# FSCEP: A New Model for Context Perception in Smart Homes

Amina Jarraya[1(✉)], Nathan Ramoly[2], Amel Bouzeghoub[2], Khedija Arour[3], Amel Borgi[4(✉)], and Béatrice Finance[5]

[1] LIPAH, Université de Tunis El Manar, 1068 Tunis, Tunisia
amina.jarraya@telecom-sudparis.eu
[2] SAMOVAR, Télécom SudParis, CNRS, Université Paris-Saclay, Evry, France
{nathan.ramoly,amel.bouzeghoub}@telecom-sudparis.eu
[3] Université de Carthage, 1054 Tunis, Tunisia
Khedija.Arour@issatm.rnu.tn
[4] Université de Tunis El Manar, 1068 Tunis, Tunisia
Amel.Borgi@insat.rnu.tn
[5] DAVID, University of Versailles Saint-Quentin-en-Yvelines, Versailles, France
beatrice.finance@uvsq.fr

**Abstract.** With the emergence of the Internet of Things and smart devices, smart homes are becoming more and more popular. The main goal of this study is to implement an event driven system in a smart home and to extract meaningful information from the raw data collected by the deployed sensors using Complex Event Processing (CEP). These high-level events can then be used by multiple smart home applications in particular situation identification. However, in real life scenarios, low-level events are generally uncertain. In fact, an event may be outdated, inaccurate, imprecise or in contradiction with another one. This can lead to misinterpretation from CEP and the associated applications. To overcome these weaknesses, in this paper, we propose a Fuzzy Semantic Complex Event Processing (FSCEP) model which can represent and reason with events by including domain knowledge and integrating fuzzy logic. It handles multiple dimensions of uncertainty, namely freshness, accuracy, precision and contradiction. FSCEP has been implemented and compared with a well known CEP. The results show how some ambiguities are solved.

**Keywords:** Situation identification · CEP · Fuzzy logic · Context ontology

## 1 Introduction

Smart homes are currently becoming increasingly popular in society. Indeed, many companies are now proposing sensors and actuators to help people in their everyday life. Those devices can work not only individually, but can also be used globally for multiple applications including situation identification. Sensors generate events as they observe the environment. For example, a motion sensor sends a signal when a movement is detected or a thermometer can send an event when the temperature varies too much. All of these events carry data that can be analyzed to generate further information.

To do this analysis, we can rely on the Complex Event Processing (CEP) methods. A CEP allows us to define continuous queries and to analyze events from multiple sources in order to generate a high level event (complex event). Thanks to this method, it is possible to analyze the meaning of different events and have a better understanding of the context. Such data are then used by various applications. For example, user's location can be inferred thanks to a CEP and then used for situation identification.

In everyday life environment, in particular smart homes, uncertainty is a serious issue. In fact, we should expect that sensors will provide non-perfect data. For instance, a thermometer may not be accurate or a motion sensor may be intempestively triggered. In such cases, uncertainty can lead to the production of complex events that are themselves uncertain. Subsequently, this may cause problems or miscomputations for the application relying on CEP outputs. For instance, if the CEP returns that the user is in room A while he is in room B, a wrong situation may be recognized.

There is multiple definitions of uncertainty in the literature. In this work, we will stick to the Ye et al. definition [15]. We consider uncertainty that can be decomposed into multiple facets or dimensions, among which we can mention: (1) *Freshness:* If a data is too old, it may be outdated; (2) *Precision:* Imprecise data is correct, yet inexact. For instance, localizing a user by coordinates is more precise than localization per rooms; (3) *Accuracy:* An inaccurate data is completely or partially wrong; (4) *Contradiction:* A contradiction occurs when two pieces of data provide a contradictory information. For example, one sensor locates a user in room A while another asserts he is in room B.

Uncertainty is not commonly addressed in existing CEP studies. The few works tackling this challenge often just deal with a subset of the dimensions. In this work, we consider the four aforementioned uncertainty dimensions as requirements to achieve. The main goal of this paper is to propose a new CEP called FSCEP (Fuzzy Semantic Complex Event Processing) that carries out each of these requirements. It relies on a semantic background knowledge and manages fuzzy events. Our contributions can be summarized with respect to the four defined requirements as follows:

1. Freshness Requirement (FR): This requirement is satisfied by construction. Indeed, CEP is able to take into account only the events triggered during a query defined time window, ensuring the freshness of the events.

