Event model to facilitate data sharing among services

Olivera Kotevska, Ahmed Lbath, Judith Gelernter

Event model to facilitate data sharing among services

Closing the Gap between Research and Implementation

O. Kotevska, A. Lbath*, J. Gelernter
National Institute of Standards and Technology
University of Grenoble Alpes, France*
{olivera.kotevska, gelern} @nist.gov, ahmed.lbath@imag.fr*

***Abstract*—Development of smart city services is presently hindered because the data is too heterogeneous, despite the increasing availability of data in open data initiatives. We determined metadata fields and semantics for our proposed model after a survey of other event models, and after its trial implementation in actual applications. Our event metadata model is unique among event models in being extensible as needed. We illustrate our model by showing it with different data types, and we validate it using real-world data in a prototype smart cities service for pedestrian safety. Wide adoption of our event metadata model has the potential to broaden the number and scope of smart city services.**

***Keywords*—Complex Event Processing, CEP, event metadata model, event model, smart city service**

I. INTRODUCTION

Problem. Half of the world population lived in cities by 2013. Historically, people live in and work for their cities; we want to make the cities work for them. Or rather, to make their devices work for them by creating city service applications. Complex event processing applications correlate data streams in real time as events occur, using pre-defined rules to identify events of interest. A common data structure will allow different data streams to be used by multiple services more easily.

Others' approaches. Event metadata models for Complex Event Processing provide the time and data backbone to allow those data streams to work together. Event models tend to be rich in metadata fields which may or may not be relevant to the service at hand. The impracticality of the metadata-rich models is not recognized because use cases are rare. Also, when data is provided, it is often simulated to match a model – which does not demonstrate whether or not a model is robust for a particular data stream.

Our approach. Our event model defines only necessary fields, and gives parameter wildcards for potential metadata expansion in three categories (*EventProfile* for the data stream, *EventSource* for the device that provides the data stream, and *Event*). These parameters are added based on the service by the developer at the time of model implementation. Our model is validated in a smart cities case study.

Significance. Developers' adoption of a standard event model – that is, a package that any data could fit into for delivery into applications in the upper layer of the Internet – could advance smart city services. Different services might access the same data concurrently, so the model would facilitate data sharing as well as integration.

Case study in city safety. We created an algorithm for pedestrian safety using real-world data, and then supplied the algorithm with data from our event model to test our model's usefulness.

We devised the case study in answer to an on-going need by local government to improve safety in Montgomery County, Maryland, U.S.A. In 2007, a Pedestrian Safety Initiative was introduced in Montgomery County that used education, streetscape and road signs, as well as traffic calming and road rule enforcement to lower the number of accidents.[1] Our application is intended for county representatives, and adds to Pedestrian Safety Initiative by showing where the most accidents are predicted to occur. The county can then better deploy resources to change street signs or deploy traffic police, for example, to heighten safety at the times and locations predicted to be unsafe.

This Introduction is followed with Section 2 on Background and Related work. Section 3 presents our event model; in Section 4 we demonstrate evaluation scenario; in Section 5 we discuss about our results and Section 6 summarizes the conclusions and future work.

Our contributions are (1) a generic, highly-scalable semantic model as a core component of complex event processing architecture, (2) validation of the proposed event model in an algorithm for a smart cities problem of pedestrian safety, and (3) bringing to the attention of the research community an Internet of Things testbed that includes both sensor and human data.[2]

[1] http://www.montgomerycountymd.gov/DOT-PedSafety/overview.html; https://volunteer.truist.com/mcvc/org/opp/10610402110.html

[2] https://data.montgomerycountymd.gov/, with data updated regularly

II. Background and Related Work

A. Complex Event Processing (CEP)

Complex event processing is a strategy to create applications, or *services* in which data streams containing events produced by *resources* such as people, devices or sensors are filtered for particular event changes of interest, called *instances*. These changes then automatically trigger some response. Events are given names so that they can be shared among services. To enable sharing, common metadata and sometimes also vocabulary for the names comes from *ontologies*.

The diagram below shows event data in basic Complex Event Processing. Data is received from multiple streams, events are detected, and an appropriate response is triggered. This sequence diagram the Figure resembles the standard event driven architecture proposed by Fujitsu [11], Microsoft [5], IBM [20] and Oracle [15].

