Managing Uncertain Complex Events in Web of Things Applications

Nathalie Moreno[1], Manuel F. Bertoa[1], Gala Barquero[1], Loli Burgueño[1], Javier Troya[2], Adrián García-López[1], and Antonio Vallecillo[1]

[1] Universidad de Málaga, Spain
{moreno,bertoa,gala,loli,adrian,av}@lcc.uma.es
[2] Universidad de Sevilla, Spain jtroya@us.es

Abstract. A critical issue in the Web of Things (WoT) is the need to process and analyze the interactions of Web-interconnected real-world objects. Complex Event Processing (CEP) is a powerful technology for analyzing streams of information about real-time distributed events, coming from different sources, and for extracting conclusions from them. However, in many situations these events are not free from uncertainty, due to either unreliable data sources and networks, measurement uncertainty, or to the inability to determine whether an event has actually happened or not. This short research paper discusses how CEP systems can incorporate different kinds of uncertainty, both in the events and in the rules. A case study is used to validate the proposal, and we discuss the benefits and limitations of this CEP extension.

1 Introduction

The Internet of Things (IoT) is a system of interrelated mechanical and digital devices, computing objects, and people, provided with unique identifiers and the ability to autonomously communicate and transfer data over a network. The Web of Things (WoT) aims at providing approaches, software architectural styles and programming patterns to build the IoT in an open, flexible, and scalable way.

Processing and analysing the steadily growing number of information sources that continuously produce and offer data in any complex system is one of the current challenges the WoT faces. To address this issue, *stream processing systems* are becoming widespread, specially in the IoT domain [8,9], where applications should process and react to events arriving from various kinds of sources including wireless sensor networks, RFID devices, GPS, etc.

Complex Event Processing (CEP) is a kind of stream-processing system for analyzing and correlating streams of information about real-time events that happen in a system, and deriving conclusions from them [3,7,12,13]. A distinguishing feature of CEP, not present in most stream-processing systems, is that it permits defining complex events or patterns on top of the primitive events, to identify elaborate meaningful circumstances and to respond to them as quickly as possible. Such event types and event patterns are defined using Event Processing Languages (EPLs).

When dealing with physical systems and networks, the events are not free from uncertainty. It may be due to different causes, including unreliable data sources and networks, measurement uncertainty, or to the inability to determine whether an event has actually happened or not. Some rules may also have some associated uncertainty, when we are not 100% confident on them. Other authors, e.g. [1,16,15,4], have addressed these issues, using different techniques and covering some of the aspects related to the representation, management and propagation of uncertainty. However, their proposals usually suffer from two main limitations. First, they normally focus on particular aspects of uncertainty in CEP systems, with only partial coverage of the problem [1]. Second, propagation of attributes' uncertainty through operations is manually done, which poses the burden of such a cumbersome task on the system modeler.

This paper identifies and classifies different kinds of uncertainties that may happen in CEP systems, and discusses how to incorporate them into CEP events and rules. More precisely, we address uncertainty in the occurrence of the events, on the values of their attributes (including their timestamps), and on the confidence of the derivation rules. We make use of an extension of UML and OCL basic datatypes [14,2] to represent uncertain values and to transparently deal with measurement uncertainty, greatly simplifying these tasks. A case study is used to describe the proposal, and an implementation on top of the Esper language [6] is also presented.

2 Preliminaries

2.1 Complex Event Processing

CEP [7,12] is a form of Information Processing [3] whose goal is the definition and detection of situations of interest, from the analysis of low-level event notifications [5]. Here we use the term *simple* events to refer to the low-level primitive event occurrences, and *complex* events to those that summarize, represent, or denote a set of other events. Complex events are derived by *rules* that define the relevant *patterns* of (simple or other complex) events, their contents, and their temporal relations. Something important to note is that most CEP systems share a common structure in their rules:

- A **Selection** phase that identifies first the occurrence—in the selected window—of the source events (simple or complex) that will trigger the rule. These events constitute the *antecedents* of the derived event produced by the rule.
- A **Matching** phase that decides whether values of the attributes of the selected events fulfill the rule requirements, and evaluates the combination of all rule conditions (using boolean operators `and`, `or`, `->`, etc).
- A **Production** phase that generates the derived event and calculates the values of its attributes. This step may also include the computation of aggregated values of source events using sums, averages, etc.

