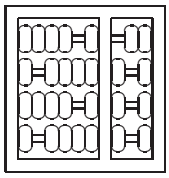

INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

Patterns in Environmental Event Processing

I. Koga *C. B. Medeiros*

Technical Report - IC-13-16 - Relatório Técnico

July - 2013 - Julho

Patterns in Environmental Event Processing

Ivo Koga * Claudia Bauzer Medeiros

Abstract

Environmental scientists have to deal with large amounts of heterogeneous data. To accomplish their scientific goals, they have to overcome data management problems that range from the acquisition to the storage of data.

This paper is concerned with solving some of the problems at the stage of data analysis. We developed a framework that enables the integration of different types of data sources and detection of relevant events considering the integrated data that flows through the framework. A key aspect of our approach is that it allows treating static and dynamic data sources homogeneously.

A case study using real data from meteorological stations is discussed showing the feasibility of our framework.

1 Introduction

Environmental monitoring applications allow scientists to detect otherwise unobserved situations, which can be of interest to their studies [1]. 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., [2].

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

*Institute of Computing (IC), University of Campinas (UNICAMP). FAPESP (grant 09/52336-7)

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, 4, 5, 6]. 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 drouth 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 [7], 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.

Our approach integrates data sources using Enterprise Service Bus (ESB), a combined infrastructure that provides integration between systems [8, 9]. Static and stream data sources are integrated using ESB adapters. Data are materialized in messages and treated as events, using Complex Event Processing (CEP) [10, 11] which provides means to deal with rules and events' causal relationships. We thereby provide ways to create a multi-layered architectures of events, corresponding to multiple kinds of data aggregations.

On the capture side, our proposal integrates distinct kinds of sources, thanks to ESB properties. 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 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 3 provides some related work that deals with environmental data. Section 4 presents an overview of our framework for environmental data management. This framework is detailed in [12]. Section 5 provides examples of use of event patterns. Section 6 presents our implementation showing the practical use of our framework and patterns, and section 7 concludes the paper.

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 [13] 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" [14]. 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) [15]. 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 [15]. It is created using event patterns rules or aggregation rules since they aggregate set of events. A complex event, unlike its name, is more understandable to humans since it 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 Related Work

There are countless studies that involve static data sources in environmental sciences (see, for instance, several papers in EOLSS [16]). 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 [17] 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 [1] 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 do not occur, since our emphasis is on providing users with tools to detect events. Error detection (and correction) are left to future work.

While the above concern the quality of stream data, other researchers deal with specific kinds of data items. For instance, reseach conducted by Rundel [18] 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 [19] 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] [20], Geographic Markup Language (GML) [21], Sensor Model Language (SensorML) [22], Ecological Metadata Language (EML) [23]. They all are creating standards to facilitate the data exchange, integration and interoperability between systems.

Similar to these studies, we are interested in the integration and processing of data from distinct kinds of sensors, but unlike them, we are also concerned in detecting event patterns from the data. Moreover, static sources are also treated together with dynamic streams.

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. [24]. 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 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

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

data are then treated as event streams, and hence events of interest can be processed using event patterns.

Figure 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.

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 wants 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) [25], deployed and processed using the Esper [26] Event Processing Engine.