2. Precision Requirement (PR): Thanks to event semantization and event fuzzification processes, events are enriched with semantic data provided by an ontology and a fuzzy value, respectively. This enrichment allows us to refine data precision.

3. Accuracy Requirement (AR): By assigning to each sensor a trust value provided by an expert in the knowledge background, the events are therefore weighted with a confidence degree. Events provided by a sensor having a trust value under a defined threshold are considered as wrong.

4. Contradiction Requirement (CR): The fuzzification process of complex events allows us to provide data with multiple valid interpretations, which can be contradictory at first.

Our approach is implemented and tested using a simulation framework. The rest of the paper is divided as follows. In Sect. 2, we present some definitions and explanations of the problem. In Sect. 3, we synthesize the different recent studies. To illustrate our requirements, we set up a scenario in Sect. 4. In Sect. 5, we describe our contribution and its multiple layers. In Sect. 6, we present the FSCEP prototype implementation and its evaluation. Section 7 synthesize the requirements dressed. The conclusion section briefly summarizes and describes the next steps of our work.

## 2 Preliminaries

Before addressing the contribution itself, this section introduces definitions and explanations for a better understanding of the rest of the paper.

Complex Event Processing (CEP) is a method for managing event streams of data from multiple sources. It aims to analyze and correlate low level events in order to generate high level events. For example, if there is an event from a motion sensor in room A followed by one event from a motion sensor in a room B, the CEP can generate an event '*user moves from room A to room B*'. In CEP, these computations are defined by a designer through queries. Queries allow to filter events by defining a time sliding window and applying constraints. Events are representative of an environment context, in our case the smart home.

**Definition 1.** *Context: The context is the set of all data describing an environment (or a smart home in our case) provided by sensors. For example, this includes temperature or user's location.*

**Definition 2.** *Event: An event is a punctual change of a context data occurring at a precise moment.*

CEP considers two types of events: simple and complex. Simple events may be seen as raw events whereas complex events are the result of the analysis of the input events.

**Definition 3.** *Simple event: A simple event is provided by a source external from the CEP. It can be seen as a raw event provided by a sensor. The carried data are usually basic. For example, a simple event can be a motion sensor signal.*

**Definition 4.** *Complex event: A complex event results from the abstraction of other several events. In CEP, complex events are often the output of the queries. The carried data is usually higher level than simple events. For instance, a complex event can be the position of a user determined through a query and multiple motion sensors events.*

Usually, CEP assumes an event cloud as input.

**Definition 5.** *Event cloud: An event cloud is a partially ordered set of events where dependencies between events are set by criteria other than time. An event cloud is distinguished from an event stream, which is an ordered sequence of events.*

**Definition 6.** *Event stream: An event stream is an ordered time sequence of events.*

CEP is widely used by the industry in various domains including decision making, finance or IT security. Among existing CEP systems, we can cite Streambase [11], Coral8 [9] or ESPER [7].

Even though it is commonly used, CEP has its limits. Indeed, CEP only analyzes temporal and causal relations between events. However, in some applications, using a more complete background knowledge is essential. For instance, in our case, the information about the sensors and the environment can be used to query and generate high level events. To fulfill this need, Semantic Context Event Processing (SCEP) was proposed, for example in ETALIS [1] and SCEPter [17]. The idea is to combine the features of CEP with the reasoning possibilities of ontologies. By this way, it is possible to find implicit relations between events. Furthermore, SCEP are able to provide semantic events. These events are richer, interoperable and easy to integrate in a knowledge base.

**Definition 7.** *Semantic event: A semantic event is an event described and provided as a small RDF*[1] *(Resource Description Framework) graph.*

In the following section, we synthetize the different recent studies.

## 3 Related Work

Nowadays, CEP is a quite popular topic. The literature around CEP is rich and there are numerous works for various applications. However, the main preoccupation around CEP is the performance. The large majority of CEP works address this issue [3,12,14,16] as underlined by Cugola et al. [4]. There are a few works addressing the problem of uncertainty of data and rules but they mostly focus on the accuracy of data rather than the imprecision. Nevertheless, in real life environment, uncertainty is a serious issue and ignoring its existence is naive. Furthermore, to the best of our knowledge, the integration of fuzzy logic within CEP to handle precision and contradiction of context data has not yet been

[1] https://www.w3.org/RDF/.

studied. As motivated earlier, relying on fuzzy data implies multiple advantages and has particularly been used for activity recognition applications [10]. Even if there are no works directly comparable to ours, we review some recent works addressing different dimension of uncertainty. Some works added 'uncertainty management' into CEP.