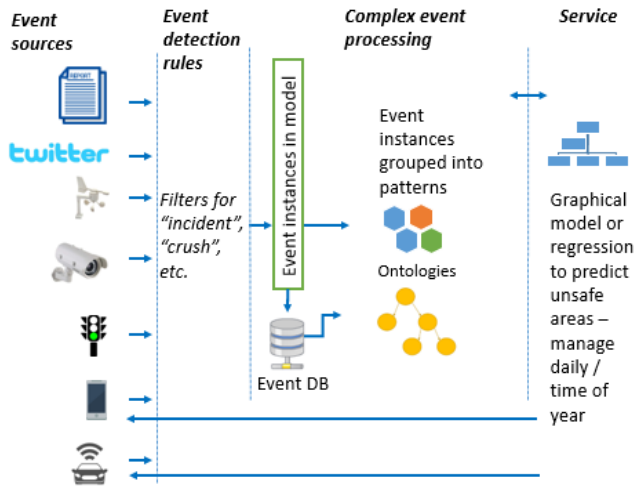

Fig. 1. Complex event processing architecture, with arrows showing event data sequence direction. An event model is a component.

Note that the vertical box in Fig. 1, "Event instances in model," where the data is organized to be stored in database server or event cloud, is central to the process.

Queries to the event processing system might be in SPARQL for RDF, TESLA–a formally defined Event Specification Language [6], or EP-SPARQL [2], or in Event Processing Language,[3] or Big Data Processing Language (BDPL) [24].

These rules have the ability not only to indicate events relevant to the application, but also when data is faulty or noisy or missing information, or includes non-events [1]. Response to data uncertainty can make the system more robust. For example, events missing values can be marked incomplete, and system maintenance would analyze marked records to determine what had gone wrong.

[3] https://docs.oracle.com/cd/E13157_01/wlevs/docs30/epl_guide/overview.html

B. Alternatives to this architecture

The event model is one part of the CEP architecture. Other proposed middleware solutions ensure that low level signals can work together without use of an event model. For example, the IoT Semantic Smart Gateway Framework [14], and the Smart M3 open source software platform that provides infrastructure between resources [23]. Even these solutions require common terms so that different devices can communicate – in what makes the solutions "smart".

C. Event model in Complex Event Processing

An event model is core to Complex Event Processing architecture, so it is understandable that more than a dozen event models have appeared in the academic literature since 2000. Some are specific to data type or domain type, or must be paired with certain controlled vocabularies in ontologies, or with particular query languages.

Event models specify metadata fields for the data streams that carry or pertain to events. The most basic metadata fields for event models are data type, and time. The idea of the model primitives is that the basic data message is independent of any application. What metadata fields go beyond the primitive? Authors from [17] explain that data fields such be considered during the "thought process" of setting up a system. This is the logic we follow when specifying "Parameters" for our model. Other researchers [19] assume that more primitive are better. See the comparison chart with evaluation metrics in the Appendix.

D. Role of the ontologies in event models

Ontologies are required in event models such as Event F [19][19], OntoEvent [16], REseT [25], and LODE [21]. In some cases, as in OntoEvent and REseT, the ontology was created specifically for the event model.

The metadata fields in our model, for the most part, align with those in the DOLCE ontology (See Fig.1) That will make it easier for data in different streams to fit into the same event model, and it will help insure interoperability. These event models are called semantic because the meaning of the data field titles is essential to the model.

Requiring an ontology on a particular topic for an event model can restrict data types that can be used, as well as the domain. Authors in [13] have a method to develop an ontology for traffic violations. In our particular application, the World Wide Web Consortium is developing a Traffic Event Ontology, which we could use for at least one data stream, but this ontology is not available as of the writing of this paper.[4] A domain-specific ontology might be useful to extract events from a natural language data stream, but it is probable that Named Entity Recognition tools such as the Stanford NER

[4] https://www.w3.org/community/traffic/

toolkit would do well in extracting proper nouns and locations required.[5]

E. How event models have been validated

Many event models in the academic literature have not been validated; the models are either entirely theoretical, or use data that has been created to fit the model. EventShop is an event processing platform that can be used to validate components of the architecture. It accepts data streams and includes modules to query the data that will isolate event instances relevant to the application. Their events are put into a grid structure called E-mage, that can integrate events from heterogeneous data streams [17].

By contrast, to validate our event model, we have devised a service that incorporates multiple data streams. We have vetted our event model when real-world data streams fit the model and then are accepted into the service.

III. Our Event Model

Our model shown in Fig. 2 below in Unified Modeling Language (UML) has three classes: EventProfile, EventSource, and Event. Location metadata is required but is not in the diagram because location metadata changes classes depending upon the service. For stationary devices, "Location" is stored with the device information, whereas for mobile devices, location will be needed continually to correspond to events. Metadata fields within the three classes of our event model are defined below.

