ESB. An adapter is a piece of software that provide connection between the provider/consumer and the ESB.

According to Rademakers and Dirksen [64] some of the core functionalities of an ESB include location transparency (the service consumer does not need to know where the provider is), transport protocol conversion (an ESB should be capable of converting different transport protocols), message transformation (e.g., from SOAP to an custom XML format), message routing (where the message comes from and where to send it), message enhancement (add additional data to the incoming message), security and monitoring and management.

An event is "a thing that happens, especially one of importance" [17]. It is a notable thing that has significance in a context in a system. In CEP, an event is an object signifying an activity with three aspects (Form, Significance and Relativity) that a computer can process. The *form* is the representation of the event (e.g., an object with its properties describing an event). The *significance* is the description of what an event signifies (e.g., a string with the event's description). The *relativity* is the relation of the event with others such as time period and causality (e.g. an array with the ids of other events that it aggregates) [49]. To process events, event processing agents (EPA) monitor a system's execution to detect patterns and process events.

A complex event is an aggregation of events [49]. It is created using event patterns rules or aggregation rules since they aggregate set of events. A complex event provides an aggregated event from among a cloud of low-level events.

Events can be matched by an *event pattern* that is a template that describes the event and all the appropriate context descriptions such as causal dependencies, timing, etc. Event Patterns can produce aggregations of events, creating a hierarchy with a sequence of levels. Each level has its own rules that specify how one can infer the higher layer events from lower level events. For instance, in environmental studies, an event may be a sudden drop in temperature. A more complex event, for instance ‘frost’, can be defined as a combination of ‘drop in temperature’ events within a window in space and time, and so on. CEP allows specifying (and checking for) such a hierarchy of events.

3.3 Related Work

There are several reports of deployment of sensor networks for monitoring and analysis of environmental data in the literature. Most of this research relates to the aforementioned sensor layer which deals with problems such as energy efficiency, reliability and capacity. Once sensor layer issues are solved, research tends to focus on application specific problems.

One example is the National Ecological Observatory Network (NEON) [38] which integrates a network of observatories to gather data on ecological biosphere. Despite the fact that they mention that their prototype systems are working towards event specification, detection and response capabilities [28], to the best of our knowledge, there is no evidence that they are dealing with events the same way as we do, i.e. providing means for researchers to insert event patterns and receive notifications when there is a match.

FLUXNET [5], another example, uses a network of micrometeorological towers to measure exchanges of CO2, water vapor, and energy between terrestrial ecosystems and the atmosphere. This network aims at integrating networks into a global effort.

Global Lake Ecological Observatory Network (GLEON) [25], yet another example, aims at the observation of lakes to understand the processes involved and effects of climate and land-usage change in a global fashion. They deal with sensor stream data management using the RBNB DataTurbine, which is a middleware that provides means to integrate heterogeneous instruments and services [77]. This middleware provides a set of features to accomplish sharing of datasets in real-time between different sites with authorized users. Although GLEON provides integration facilities, it does not focus in dealing with event processing. This approach is similar to our use of ESB for integration, but without considering pattern identification.

Consortium of Universities for the Advancement of Hydrologic Science, Inc. (CUAHSI) Hydrologic Information System (HIS) [15] provides an infrastructure that relies on service oriented architecture (SOA) for sharing hydrologic data, i.e. publication, discovery and access of hydrologic data. A desktop software that can be used to access, visualize and analyze hydrologic data published in CUAHSI called HydroDesktop [3] was also developed. While providing useful features to overcome challenges (e.g. integration and sharing of data sources), CUAHSI-HIS is only focused on Hydrological studies.

Hughes et. al. [34] developed and deployed a middleware system to support environmental monitoring based on wireless sensor network (WSN). Their software component infrastructure, named LooCI [33], provides event processing at the WSN level, where each node acts as a broker, consuming and producing events. In summary, their work deals with events and integrates WSN stream data at the sensor level (i.e., before leaving the network). We instead, are concerned with processing data only after they arrive at the framework, and perform event processing at a different application level. Our framework integrates stream and static data using a filtering/pre-processing phase before data enter the ESB. Once data are encapsulated into ESB messages, they are integrated and processed within the framework as events.

All these efforts concentrate on processing and integrating the data to allow the usage of the associated variables. Although they provide means to share datasets from their measurement sites, they do not provide means to deploy and detect generic patterns through these datasets, i.e. they do not provide ways to efficiently acquire only essential datasets/events for the researchers experiment.

While these initiatives treat data integration problems using database-like approaches, we are more concerned with the events captured by the sensors, as a step that follows integration processes. So, rather than only integrating the data streams, we process the streams looking for events that are reported to users, according to user requests. Related work that intersects ours in terms of event detection includes, for instance [18], [13], [27]. These papers provide good solutions in different aspects in dealing with sensor data, integration and events. On the other hand, they do not provide a solution capable of integrating static and dynamic types of data.

As will be seen next, we process sensor data in two different steps. First, using ESB, we integrate sensor data sources, but at the same time consider other kinds of data as integration inputs, therefore differentiating ourselves from intersection efforts such as NEON or FLUXNET.

Next, we treat these integrated data as sources of event streams, looking for patterns and generating higher level semantics. Again in this point, we distinguish our efforts from those that process events in sensor network data streams.

3.4 The Data Management Framework

Environmental sciences are among the research fields being transformed by the ability to retrieve observations at high spatial and temporal granularity through the deployment of in situ sensor network technologies [7]. However, to make sense of the data collected is not a trivial task. As outlined by [76], there are several challenges in this context due to data complexity, spatial and temporal context sensitivity, data heterogeneity and volume, among others.

Environmental studies usually involve spatiotemporal factors which present several other challenges. Our proposal is based on the following premises: (a) sensor data are treated as event streams, and hence data of interest can be processed as events; (b) heterogeneous streams can be jointly processed using an ESB platform, which also enables the inclusion of data from arbitrary (non-sensor based) sources.

Figure 3.1, adapted from [39], shows an overview of the framework. It uses ESB and CEP to facilitate the integration of data providers and consumers. Data from providers and to consumers are input and output, using ESB adapters, i.e. a piece of software that provides connection between the provider (or consumer) and the ESB. Once providers are connected via ESB into our framework, they can input data. These data enter the framework and are encapsulated into a standard event format, defined by us. Data can go through filtering, routing, translation (to the standard event format), transformation, aggregation and event pattern detection. At the end, data are translated back into formats expected by consumers and disseminated to them.

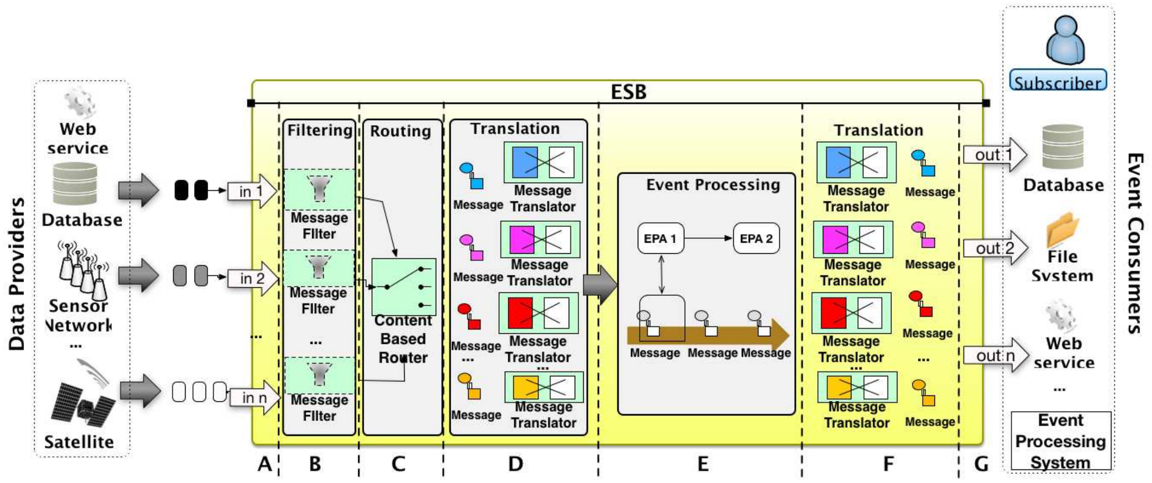


Figure 3.1: Architecture of the framework

A key issue in our framework is the possibility to detect patterns. Patterns can help acquire knowledge about environmental observations and can be tailored to distinct user needs. In our framework, patterns are described in Event Processing Language (EPL), deployed and processed using an Event Processing Engine called Esper [20]. For details on issues concerning the ESB and framework internals, the reader is referred to [39]. This paper emphasizes the management of environmental data and patterns.

Another key issue is that since this framework is based on events and presents a homogeneous input (provider) and output (consumer) interface. Other solutions may input events, but do not necessarily output events.

Finally, ESB provides location transparency, i.e., the consumer does not need to worry about where the producer is. This provides independence of consumers from producers, which is another advantage of this proposal.

3.5 Environmental Data and Event Patterns

3.5.1 Overview and basic definitions

In our work events are represented as event objects which are instances of event types. Event types are abstract representation of real instances of an event. For example, consider a stream of measurements coming from a temperature sensor. All the measurements reported share the same kind of information structure which can be defined as an event type [21]. In this paper, we use the term event object and event as synonyms. As will be seen, our events are encapsulated in Java objects.

We denote an event as a quadruple: <*measured-value*, *nature*, *spatial-variable*, *timestamp*> = <v, n, s, t>, where:

- v is the value of the variable captured (e.g. 40 [degrees celsius], 80% [humidity]);
- n the nature of the variable (e.g. temperature, humidity);
- s is the location (coordinates) of the measurement;
- t is a time interval [ts, tf][1] for which v was valid at that location (s). If ts = tf this is an instantaneous event (e.g., sensor measurement).

An *event pattern*, *pattern* for short, here, is the specification of one or more events concerning the variables being monitored. A more general pattern definition that is suitable to our work is: a description of how an event or a set of events of interest looks like [49].

A pattern can involve a single variable – e.g., precipitation – at a given location or set of locations, in a given moment or period of time. It may also involve more than one variable – e.g., precipitation, temperature and solar radiation – again with varying spatio-temporal attributes. Pattern detection can serve to alert about given conditions, or point out variations in collections of events. Thus, looking for patterns enables a scientist to detect some combinations of events occurring in a particular order, or if a particular event did not emerge when/where it was expected to occur.

[1]ts is the start time and tf is the final or end time.

Pattern elements Patterns are expressed as combinations of predicates over value, nature, space and time. We use the term predicate as the property that a subject can have [66], i.e. if we have a statement like "$x > 5$", the variable x is the subject while the "is greater than 5" is the predicate. Once a value is assigned to a variable, e.g. $x = 10$, it becomes a proposition and has a truth value (*true* or *false*). The proposition will have to return *true* in order to match a predicate.

The notation used for predicates that forms a pattern is: V(x), N(x), S(x), T(x) applied to respectively the value, the nature, the space and the time component of event x.