Wasserkrug et al. [13] proposed a framework for knowledge representation in CEP supporting uncertainty. They address two types of uncertainty: the imprecision of events and the 'uncertain relations between events'. To do so, based on probability theory, the authors defined a 'probability space' and relied on dynamically generated Bayesian Networks to compute the relevant probabilities. This is one of the first works addressing uncertainty in CEP. However, compared to our requirement, they only tackle the imprecision. We did not identify uncertain relations as a requirement for smart home applications.

Artikis et al. [2] aims to identify uncertainty dimensions that may appear in CEP and discussed of possible solutions to extend traditional events. They identify the following types of uncertainty:

– Incomplete event streams: some events are not triggered while they should, for instance due to the occlusion of a camera.
– Insufficient event dictionary: some type of events may not be taken into account.
– Erroneous event recognition: matches our definition of imprecision.
– Inconsistent event annotation: annotation on training data for machine learning are inconsistent.
– Imprecise event patterns: imprecision of the query.

Even if they deal with multiple uncertainty dimensions, these are different from the ones we identified for smart home applications. In fact, we consider all possible types of event to be known and provided to the FSCEP, excluding the second and fourth types. Furthermore, since we aim to operate in smart homes, we suppose that user queries are simple and do not imply complex event pattern recognition, putting away the last dimension. In their work, they proposed multiple short solutions to apply in various aspects of CEP, in particular probabilistic approaches. However, none of them use fuzzy logic.

More recently, Cugola et al. [4] proposed a model for dealing with uncertainty in CEP: CEP2U (Complex Event Processing under Uncertainty). They consider two types of uncertainty. Firstly, they model the accuracy on events themselves, in other words, the accuracy of the information. It's important to underline that they refer to 'precision', however, their definition differ from ours and is actually matching our definition of accuracy. For a better understanding, we will stick to our definition. For events, there are two types of uncertainty: one for the 'content' and one for the 'occurrence'. Secondly, they consider the accuracy of the rules: that is to say "the possibility that rules do not completely reflect the behavior of the monitored environment". To model the uncertainty, CEP2U uses theory of probability for events and Bayesian Networks (BN) for rules. Each event is associated with a probability distribution function. On the other hand, rules are linked to a BN. CEP2U relies on a fully probabilistic approach and does not handle the fuzziness of data. Compared to our approach, it actually solves a different aspect of uncertainty. In fact, a probabilistic method allows us to handle the accuracy of sensors while fuzzy logic tackles imprecision [15]. In our case, we aim to support multiple interpretations of one data. Thus, FSCEP and CEP2U are complementary. A further work could be to combine both approaches.

Lee et al. [8] aims to tackle the problem of dynamic changes in the domain environment or expert mistakes. This can be seen as a problem of freshness, but applied on the rules. To overcome this, they have proposed a tool known as Sequence Clustering-based Automated Rule Generation (SCARG) that generates rules by analyzing decision-making history. SCARG relies on four steps. First, it collects event sequence samples from an expert. Then, it computes a cluster model thanks to sequence clustering. Once the clusters are set, each cluster is graphically modeled through Markov Probability Transition Model (MPTM). Finally, using node centrality measure, those models are pruned. By doing so, SCARG is able to support dynamic changes. Although they are not handling data uncertainty from sensors, their approach can be applied in smart homes which are also subjected to evolution. Again, compared to our approach, Lee et al. solves another problem of uncertainty and the two approach can be seen as complementary.

Uncertainty is addressed in different ways in CEP studies. However, dealing with imprecise and contradictory data using a CEP enhanced with fuzzy logic is novel. Furthermore, this novelty is emphasized by the usage of semantic data.

In order to illustrate how we tackle the four requirements pointed out in Sect. 1, we present the following scenario which sketches the main ideas.