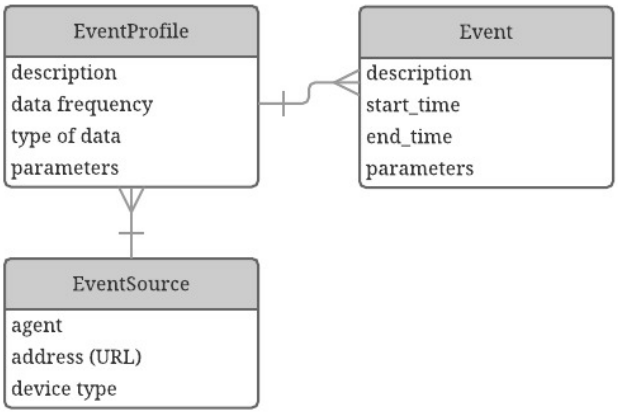

Fig. 2. Our model for event data shown in Universal Modeling Language (UML). EventProfile and EventSource parameters could include data rate for system failure checking, and parameters could include location for devices that are mobile; otherwise, location would be a parameter for Event.

Our model is extensible. Some services will require only its specified metadata fields, whereas other services will require consideration of which parameters are relevant. *EventProfile* and *EventSource* send their data at system start or re-start, whereas the Event class sends data continually. Data from the Event class is filtered using rules or machine learning into instances relevant to the service.

The *EventProfile* class holds metadata about the data stream. "Description" explains what data is being stored for services. "Data frequency" presents the rate data streams are expected to be received, while "type of data" provides information about the data that has been collected. "Parameters" represent any additional information related to the data stream that are important, like measurement type, expected states, or frequency.

The *EventSourc*e class holds the data related to devices or sensors. It has the following attributes:" Agent" provides information for the event source, like county police or weather channel. "Address" is the address from where data was collected or original source of observed data. "Device type" is the type of device, such as temperature sensor, mobile device etc. "Parameters" represents additional information related to the device, such as serial number, model, and battery expiration date.

The *Event* class holds the data related to events in the data stream, like temperature, heart rate, pedestrian movement, or car accident. It has the following attributes: "Description" is auto-filtered from an event text stream or is set up beforehand to describe the events. "StartTime" is the date and time when the event started, "EndTime" is the date and time when event stopped. The frequency of "StartTime" and "EndTime" can be set by adding a parameter to the *EventProfile*. Parameters for the event represent additional information related to the event that are useful for a particular service. The events could be sent in as received, or they could be aggregated and then sent at some pre-determined frequency.

*Ontology***.** The system designer should choose an ontology used by the other applications for wide data sharing among applications. Rather than share or link ontologies [12], an upper level ontology such as DOLCE could be used to bridge domain ontologies.[6] DOLCE includes reasoning without controlled vocabulary, and it was used in the F event model [19]. In aligning our model with DOLCE, Feature is the category for our Description, and Physical Quality is the category for Type.

Uniqueness and benefits. We conducted an extensive survey of event models and found that our event model differs from others in two main aspects: (1) fewer required metadata fields, and (2) some of the metadata is sent once only. Fewer metadata fields allows the model to be maximally re-usable for different data and domain types, although it means that some thought will be needed each time the model is implemented. Sending some metadata once only lessens the data load on the network somewhat to speed processing.

Instances in the event stream. Recall that instances or occurrences are the events being monitored for. A script apart from the model filters the event data stream to recognize instances relevant to the application, and to set bounds on what belongs in the event model. For example, an expected range of values for some metadata field could be specified, and data that

[5] http://nlp.stanford.edu/ner/

[6] http://www.loa.istc.cnr.it/old/q.html

went beyond that range would be dismissed. Filtering for the model also could be handled on a data stream-by-stream basis so that a data stream that did not flow at an expected frequency, might indicate sensor or device malfunction.

IV. Case Study

This experiment shows how different forms of event data and metadata in the model are suitable for a smart cities application.

A. Objective

We want to use our event model to supply data for an actual application. Our research question is: *Is it possible to show the influence of time, weather, and community happenings on pedestrian safety incidents in a given location by postal zip code?*

B. Use case and data for the experiment

Events for 2015 traffic incidents, weather, and community happenings for Montgomery County, Maryland are posted on the open web.[7] The county updates the data regularly. We intend to supplement the Pedestrian Safety Initiative in Montgomery County, Maryland, by offering an algorithm to predict unsafe event areas according to zip code. To show differences in time-of-day, the service could output predicted locations every few hours.

C. Method

Event processing methods work well for passive data collection because they allow large quantities of data to be processed if necessary in different locations (also called distributed systems). Event processing also is scalable, and using corrective rules can help make the system fault tolerant. For example, only accept into the model if the time is between 1pm and 6pm Eastern Time.