3.5.2 Event Patterns

Patterns can be simple or composite. Simple ones are patterns that concern one event and may be of type Value, Spatial, Temporal and Spatiotemporal. Value patterns only consider the value in the predicate. Spatial patterns consider values over a spatial distribution. Temporal patterns focus in variations of values in time. Spatiotemporal patterns regard variations in value over time and spatial distributions. Composite patterns comprise combinations of two or more simple patterns. Patterns are run against sliding windows defined by the users.

Here we present a description and the notation of these patterns.

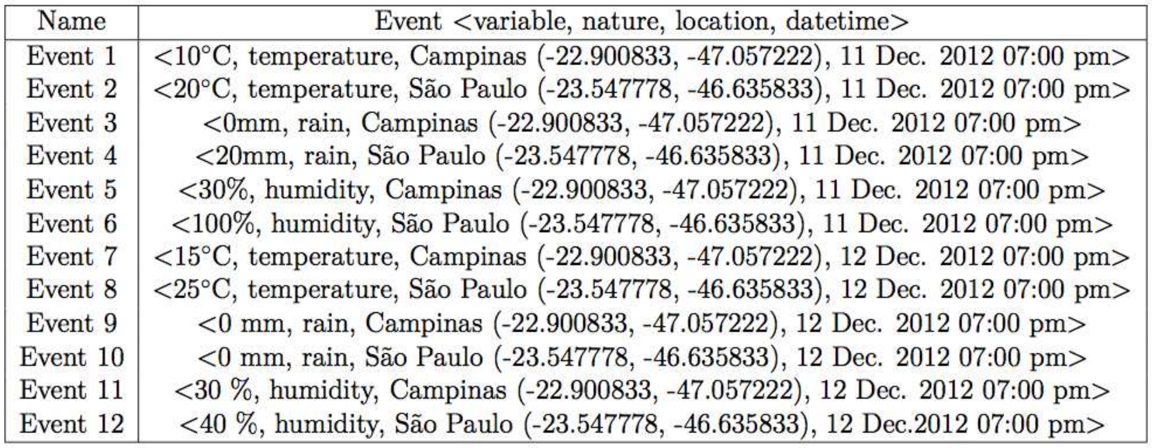
Some examples are given in order to subsequently present how they detect events. The examples concern Events 1 through 6 which provide measurements of temperature, rain and humidity in the cities of Campinas and São Paulo, at date and time 11 December 2012 07:00 pm, 12 December 2012 07:00 pm, 13 December 2012 08:00 am and 13 December 2013 09:00 am depicted in Table 3.5.2. In what follows, the notation $x.v, x.n, x.s, x.t$ refers respectively to the value, nature, space and time components of event x. The spatial component contains the name and coordinates of the city.

Value pattern

In value patterns, predicates consider only the value of an environmental variable measured by a sensor, weather station, etc (for example: temperature, humidity, light, solar radiation, among others). Consider patterns:

- $P_1(x)$: $[x.v > 10°\text{C} \wedge x.n \;\epsilon$ temperature$]$ – detects temperature above 10°C
- $P_2(x_1, x_2)$: $[x_1.v > 10°\text{C} \wedge x_1.n \;\epsilon$ temperature $\vee\; x_2.v \geq 90\% \wedge x_2.n \;\epsilon$ humidity$]$ – detects temperature above 10°C or humidity above 90% regardless of time or space variables

Table 3.1: Event Examples



Name	Event <variable, nature, location, datetime>
Event 1	<10°C, temperature, Campinas (-22.900833, -47.057222), 11 Dec. 2012 07:00 pm>
Event 2	<20°C, temperature, São Paulo (-23.547778, -46.635833), 11 Dec. 2012 07:00 pm>
Event 3	<0mm, rain, Campinas (-22.900833, -47.057222), 11 Dec. 2012 07:00 pm>
Event 4	<20mm, rain, São Paulo (-23.547778, -46.635833), 11 Dec. 2012 07:00 pm>
Event 5	<30%, humidity, Campinas (-22.900833, -47.057222), 11 Dec. 2012 07:00 pm>
Event 6	<100%, humidity, São Paulo (-23.547778, -46.635833), 11 Dec. 2012 07:00 pm>
Event 7	<15°C, temperature, Campinas (-22.900833, -47.057222), 12 Dec. 2012 07:00 pm>
Event 8	<25°C, temperature, São Paulo (-23.547778, -46.635833), 12 Dec. 2012 07:00 pm>
Event 9	<0 mm, rain, Campinas (-22.900833, -47.057222), 12 Dec. 2012 07:00 pm>
Event 10	<0 mm, rain, São Paulo (-23.547778, -46.635833), 12 Dec. 2012 07:00 pm>
Event 11	<30 %, humidity, Campinas (-22.900833, -47.057222), 12 Dec. 2012 07:00 pm>
Event 12	<40 %, humidity, São Paulo (-23.547778, -46.635833), 12 Dec.2012 07:00 pm>

If applied to the set of events of Table 3.5.2, pattern P_1 will match events 2, 7 and 8. Pattern P_2 will match events 2, 7, 8 and 6.

Spatial patterns

Spatial patterns are based on the spatial properties of events, considering the value of environmental variables given a spatial context. For example, temperature measured in a place near the city of Campinas. Consider patterns:

- $P_1(x)$: $[x.v > 10°\mathrm{C} \wedge x.n \in$ temperature $\wedge\ x.s =$ Campinas $]$ – detects temperature above 10°C in the city of Campinas

- $P_2(x_1, x_2)$: $[x_1.v > 10°\mathrm{C} \wedge x_1.n \in$ temperature $\wedge\ x_1.s =$ Campinas $\wedge\ x_2.v > 20°\mathrm{C} \wedge x_2.n \in$ temperature $\wedge\ x_2.s =$ São Paulo$]$ – detects temperature above 10°C in Campinas and temperature above 20°C in São Paulo.

If applied to the set of events of Table 3.5.2, pattern P_1 will match event 7. Pattern P_2 will match Event 7 and Event 8.

Temporal patterns

Temporal patterns are the ones that are concerned with variation of events in time, i.e. when time plays a major role [21]. They consider the value of environmental variables and temporal assumptions about them. They can be applied, for instance, after filtering data from a certain region (e.g., using a spatial pattern). For example, temperature measured in the last decade or rain rate in the last month. Consider patterns:

- $P_1(x)$: [$x.v$ = 10°C $\wedge$ $x.n$ ϵ temperature $\wedge$ $x.t$ < 12 December 2012] – detects temperature equals 10°C when date is before 12 December 2012
- $P_2(x_1, x_2)$: [$x_1.v$ > 10 mm $\wedge$ $x_1.n$ ϵ rainfall $\wedge$ $x_1.t$ = 11 December 2012 $\vee$ $x_2.v$ <= 0 mm $\wedge$ $x_2.n$ ϵ rainfall $\wedge$ $x_2.t$ = 12 December 2012] – detects rain above 10 mm when date and time is equals 11 December 2012 and rain less than or equals 0 mm when date and time is equal 12 December 2012.

If applied to the set of events of Table 3.5.2, pattern P_1 will match Event 1, and pattern P_2 will match Event 4, Event 9 and Event 10.

Spatiotemporal patterns

Spatiotemporal patterns are those specific to the evaluation of spatial trends over time [21]. They combine measured variables, spatial and time properties. For example, rain measured in the last decade in the state of São Paulo. Consider patterns:

- $P_1(x)$: [$x.v$ = 10°C $\wedge$ $x.n$ ϵ temperature $\wedge$ $x.s$ = *Campinas* $\wedge$ $x.t$ = 11 December 2012 07:00 pm] – detects temperature equals 10°C in the city of Campinas when date and time is equals 11 December 2012 07:00 pm
- $P_2(x_1, x_2)$: [$x_1.v$ > 8°C $\wedge$ $x_1.n$ ϵ temperature $\wedge$ $x_1.s$ = Campinas $\wedge$ $x_1.t$ = 11 December 2012 $\wedge$ $x_2.v$ > 0 mm $\wedge$ $x_2.n$ ϵ rain $\wedge$ $x_2.s$=São Paulo $\wedge$ $x_2.t$ = 11 December 2012] – detects temperature above 8°C in Campinas while there is rain in São Paulo in the same day.

If applied to the set of events of Table 3.5.2, pattern P_1 will match Event 1, and pattern P_2 will match Event 1 and Event 4.

All the aforementioned patterns execute considering a sliding windows defined by pattern creator. Thus, if a scientist wants to detect environmental patterns considering time, the more appropriate sliding window will be time-based, or if the number of events are more important, an event-number-based window will be more appropriate.

As outlined in section 3.4, there is a possibility to combine data from different sources and detect patterns from them. Since patterns detailed here are generic, i.e. they are not tied to a specific data type or source, they can be used to detect variations in different types of environmental variables that comes from stream and non-stream sources. This combination can produce different correlations, e.g. given a region where we usually find some meteorological measurements, produce a notification event when measurements diverge from the expected value by a given factor.

3.6 Example of Pattern Detection

Floods are one of the most frequent natural disasters that cause damage worldwide. Flood-prediction is not a trivial task [46].

Suppose that a given region, e.g. a national park or a protected area (PA), is being monitored as concerns environmental variables temperature, humidity and rainfall so that for some given specific conditions (e.g. flood) animals in that region may be due to special treatment (e.g. be evacuated). Event processing occurs in two levels:

- filtering – forward to pattern matching only data from rain sensors within that region
- pattern matching – find a sequence of rain sensor events that, over a period of time defined by researchers indicate that a flood could occur

The spatiotemporal pattern that follows assume the sensor data were already selected by the filtering phase, e.g. using a buffer operator around the region and type of measurement.

False positives should not be tolerated, since an evacuation of an area can be costly. Thus, we insert 2 rain (x_1 and x_2) and 2 river level (x_3 and x_4) predicates in the pattern to be deployed in four different regions of a polygon comprising the region to be protected. Then, the pattern to be considered is the following:

$P(x_1, x_2, x_3, x_4)$: $[x_1.v > \mathit{threshold\_rain} \wedge x_1.n \in \text{rain}, x_2.v > \mathit{threshold\_rain} \wedge x_2.n \in \text{rain}, \text{dist}(x_1.s, x_2.s) > 1 \text{ km}, x_3.v > \mathit{threshold\_river\_level} \wedge x_3.n \in \text{river\_level}, x_4.v > \mathit{threshold\_river\_level} \wedge x_4.n \in \text{river\_level}\ ]$

This pattern takes 2 rain sensors and 2 river level sensors that can inform if the levels of rain or river go above a given threshold (*threshold_rain* and *threshold_river_level*, user-given). The distance between x_1 and x_2 was considered to be above 1 km. If this pattern is evaluated to true, a flood event is generated and an alert is triggered.

We know that in a real operation scenario a flood alert has to be calculated using a combination of affected population, vulnerability and the size of the flood to provide a risk score [46]. However, in this example, we show the use of patterns to provide the required data in near real-time, combining user-given values with rain and river level stream data. This approach enables researchers to define patterns and receive notifications as fast as data are being received.

3.7 Conclusions and Future Work

This paper presented our work in specifying patterns of events to capture specific environmental conditions from sensor networks in space and time. Patterns are used to construct queries that are posed against sensor data that is collected, standardized and stored in our framework. This work needs to be extended with other case studies, e.g. investigating more adapters.