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

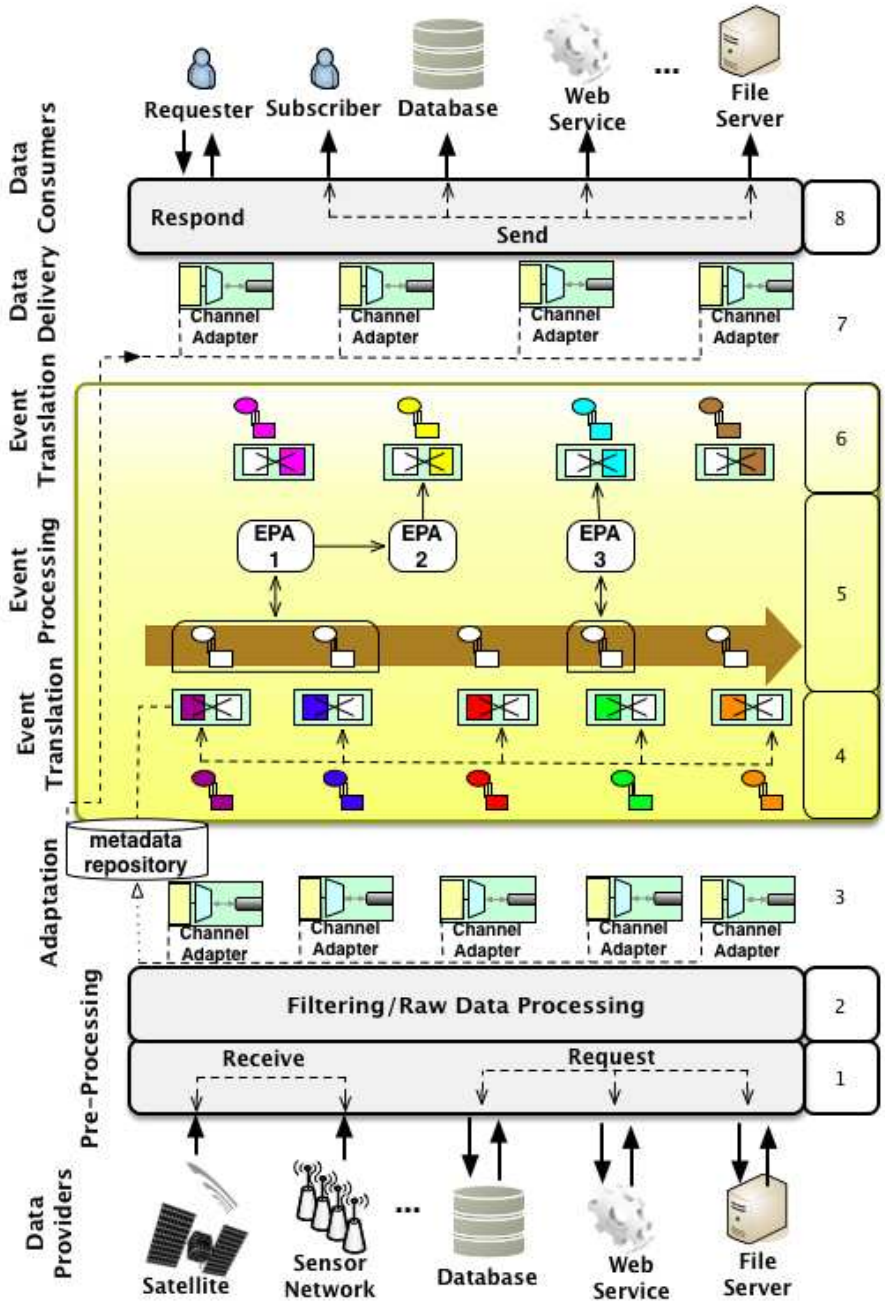

Figure 1: Architecture of the framework.

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.

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 [27]. 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 of the form:

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

- v is the value of the variable captured (e.g. 40 degrees celsius, 80% humidity)
- s is the location (coordinates) of the measurement
- t is the timestamp when the variable was measured
- n the nature of the variable (e.g. temperature, humidity)

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. species information. Information can also be retrieved from satellite images, which are cut in polygons and transformed into values of radiation or calculated indices such as Normalized Difference Vegetation Index (NDVI)[2]. These values can be taken in several different ways, i.e. the centroid of the polygon, an average of all 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.

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

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 sake. Our actual patterns are implemented in EPL, an SQL-like domain specific language provided by Esper – see section 6.

Many kinds of frogs only croak when it rains and spawn their eggs in temporary ponds. 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 [28]. 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 [29] 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 for frog reproduction:

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

Since this pattern has value, space and time predicates, it is a spatiotemporal pattern. From this composite pattern we can derive the other simple types of pattern:

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

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.

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 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. In this Figure, two weather stations were selected using polygons in order to visualize their measurements.