Getting data into the model. Our data included road violations, weather, and community happenings. Of these, the road violations and community happenings include a lot of text. But event recognition in text data can be difficult due to human language ambiguity. To recognize which events were relevant instances, we experimented by comparing a non-domain specific ontology, the Freebase knowledge-base, and a classifier we trained on a portion of the data. We found that the classifier was the most accurate filter to find relevant instances.

Handling data uncertainty. Uncertainty in the general case could be minimized with the combination of rules (for instance, only accept into the model if it includes a beginning time). Specifically, in our application, we lacked zip codes for some of the community happenings that were city-wide (such as elections), so for these, we entered zip codes for the entire city. Also for the case of missing data in the location fields, we used alternative data entry, like longitude and latitude and get the zip code and city information.

[7] https://data.montgomerycountymd.gov/

Example of data in the model. The following Figures 3,4,5 show how community happenings, weather and traffic violation metadata fit into our event model. We took this data from the Montgomery County open data website for this case study, but in a real time application, the *EventSource* would provide the actual source of the data flow with its parameters.

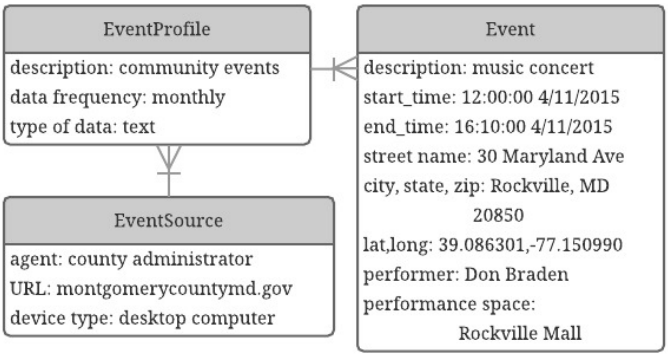

Fig. 3. Community happening metadata in our event model

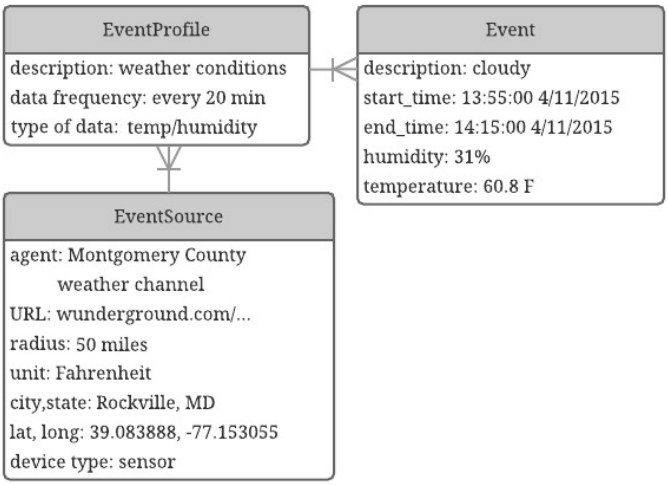

Fig. 4. Weather condition metadata in our event model

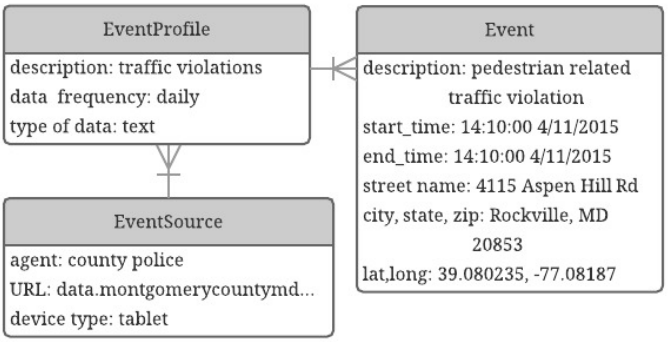

Fig. 5. Traffic violation metadata in our event model

Our metadata fields correspond to categories in the DOLCE ontology. DOLCE allows natural language descriptions, so we leave our text data streams as sent.

D. Data analytics and findings

The proposed method predicts which county region by zip code will be safest based on the past number of pedestrian incidents per zone. Analysis is performed using the R software environment for statistical computing and graphics. We experimented with both Poisson Regression and Probabilistic Graphical models (PGM) to determine which was more suitable for prediction with our data. Poisson regression is used for point data, such as we have, and it is commonly used for predicting the probability that an event will occur based on several predictor variables that may either be numerical or categorical. PGM has the ability to represent dependencies among events on a graph. The table shows that the statistical method selected, whether a PGM or the Poisson Regression, is not a significant factor in predictive accuracy.