Although several CEP systems and languages exist, they all share the same basic concepts, mechanisms and structure. In this paper we will use Esper [6] to write the run and perform the executions.

2.2 Uncertainty in CEP

There are different kinds of uncertainty that may happen in CEP systems. Let us describe them using the common structure of the rules. Starting with the selection phase:

- Uncertain events in the stream: missing events in the stream, despite the fact that they actually happened (false negatives, FN); or events in the stream that were wrongly inserted (false positives, FP).
- Lack of precision in the values of the attributes of the basic events in the stream, including lack of precision in the events timestamp [18], due to imprecision of the measuring methods or tools (measurement uncertainty, MU).

In the Matching phase:

- Lack of precision due to uncertainty of comparison operators (`=`, `<`, `>`, `->`,...) between uncertain values of attributes of matched events.
- Lack of precision due to uncertainty of logical composition operators (`or`, `and`, `not`) between uncertain statements.

In the production phase:

- Lack of precision in the values of the attributes of derived events, due to the propagation of uncertainty in their computation from the events' attributes.
- Lack of precision in the probability of the occurrence of the derived event, due to incomplete or erroneous assumptions about the environment in which the system operates, which many influence the confidence of the rule.

Here we are interested in assigning probabilities to derived events, expressing the *confidence level* on their occurrence, and on the values of their attributes. Note that the probability we assign to an event does not mean how probable it is to happen in reality—something that for simple events can be expressed in terms of a probability distribution, or for derived events can be estimated using, e.g., Monte-Carlo simulations [6]—, but the confidence level that we have that the event will actually occur if the CEP system predicts its occurrence.

2.3 Running example

To illustrate our proposal, suppose we have a smart house, with sensors that permit detecting three basic parameters: temperature, carbon monoxide (CO) level, and whether the main door is open or not. These values are periodically sensed and notified by means of `Home` events, which also include information about the house `id`, the time at which the event was issued, the coordinates of the house, and its size in square meters. We also want to monitor whether the people living at a house are inside or not. Therefore, we suppose that they periodically issue `Person` events with their location (coordinates `x` and `y`).

The following two events show how `Person` and `Home` events are issued.

```
Person(id:1,ts:1519772340,x:50,y:50)
Home  (id:3,ts:1519772340,x:0,y:0,sq:100,temp:20,co:4000,dopen:false)
```

Using these simple events with the information provided by the sensors, we are interested in the following complex events:

- **TemperatureIncrease**: The temperature of the house has increased 2 or more degrees in less than one minute.
- **TemperatureWarning**: There are 4 `TemperatureIncrease` events is less than 5 minutes, and the temperature is always above 33 degrees.
- **COHigh**: CO levels exceed 5000 units.
- **FireWarning**: A `COWarning` event is detected, followed by a `Temperature-Warning` event, everything within less than 5 seconds.
- **NobodyHome**: The main door of the house is closed and there is nobody within the perimeter of the house.
- **CallFireDept**: A `FireWarning` event occurs after a `NobodyHome` event is detected.

To illustrate how CEP rules are written in Esper, the following listing shows the `TempIncrease` rule:[3]

```
insert into TempIncrease
select h2.ts as ts, h1.id as id, h2.temp as temp, h2.temp-h1.temp as incr
from pattern [(every (
               h1=Home() -> h2=Home(h2.temp-h1.temp>=2 and h2.id=h1.id)))
     where timer:within(1 minutes)];
```

3 Extending CEP to deal with Uncertainty

3.1 Assigning probabilities to events

Confidence in the events. As previously mentioned, we are interested in assigning probabilities to the derived events, i.e., those produced by the rules. These probabilities represent the *confidence level* on their occurrence. Here it is interesting to distinguish between the *real* event and the *digital* one: the former one happens in reality, while the latter is produced by the CEP system. It may be the case that the real event has not happened but the CEP rules forecast its occurrence (i.e., a false positive). With this, if `e` is a derived event produced by a rule of our CEP system, then $P(\texttt{e}) = 0.99$ means that you are 99% sure that the actual event has indeed happened. Note that for a simple event `e`, this probability coincides with $(1 - P_{fp}(\texttt{e}))$, where $P_{fp}(\texttt{e})$ is the probability of a false positive for that event. In this proposal, we will add an attribute `prob` to every event, indicating the confidence we have on its occurrence.