We are investigating extending this work to composite patterns over sensor data combined with data on species observations (e.g., from collection catalogues). In environmental monitoring studies, it is possible, for example, to model regions where an endangered or rare specimen may occur by looking at the specimen occurrence database and determine if a weather condition pattern is met when applied to other regions. Thus, integrating species occurrence databases and meteorological databases can point out important correlations such as detection of the weather conditions where a specimen occurred or detect regions likely to find a species given spatiotemporal patterns.

Chapter 4

Patterns in Environmental Event Processing

4.1 Introduction

Environmental monitoring applications allow scientists to detect otherwise unobserved situations, which can be of interest to their studies [26]. These applications support observations in spatial and/or temporal granularities not available using static or historical sources. However, such dynamic data have to be combined with static sources.

If these applications open new possibilities in research, scientists on the other hand have to deal with many obstacles concerning the quality and management of the data used (and produced by) their studies. The data management cycle (e.g. acquisition, analysis and storage) presents many challenges, e.g., [61].

In acquiring data, the scientist has to deal with different types of data from various sources. While many studies are concerned with static data sources (e.g. historical files), there is an increasing interest in adopting dynamic sources (typically from sensors). As sensors become popular due to many reasons (e.g. lowering prices), data providers are increasingly publishing real-time stream data; this aggravates the issues of heterogeneity.

To analyze data, scientists have to select the kinds of data they want to treat, and aggregate the different data sources. Since there are intrinsic heterogeneity issues and static and stream data involved, this type of task increases in complexity as the number of data sources grows.

Storage is yet another concern, since research data does not fit completely in one type of format (e.g. relational database table or spreadsheet) or media. Stream data are often useful for a short period of time, but storing this type of data may also be useful for forecasting and long term analysis.

There is plenty of work that deals with environmental data and that handles the issues of their capture, analysis and storage such as [3, 9, 11, 71, 72]. In acquiring data, such research aims at sampling, data routing, reliable communication and energy-efficient acquisition of data. In analyzing data, the focus is in data exploration, classification, statistics. In storing data, research concentrates in the capacity of data storage and search in data repositories.

An increasing number of environmental studies must cope with static and dynamic data. Solutions are geared towards specific problems, and particular data sources. Typically, each study is directed to a given geographic region, having a specific focus (e.g., analyzing drought effects). Once region and focus are defined, researchers specify the data sources of interest and provide an approach that takes advantage of the characteristics of these sources (e.g. in [4], satellite images, soil water and radiation sensors were combined to evaluate drought stress and carbon uptake in a specific region of the Amazon forest). Even when they manage to integrate static and dynamic sources, these approaches are seldom scalable to other regions, or phenomena, or other kinds of data sources. Our approach, instead, is generic. To the best of our knowledge, there is no generic approach for joint integration of stream and static data for environmental sciences.

The generic nature of our solution is supported by the combination of two technologies:

- On the capture side, our proposal integrates data sources using Enterprise Service Bus (ESB), a combined infrastructure that provides integration between systems [10, 70]. Static and stream data sources are integrated using so-called ESB adapters. Data are materialized in messages and treated as events, using Complex Event Processing (CEP) [18, 50] which provides means to deal with rules and events' causal relationships. We thereby provide ways to create a multi-layered architecture of events, corresponding to multiple kinds of data aggregations;

- On the analysis perspective, it allows identification of patterns of interest, using an event-based paradigm. Once patterns are detected, we use ESB adapters to disseminate the results to different destinations. Ingested data are moreover maintained in an object-relational database for subsequent processing of historical data and patterns that compare the present to the past.

This paper shows how our framework deals with the aforementioned problems and how it can help scientists in managing their environmental data to extract useful information. The rest of the paper is organized as follows. Section 4.2 provides some basic definitions we use throughout the paper. Section 4.3 provides some related work that deals with environmental data. Section 4.4 presents an overview of our framework for environmental data management. This framework is detailed in [39]. Section 4.5 provides examples of use of event patterns. Section 4.6 presents our implementation showing the practical use of our framework and patterns, and section 4.7 concludes the paper.

4.2 Basic Concepts

This section provides some of the concepts we use throughout the paper in order to clarify and standardize definitions.

An ESB is an infrastructure focused in integration of systems that communicate via messages. It provides routing, invocation, mediation and other capabilities to facilitate communication between systems. Therefore if a new system is added to this infrastructure it has only to provide an adapter that fits its message format to couple with other systems that are already coupled with the ESB. An adapter is a piece of software that provide connection between the provider/consumer and the ESB.

According to Rademakers and Dirksen [64] some of the core functionalities of an ESB include location transparency (the service consumer does not need to know where the provider is), transport protocol conversion (an ESB should be capable of converting different transport protocols), message transformation (e.g., from SOAP to an custom XML format), message routing (where the message comes from and where to send it), message enhancement (add additional data to the incoming message), security and monitoring and management.

An event is "a thing that happens, especially one of importance" [17]. It is a notable thing that has significance in a context in a system. In CEP, an event is an object signifying an activity with three aspects (Form, Significance and Relativity) that a computer can process. The *form* is the representation of the event (e.g., an object with its properties describing an event). The *significance* is the description of what an event signifies (e.g., a string with the event's description). The *relativity* is the relation of the event with others such as time period and causality (e.g., an array with the ids of other events that it aggregates) [49]. To process events, event processing agents (EPA) monitor a system's execution to detect patterns and process events.

A complex event is an aggregation of events [49]. It is created using event patterns rules or aggregation rules since they aggregate set of events. A complex event provides an aggregated event from among a cloud of low-level events.

Events can be matched by an *event pattern* that is a template that describes the event and all the appropriate context descriptions such as causal dependencies, timing, etc. Event Patterns can produce aggregations of events, creating a hierarchy with a sequence of levels. Each level has its own rules that specify how one can infer the higher layer events from lower level events. For instance, in environmental studies, an event may be a sudden drop in temperature. A more complex event, for instance 'frost', can be defined as a combination of 'drop in temperature' events within a window in space and time, and so on. CEP allows specifying (and checking for) such a hierarchy of events.

4.3 Related Work

There are countless studies that involve static data sources in environmental sciences (see, for instance, several papers in EOLSS [80]). They concern issues that cover the entire data life cycle and distinct analyses and models that such data feed. We concentrate on related work in sensor streams, a relatively more recent research domain. We recall that our approach allows considering both kinds of data within a single framework.

As data streams are becoming more and more popular on the Web, lots of work in processing and publishing environmental data have appeared. Platforms such as Cosm [14] facilitate sharing data streams on the web.

This, in turn, has prompted research on detecting errors in the stream. An event may contain faulty information (e.g., due to sensor malfunctioning), and a sequence of such events may lead experts to wrong conclusions. Conversely, correct values may be treated as errors (e.g., outliers when extreme conditions actually occur). For instance, the work of Gupchup [26] comments that 45% of sensor measurements are misclassified as faults. That work also shows that simply tuning fault-processing algorithms is not enough, since tuning may inversely not recognize actual errors. Our work assumes that stream errors are treated before entering the ESB. We point out that our emphasis is on providing users with tools to detect events.

While the above concern the quality of stream data, other researchers deal with specific kinds of data items. For instance, research conducted by Rundel [67] provides many examples of ecological data acquisition using terrestrial, soil and aquatic sensor networks. It points out that integration of sensor data can reveal previously unobserved phenomena.

In the same spirit, Hart and Martinez [29] provide more than 40 different types of environmental sensor network deployments. They point out that environmental sensor networks, i.e. sensor networks specifically tuned to an environmental application, will be key to provide new approaches in the study of environmental processes. However, there are some problems to overcome in dealing with automatic data gathering such as different sources of data and formats. Several efforts are concerned with these problems such as Open Archives Initiative[1] [47], Geographic Markup Language (GML) [63], Sensor Model Language (SensorML) [8], Ecological Metadata Language (EML) [22]. They all are creating standards to facilitate the data exchange, integration and interoperability between systems.

Consortium of Universities for the Advancement of Hydrologic Science, Inc. (CUAHSI) Hydrologic Information System (HIS) [15] provides an infrastructure that relies on service oriented architecture (SOA) for sharing hydrologic data, i.e. publication, discovery and access of hydrologic data. A desktop software that can be used to access, visualize and

[1] http://www.openarchives.org

analyze hydrologic data published in CUAHSI called HydroDesktop [3] was also developed. While providing useful features to overcome challenges (e.g. integration and sharing of data sources), CUAHSI-HIS is only focused on Hydrological studies.

Hughes et. al. [34] developed and deployed a middleware system to support environmental monitoring based on wireless sensor network (WSN). Their software component infrastructure, named LooCI [33], provides event processing at the WSN level, where each node acts as a broker, consuming and producing events. In summary, their work deals with events and integrates WSN stream data at the sensor level (i.e., before leaving the network). We instead, are concerned with processing data only after they arrive at the framework, and perform event processing at a different application level. Our framework integrates stream and static data using a filtering/pre-processing phase before data enter the ESB. Once data are encapsulated into ESB messages, they are integrated and processed within the framework as events.

Similar to these studies, we are interested in the integration and processing of data from distinct kinds of sources, but unlike them, we are also concerned in detecting event patterns from static and stream data.

A few papers explore non-standard kinds of sensors. One such example of event processing for environmental monitoring is the work of Sakaki et al. [68]. In this work, they developed a system using events of Twitter messages to detect earthquakes with high probability and much faster than Japan Meteorological Agency (JMA). They consider a Twitter user as a sensor and process the message to only take relevant earthquakes notifications into account. This work has similarities with ours considering the use of event detection, but on the other hand they do not provide combination of events or stream data with other sources of data, i.e. possibility to access Web Services, files, databases like us.

4.4 The Data Management Framework

Environmental studies usually involve spatiotemporal factors which present several challenges, including spatial and temporal variability. Our work is specifically concerned with data integration and pattern detection, i.e. it identifies patterns out of data that are fed into our framework from different sources. Our proposal is based on the following premises: (a) data from heterogeneous sources are absorbed by our framework, thanks to the ESB features; (b) data are filtered with the focus to retrieve only interesting data; (c) all such data are then treated as event streams, and hence events of interest can be processed using event patterns.

Figure 4.1 shows an overview of the framework that we have implemented, where data flow from the bottom (providers) to the top (consumers) being encapsulated into messages and treated by CEP in the middle. It uses ESB to facilitate the integration of data providers and consumers. To use ESB in environmental studies, it is important to have a pre-processing phase since ESB are usually applied to enterprise applications where data granularity is not usually as diverse as found in this context. In our approach, a pre-processing phase consists in ingesting and extracting interesting data to be submitted to event pattern detection.

Data from providers and to consumers are input and output, using ESB adapters. Once providers are connected via the ESB into our framework, the framework can absorb data (1) by pushing or pulling data, i.e. in (1) our framework acts as a client by receiving data from providers or continually requesting new data.

At the same time data are collected, filtering (2) is required before data enter the ESB, since non relevant data could flood our framework unnecessarily. This filtering consists in selecting data sources and the items of interest from such sources. We may, for example, extract moisture and temperature from a set of meteorological stations, recover just a piece (region) of a satellite image or issue a query to retrieve data for the year 2012 from a database. This is particularly interesting to tailor the framework to specific contexts. For instance in environmental applications, it is usual to consider temporal series of satellite images, where just part of each image may be needed. This initial preprocessing would save data traffic within the ESB. As well, this phase can eliminate data that do not meet minimum quality requirements.