TABLE I. ACCURACY OF TWO METHODS TO PREDICT UNSAFE LOCATIONS

	Probability of a (negative) pedestrian event in a location	
	Probabilistic Graphical Model	***Poisson Regression***
Average accuracy	0.864	0.869

E. Evaluation and visualization

To show the relative effectiveness of the pedestrian safety algorithm on data from the event model, two means of evaluation are used: comparative maps (Fig.6 and 7), and the confusion matrix as performance measures for accuracy calculated in Table 1. Figure 6 show maps for 2015 actual pedestrian incidents for the period from January to March and incidents predicted for the same period using Probabilistic Graphical Models.

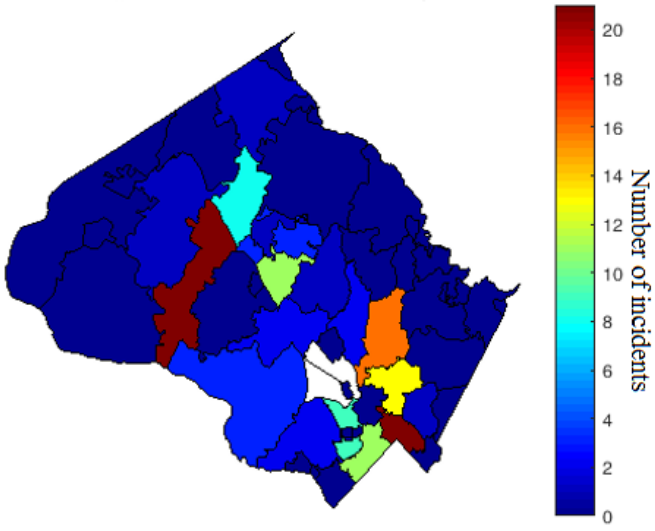

Fig. 6. Actual traffic violations in Montgomery County, Maryland related to pedestrian safety events January –March 2015 by zip code

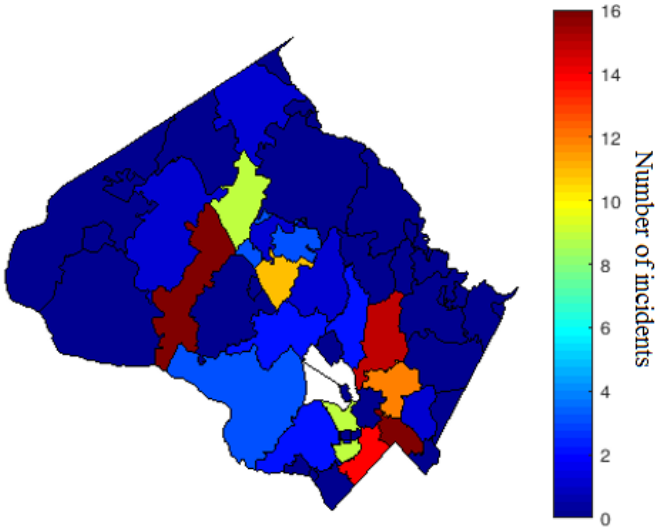

Fig. 7. Predicted traffic violations in Montgomery County, Maryland, related to pedestrian safety events January-March 2015 by zip code

Comparing output colors in the actual (top map) and predicted (bottom map) shows that our algorithm achieved excellent results. We split the 2015 data into four 3-month segments and show the map pair only for the first 3-month segment due to space limitations, even though mapped results throughout the year were equally good.

The high quality mapping results were confirmed by the .86 predictive accuracy shown in Table 1 of both the Probabilistic Graphical Model and the Poisson Regression. Based on the fact that the different data streams fed easily from the event model to the pedestrian safety algorithm shows that the event model is robust to at least three data types.

In creating an algorithm, we tried several statistical models for prediction. We evaluated results by creating a mapped output comparing actual pedestrian events to predicted events (Fig. 6 and 7), and by using the accuracy statistic averaged over a year of predictions (Table 1). The accuracy match between our prediction model and the real world data shows our service using data from our event model is accurate to .86 using either Probabilistic Graphical Models or a Poisson Regression. We can extend the algorithm to predict time as well as location.

Our pedestrian safety algorithm is limited by the fact that we do not have a record of pedestrian safety incidents in the county for 2015 that is entirely complete since we are limited to data that is available publically. However, it is strengthened by the fact that we found event instances to a high degree of accuracy. In a real-time application, there will likely be additional errors identifying event instances.