## 4 Scenario

*Monika lives in a smart home equipped with multiple sensors and actuators to monitor and control several parameters. The environment relies on cameras, pressure sensors, motion sensors and beacons (locating her phone in a room). In particular, the living room and the office, each one possesses beacons, pressure sensors, one motion sensor and one camera. Monika is working in her office and is detected by the camera. However, she left her phone in the living room: the phone is detected by the beacon, sending a signal. The window in the living room is open and the wind is moving the curtain.*

Several events are received from sensors:

- In the living room: one from the phone and three from the motion sensor. The camera doesn't send any event.
- In the Office: one from the pressure sensor (related to the chair), one from the motion sensor and one from the camera.

In this case, two interpretations of Monika's location are possible from the triggered events: she can be either in the living room or in the office.

With a classic non-fuzzy CEP, only one interpretation would remain in the output complex event. Selecting which one is to be kept can be hazardous and may lead to a wrong interpretation of the user's activity.

By using fuzzy semantic events, no interpretations are removed. In this case, it will be believed that Monika is in the living room AND in the office, with a trust value for each. The FSCEP returns a complex event carrying the location with the two possible interpretations.

Furthermore, as sensors may not be accurate, we propose to add a trust index for sensors in the background ontology. Indeed, a motion sensor may send a signal while there was no movement, for instance due to an untimely light change. This index is defined by an expert and determines how much a sensor is trustful. It is then used by the FSCEP to determine trust values of fuzzy events. For example, the camera is the most reliable than other sensors. The definition of trust values is outside of the scope of this paper.

## 5 FSCEP Model for Situation Identification Approach

In this section, we describe the three layers of FSCEP model (as depicted in Fig. 1) and we focus on the perception layer.

– *Sensing* is devoted to data gathering from sensors deployed in the environment.

– *Perception* is the interpretation of the sensing phase returned data, providing a further abstraction operation aiming at transforming raw sensor data in more significant pieces of information.

– *Application* is the execution of a set of actions according to the perception results. Pervasive applications must constantly be adapted to a highly dynamic context changes and are designed to support users in their daily lives. Sensing and perception phases constitute primordial phases and their outputs serve as inputs to application layer, since they provide context information at a high level of abstraction.

These three processes are detailed in the following sections.

### 5.1 Sensing Phase

The sensing phase is achieved through sensors deployed in the environment. Sensors are devices aiming to analyze the environment in order to provide data about it. Each sensor is responsible for gathering various context data, such as user's location or temperature. Cameras, thermometers or motion sensors are typical examples. There are multiple types of sensors. Some are pretty basic and return raw data while others are smarter and are able to compute and provide processed data.

We distinguish two types of sensors:

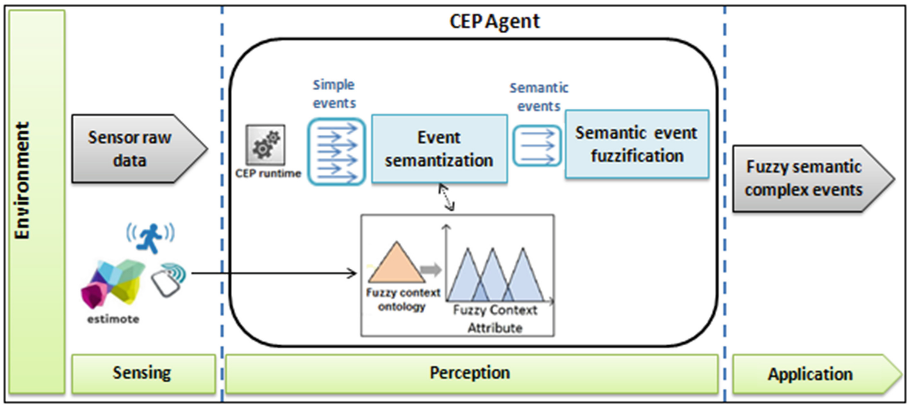
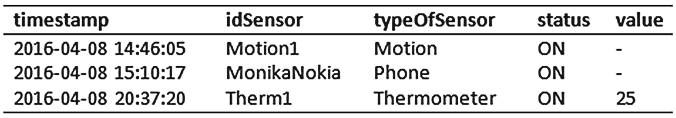


**Fig. 1.** Fuzzy Semantic CEP approach for environmental perception

– *Active sensor (push mode)*: an active sensor **sends** (pushes) the information to the system. This is typically the case for event oriented sensors, such as motion detector.
– *Passive sensor (pull mode)*: a passive sensor **is asked** to provide a data. In other words, the data is pulled. This is typically the case for sensors that continuously analyze the context, such as thermometers.