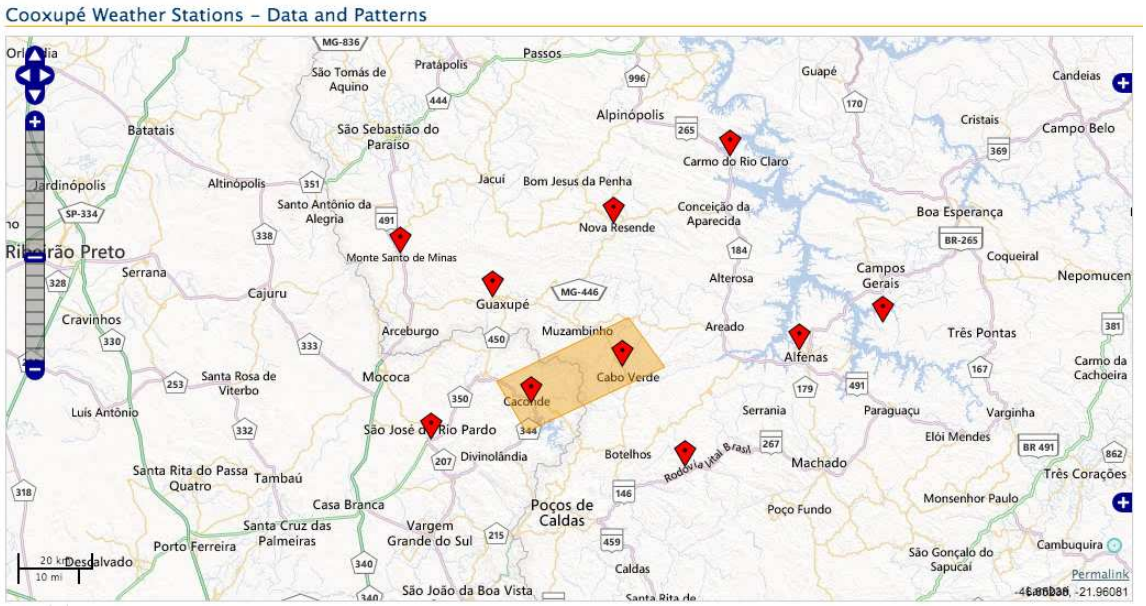

Figure 2: Weather Stations deployment - selected 2 stations

Each weather station continuously collects at least 26 different types of measurements as shown in Table 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).

Figure 3 shows the steps performed to collect sensor data. 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 [30] and patterns are identified using the Esper [26] 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

Table 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

possibility to retrieve historical data.

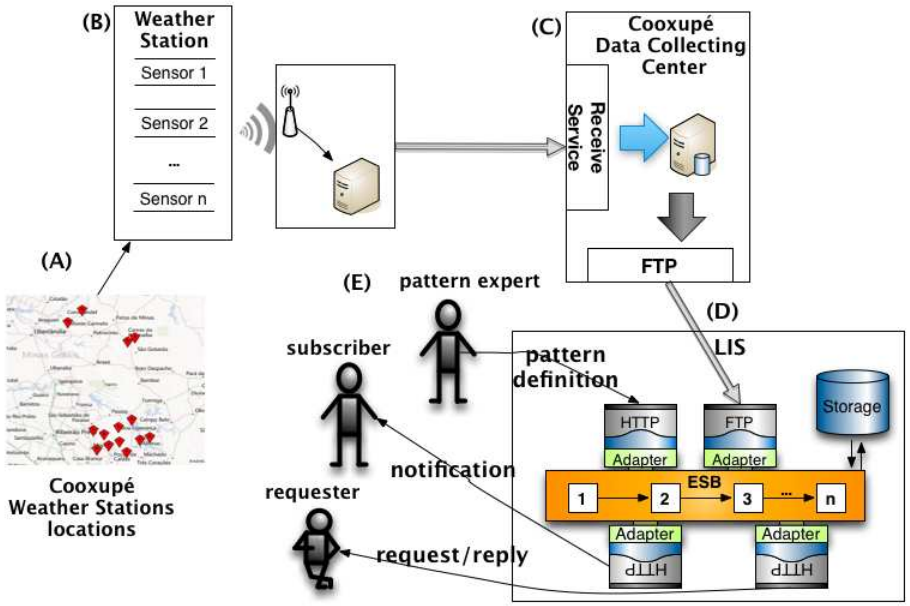

Figure 3: Steps of data collecting, processing and storage of our case study.

Figure 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 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 in programming (biological) field trips, planning collection activities and analyzing environmental conditions to help in biodiversity studies.

Figure 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 all weather stations and a data window of 10 elements. Both are value patterns since they are not concerned with time or space (even though the values are implicitly linked to space and time).