F. How government officials could use the system

Data from the Montgomery County, Maryland open government stream in JSON/xml first would be set to flow into the application. We would create a web-based interface for the application that would be active around the clock.

Users could examine the county map by zip code in the interface to see which areas in the county are having pedestrian incidents at that time (as in Fig. 7), and can then make informed decisions where county resources should be deployed to heighten safety. Our experiments have shown that the prediction algorithm is up to 86% accurate, and so the officials should feel confident that the resources would be deployed effectively. It would straightforward for us to add analytics to the interface as well, so that the officials could see trends over time.

V. Discussion

A. Semantic model for event processing architecture: toward a standard

Numerous event models in the literature vary with respect to required data fields and the role of ontologies. It is rare to find research that discusses deploying an event model: how to get data in and out, and how to handle missing data, for example. That makes it impossible to "plug in" other event models into our architecture for comparison. This paper has covered both the data modeling and some flexibles rules to guide deployment of our event model in a city service.

Our model requires a minimum of metadata fields, and it is efficient, minimizing the amount of network traffic by linking a continuous time dependent data stream to non-time dependent meta-information stored on an event server or cloud.

Overlap among different types of data sources should be handled by an upper level ontology such as DOLCE. We made the data field types in our model align with DOLCE categories in order to facilitate getting data from different streams into the same model.

A one-set-of-metadata-fits-all model would be more convenient than ours, but it would not necessarily capture what each particular service requires. That is why some thought will still be required to add parameters to suit the service being implemented.

Relying on a standard event model will make it easier for developers to build Complex Event Processing systems. In that our event model is generic in suiting a variety of data streams and a variety of domains, we propose it as basis for an event model standard.

B. Event model in Complex Event Processing architecture

Event models in complex event processing diagrams are ubiquitous. Actual systems that perform event processing are less common, and in fact, input of heterogeneous data fields into a single system is still coming into focus. The event model fits different data types into the same framework, thus making data integration easier.

Getting data into the model for highly structured can be as simple as matching metadata field names. But when the data flow does not include metadata, or when it is a natural language stream -- as in the police reports used here -- Natural Language Processing tools will employ data extraction. The most recent Named Entity Recognition (NER) tools to extract person, organization, location, and other objects that are proper nouns, are well advanced, achieving accuracies in the 90% range.

C. Publically-available, updated Internet of Things testbed

We applied Internet of Things data to help solve an actual problem: determine which areas of Montgomery County, Maryland require more monitoring and local resources to improve pedestrian safety. We used that algorithm to test our event model.

Few Internet of Things data sets at the time of writing are publically available for experimentation. It is not surprising, therefore, that we found that few of the research papers on event models use real-world data. Many papers are entirely theoretical, and some use synthetic data sets which were created or assembled from the web for the purposes of demonstrating a point.

Internet of Things data sets are obscured from a general web search because they are not referred to as Internet of Things, or sensor data – the keywords we use in computer science literature. The scientific community should look for human and sensor data relevant for Internet of Things research in open city projects. Our data come from Maryland, and New York has an even broader data set,[8] and the Open Data Portal has data from around the world.[9]

VI. Conclusion

An event model with an ontology to control word descriptions will bridge smart cities services by reusing data streams across applications. This approach will make applications more financially viable and expand the potential number and scope of smart city services. Our event model should be tested with more data stream types. Even better to test the event model would be to create services complementary to this in pedestrian safety to ensure that the same data can be used in multiple ways.

Acknowledgment

We are grateful for Maxence Lefort's advice on the design of our event model. The work was done in conjunction with the Information Technology Laboratory of the National Institute of Standards and Technology. The identification of any commercial product or trade name does not imply endorsement or recommendation by the National Institute of Standards and Technology, nor is it intended to imply that the materials or equipment identified are necessarily the best available for the purpose.

References

[1] Alevizos, E. Skariatidis, A., Artikis, A., Paliouras, G. (2015). Complex event recognition under uncertainty: a short survey.

[8] https://data.ny.gov/

[9] dataportals.org/search

Workshop Proceedings of the EBDT/ICDT 2015 Joint Conference, March 27, 2015, Brussels, Belgium. [7 p.]

[2] Anicic, D., Fodor, P. Rudolph, S., Stojanovic, N. (2011). EP-SPARQL: a unified language for event processing and stream reasoning. In S. Srinivas, K., Ramamrithan, A. Kumar, M.P. Ravindra, E. Bertino, and R. Kumar Eds. *Proceedings of the 20th International Conference on World Wide Web WWW '11*, 635-644.