After the pre-processing, data are encapsulated into messages (3) by channel adapters. Here is where the data are standardized to flow through the ESB. In our framework, these messages are presented in Java objects. Temperature and humidity data are transformed into real numbers, satellite images can become descriptors, link to the image file and variable values, result of queries to databases can be transformed into a set of numbers and strings.

Once data are available in the ESB, it is necessary to choose the data to be processed (4) by the event patterns, i.e. for each type of study, scientists are interested in a certain type of data and will create event patterns considering the messages they want to observe. For example, given all messages flowing in the ESB, a scientist may be interested only in temperature data from a local sensor network deployed in a given region. For event detection, each ESB message is treated as an event and submitted to CEP mechanisms.

Event patterns are created and implemented in EPAs. Each EPA will evaluate events and trigger a new event if the event set satisfies the pattern. Patterns can help identify distinct events in environmental observations. Once an expert specifies a pattern, the framework is able to detect it and perform some action, e.g., notify user(s) or store data for later analysis. In our framework, patterns are described in the Event Processing Language (EPL) [19], deployed and processed using the Esper [20] Event Processing Engine.

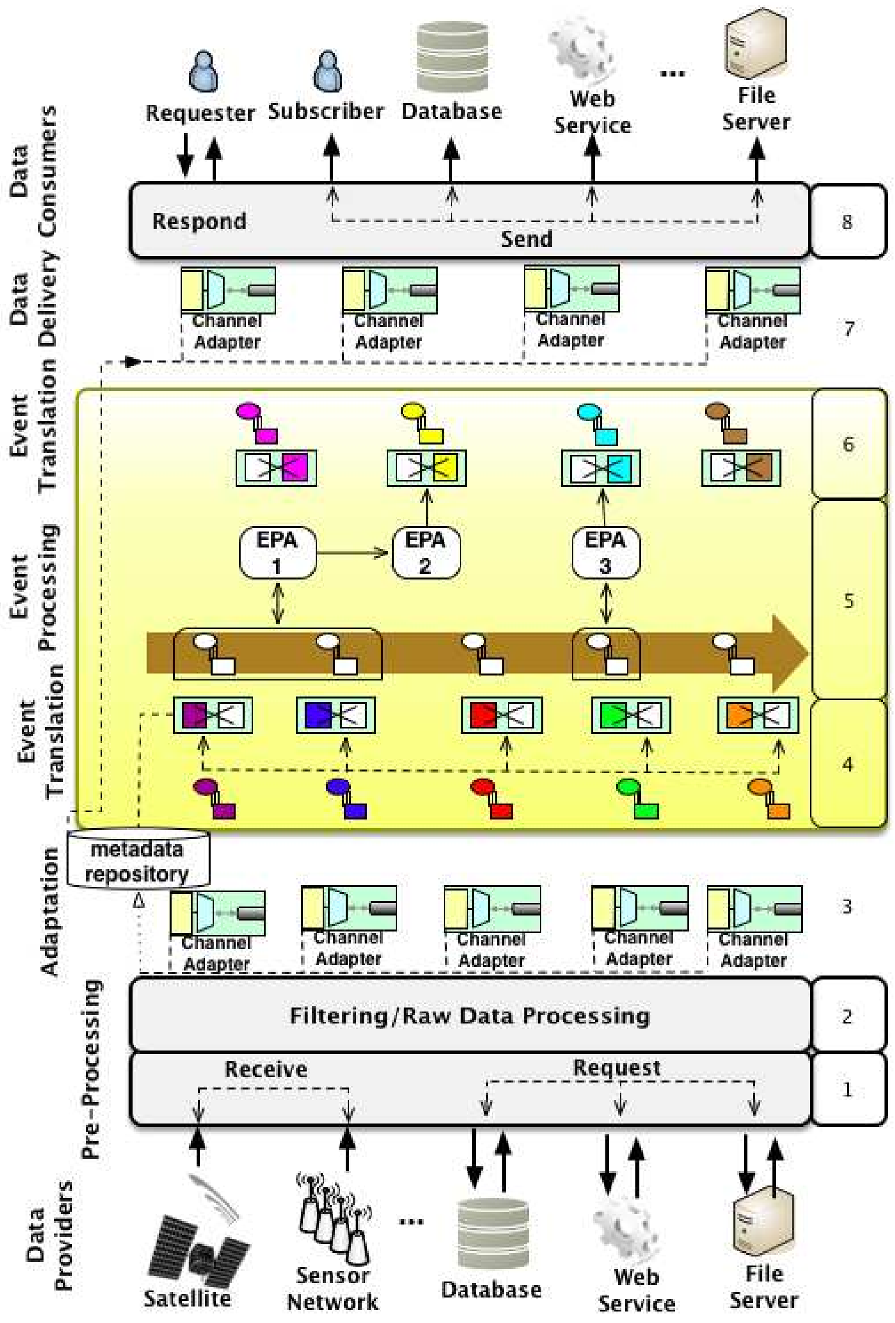


Figure 4.1: Architecture of the framework.

After the translation of messages into events (4), event pattern detection is performed in (5). Once an event pattern is found, an event is produced and encapsulated in a new ESB message (6). The ESB then takes care of sending this encapsulated event to interested users (7), e.g. scientist A or B.

In order to notify users or store events (8), the output can occur in two ways: notification or request/reply. Events are sent to subscribed users (e.g. Scientists) to receive notification or sent for storage in databases, file servers, etc. Subsequently, other user can make a request for these stored data. This request goes through the ESB and recovers the repository where the data lies.

In environmental studies, it is possible that two scientists are interested in studying certain types of variables in different ways. They can use our framework defining different events and event patterns they want to detect, e.g. scientist A is interested in detecting rain and temperature below 10 degrees celsius from a local sensor network deployed in region R_1 whereas scientist B is interested in receiving a piece of a satellite image deployed in region R_2 when temperature sensors from the region R_2 are above 20 degrees celsius. Scientist A has to consider messages flowing in the ESB of type rain and temperature from the sensor network deployed in region R_1 and create an event pattern to detect temperature < 10 and rain = true. Scientist B will have to consider messages created from pieces of satellite images and measurements from temperature sensors deployed in region R_2 and specify an event pattern to detect when those temperature sensors are above 20 degrees celsius.

We point out that we do consider factors such as data quality, or degree of fuzziness of spatial information. We are aware that these are very relevant issues to be taken into account when processing environmental data. For the purpose of this work, and its implementation, the following hypotheses are made:

- data quality can be inferred from metadata associated with data sources (e.g., spatial coverage, precision, accuracy and others), and only data items that meet a certain quality level are allowed through the filtering phase;
- fuzziness can be partially dealt with EPAs, using some of the operations on events proposed in [21] in particular translation[2]. The analysis and processing of fuzzy environmental data is in itself an open problem, and a thorough treatment of this issue is outside of the scope of this research.

[2]Other EPA operations include, among others, project, aggregate, and compose

4.5 Environmental Patterns

Patterns are used to detect significant variations or behaviors in a particular type of study. This section exemplifies some of the environmental patterns described in [41]. In that paper, we specify generic patterns that can be combined and extended to capture arbitrary combinations of events. Patterns are specified against data, which are basically treated as streams of events that take into account spatio-temporal nature of environmental data. Events have the form:

$<$*measured-value*, *nature*, *spatial-variable*, *timestamp*$>$ = $<v, n, s, t>$, where:

- v is the value of the variable captured (e.g. 40 [degrees celsius], 80% [humidity]);
- n the nature of the variable (e.g. temperature, humidity);
- s is the location (coordinates) of the measurement;
- t is a time interval [ts, tf][3] for which v was valid at that location (s). If ts = tf this is an instantaneous event (e.g., sensor measurement).

Values can be retrieved in different ways. Numerical flows from meteorological stations (e.g. temperature and humidity readings) are encapsulated in integer or double values; stored data can be retrieved from databases, e.g. information on species. Information can also be extracted from satellite images, which are cut in polygons and transformed into values of radiation or calculated indices such as Normalized Difference Vegetation Index (NDVI)[4]. These values can be taken in several different ways, i.e. from the pixel at the centroid of the polygon, an average of all pixel values inside the polygon, the maximum or minimum value of the pixels in the polygon among other possibilities that are application specific. If it is possible, a link to the original file or image is kept in order to trace provenance.

We consider two types of patterns: simple and composite. The former denotes patterns for one event, the latter are patterns comprising combinations of events. We also classify event patterns as Value, Spatial, Temporal and Spatiotemporal patterns. Value patterns only consider the value in the predicate. Spatial patterns consider values of predicates over a spatial distribution. Temporal patterns focus in variations of predicates in time. Spatiotemporal patterns combine all of the aforementioned patterns, i.e. combine measured variables, spatial and time properties.

Let us now exemplify each kind of pattern using an example. We point out that we write the patterns, in this section, in an informal logic-like rule language, for readability

[3]ts is the start time and tf is the final or end time.

[4]NDVI – helps in assessing the level of green vegetation using spectral reflectance measurements acquired in the visible and near-infrared regions.

sake. Our actual patterns are implemented in EPL, an SQL-like domain specific language provided by Esper – see section 4.6.

Many kinds of frogs only croak when it rains and spawn their eggs in temporary ponds[5]. Rain is not the only necessary condition for them to reproduce, the ponds must have enough water. In other words, rain must fall in sufficient volume to create a pond. There is a need to know how many inches of rain in an area need to fall for ponds to form. This varies with soil type (in clay, it would be easier and in sand it would be more difficult).

Estimating soil wetness is not a trivial task. There are techniques to use remote sensing to compute soil moisture such as the work of Jackson [36]. He showed that it is possible to use passive microwave remote sensing to measure soil moisture, parameterizing an algorithm with surface, type of soil, vegetation indices, etc. The work of [82] showed that in a geographically homogeneous region, it is feasible to compute soil moisture from scanning multichannel microwave radiometer (SMMR) considering some restrictions such as dense vegetation and extreme hydrological conditions.

Given these facts, we can adopt some premises to derive a pattern: (1) soil moisture is computed from satellite images, (2) rainfall measurements come from weather stations, (3) frog species and their information come from a database. Let us assume the following composite pattern P for frog reproduction:

$P_{frog\_reproduction}(sm, rain)$:
[$sm.v > 80\% \wedge sm.n \in$ soil moisture $\wedge$ November $< sm.t <$ January $\wedge$ $s.v$ in southeast of Brazil $\wedge$
$rain.v =$ true $\wedge$ $rain.n \in$ rainfall $\wedge$ November $< rain.t <$ January $\wedge$ $rain.s$ in southeast of Brazil]

where sm is soil moisture, $rain$ is rainfall, t is the month timestamp and s is region location.