In our scenario, we used active sensors, namely motion sensors, pressure sensors, beacons and cameras. Each sensor event (a sensor raw data) indicates a change in the state that should be identified and is modeled by a *timestamp* (represented in milliseconds) used to establish temporal relations in order to correlate information from different sensors. On top of the *timestamp*, an event carries the following properties: an *idSensor* which is the sensor's identifier; a *typeOfSensor* describing the sensor's type (e.g. *motion*, *camera*, or *phone*); a *status* denoting the sensor's status (*ON/OFF*) and a *value* which represents the asked value (this property concerns passive sensors). An example of a listening of sensor events is shown in Fig. 2. An event is triggered when the *status* or the *value* is changed.



| timestamp | idSensor | typeOfSensor | status | value |
|---|---|---|---|---|
| 2016-04-08 14:46:05 | Motion1 | Motion | ON | - |
| 2016-04-08 15:10:17 | MonikaNokia | Phone | ON | - |
| 2016-04-08 20:37:20 | Therm1 | Thermometer | ON | 25 |

**Fig. 2.** An example of time-stamped sensor events

### 5.2 Perception Phase

This phase constitutes the core of our approach. It consists of a multi-agent system where agents are responsible for gathering context data called **sensing agents**.

Each sensing agent takes as input simple events and provides, as output, fuzzy semantic complex events. Multiple sensing agents are deployed to observe different context data. Each sensing agent is related to one or more sensors that observe the same context data.

For example, considering a given context data such as a user's location, a sensing agent can be related to three sensors to observe the user's location (as described in the scenario above) namely motion sensor, phone's beacon and camera (cf. Fig. 3).

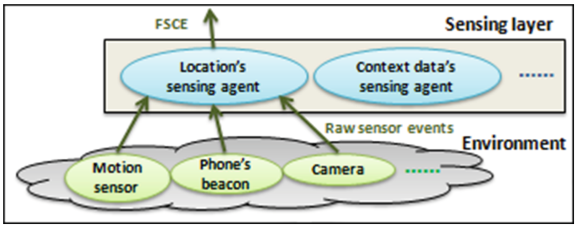


**Fig. 3.** Example illustrated for user's location perception

A sensing agent integrates a CEP engine that identifies, processes, and responds to discrete events from multiple sensors. In order to deal with a continuous sensor sequence, CEP usually uses the two most common types of sliding window implementations, time sliding windows and sensor event sliding windows.

*A Time sliding window* aims to partition a sensor sequence into a fixed time interval, whereas *a sensor event sliding window* aims to partition a sensor sequence into windows having the same number of sensor events. In this paper, we choose the time sliding window technique since the sensor readings that we use are characterized by discrete sensor events.

The perception phase is divided into two steps: *event semantization* and *semantic event fuzzification*. Details are provided in the following.

**Event Semantization.** Event semantization aims to improve CEP's output by incorporating ontologies into the process of complex event detection. In fact, SCEP engines enriches CEP queries with semantic data from ontologies. This step describes how event attributes are matched to semantic entities and linked to an external ontology in order to enrich the information of the events. As a result of *event sematization* step, incoming simple events (sensor raw events) are transformed into semantic events that point towards an existing ontology.

We propose the following definitions in order to formalize our approach. Please note that these are the formal definitions specific to FSCEP.

**Definition 8 (A simple event).** *A simple event is a vector of sensor raw data defined as* $e = \{t, id, type, st, v\}$ *where* $t$ *is the timestamp, id is the sensor's identifier, type is the sensor's type (e.g. motion, camera, or phone), st is the sensor's status (ON/OFF) and* $v$ *is the asked value (e.g. a return value from a thermometer).*

**Definition 9 (A semantic event).** *A semantic event* **se** *is modeled as a finite set of triples* $(s, p, o)$ *of a graph RDF where subject, predicate, object respectively* $s, p, o \in U$ *where* $U$ *is an infinite set of URI's*[2].