[3] Baekgaard, L, (2002) Event modeling in UML. Issues & Trends of Information Technology Management in Contemporary Organizations. Ed. Mehdi Khosrow-Pour (Information Resources Management Association, USA) [3 p.]

[4] Chen, H., Finin, T., & Joshi, A. (2005). The SOUPA ontology for pervasive computing. In Ontologies for agents: Theory and experiences (pp. 233-258). Birkhäuser Basel.

[5] Chiu, O. (2014). Announcing Azure Stream Analytics for real-time event processing. https://azure.microsoft.com/en-us/blog/announcing-azure-stream-analytics-for-real-time-event-processing/. Last time accessed: 26/02/2016.

[6] Cugola, G., Margara, A. (2010). TESLA: a formally defined event specification language. In *Proceedings of the Conference on Distributed Event-Based Systems (DEBS),* Cambridge, United Kingdom, 50-61.

[7] Doerr, M, Ore, C.E. & Stead, S. The CIDOC conceptual reference model: a new standard for knowledge sharing. In Conceptual modeling. Australian Computer Society, Inc., 2007.

[8] Ekin, A, Tekalp, A. M, & Mehrotra, R. Integrated semantic-syntactic video modeling for search and browsing. IEEE Transactions on Multimedia, 6(6), 2004.

[9] Fowler, C., & Qasemizadeh, B. (2009). Towards a common event model for an integrated sensor information system. In 1st International Workshop on the Semantic Sensor Web (SemSensWeb), [13 p.]

[10] Francois, A. R. J., Nevatia, R, Hobbs, J, & Bolles, R. C. VERL: An ontology framework for representing and annotating video events. IEEE MultiMedia, 12(4), 76-86.

[11] Fujitsu Develops Distributed and Parallel Complex Event Processing Technology that Rapidly Adjusts Big Data Load Fluctuations Online. Available: http://www.fujitsu.com/global/news/pr/archives/month/2011/20111216-02.html. Last time accessed: 26/02/2016.

[12] Gyrard, A., Serrano, M., Atemezing, A. (2015) Semantic web methodologies, best practices, and ontology engineering applied to Internet of Things. *IEEE Second World Forum on Internet of Things, Milan, Italy*, 412-217.

[13] Jiang, Y., Xu, Z., Wang, X. (2014). The construction of ontology in the area of traffic violations. J.J. Park et al (Eds). *Future Information Technology. Lecture Notes in Electrical Engineering 309*, 379-384.

[14] Kostis, K. and Katasonov, A. (2013). Semantic interoperability on the Internet of Things: The Semantic Smart Gateway Framework. *International Journal of Distributed Systems and Technologies 4*(3), 47-69.

[15] Kress, J., Maier, B., Normann, H., Schmeidel, D., Schmutz, G., Trops, B., Utschig-Utschig, C., Winterberg, T. (2016). Event-Driven SOA. http://www.oracle.com/technetwork/articles/soa/ind-soa-events-2080401.html. Last time accessed: 26/02/2016.

[16] Ma, M., Wang, P., Yang, J., Li, C., (2015). OntoEvent: An ontology-based event description language for semantic complex event processing. J. Li and Y. Sun (Eds). *WAIM 2015, LNCS 9098*, 448-451.

[17] Pongpaichet, S., Singh, V.K., Gao, M., Jain, R., (2013). EventShop: Recognizing situations in web data streams. *WWW2013 Companion, May 13-17, 2013, Rio de Janeiro, Brazil*, 1359-1367.

[18] Raimond, Y. Abdallah, S., Sandler, M., Giasson, F., (2007). The music ontology. Austrian Computer Society, [6 p.] http://raimond.me.uk/pubs/Raimond-ISMIR2007-Submitted.pdf

[19] Scherp, A., Franz, T. Saathoff, C. and Staab, S. (2009). F—A Model of Events based on the foundational ontology DOLCE+DnS Ultralite. *Knowledge Capture, '09, September 1-4, Redondo Beach, California*, 137-144.

[20] Selman, D., Ritchie, A., (2015). Event Driven Architecture and Decision Management. IBM's Operational Decision Manager. Available:https://developer.ibm.com/odm/docs/odm-capabilities/odm-advanced-decision-server-insights/event-driven-architecture-and-decision-management/. Last time accessed: 02/26/2016.

[21] Shaw, R., Troncy, R., Hardman, L. (2009). LODE: Linking Open Descriptions of Events in *'The Semantic Web*', Springer Berlin / Heidelberg, pp. 153-167.

[22] Shanahan, M. (1999). The event calculus explained. In Artificial intelligence today (pp. 409-430). Springer Berlin Heidelberg.