Since this pattern has value, space and time predicates, it is a spatiotemporal pattern that considers a moisture event (sm), a rain event ($rain$), ranging over a time period t in a region s. From this composite pattern we can derive the other simple types of pattern:

- value patterns – $P_{frog\_reproduction}(sm)$: [$sm.v > 80\% \wedge sm.n \in$ soil moisture]; $P_{frog\_reproduction}(rain)$: [$rain.v =$ true $\wedge$ $rain.n \in$ rainfall]
- spatial pattern – $P_{frog\_reproduction}(rain)$: [$rain.s$ in southeast of Brazil]
- temporal pattern – $P_{frog\_reproduction}(sm)$: [$November < sm.t < January$]

[5]We thank Dr. Felipe Toledo, from the Institute of Biology, for this and other examples.

In the event processing engine, the aforementioned patterns are executed over a slide window, i.e. some pre-determined (e.g. 10 repeated events) characteristic will trigger the creation of a frogs reproducing event.

4.6 Implementing and checking patterns – examples with real data

Our tests take advantage of sensor data streams that have been made available to us by Cooxupé, the largest coffee cooperative in the world. These data come from 14 weather stations at different locations in the states of Minas Gerais and São Paulo, Brazil. Figure 4.2 is a screen copy from our system that shows where these stations are deployed. This is a screen copy that illustrates how our system can show, upon user request, the geographic location of the data sources being processed. Two weather stations were selected using polygons in order to visualize their measurements.

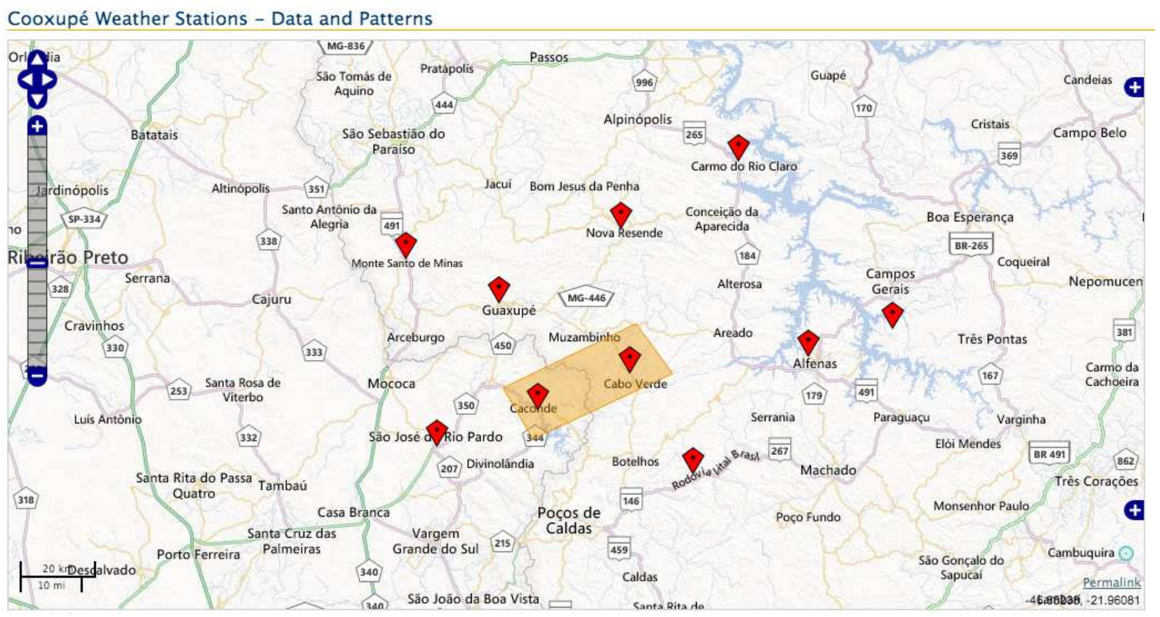
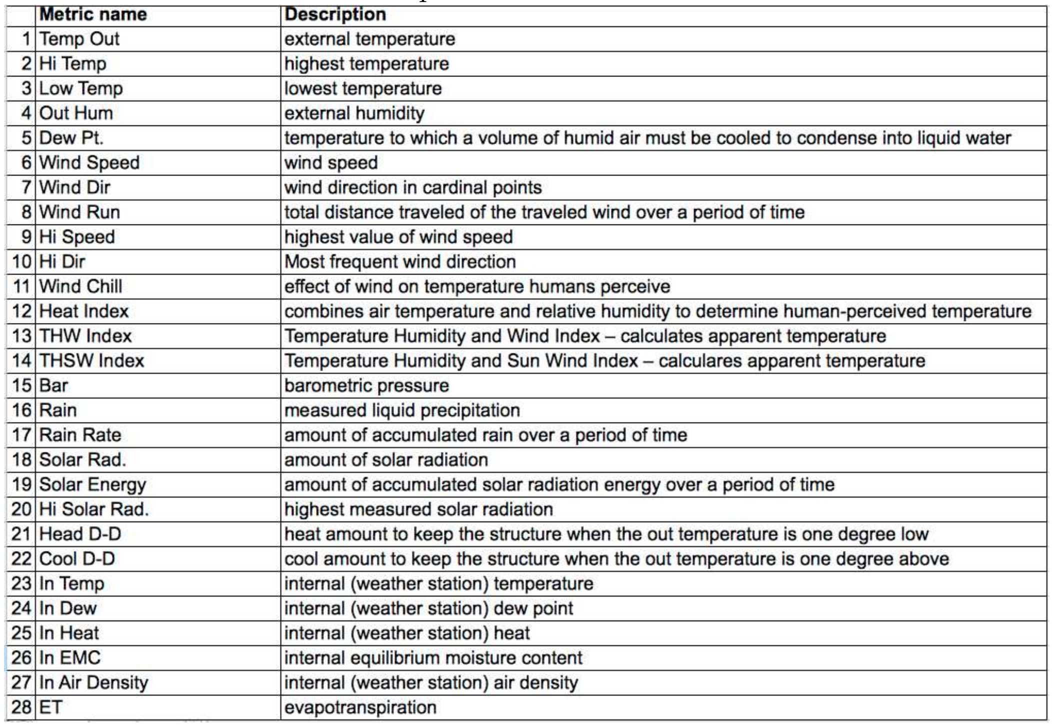


Figure 4.2: Weather Stations deployment - selected 2 stations

Each weather station continuously collects at least 26 different types of measurements as shown in Table 4.1. Twelve of the 14 stations provide 28 types of measurement, and 2 stations (located at the cities of Guaxupé and Alfenas) provide 26 (excluding internal equilibrium moisture content and weather station internal air density).

Table 4.1: Cooxupé Weather station measurements.



	Metric name	Description
1	Temp Out	external temperature
2	Hi Temp	highest temperature
3	Low Temp	lowest temperature
4	Out Hum	external humidity
5	Dew Pt.	temperature to which a volume of humid air must be cooled to condense into liquid water
6	Wind Speed	wind speed
7	Wind Dir	wind direction in cardinal points
8	Wind Run	total distance traveled of the traveled wind over a period of time
9	Hi Speed	highest value of wind speed
10	Hi Dir	Most frequent wind direction
11	Wind Chill	effect of wind on temperature humans perceive
12	Heat Index	combines air temperature and relative humidity to determine human-perceived temperature
13	THW Index	Temperature Humidity and Wind Index – calculates apparent temperature
14	THSW Index	Temperature Humidity and Sun Wind Index – calculares apparent temperature
15	Bar	barometric pressure
16	Rain	measured liquid precipitation
17	Rain Rate	amount of accumulated rain over a period of time
18	Solar Rad.	amount of solar radiation
19	Solar Energy	amount of accumulated solar radiation energy over a period of time
20	Hi Solar Rad.	highest measured solar radiation
21	Head D-D	heat amount to keep the structure when the out temperature is one degree low
22	Cool D-D	cool amount to keep the structure when the out temperature is one degree above
23	In Temp	internal (weather station) temperature
24	In Dew	internal (weather station) dew point
25	In Heat	internal (weather station) heat
26	In EMC	internal equilibrium moisture content
27	In Air Density	internal (weather station) air density
28	ET	evapotranspiration

Figure 4.3 shows the steps performed to collect sensor data. Our framework is indicated in the figure (bottom right), shown as running in our LIS laboratory. This figure illustrates an example of our pre-processing implementation effort, i.e., the kind of modules that had to be developed to get data, before it entered the framework. In (A), Cooxupé deployed 14 weather stations and at each station there are sensors (B) that collect data at one hour intervals. The Cooperative's data collecting center (C) fetches data from the stations at 15 minute intervals to prevent synchronization problems, and stores them in a flat table, which is then incrementally retrieved by us. These data are integrated into our framework (D) at our Laboratory of Information Systems (LIS) using the FTP adapter from Mule ESB [55] and patterns are identified using the Esper [20] CEP engine. In (E), an expert can write a pattern and insert it into our framework which will notify a subscriber. There is also a possibility to retrieve historical data.

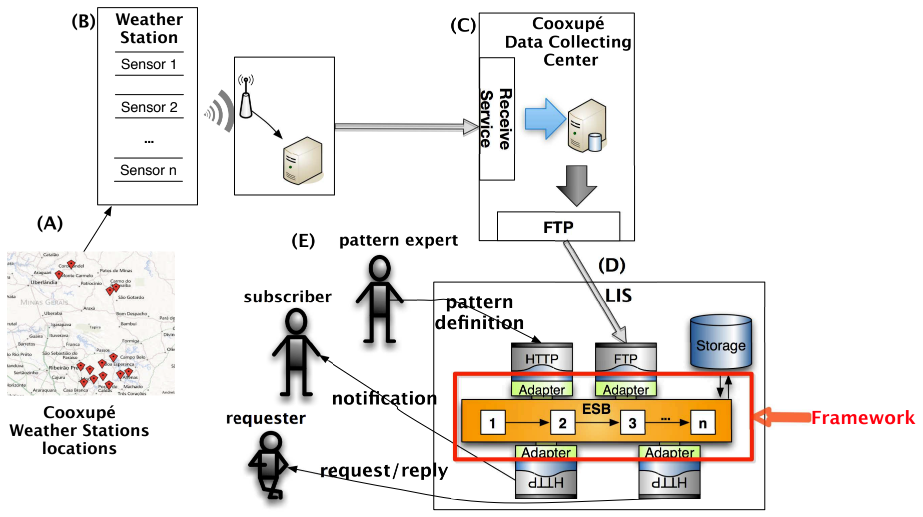


Figure 4.3: Steps of data collecting, processing and storage of our case study.

Figure 4.4 shows part of a screen copy of our system, with measurement charts of temperature, humidity and rain for the station located at the city of Campos Gerais. Experts can visualize data from specific stations by clicking on the map (see Figure 4.2) and see the variations of the desired variables.

Such interaction and visualization facilities are common to many monitoring systems. We differentiate ourselves from related work in the ability to support in our system pattern specification and detection, as well as combination of heterogeneous data sources. We proceed by showing examples of where such patterns can help the analysis of environmental conditions to help in biodiversity studies.