*Sensor data, extracted from simple events, are linked to additional semantic data such as the sensor's location (MotionSensor:Motion1, IsLocatedIn, Location:office1) and the sensor's trust level (MotionSensor:Motion1, hasTrust, "0.4") shown in Fig. 5.*

**Definition 10 (Event semantization process).** *The event semantization process consists on transforming a simple event* $e_i$ *to a semantic event* $se_i$ *that point into an existing ontology.*

$$EventSemantization(\{e_1, e_2, \ldots, e_n\}) \longrightarrow \{se_1, se_2, \ldots, se_n\}$$

*where* $0 < i <= n$ *and* $n$ *is the number of incoming simple events per time sliding window.*

In this paper, we used a context ontology for human activity representation[3] [10] and adapted it to our needs by adding:

– Several subclasses of the concept *Sensor* such as *MotionSensor* (Fig. 4).
– A data property *hasTrust* associated to *Sensor* subclasses that expresses the trust degree of a sensor type given by a user. Therefore, a semantic event contains necessarily a trust value related to the corresponding sensor. Trust's values are static in the ontology and are supposed to be provided by an expert.

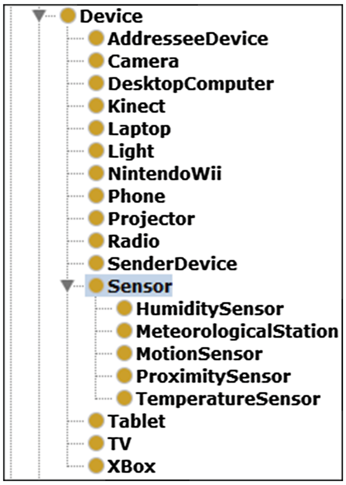


**Fig. 4.** Excerpt of Sensor subclasses in the ontology (partial).

*Example 1* Semantic events are dynamically generated from the raw events. Considering the scenario described in Sect. 4 and the context ontology shown in Fig. 4, a semantic enrichment is fulfilled by adding RDF triples giving higher information concerning the sensors'

[2] https://www.w3.org/TR/2004/REC-rdf-concepts-20040210/.
[3] http://users.abo.fi/ndiaz/public/FuzzyHumanBehaviourOntology/FuzzyHumanBehaviourV11.owl.

location and the sensors' trust. For example, events received from the phone are semantized as follows: (phone:MonikaNokia, islocatedIn, Livingroom) and (phone:MonikaNokia, hasTrust, 0.4). A new RDF graph is then constructed as it is shown in Fig. 5.

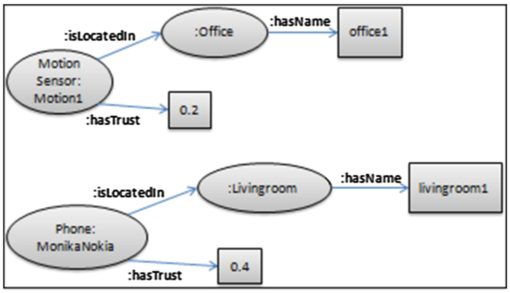


**Fig. 5.** Event semantization through RDF graph's scenario

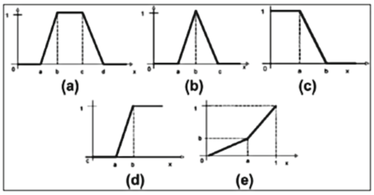


**Fig. 6.** Membership functions for fuzzy data type

**Semantic Event Fuzzification.** Fuzzy logic is widely used to deal with uncertain knowledge, especially imprecise knowledge. In pervasive computing, fuzzy logic is used to map sensor raw data to fuzzy attributes in order to manage sensor unreliability and sensor data imprecision.

Therefore, this step aims to compute one fuzzy event from multiple semantic events observing the same context data. The output data then carries fuzzy values for this context data.

To apply fuzzification on semantic events, we use the Fuzzy OWL2 formalism. In this formalism, fuzziness is modeled through annotations. With such knowledge, it is possible to use dedicated tools to handle imprecise and contradictory data.

In [10], the authors consider data types, concepts, properties, and relations as fuzzy. For fuzzy data types, they defined membership functions: (a) Trapezoidal function; (b) triangular function; (c) left-shoulder function; (d) right-shoulder function; and (e) linear function (Cf. Fig. 6).