Uncertainty in measurements and attributes values. Any measurement in a physical setting is not free from *uncertainty*, which can come from different reasons, including variability of input variables, numerical errors or approximations of some parameters, observation errors, measurement errors, or simply lack of knowledge of the true behavior of the system or its underlying physics [10].

[3] The rest of the rules, together with all implementations and project data, is available from `http://atenea.lcc.uma.es/projects/UCEP.html`.

In a previous paper [2] we extended the basic UML and OCL datatypes so they incorporate the associated uncertainty. Thus, `Real` values with uncertainty are represented in terms of `UReal` values, composed of pairs (x, u), also noted as $x \pm u$, where x is the value, and u represents its uncertainty as the standard deviation of its possible variations. Likewise, a `Boolean` value b is lifted to an `UBoolean`, which is a pair (b, c) in which c is a real number between 0 and 1 that represents the confidence we assign to b. Comparison operators between `UReal` variables return `UBoolean` values. For example, if $a = 2.0{\pm}0.3$ and $b = 2.5{\pm}0.25$, we obtain that $a < b$ with a confidence of 0.893 [2].

Assigning probabilities to complex events. Our approach assumes that the probability of a derived event in CEP depends on three main issues: the confidence level that we have on the occurrence of the input events of the rule (*antecedents*); the confidence level that we have on matching and comparison operations performed by the rule to trigger the production of the derived event, as well as the computation of its attributes; and, finally, the confidence that we have on the rule itself.

This means that, given a rule R whose antecedents are events $\mathtt{e}_1, ..., \mathtt{e}_n$ (they can be either simple or complex events), that performs a matching process m_R and produces a derived event $\mathtt{e}$, the probability of event $\mathtt{e}$ is given by

$$P(\mathtt{e}) = P(\mathtt{e}_1, ..., \mathtt{e}_n) \cdot P(m_R) \cdot P(R) \tag{1}$$

where:

– $P(\mathtt{e}_1, ..., \mathtt{e}_n)$ is the combined probability of the events. For example, in case they are all independent, $P(\mathtt{e}_1, ..., \mathtt{e}_n) = P(\mathtt{e}_1) \cdot ... \cdot P(\mathtt{e}_n)$. Otherwise conditional probabilities should be used, as detailed in, e.g., [15].
– $P(m_R)$ is the confidence level of the matching process, which not only accounts for the uncertainty in the comparison operations between uncertain values and the combination of these comparisons using logical connectors (`or`, `and`, `->`, etc.), but also the propagation of uncertainty in the operations that calculate the values of the attributes of the derived event.
– $P(R)$ is the rule confidence, represented by a probability that captures the possible imprecision of the rule due to incomplete or erroneous assumptions about the environment in which the system operates, or other factors which many influence the confidence on the rule. This confidence can be calculated by Bayesian Networks, as in [4], by expert knowledge, or by any other means.

As we can see, the inputs of this method are the confidence level of the simple events (i.e., the probabilities of being false positives), and the probability of the rule, estimated by the expert users.

3.2 The Smart House example with probabilities

Let us see how the probabilities of the simple and complex events for the Smart House are calculated. Starting from the simple events, they are now enriched with the measurement uncertainty of their attributes, and with the probability of the event. Remember that, for simple events, what we have called `prob` is calculated as $(1 - P_{fp}(\mathtt{e}))$, where $P_{fp}(\mathtt{e})$ is the probability of a false positive for that event. The following two events are examples of the ones we will be feeding now to the CEP system:

```
Person(id:1, ts:(1519772340,1), x:(50,0.1), y:(50,0.1), prob:0.999)
Home  (id:3, ts:(1519772340,1), x:(0,0.1), y:(0,0.1), sq:(100,0.1),
       temp:(20,0.3), co:(4000,20), dopen:(false,0.999), prob:0.998)
```