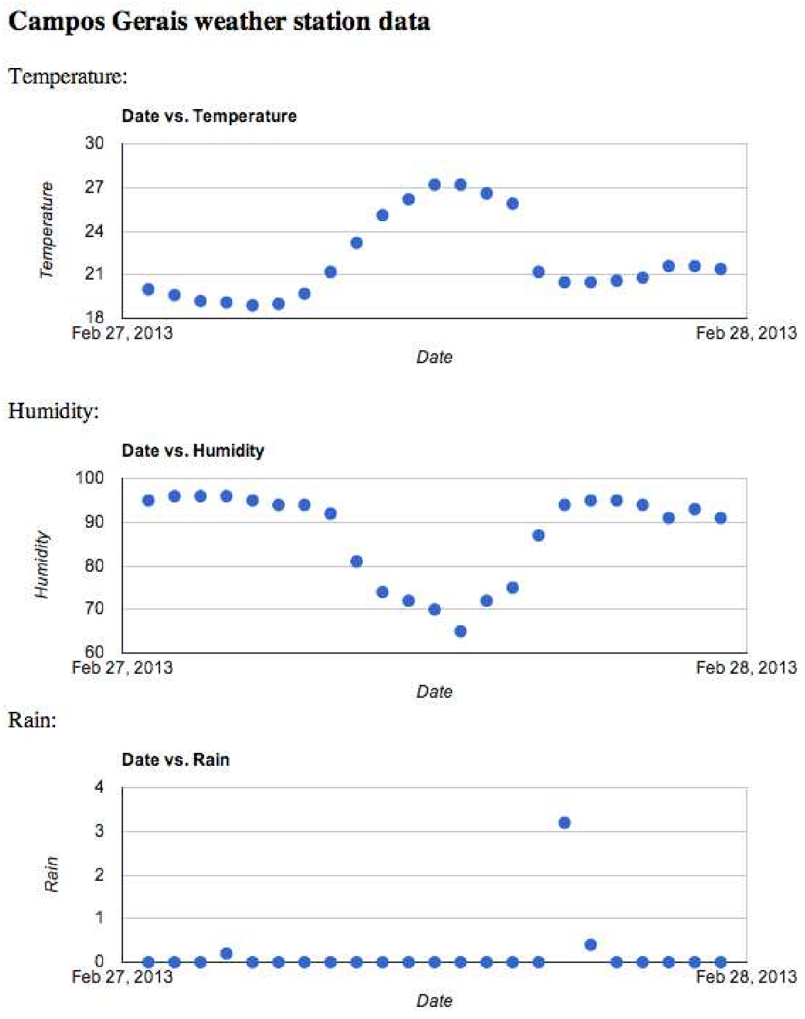
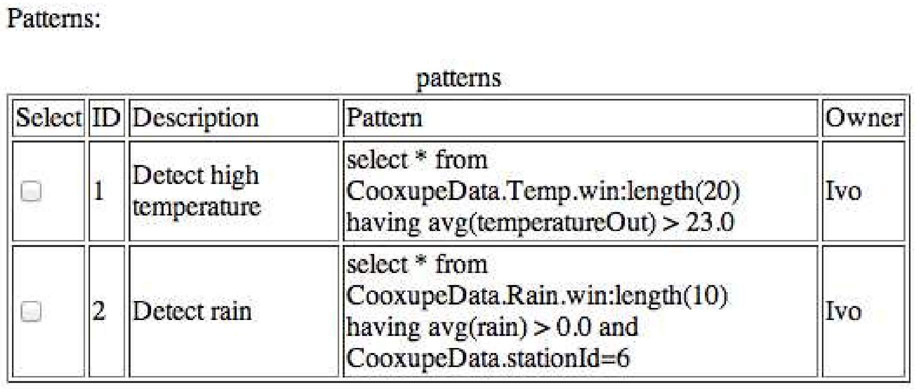


Figure 4.4: Measurements over 24 hours - variables selected for visualization

Figure 4.5 is a screen copy of our system showing some EPL statements. The first EPL detects temperature above 23.0 degrees celsius considering a data window of 20 elements, i.e. get consecutive values of 20 measurements. The second EPL detects if there was rain considering weather station at Campos Gerais and a data window of 10 elements. The first is a value pattern since it is not concerned with time or space (even though the values are implicitly linked to space and time). The CooxupeData qualifier in the queries restricts data to those collected by Cooxupé weather stations, the second query furthermore is limited to the station in Campos Gerais, thus implicitly imposing a spatial predicate.


Patterns:

patterns

Select	ID	Description	Pattern	Owner
▢	1	Detect high temperature	select * from CooxupeData.Temp.win:length(20) having avg(temperatureOut) > 23.0	Ivo
▢	2	Detect rain	select * from CooxupeData.Rain.win:length(10) having avg(rain) > 0.0 and CooxupeData.stationId=6	Ivo

Figure 4.5: Pattern definition window.

Let us now go back to our frog example. Consider the pattern for frogs' reproduction mentioned in section 4.5. First, frogs and their habitat regions are retrieved from databases, e.g. [23, 57]. Then satellite images, e.g. from National Oceanic and Atmospheric Administration (NOAA)[6], are selected: only the regions where the species of frog of interest can be found are taken into account. The soil moisture in such regions can be calculated using techniques from [36, 82]. The value that goes into the framework is the minimum soil moisture computed (since we want to be sure that ponds were formed) and information from satellite images and frogs' habitats, i.e. timestamp and location of the event.

Rainfall measurements are gathered from weather stations; again, only measurements from regions where frogs live are retrieved. Thus, our framework can produce soil moisture, rainfall measurements and type of frogs events from each region and send these information for event processing.

[6]An United States Department of Commerce scientific agency focused on the conditions of the oceans and the atmosphere.

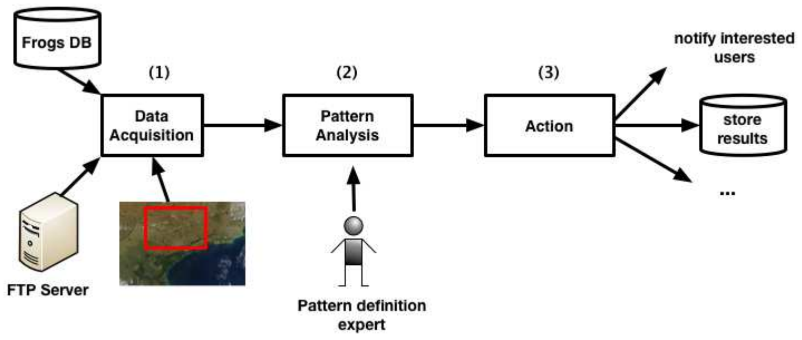


Figure 4.6: Frogs reproduction pattern detection.

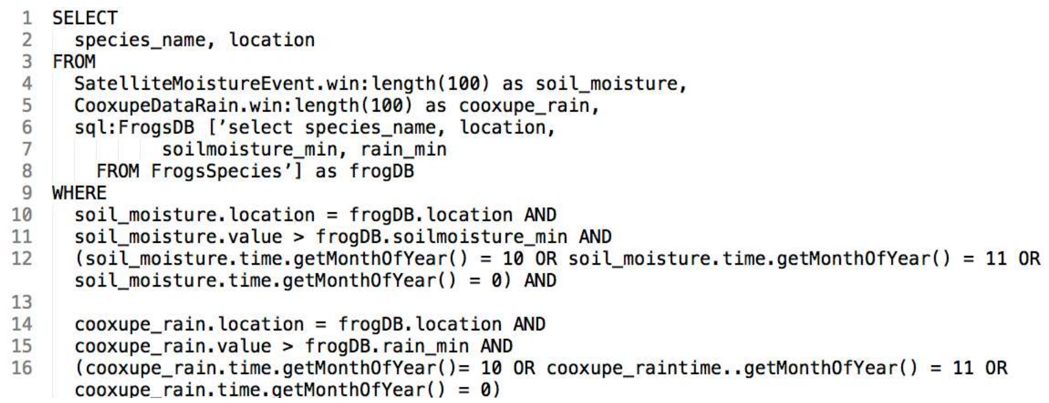
Figure 4.6 shows the steps to perform this task. In step (1) data are acquired from databases (frogs data), satellite images and weather stations. Data can come from real-time streams from the Internet or sensors, or stored repositories such as databases, Web Services or files. Step (2) relies on the knowledge of an expert to create a pattern useful to the domain, e.g. a frog expert wants to verify time and locations in which a specific kind of frog may reproduce.

Patterns rely on the assumption that data are integrated via adapters into the framework and encapsulated in messages. Once this is accomplished, the event processing engine can detect the pattern. In (3), an action can be performed following detection of the event pattern such as notification, storage of results, or others.

Considering this approach, we created an EPL statement, shown in Figure 4.7, to estimate what frogs are prone to reproduce for a region. To do this, we put together the computed values of soil moisture from satellite images, weather station meteorological measurements and frogs' information stored in a database (FrogsDB).

The pattern's spatial extent is restricted to locations where frogs live in southeast of Brazil. For these locations, soil moisture is computed from satellite images and rain measurements are provided by Cooxupé stations. The pattern takes minimum values of rain and soil moisture from the frogs' database and compares to the events that flow into the framework. The current month (a system variable which stores the current month) must be between November and January, i.e. the time period where usually there is rain in southeast of Brazil.

This query does not take any window into consideration. Furthermore, an event will be raised by the framework each time the pattern has a match. Since it is difficult to find a match in this type of event, we want to verify each time these characteristics match with the minimum requirements of the different species of frog to reproduce in the months between November and January.



```
1  SELECT
2    species_name, location
3  FROM
4    SatelliteMoistureEvent.win:length(100) as soil_moisture,
5    CooxupeDataRain.win:length(100) as cooxupe_rain,
6    sql:FrogsDB ['select species_name, location,
7            soilmoisture_min, rain_min
8      FROM FrogsSpecies'] as frogDB
9  WHERE
10   soil_moisture.location = frogDB.location AND
11   soil_moisture.value > frogDB.soilmoisture_min AND
12   (soil_moisture.time.getMonthOfYear() = 10 OR soil_moisture.time.getMonthOfYear() = 11 OR
     soil_moisture.time.getMonthOfYear() = 0) AND
13
14   cooxupe_rain.location = frogDB.location AND
15   cooxupe_rain.value > frogDB.rain_min AND
16   (cooxupe_rain.time.getMonthOfYear()= 10 OR cooxupe_raintime..getMonthOfYear() = 11 OR
     cooxupe_rain.time.getMonthOfYear() = 0)
```

Figure 4.7: Frogs' reproduction EPL.

Let us now explain the details of Figure 4.7. Lines 1 and 2 define what needs to be investigated: species name and location. Lines 3 to 8 depict the sources of events, i.e. soil moisture values computed from satellite images, meteorological station data (line 5) from Cooxupé stations, frogs database. Lines 6 through 8, in particular, extract from that database the minimum values to be considered. Lines 10 and 11 compare these minimum values with these computed over satellites images. Lines 14 and 15 take the Cooxupé meteorological station data, for a given location, and compare those data to data obtained from the frogs database, to check if minimum rain rate is reached. Lines 12 and 16 verify if the soil moisture and rain events, respectively, occur in November, December or January (10, 11 and 0).

If the pattern has a match (i.e. soil moisture, rain greater than the minimum rate from frogs database and the current month is between November and January), it will raise an event that will contain the frog species name, location and when Cooxupé meteorological station event occurred, i.e. attributes defined in line 2. This information can inform where and when a specific frog species is prone to reproduce.