**Definition 11 (A fuzzy semantic complex event FSCE).** *For an observed context data* ***cd****, a fuzzy semantic complex event $FSCE_{cd}$ is an extended graph RDF of* ***se*** *with additionnal semantic data which is the fuzzy value. We denote $FSCE_{cd} \subseteq se$.*

*A $FSCE_{cd}$ is an event having multiple interpretations with a fuzzy value for one observed context data. Especially, a $FSCE$ presents one or more detected values for an observed context data with trust weights.*

*Observed context data can be pressure, user's presence, temperature, etc.*

**Definition 12 (Fuzzification process).** *The fuzzification process can be seen as a function that takes as input a list of semantic events and generates a $FSCE_{cd}$ for an observed context data.*

$$Fuzzification(\{se_1, se_2, \ldots, se_n\}_{cd}) \rightarrow FSCE_{cd}$$

To generate a $FSCE_{cd}$, we use a dedicated membership function $MF$. This function uses as inputs detected values for the observed context data and the set of semantic events matching this value. As a result, it provides a trust weight for each value detected meaning that the context data has been observed for different values with a certain weight. The *fuzzification process* calls the membership function for all detected values (see Algorithm 1).

Let $MF_{cd}$ be the membership function:

$$MF_{cd}(SE, val) \Rightarrow WT_{val}$$

where $SE$ is a set of semantic events, $val$ is a common detected value for these events and $WT_{val}$ is the weighted trust determined for $val$.

$MF_{cd}$ is the weighted sum of the trust weight of those semantic events corresponding to the value $val$ for an observed data context.

In fact, this function uses the trust index of sensors carried in the ontology. The membership function $MF_{cd}$ is denoted below:

$$MF_{cd}(SE, val) = [\textstyle\sum_{i=1}^{n}(\alpha_i)]/[\sum_{j=1}^{N}(\alpha_j)]$$

where $N$ is the total number of events in the environment; $n$ is the total number of events for value $val$ and $\alpha$ is the trust value of a sensor belonging to $SE$.

The $FSCE_{cd}$ computed by the fuzzification function can be defined as follows:

$$FSCE_{cd} = \cup MF_{cd}(SE_{val}, val) \forall val \in VAL$$

where $VAL$ is the set of all possible different values in the smart home.

*Example 2* Fuzzification function

Considering the example scenario, if we assume that we have several triggered semantic events:

– In the living room: one semantic event from the phone's sensor (trust degree:0.4), three semantic events from the motion sensor (trust degree:0.2).
– In the office: one semantic event from the pressure sensor (trust degree:0.6), one semantic event from the motion sensor and one semantic event from the camera (trust degree:0.8).

Then, we apply the *fuzzification process* and generate a FSCE for each observed context data namely user's presence, user's pressure and user's movement as below:

$$FSCE_{presence} = \{[livingroom; 0.33], [office; 0.67]\}$$
$$FSCE_{pressure} = \{[livingroom; 0], [office; 1]\}$$
$$FSCE_{movement} = \{[livingroom; 0.75], [office; 0.25]\}$$

The trust weights $WT_{(livingroom)})$ and $WT_{(office)}$ are computed by applying the $MF$ function. For the user's presence example:

$$WT_{livingroom} = MF_{cd}(SE, livingroom) = 0.4/(0.4 + 0.8) = 0.33$$
$$WT_{office} = MF_{cd}(SE, office) = 0.8/(0.4 + 0.8) = 0.67$$

In this example, we conclude that the user is performing an activity in the living room with 0.33 of trust AND in the office with 0.67 of trust.

In this work, we have used $MF$ as the weighted sum function but we can apply other different functions proposed in literature to compute trust weights.

The fuzzification of the observed context data value through the membership function is shown in Fig. 7.

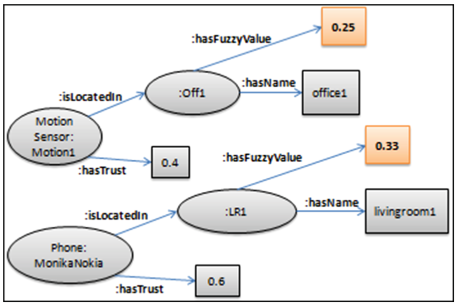


**Fig. 7.** Fuzzification of location's context data in RDF graph

## 5.3 Application Phase

Sensing and perception phases constitute primordial phases to interpret low level abstraction of sensor raw data into high level abstraction of context information. Pervasive computing applications aims to deal with context information to provide users with appropriate behavior in their environment.