Let us now go back to our frog example. Consider the pattern for frogs' reproduction mentioned in section 5. First, frogs and their habitat regions are retrieved from databases. Then satellite images, e.g. from National Oceanic and Atmospheric Administration (NOAA)[3], are selected: only the regions where the species of frog of interest can

[3]An United States Department of Commerce scientific agency focused on the conditions of the oceans

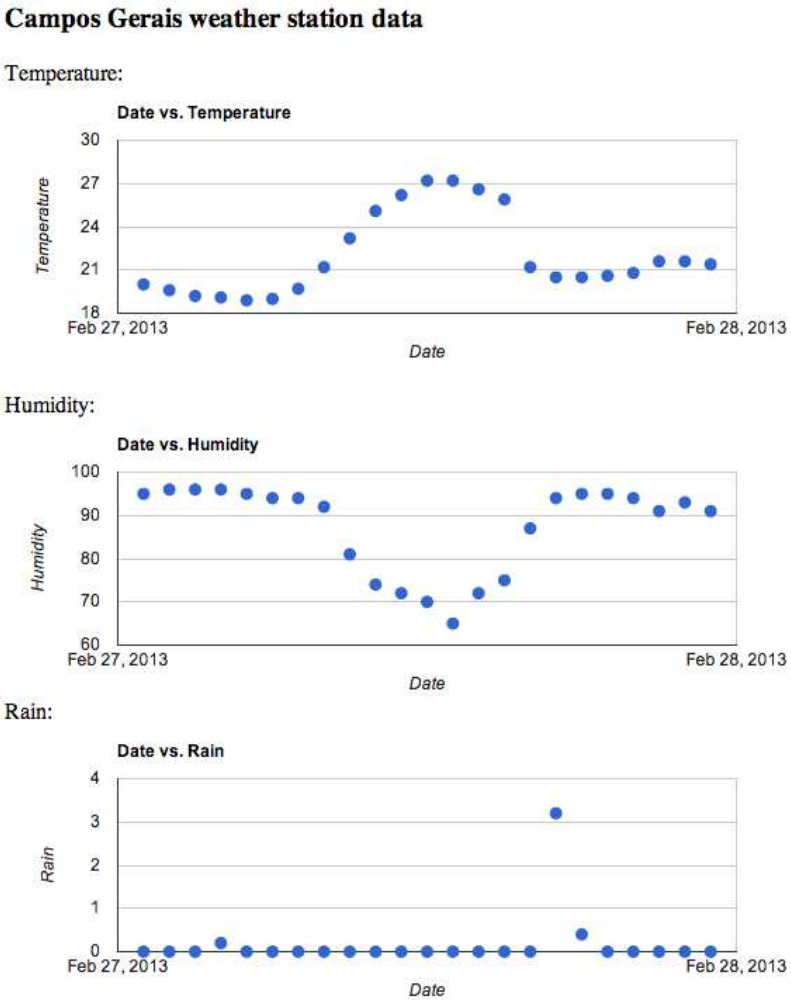

Figure 4: Measurements over 24 hours - variables selected for visualization

Patterns:

patterns

Select	ID	Description	Pattern	Owner
☐	1	Detect high temperature	select * from CooxupeData.win:length(20) having avg(temperatureOut) > 23.0	Ivo
☐	2	Detect rain	select * from CooxupeData.win:length(10) having avg(rain) > 0.0	Ivo

Figure 5: Pattern definition window - 1

be found are taken into account. The soil moisture in such regions can be calculated using techniques from [28, 29]. The value that goes into the framework is the minimum soil moisture computed (once 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. Figure 6 shows the steps to perform this task.

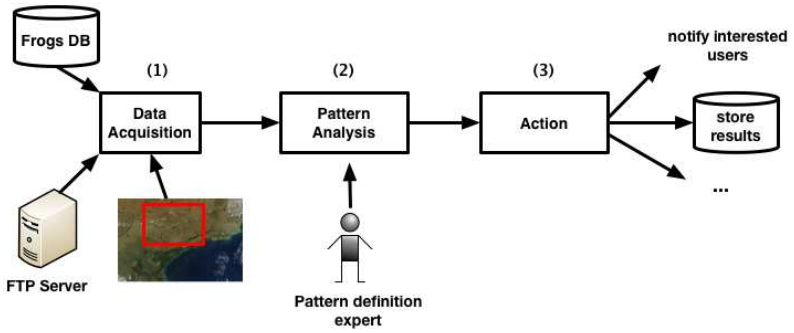

Figure 6: Frogs reproduction pattern detection.