4.7 Conclusions

A framework to provide environmental sciences data integration and pattern detection was shown in this paper. This framework was designed and implemented focusing on the need of environmental scientists to detect important events out of the highly heterogeneous data environment they work with.

Considering data collection frequency at the sensor level. Many sensors collect data at short intervals, but only transmit at specific rates to save energy. Thus, users are notified based on data transmitted. It might be interesting to estimate collection rate and make the user aware that more data might be available if transmission rates increased.

Although the framework brings together facilities to integrate and process events, experts are needed to depict appropriate patterns. Experts have to carefully examine data that comes from different sources to define the ideal patterns for their scientific studies. Moreover, pre-processing and filtering require development of specific algorithms, that take the nature of the data into account.

As future work, we suggest the use of data mining and machine learning techniques to assist scientists in discovering and creating patterns in the framework.

Extracting and deploying patterns in the framework is useful, but having a repository of patterns where experts could exchange patterns would be useful for reproducibility of their findings. One key issue in the specification of this repository is the use of semantics in event handling – *e.g.* [75].

Chapter 5

Conclusions and Extensions

5.1 Main Contributions

This PhD research addressed some Computer Science issues involving data management – more specifically, handling heterogeneity in environmental data. The focus of this work was to provide means to help environmental scientists cope with heterogeneity.

Aiming at overcoming challenges in environmental data management, experiments were performed in storage and publication of sensor data, extracting information for different types of users.

The publication of sensor data, on the Web, involves issues that go beyond the nature of the data being collected and are intimately related with problems of the Web itself – such as data heterogeneity, privacy, volume of data and user requirements.

From our first attempt in solving environmental data management challenges we had some contributions.

A review in architectural solutions to cope with heterogeneity and interoperability in sensor data management was performed. We examined generic solutions, aimed at retrieval and processing of sensor data without specifying the application domain such as [1, 12, 62] and other solutions that are targeted to specific applications such as GeoCENS [48], focused in local scale sensor networks, SEAMONSTER [30], developed to study glacier watersheds and Life Under Your Feet [73], which was developed for soil monitoring. Nevertheless, all these efforts provide isolation layers between the sensor network and users, enabling the development of custom layers in order to overcome problems in each layer at a time.

Envisioning link the best of each proposal and provide a more flexible architecture, we initially developed a solution based on Web services and components. The former provides means for different systems to communicate, increasing the accessibility and the latter enables creation of customizations to access and visualize sensor data.

Continuing our work, following our concern with the Web Science issue of visibility: data has to be found and correctly interpreted; we verified which alternatives could be used to accomplish our goals.

In chapter 3, we built the foundation of our solution to this kind of problem. We figured out that integrating data sources and treating data as events proved to be a good alternative to work on. We defined the notation of event patterns (descriptions of events of interest), how it can be retrieved and classified them in simple or composite (combinations of simple patterns).

This approach allows the generic management of data sources (static and dynamic, from different regions and considering different phenomena). It integrates data using ESB adapters and ESB messages are treated as events that are extracted from the sources using CEP which enables the creation of hierarchies of event patterns.

In chapter 4, we pointed out that the majority of the studies in environmental sciences concentrate in dealing with static sources and we focused our work in combining both static and stream data in a single framework. To properly import the concepts of the CEP and ESB to environmental sciences context, we introduced a pre-processing phase before data enter the ESB, since ESB is generally applied to enterprise applications where data granularity is not usually as diverse as found in this context. For example: satellite images may be cut to consider only regions to be studied or measurements from meteorological stations may be filtered to consider air humidity from a specific region.

After pre-processing, data are encapsulated in messages to flow through the ESB. For each type of study, scientists will create event patterns to match the phenomena they want to observe. Each event pattern will be deployed inside the framework and if there is a match it will trigger the creation of a new event and an action will be performed (e.g. notify users, store data for later analysis).

The final patterns notation we developed is of the form:
<*measured-value*, *nature*, *spatial-variable*, *timestamp*> = <*v*, *n*, *s*, *t*>, where:

- *v* is the value of the variable captured (e.g. 40 degrees celsius, 80% humidity);
- *n* the nature of the variable (e.g. temperature, humidity);
- *s* is the location (coordinates) of the measurement;
- *t* is a time interval [ts, tf][1] for which v was valid at that location (s). If ts = tf this is an instantaneous event (e.g., sensor measurement).

We classify event patterns as Value, Spatial, Temporal and Spatiotemporal patterns. Value patterns only consider the value in the predicate. Spatial patterns consider values of

[1]ts is the start time and tf is the final or end time.

predicates over a spatial distribution. Temporal patterns focus in variations of predicates in time. Spatiotemporal patterns combine all of the aforementioned patterns, i.e. combine measured variables, spatial and time properties. This framework presents additional advantage to users. They can adapt patterns dynamically, and therefore test distinct hypotheses when checking for environmental conditions.

Our main contributions are:

- Considering data collection frequency at the sensor level. Many sensors collect data at short intervals, but only transmit at specific rates to save energy. Thus, users are notified based on data transmitted. It might be interesting to estimate collection rate and make the user aware that more data might be available if transmission rates increased;
- A framework to help environmental scientists cope with heterogeneity problems that allows integration of static and stream data sources in a generic way (as opposed to specific solutions in the literature);
- This framework presents an additional advantage to users. They can adapt patterns dynamically, and therefore test distinct hypothesis when checking for environmental conditions;
- Treatment of environmental data events, and processing patterns;
- Application of the aforementioned findings in ecological studies showing how scientists can use our proposal to acquire data of interest from the available data sources.

5.2 Extensions

There are many possible extensions to this work involving theoretical and practical proposals. Examples of some of these extensions are:

- Investigation and specification of easier means to build and deploy new event patterns inside the framework. In this direction experts can use data mining techniques such as [16, 35, 37, 78];
- Pattern management: extracting and deploying patterns in the framework is useful, but having a repository of patterns where experts could exchange patterns would be useful for reproducibility of their findings. One key issue in the specification of this repository is the use of semantics in event handling – *e.g.* [75];

- Performance and security/privacy issues must be addressed in order to use this framework. Integrating multiple data sources with a large amount of data flows can generate performance problems. Scalability studies must be performed to allow the framework to handle large amounts of data. Similarly, sensitive data can flow through the framework; it must be ready to use mechanisms of data security such as data encryption;

- Investigation of data provenance and data quality applications in the framework is another possible extension. Recording data processing from the input to the output can allow traceability and provide means for experts to detect errors and may allow determination of data quality – e.g. as in [51];

- Considering data collection frequency at the sensor level. Many sensors collect data at short intervals, but only transmit at specific rates to save energy. Thus, users are notified based on data transmitted. It might be interesting to estimate collection rate and make the user aware that more data might be available if transmission rates increased.

Bibliography

[1] K. Aberer, M. Hauswirthand, and A. Salehi. Infrastructure for data processing in large-scale interconnected sensor networks. In *Proceedings of the Mobile Data Management (MDM 2007)*, Mannheim, Germany, 2007. IEEE Computer Society.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38:393–422, 2002.

[3] D. P. Ames, J. S. Horsburgh, Y. Cao, J. Kadlec, T. Whiteaker, and D. Valentine. Hydrodesktop: Web services-based software for hydrologic data discovery, download, visualization, and analysis. *Environmental Modelling & Software*, 37(0):146 – 156, 2012.

[4] G. P. Asner, D. Nepstad, G. Cardinot, and D. Ray. Drought stress and carbon uptake in an amazon forest measured with spaceborne imaging spectroscopy. *PNAS*, 101(16):6039–6044, 2004.

[5] D. Baldocchi and et al. FLUXNET: A New Tool to Study the Temporal and Spatial Variability of Ecosystem–Scale Carbon Dioxide, Water Vapor, and Energy Flux Densities. *Bull. Amer. Meteor. Soc.*, 82(11):2415–2434, November 2001.

[6] T. Berners-Lee, W. Hall, J. A. Hendler, K. O'Hara, N. Shadbolt, and D. J. Weitzner. A framework for web science. *Foundations and trends in Web Sci.*, 1(1):1–130, 2006.

[7] C. L. Borgman and et al. Drowning in data: digital library architecture to support scientific use of embedded sensor networks. In *Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries*, JCDL '07, pages 269–277, New York, NY, USA, 2007. ACM.

[8] M. Botts and A. Robin. Opengis sensor model language (sensorml) implementation specification. *OpenGIS Implementation Specification OGC*, pages 07–000, 2007.

[9] M. Chang and P. Bonnet. Meeting ecologists' requirements with adaptive data acquisition. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, SenSys '10, pages 141–154, New York, NY, USA, 2010. ACM.

[10] D. A. Chapell. *Enterprise Service Bus.* O'Reilly Media Inc., 2004.

[11] C. Chong and S. P. Kumar. Sensor networks: evolution, opportunities, and challenges. *Proceedings of the IEEE*, 91(8):1247–1256, 2003.

[12] X. Chu, T. Kobialka, B. Durnota, and R. Buyya. Open sensor web architecture: Core services. In *Proceedings of the 4th International Conference on Intel ligent Sensing and Information Processing (ICISIP 2006)*, pages 98–103, Piscataway, New Jersey, USA, 2006. IEEE Press.

[13] G. E. Churcher and J. Foley. Applying complex event processing and extending sensor web enablement to a health care sensor network architecture. In *Sensor Systems and Software*, volume 24 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 1–10. Springer, 2010.

[14] Cosm Ltd. Cosm platform. `https://cosm.com`, March 2013.

[15] CUAHSI-HIS. CUAHSI's Hydrologic Information System (CUAHSI-HIS). http://his.cuahsi.org/, 2013.

[16] T. De Bie. An information theoretic framework for data mining. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '11, pages 564–572, New York, NY, USA, 2011. ACM.

[17] Oxford Dictionaries. Oxford dictionaries. `http://oxforddictionaries.com/definition/american_english/event`, December 2012. accessed December 10, 2012.

[18] J. Dunkel. On complex event processing for sensor networks. In *Autonomous Decentralized Systems, 2009. ISADS '09. International Symposium on*, pages 1 –6, March 2009.

[19] EsperTech Inc. Esper Reference 4.9.0. `http://esper.codehaus.org/esper-4.9.0/doc/` (Accessed April, 2013), 2012.

[20] EsperTech Inc. Esper website. `http://esper.codehaus.org/` (Accessed March, 2013), 2013.

[21] O. Etzion and P. Niblett. *Event Processing in Action.* Manning, 2011.

[22] E. H. Fegraus, S. Andelman, M. B. Jones, and M. Schildhauer. Maximizing the value of ecological data with structured metadata: an introduction to ecological metadata language (EML) and principles for metadata creation. *Bulletin of the Ecological Society of America*, 86(3):158–168, 2005.

[23] International Union for Conservation of Nature and Natural Resources. IUCN Red List. `http://iucnredlist.org/`.

[24] Brazilian Institute for Web Science Research. Brazilian institute for web science research. `http://webscience.org.br`, accessed in august 2010.

[25] Global Lake Ecological Observatory Network. Global Lake Ecological Observatory Network (GLEON). `http://www.gleon.org/` (Accessed Oct, 2012), 2012.