[23] Smirnov, A., Kashevnik, A., Shilov, N., Balandin, S., Oliver, I., Boldyrev, S. (2011). Development of the on-the-fly ontology matching model for smart spaces. In *Consumer Communications and Networking Conference (CCNC), 9-12 January, Las Vegas, Nevada, U.S.A.,* 808-809.

[24] Stühmer, R., Vergianadis, Y., Alshabani, I. Morsellino, T., Aversa, A. (2013). PLAY: Semantics-based event marketplace. *14th IFIP WG 5.5 Working Conference on Virtual Enterprises, PRO-VE Sept 2013, Dresden, Germany*.

[25] Uma, V., and Aghila, G. (2014). Event order generation using Reference Event based qualitative Temporal (REseT) relations in Time Event Ontology. *Central European Journal of Computer Science 4*(1): 12-29.

[26] Wang, X, Mamadgi, S, Thekdi, A., Kelliher, A., & Sundaram, H. Eventory - an event based media repository. In Semantic Computing. IEEE, 2007.

[27] Westermann, U, and Jain, R. (2007). Toward a common event model for multimedia applications. IEEE multimedia 14(1): 19-29.

[28] Zhong, Z., Liu, Z., Li, C., & Guan, Y. (2012). Event ontology reasoning based on event class influence factors. International Journal of Machine Learning and Cybernetics, 3(2), 133-139.

Appendix: Comparison chart for event models *

	Processing				Required fields								Other requirements	
Event Model	Raw data into model	Handle missing data	Handle uncertainty	Data types	Static/dynamic	Action/verb/event	Participant/actor/subject/Agent	Object	Time	Location	Device associated	Device affected	Ontology	Processing language
Event Modeling in UML (Baekgaard, 2002) [3]	ns	ns	ns	ns	ns	Y	Y	Y	ns	ns	ns	ns	no	no
Eventory (Wang et al, 2007) [26]	Y	ns	ns	ns	ns	Y	Y	ns	Y	Y	ns	ns	no	no
Event ML (IPTC, release 2014)	ns	ns	ns	ns	ns	Y	Y	ns	Y	Y	ns	ns	no	XML
SsVM (Ekin et al, 2004) [8]	ns	ns	ns	video	ns	Y	Y	Y	Y	Y	ns	ns	no	SQL
Event E (Westermann et al, 2007) [27]	ns	ns	Y	Multi-media	ns	Y	Y	ns	Y	Y	ns	ns	no	no
Event F (Scherp et al, 2009) [19]	ns	Y	ns	Any	ns	Y	Y	ns	Y	Y	ns	ns	DOLCE+DnS	RDF/XML
OntoEvent (Ma et al, 2015)	ns	ns	ns	Location, Common data types	ns	Y	Y	ns	Y		ns	ns	Created by them	OntoEvent lang.
REseT (Uma et al, 2014) [25]	Y	ns	ns	Common data types	ns	Y	Y	ns	Y	Y	ns	ns	Created by them	DL
Common Event Model (Fowler et al, 2009) [9]	ns	ns	ns	Location, common data types	ns	Y	Y	Y	Y	Y	Y	ns	Created by them	RDF/XML
Event ontology (Zhong et al, 2012) [28]	ns	ns	ns	Common data types	ns	Y	Y	ns	Y	Y	ns	ns	Created by them	RDF/XML
VERL (Francois et al, 2005) [10]	ns	ns	ns	Multi-media	ns	Y	Y	Y	Y	Y	ns	ns	VEML 2.0/OWL	VEML/OWL-DL
CIDOC CRM (Doerr et al, 2007) [7]	ns	ns	ns	ns	ns	ns	Y	ns	Y	Y	ns	ns	ISO 21127	XML
SOUPA (Chen et al, 2005) [4]	ns	ns	ns	Common data types	ns	Y	Y	ns	Y	Y	Y	ns	COBRA-ONT/OWL	RDF/XML
LODE (Shaw et al, 2009) [21]	Y	ns	Y	ns	ns	ns	Y	ns	Y	Y	ns	ns	Created by them	RDF/XML
EventOntology (Raimond et al, 2007) [18]	Y	ns	ns	Multi-media	ns	Y	Y	Y	Y	Y	ns	ns	Created by them	RDF/XML
Event Calculus (Shanahan, 2001) [22]	ns	ns	ns	ns	ns	Y	ns	ns	Y	Y	ns	ns	no	Some logic language
Ours (Kotevska et al, 2016)	Y	Set by coder	Y	any	Y	Y	no	no	Y	Y	Y	no	any	any

*ns = not specified, Y = yes