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 scientist wants to verify what time and locations where a specific kind of frog may reproduce.

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

Considering this approach, we create an EPL statement, shown in Figure 7, to estimate what frogs are prone to reproduce for a region. To do this we join the computed values of soil moisture from satellite images, weather station meteorological measurements and frogs' information stored in a database (FrogsDB).

This pattern considers locations where frogs live to take soil moisture values computed from satellite images and Cooxupé rain measurements. 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.

Let us now explain the details of Figure 7. Lines 1 and 2 select species name, location

and the atmosphere.

```
1  SELECT
2      species_name, location, cooxupe_event.timestamp
3  FROM
4      SatelliteMoistureEvent as soil_moisture,
5      CooxupeDataEvent as cooxupe_event,
6      CooxupeStation as cooxupe_station,
7      sql:FrogsDB ['select species_name, location,
8                    soilmoisture_min, rain_min
9          FROM FrogsSpecies'] as frogDB
10 WHERE
11     soil_moisture.location = frogDB.location AND
12     soil_moisture.value > frogDB.soilmoisture_min AND
13
14     cooxupe_event.station = cooxupe_station.id AND
15     cooxupe_station.location = frogDB.location AND
16     cooxupe_event.rain > frogDB.rain_min AND
17
18     'November' < current_month_var < 'January';
```

Figure 7: Frogs' reproduction EPL.

and timestamp where each kind of frog are subject to reproduction. Line 3 to 9 depict the sources of events, i.e. satellite soil moisture computed values, meteorological station data, frogs database. Lines 11 and 12 take the soil moisture and values where there are species of frogs in the database and compare if the moisture value is greater than the values stored in FrogsDB. Lines 14 to 16 take the Cooxupé meteorological station data and compares to the frogs database values to see if the minimum rain rate is reached. Line 18 verifies if the current month is between November and January.

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.

7 Conclusion

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.

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.

References

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

[2] 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.

[3] C. Chong and S. P. Kumar, "Sensor networks: evolution, opportunities, and challenges," *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1247–1256, 2003.

[4] 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, (New York, NY, USA), pp. 141–154, ACM, 2010.

[5] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, and D. Estrin, "Habitat monitoring with sensor networks," *Commun. ACM*, vol. 47, no. 6, pp. 34–40, 2004.

[6] 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, (New York, NY, USA), pp. 214–226, ACM, 2004.

[7] 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*, vol. 101, no. 16, pp. 6039–6044, 2004.

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

[9] M.-T. Schmidt, B. Hutchison, P. Lambros, and R. Phippen, "The enterprise service bus: Making service-oriented architecture real," *IBM Systems Journal*, vol. 44, no. 4, pp. 781–797, 2005.

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

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

[12] 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.

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

[14] O. Dictionaries, "Oxford dictionaries." http://oxforddictionaries.com/definition/event?region=us, December 2012. accessed December 10, 2012.

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

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

[17] Cosm Ltd., "Cosm platform." https://cosm.com, Mar. 2013.

[18] 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*, vol. 182, pp. 589–607, Jan. 2009.

[19] J. K. Hart and K. Martinez, "Environmental Sensor Networks: A revolution in the earth system science?," *Earth Science Reviews*, vol. 78, no. 3-4, pp. 177–191, 2006.

[20] 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, (New York, NY, USA), pp. 54–62, ACM, 2001.

[21] Z. 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*, vol. 6, no. 2, pp. 95–116, 2004.

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

[23] 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*, vol. 86, no. 3, pp. 158–168, 2005.

[24] 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, (New York, NY, USA), pp. 851–860, ACM, 2010.

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

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

[27] I. Koga and C. B. Medeiros, "Managing Environmental Data from Sensor Networks," *Journal of Ecological Informatics and Computational Ecology, submitted*, 2012.

[28] T. J. Jackson, "III. Measuring surface soil moisture using passive microwave remote sensing," *Hydrological Processes*, vol. 7, no. 2, pp. 139–152, 1993.

[29] 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*, vol. 104, no. D4, pp. 4145–4168, 1999.

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