The probability of the complex events was calculated as the product of three factors: (1) the confidence level on its *antecedents*, (2) the confidence level on matching and comparison operations, and (3) the confidence level on the rule itself. For example, the probability of the previous `TempIncrease` event, created by the rule with the same name, is given by:

$$\begin{array}{ll} P(\,\mathtt{TempIncrease}) = & \\ \quad P(\mathtt{Home})^2 \cdot & //\textit{Antedecents} \\ \quad P\left(\mathtt{h2.temp} - \mathtt{h1.temp} \geq 2.0\right) \cdot & //\textit{Matching operations} \\ \quad P(\mathtt{h1.ts} < \mathtt{h2.ts}) \cdot & //\textit{Comparison operations} \\ \quad P\left(TempIncreaseRule\right) & //\textit{Rule confidence} \end{array} \tag{2}$$

This has been implemented in Esper, using the Java implementation we have developed for the library of OCL types extended with uncertainty [2]. With this, the `TempIncrease` rule can be written to account for uncertainty as follows.

```
@Name("TempIncrease")
insert into TempIncrease
select h2.ts as ts, h1.id as id, h2.temp as temp,
   UReals.minus(h2.temp, h1.temp) as incr,
   h1.prob * h1.prob *
   UReals.ge(UReals.minus(h2.temp,h1.temp), 2.0).getC() *
   UReals.lt(h1.ts,h2.ts).getC() * P(TempIncreaseRule) as prob
from pattern [(every (h1 = HomeEvent() ->
   h2=HomeEvent(UBooleans.toBoolean(
   UReals.ge(UReals.minus(h2.temp, h1.temp), 2)) and h2.id=h1.id)))
where timer:within(1 minutes)];
```

We can see how the rule computes all attributes of the complex event, including attribute `prob` with the associated confidence, using formula (2) above. The rest of the rules are available from `http://atenea.lcc.uma.es/projects/UCEP.html`.

4 Running the system

One of the common issues of all the proposals that incorporate uncertainty into CEP systems is related to the degradation of performance [1] due to the operations required to accomplish the computation of the aggregated uncertainties and the probabilities of the rules. To evaluate our proposed solution, we executed with Esper the case study presented in Section 2.3, with and without incorporating uncertainty. Simulations were carried out on a machine with Windows 8.1 64 bits, 8Gb of RAM memory, and one Intel Core i7 processor with 4 hyperthreaded 3.07GHz cores (8 threads).

Table 1. Esper execution times (in seconds), with and without uncertainty.

# Events	3,600	36,000	360,000	3,600,000
Plain CEP rules	3.990	3.989	8.136	13.700
CEP rules with Uncertainty	5.586	3.304	9.349	15.210

Table 1 shows the results, for different input event stream sizes. Although this is only an initial experiment, the average overhead is just of 12%, which seems an encouraging result. We think that the reasons are twofold. First, users do not need to worry about the calculation and propagation of uncertainty in operations, our library of uncertain types takes care of it. Second, the probability of rules is not re-calculated in every rule execution, only once before the system is started.

5 Related Work

This proposal is inspired by existing works in the field of CEP. The work [1] provides a very interesting summary on the proposals that address uncertainty in CEP systems, and that partially cover some of the ideas we have mentioned here. Among them, [15,17,11] cover the uncertainty in the selection of the antecedents, while Wasserkrug [16] and Cugola [4] deal with the other two phases using Bayesian Networks, although each one using different approaches. We depart from these approaches in two main ways. First, we use the extended type system for reals numbers and boolean values defined in [2]. This greatly simplifies the representation of the uncertainty and its propagation, as we have seen in Section 3. Furthermore, we separate the process of calculating the probability of every CEP rule from its application. In this way, we use a variable (that can be a constant or just a function) that determines the probability of a rule, independently from how such variable is calculated. It can be using Bayesian Networks, such as in [16] or in [4] or by any other means. Only one work [18] seems to deal with uncertain timestamps, as we also do, given that timestamps are modeled in our proposal as uncertain integers and hence naturally handled.

