Event Processing in Sensor Webs

Thomas Everding, Johannes Echterhoff

Short Summary

The submission describes new achievements in the area of Spatio-temporal Stream Processing, a new and powerful approach for geodata processing in Sensor Webs. We introduce the Event Pattern Markup Language which can be integrated in SDIs to enable stream processing functionality and provide examples of innovative applications.

Abstract

Introduction

Geospatial data has become a prominent source of information in our everyday life. Tools like Geoportals and Geobrowsers provide access (mostly visual) to vast amounts of geodata. This data is usually stored in server plants which can be part of a Spatial Data Infrastructure (SDI). Most if not all of this data originates from observations and measurements performed by (geo-) sensors. It is therefore a vital part of our information society. The basic sensor data may be used to derive new information which itself can be subject to further processing. The entities that perform the derivation are usually automated processes but can also be humans, for example an analyst who detects a traffic jam based upon car speed readings.

Geodata is usually accessed via the Internet by pulling data from the underlying service infrastructure. This approach may be sufficient to fulfill non time critical information needs but fails to support use cases where data needs to be available in near real-time. This is especially the case for emergency and disaster management systems where rapid threat detection and response are essential. We also recognize a growing density of sensor deployments which on the one hand affords an increased awareness of our environment but on the other hand also means that we need to be able to handle an ever-growing flood of information. Our service infrastructures therefore need to provide functionality to pre-filter (new) geodata based upon given criteria and deliver it directly to us.

Information communities like the financial market apply stream processing techniques to filter and derive the information they are interested in near real-time. Another example is network security systems which apply stream processing to detect network intrusions of various kinds (Luckham 2002). We adapted these techniques to the geospatial domain. More specifically, we have created the Event Pattern Markup Language with which spatio-temporal event processing instructions can be encoded and used in SDIs. In this paper we introduce EML and show how it can be integrated into the Sensor Web Enablement (SWE) initiative of the Open Geospatial Consortium (OGC).

Related Work

Sensor Web Enablement

SWE provides a framework of open standards for the Sensor Web, a system of real or virtual sensors that are made accessible via the Internet (Moodley & Simonis 2006; Simonis 2004). The specifications defined in SWE can be divided into two parts: an information model for the encoding of (meta-) data and a service model which specifies the functionality to access sensors and sensor data. Botts et al. (2007) provide a good overview of the relevant standards and functionality. Two of the SWE specifications are of higher interest in this paper and will therefore briefly be introduced. The Sensor Model Language (SensorML) is used for encoding the characteristics and capabilities of a sensor (Botts 2007). Both physical as well as non-physical sensors (i.e. processes) can be described, along with the algorithms executed by these processes internally. In addition, the Sensor Alert Service (SAS) has been specified and is used for publishing of and subscribing to sensor data that matches given filter expressions (Simonis 2006).

Event Processing

At first the term event has to be defined. According to ISO 19136 an event is "an action that occurs at an instant or over an interval of time". These events are represented by event objects. Given this definition, Event Processing in common can be defined as creating, deleting, reading and editing of as well as reacting to events and their representations (Chandy & Schulte. 2007).

A special form of event processing is Complex Event Processing (CEP). It uses relations between multiple events to derive higher level information using event patterns that are executed upon all available events (the *event cloud*). The pattern matching can be seen as inverted database queries: instead of performing different queries on a static dataset CEP performs static queries on dynamic data in order to perform matching on new events as quickly as possible. Luckham (2002) provides an overview of the capabilities of CEP.

Another branch of event processing, Event Stream Processing (ESP), focuses on processing event objects that are ordered in time. These event objects are received in a data stream which can be of infinite size (Babcock et al. 2002). Therefore specialized techniques like data views that allow only a certain length of the event stream to be subject to queries (like time views and length views) have been developed. Because these time-ordered data streams are members of the whole event cloud, ESP can be seen as a subset of CEP (Luckham 2006).

While using CEP to derive higher level information relations between events are established. If for instance the higher level information is the average of two temperature measurements, these two measurement events are the causal ancestor of the average

event. In order to represent this causal relationship the higher level information produced by CEP is stored in complex events which store the relationship to their ancestors in a *causal vector* (Luckham 2002).

A Markup Language for Patterns on Event Streams

The Event Pattern Markup Language (EML) can be used to encode patterns for Complex Event Processing (CEP) and Event Stream Processing (ESP). It defines a set of features that is common to many pattern languages and is extensible to support custom functionality, like spatial operations. EML patterns act upon streams of incoming data and enable filtering as well as aggregation of this data. Patterns can be combined into a pattern network to construct multi stage filters and processing chains.

We defined four different types of patterns, the *Simple Pattern*, *Complex Pattern*, *Timer Pattern* and *Repetitive Pattern*. Each pattern defines rules which are matched against incoming data by a *pattern matcher*. If a match is detected a new event (possibly referencing the events that caused this new event) is fired. Matching events may also be directly delivered to assigned output destinations.

Let us describe the available patterns in more detail. Simple Patterns are used to pre-filter data from an external source. Only simple patterns can be connected to external sources, so they represent the gates through which data streams have to enter the pattern network. This network is created by linking different patterns via Complex Patterns. Those are also used to perform further processing of data generated by the input patterns they are connected to. Complex Patterns enable correlation of events and can take the relationship between different events, i.e. the causal vector, into account. Timer Patterns are used to create trigger events on a given regular interval. Repetitive Patterns also create trigger events, but only if a specific number of input events has been recorded.