[26] J. Gupchup, A. Sharma, A. Terzis, A. Burns, and A. Szalay. The perils of detecting measurement faults in environmental monitoring networks. In *DCOSS*, 2008.

[27] L. Gurgen and et al. Sstreamware: a service oriented middleware for heterogeneous sensor data management. In *Proc. 5th Intl conf. on Pervasive services*, ICPS '08, pages 121–130, 2008.

[28] M. P. Hamilton and et al. New approaches in embedded networked sensing for terrestrial ecological observatories. *Environmental Engineering Science*, 24(2):192–204, 2007.

[29] J. K. Hart and K. Martinez. Environmental Sensor Networks: A revolution in the earth system science? *Earth Science Reviews*, 78(3-4):177–191, 2006.

[30] M. Heavner, D. R. Fatland, E. Hood, and C. Connor. SEAMONSTER: A wireless Sensor Web prototype applied to studying glaciated watersheds. *ESTF 2010*, 2010.

[31] T. Hey, S. Tansley, and K. Tolley, editors. *The Fourth Paradigm: Data-Intensive Scientific Discovery.* Microsoft Corporation, 2009.

[32] Chi-Fu Huang, Yu-Chee Tseng, and Hsiao-Lu Wu. Distributed protocols for ensuring both coverage and connectivity of a wireless sensor network. *ACM Transactions on Sensor Networks*, 3:1–24, 2007.

[33] D. Hughes, K. Thoelen, W. Horré, N. Matthys, J. D. Cid, S. Michiels, C. Huygens, and W. Joosen. Looci: a loosely-coupled component infrastructure for networked embedded systems. In *Proceedings of the 7th International Conference on Advances in Mobile Computing and Multimedia*, pages 195–203. ACM, 2009.

[34] D. Hughes, J. Ueyama, E. Mendiondo, N. Matthys, W. Horré, S. Michiels, C. Huygens, W. Joosen, K. L. Man, and S. Guan. A middleware platform to support river monitoring using wireless sensor networks. *Journal of the Brazilian Computer Society*, 17(2):85–102, 2011.

[35] E. Ikoma, K. Taniguchi, T. Koike, and M. Kitsuregawa. Development of a data mining application for huge scale earth environmental data archives. *Int. J. Comput. Sci. Eng.*, 2(5/6):262–270, August 2006.

[36] T. J. Jackson. III. Measuring surface soil moisture using passive microwave remote sensing. *Hydrological Processes*, 7(2):139–152, 1993.

[37] J. M. Kang. *Spatio-temporal data mining for environmental science.* PhD thesis, University of Minnesota, Minneapolis, MN, USA, 2010.

[38] M. Keller and et al. A continental strategy for the National Ecological Observatory Network. *Frontiers in Ecology and the Environment*, 6(5):282–284, June 2008.

[39] I. Koga and C. B. Medeiros. Integrating and processing events from heterogeneous data sources. In *Proceedings VI eScience Workshop - XXXII Brazilian Computer Society Conference*, July 2012.

[40] I. Koga and C. B. Medeiros. Patterns in Environmental Event Processing. Technical Report IC-13-14, Institute of Computing, University of Campinas, July 2013.

[41] I. Koga and C. B. Medeiros. Managing environmental data from sensor networks. In *International Conference on Web Information Systems and Technologies (WEBIST), submitted for publication*, 2014.

[42] I. Koga, C. B. Medeiros, and O. Branquinho. Handling and publishing wireless sensor network data: a hands-on experiment. In *Proceedings IV eScience Workshop - XXX Brazilian Computer Society Conference*. SBC, July 2010.

[43] I. Koga, C. B. Medeiros, and O. Branquinho. Handling and publishing wireless sensor network data: a hands-on experiment. *Journal of Computational Interdisciplinary Sciences*, 2(1):13–22, March, April 2011.

[44] I. Koga, L. Sampaio, and J. A. S. Monteiro. FLAVOR: A dynamic and open framework for the development of network measurement access and visualization tools. In *XXV Brazilian Symposium on Computer Networks and Distributed Systems (SBRC)*, pages 665–678, Belém, PA, 2007.

[45] G. E. Krasner and S. T. Pope. A cookbook for using the model-view controller user interface paradigm in smalltalk-80. *J. Object Oriented Program.*, 1(3):26–49, 1988.

[46] Z. Kugler and T. D. Groeve. The global flood detection system. Technical Report 44149, JRC Scientific and Technical Reports, 2007.

[47] C. Lagoze and H. V. Sompel. The open archives initiative: building a low-barrier interoperability framework. In *Proceedings of the 1st ACM/IEEE-CS joint conference on Digital libraries*, JCDL '01, pages 54–62, New York, NY, USA, 2001. ACM.

[48] S. H. L. Liang, D. Chang, J. Badger, R. Rezel, S. Chen, C.-Y. Huang, and R.-Y. Li. Capturing the long tail of sensor web. In *International Workshop on Role of Volunteered Geographic Information in Advancing Science, part of GIScience 2010*, Zurich, Switzerland, 09/2010 2010.

[49] D. C. Luckham. *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[50] Y. Magid and et al. Industry experience with the IBM Active Middleware Technology (AMiT) Complex Event Processing engine. In *Proc. 4th ACM DEBS '10*, pages 140–149, 2010.

[51] J. E. G. Malaverri. *Supporting data quality assessment in eScience: a provenance based approach.* PhD in Computer science, Institute of Computing – University of Campinas (UNICAMP), 2013.

[52] D. Marples and P. Kriens. The open services gateway initiative: An introductory overview. *IEEE Communications Magazine*, 39:110–114, December 2001.

[53] C.B. Medeiros. Grand Research Challenges in Computer Science in Brazil. *IEEE Computer Society*, 41(6):59 –65, june 2008.

[54] Microsoft. Windows azure. `http://www.microsoft.com/windowsazure/`, accessed in august 2010.

[55] MuleSoft Inc. Mule website. `http://www.mulesoft.com/` (Accessed Apr, 2012).

[56] E. F. Nakamura, A. A. F. Loureiro, and A. C. Frery. Information fusion for wireless sensor networks: Methods, models, and classifications. *ACM Comput. Surv.*, 39(3):9, 2007.

[57] Catalogue of Life. Catalogue of life. `http://www.catalogueoflife.org/`.

[58] OGC. Sensor Web Enablement. `http://www.opengeospatial.org/projects/groups/sensorwebdwg`, 2011.

[59] OGC. Geoserver. `http://geoserver.org`, accessed in august 2010.

[60] N. A. Pantazis, S. A. Nikolidakis, and D. D. Vergados. Energy-efficient routing protocols in wireless sensor networks for health communication systems. In *PETRA '09: Proceedings of the 2nd International Conference on PErvsive Technologies Related to Assistive Environments*, pages 1–8, New York, NY, USA, 2009. ACM.

[61] G. Z. Pastorello. *Managing the lifecycle of sensor data: from production to consumptions*. PhD in Computer science, Institute of Computing – University of Campinas (UNICAMP), 2008.

[62] G. Z. Pastorello Jr, C. B. Medeiros, and A. Santanchè. Providing homogeneous access for sensor data management. Technical Report IC-07-012, Institute of Computing, State University of Campinas, May 2007.

[63] Z.n Peng and C. Zhang. The roles of geography markup language (gml), scalable vector graphics (svg), and web feature service (wfs) specifications in the development of internet geographic information systems (gis). *Journal of Geographical Systems*, 6(2):95–116, 2004.

[64] T. Rademakers and J. Dirksen. *Open-Source ESBs in Action*. Manning Publications Co., Greenwich, CT, USA, 2009.

[65] C. Reed, M. Botts, J. Davidson, and G. Percivall. Ogcsensor web enablement:overview and high level architecture. In *Autotestcon, 2007 IEEE*, pages 372–380, 17-20 2007.

[66] K.H. Rosen. *Discrete mathematics and its applications (6th ed.)*. McGraw-Hill, Inc., New York, NY, USA, 2007.

[67] P. W Rundel, E. A. Graham, M. F. Allen, J. C. Fisher, and T. C Harmon. Environmental sensor networks in ecological research. *The New phytologist*, 182(3):589–607, January 2009.

[68] T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 851–860, New York, NY, USA, 2010. ACM.

[69] A. Santanchè and C. B. Medeiros. A component model and an infrastructure for the fluid web. *IEEE Transactions on Knowledge and Data Engineering*, 19(2):324–341, February 2007.

[70] M.-T. Schmidt, B. Hutchison, P. Lambros, and R. Phippen. The enterprise service bus: Making service-oriented architecture real. *IBM Systems Journal*, 44(4):781–797, 2005.

[71] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler. An analysis of a large scale habitat monitoring application. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, SenSys '04, pages 214–226, New York, NY, USA, 2004. ACM.

[72] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, and D. Estrin. Habitat monitoring with sensor networks. *Commun. ACM*, 47(6):34–40, 2004.

[73] K. Szlavecz, A. Terzis, S. Ozer, R. Musaloiu-e, J. Cogan, S. Small, S. Ozer, R. Burns, J. Gray, and A. S. Szalay. Life Under Your Feet : An End-to-End Soil Ecology Sensor Network, Database, Web Server, and Analysis Service. In *3rd Workshop on Embedded Networked Sensors (EmNets 2006)*, pages 51–55, May 2006.

[74] A. L.C. Tavares and M. T. Valente. A gentle introduction to osgi. *SIGSOFT Softw. Eng. Notes*, 33(5):1–5, 2008.

[75] K. Teymourian and A. Paschke. Enabling knowledge-based complex event processing. In *Proc. EDBT/ICDT Workshops*, EDBT '10, pages 37:1–37:7, New York, NY, USA, 2010. ACM.

[76] D. Thau and et al. LNCS 5872 - Contemporary Challenges in Ambient Data Integration for Biodiversity Informatics. *Florida Entomologist*, 0630033:59–68, 2009.

[77] Sameer Tilak and et al. Conceptual challenges and practical issues in building the global lake ecological observatory network. In *3rd International Conference on Intelligent Sensors, Sensor Networks and Information*, pages 721 –726, 2007.

[78] V. D. Tran, L. Hluchy, and O. Habala. Data mining and integration for environmental scenarios. In *Proceedings of the 2010 Symposium on Information and Communication Technology*, SoICT '10, pages 55–58, New York, NY, USA, 2010. ACM.

[79] The Web Science Trust. The web science trust. `http://webscience.org/home.html`, accessed in august 2010.

[80] UNESCO. Encyclopedia of Life Support Systems (EOLSS). `http://http://www.eolss.net/`.

[81] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner. A break in the clouds. *ACM SIGCOMM Computer Communication Review*, 39(1):50, December 2008.

[82] K. Y. Vinnikov, A. Robock, S. Qiu, J. K. Entin, M. Owe, B. J. Choudhury, S. E. Hollinger, and E. G. Njoku. Satellite remote sensing of soil moisture in illinois, united states. *Journal of Geophysical Research: Atmospheres*, 104(D4):4145–4168, 1999.

[83] Q. Zhao and M. Gurusamy. Lifetime maximization for connected target coverage in wireless sensor networks. *IEEE/ACM Trans. Netw.*, 16(6):1378–1391, 2008.