6 Conclusions and Future Work

In this paper we have presented an initial proposal to deal with uncertainty in CEP systems that analyze streams of real-time events coming from different sources, a technology whose importance is significantly growing in the WoT. Different kinds of uncertainties have been identified and incorporated into the CEP rules and into the events, allowing modelers to represent and manage this kind of information.

The example also illustrates the need to consider probability in CEP systems. Instead of all events being equally probable, our proposal permits assigning confidence to events. This introduces an implicit priorization mechanism, very useful for instance when two or more critical events occur (e.g., `CallFireDept`). In these cases we could discriminate among them based on their probability, attending those most probable.

There are several lines of work that we plan to address next. First, we need to consider dependency between events, since in this proposal we assume they are independent. Second, we would like to consider false negatives in the event stream. Third, we want to validate our proposal with more examples and realistic case studies, in order to gain a better understanding of its advantages and limitations. Fourth, we plan to improve tool support to further automate all processes, so human intervention is kept to a minimum. Finally, we also want to apply our proposal to further EPLs, beyond Esper.

References

1. Alevizos, E., Skarlatidis, A., Artikis, A., Paliouras, G.: Complex event processing under uncertainty: A short survey. In: Proceedings of the Workshops of the EDBT/ICDT 2015 Joint Conference. Volume 1330 of CEUR Workshop Proceedings., CEUR-WS.org (2015) 97–103
2. Bertoa, M.F., Moreno, N., Barquero, G., no, L.B., Troya, J., Vallecillo, A.: Expressing uncertainty in OCL/UML datatypes. In: Submitted. (2018) Technical report available from `http://atenea.lcc.uma.es/projects/UncertainOCLTypes.html`.
3. Cugola, G., Margara, A.: Processing flows of information: From data stream to complex event processing. ACM Comput. Surv. **44**(3) (2012) 15:1–15:62
4. Cugola, G., Margara, A., Matteucci, M., Tamburrelli, G.: Introducing uncertainty in complex event processing: model, implementation, and validation. Computing **97**(2) (2015) 103–144
5. Cugola, G., Margara, A., Pezzè, M., Pradella, M.: Efficient analysis of event processing applications. In: Proc. of DEBS'15, ACM (2015) 10–21
6. EsperTech: Esper - Complex Event Processing. `http://www.espertech.com/esper/` (accessed 18 November 2017).
7. Etzion, O., Niblett, P.: Event Processing in Action. Manning Publications (2010)
8. Garcia-de Prado, A., Ortiz, G., Boubeta-Puig, J.: COLLECT: COLLaborativE ConText-aware service oriented architecture for intelligent decision-making in the Internet of Things. Expert Systems with Applications **85** (2017) 231–248
9. Greengard, S.: The Internet of Things. MIT Press (2015)
10. JCGM 100:2008: Evaluation of measurement data—Guide to the expression of uncertainty in measurement (GUM). Joint Com. for Guides in Metrology. (2008) `http://www.bipm.org/utils/common/documents/jcgm/JCGM_100_2008_E.pdf`.
11. Kawashima, H., Kitagawa, H., Li, X.: Complex event processing over uncertain data streams. In: Proc. of 3PGCIC'10, IEEE Computer Society (2010) 521–526
12. Luckham, D.C.: The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. Addison-Wesley (2002)
13. Luckham, D.C.: Event Processing for Business: Organizing the Real-Time Enterprise. Wiley (2012)
14. Mayerhofer, T., Wimmer, M., Vallecillo, A.: Adding uncertainty and units to quantity types in software models. In: Proc. of SLE'16. (2016) 118–131
15. Wang, Y.H., Cao, K., Zhang, X.M.: Complex event processing over distributed probabilistic event streams. Computers & Mathematics with Applications **66**(10) (2013) 1808–1821
16. Wasserkrug, S., Gal, A., Etzion, O., Turchin, Y.: Complex event processing over uncertain data. In: Proc. of DEBS'08, ACM (2008) 253–264
17. Xu, C., Lin, S., Lei, W., Qiao, J.: Complex event detection in probabilistic stream. In: Proc. of APWEB'10, IEEE Computer Society (2010) 361–363
18. Zhang, H., Diao, Y., Immerman, N.: Recognizing patterns in streams with imprecise timestamps. Inf. Syst. **38**(8) (2013) 1187–1211