Matching (a stream of) events against a pattern is performed in three consecutive processing steps, comparable to the select-from-where construct used in SQL (but in reverse order). At first the *guard* (filter) statements are evaluated against each new event. Guards are encoded using the OGC Filter Encoding (Vretanos 2005). Every incoming event that passes the guard of a pattern is inserted into the patterns *view*. A view (also called *window*) defines a subset of the events that passed the guards. Available subsetting criteria are time and / or length based (e.g. “events during the last three minutes” and “the last 20 events”). Events that do no longer match the view criteria (e.g. because they are no longer in the three minute period or have been followed by more than the allowed number of events) are removed from the view. Thus, views are important tools to prevent execution of pattern matching against a possibly infinite amount of events. Whenever a new event enters a view, the *select functions* of the pattern are executed against the events currently contained in the view. Select functions define which information is used – possibly executing additional computations to derive information (e.g. the complex hull of the contained events or an average value for a common property) – as the result of the matching. They also define whether the result is published to output destinations, in addition to being published to the pattern matcher as a new event.

We will show how EML can be applied with a simple example. Consider a temperature surveillance system that detects sensor readings exceeding a given threshold. A simple system would generate an alert every time the threshold is exceeded – for each measurement. Using EML one is not only able to detect threshold exceeding but also (or only) crossing. This can be achieved by applying two simple patterns, one looking for readings below the threshold, one looking for those above, and a complex pattern looking for readings below the threshold followed by those above. By only outputting the latter events, one can considerably reduce the number of events generated by the system. By adding an additional complex pattern that looks for situations in which one reading above the threshold is followed by one that is below, the system is enabled to detect “all clear” events.

Applications

Event stream processing based upon EML has been integrated in two SWE applications. Event patterns can be added to SensorML descriptions to define the internal workings of an existing sensor but also to configure tools to process incoming readings as defined by the pattern (network), thereby realizing a non-physical, i.e. virtual sensor.

The second application that makes use of EML is the Sensor Event Service (Echterhoff & Everding 2008). This service is an experimental successor of the Sensor Alert Service. It allows EML in subscriptions and thus enables usage of advanced CEP and ESP functionality for filtering sensor data encoded in Observation & Measurements (Cox 2007a; Cox 2007b) and sending notifications to subscribers.

Conclusion / Future Work

We have developed a markup language with which spatio-temporal stream processing instructions can be encoded. Stream processing functionality has been added to the Sensor Web, enabling more sophisticated detection of interesting events and reducing the amount of transferred data not relevant for clients. Our next steps are to extend the functionality of EML by integrating more spatial operators and defining spatial views for event patterns. We are also going to integrate EML and publish/subscribe mechanisms in traditional SDI services, bringing the building blocks for Event Driven SDIs in place.

References

Babcock, B., Babu, S., Datar, M., Motowani, R., Widom, J. (2002): Models and Issues in Data Stream Systems. Proceedings of 21st ACM Symposium on Principles of Database Systems (PODS 2002), Madison, Wisconsin, USA. online at http://ilpubs.stanford.edu:8090/535/1/2002-19.pdf (accessed on: 15.12.2008).

Botts, M (2007): Sensor Model Language Implementation Specification, OGC document number 07-000, online at http://portal.opengeospatial.org/files/?artifact_id=21273

Botts, M., Percivall, G., Reed, C., Davidson, J. (2007): OGC Sensor Web Enablement: Overview And High Level Architecture. OGC document number: 07-165, online at http://portal.opengeospatial.org/files/?artifact_id=25562

Chandy, K. M., Schulte, W. R. (2007): What is Event Driven Architecture (EDA) and Why Does it Matter? online at http://complexevents.com/?p=212 (accessed on: 25.02.2008)

Cox, S. (2007a): Observations and Measurements – Part 1 - Observation schema, OGC document number 07-022r1, online at http://portal.opengeospatial.org/files/?artifact_id=22466

Cox, S. (2007b): Observations and Measurements – Part 2 - Sampling Features, OGC document number 07-002r3, online at http://portal.opengeospatial.org/files/?artifact_id=22467

Echterhoff, J. and Everding, T. (2008): Sensor Event Service Interface Specification, OGC document number 08-133, online at http://portal.opengeospatial.org/files/?artifact_id=29576

ISO/DIS 19136 Geographic information – Geography Markup Language (GML)

Luckham, D. (2002): The Power of Events. Addison-Wesley, Boston, USA.

Luckham, D. (2006): What's the Difference Between ESP and CEP? online at http://complexevents.com/?p=103 (accessed on: 15.12.2008)

Moodley, D. and Simonis, I. (2006): A New Architecture for the Sensor Web: The SWAP Framework. Proceedings of the Semantic Sensor Networks Workshop, in Supplemental Proceedings of the 5th International Semantic Web Conference. Athens, Georgia, 5-9 November 2006. online at http://www.ict.csiro.au/ssn06/data/swap.pdf

Simonis, I. (2004): Sensor Webs: A Roadmap. Proceedings of the 1st Goettinger GI and Remote Sensing Days, Göttingen, Germany. online at: http://www.ggrs.uni-goettingen.de/ggrs2004/CD/Applications_in_Geography/GGRS2004_Simonis_G315.pdf (accessed on: 15.12.2008)

Simonis, I. (2006): OGC Sensor Alert Service Candiate Implementation Specification, OGC document number 06-028r3, online at http://portal.opengeospatial.org/files/?artifact_id=15588

Vretanos, P. A. (2005): OpenGIS Filter Encoding Implementation Specification. OpenGIS Implementation Specification, OGC 04-095, online at http://portal.opengeospatial.org/files/?artifact_id=8340