Among these applications, we can cite situation identification. Situation identification aims to understand what is happening in the environment. For example, if the user is sleeping while a gas leak occurs, it can be detected as a 'danger' situation. Situation identification can be seen as a generalization of activity recognition: it analyzes the whole environment and not only one user. Detecting situations is a challenge. However, it is even more complicated when taking uncertainty into consideration. Providing enriched data, in particular about uncertainty, to such application is a convenient asset. In fact, by having such information, applications can take them in consideration. In our case, we add fuzziness to semantic events, making them directly usable by Rodríguez's activity recognition approach [10]. Rodríguez et al. actually relies on fuzzy data and fuzzy operators to identify user activities. One of our future objectives is actually to enhance this approach for situations recognition, but this is not in the scope of this paper. Of course, there are other works compatible with our FSCEP, for example D'Aniello's situation awareness framework [5].

We focused our analysis on situation related work, but other applications are possible in smart homes. Security and alarming are of course interesting applications. The fuzzy logic could prevent false alerts. Personal robotics can also be an application. For instance, a robot may adjust its tasks plan according to trust weights on fuzzy data.

More generally, by relying on a FSCEP, pervasive applications can deal with real fuzzy semantic context information that resolve the main requirement handling different dimensions of uncertainty, namely freshness, precision, accuracy and contradiction.

## 5.4 The FSCEP Algorithm

In this section, we present the FSCEP algorithm that implements the proposed approach.

The FSCEP algorithm (Algorithm 1) is applied to all sensing agents of the perception phase. It takes as an input parameters an event cloud $SEQ$ for an observed data context according to a time sliding window, a query $q$ to filter simple events to batched events and a background knowledge fuzzy ontology $O$ to enrich batched events with semantic knowledge.

The FSCEP algorithm aims to provide, as an output, a fuzzy semantic complex event $FSCE$. The FSCEP algorithm proceeds in three main steps:

- *First step (line 6)*: each sensing agent integrates a CEP engine which receives events triggered from the environment and applies the query $q$ defined by the expert in order to generate a list of filtered and batched events $L_{BE}$.

- *Second step (lines 7–10)*: we enrich batched events with semantic context data namely trust degree of sensors and values of context data such as location's value (bedroom, office, etc.) and transform them into an RDF graph. $L_{se}$ contains the set of semantic events.

- *Third step (lines 11–13)*: for each value of observed context data, we measure the fuzzy value by applying our membership function and annotate the graph with this fuzzy value to obtain as result a fuzzy semantic complex event $FSCE$.

The computational complexity of the FSCEP algorithm is O(n) where n is the number of incoming simple events for an observed data context.

## 5.5 Handling FSCEs in FSCEP

Although in this paper we focus on the process to generate FSCE from simple sensor events, it's important to notice that our model shall also be able to have FSCEs as input. Thus, and similarly to classic CEP, FSCEP agents can be chained in order to generate more refined FSCEs. Handling a FSCE follows the same process as a simple event, with some variation. First, the query can use

**Algorithm 1.** FSCEP algorithm of real event cloud for an observed context data

```
Input:
$SEQ = \{e_1, e_2, \ldots, e_n\}$: an event cloud for observed context data
$q$: user defined CEP query
$O$: background knowledge fuzzy ontology
Output:
$FSCE$: a fuzzy semantic complex event (graph RDF)
1  begin
2      $initialise(L_{BE})$ // List of batched events
3      $initialise(L_{se})$// List of semantized events
4      $initialise(L_v)$// List of observed context data's value
5
       // First step: batching and filtring events with CEP
6      $L_{BE} = CEPengine(q, SEQ)$
       // Second step: event semantization
7      for $e_i \in L_{BE}$ do
8          $< v_{ph_e}, trust_e >= queryOntology(O, idSensor_e, typeOfSensor_e)$
             $graph_{e_i} = semantizeEvent(e, v_{ph_e}, trust_e)$
9          $L_{se}.add(graph_{e_i})$
10     $L_v = loadValuesFrom(L_{se})$
       // Third step: fuzzification of semantic events
11     for $v_i \in L_v$ do
12         $fv_i = MF_{v_i}(L_{se}, v_i)$
13         $FSCE.addAnnotation(v_i, fv_i)$
14     $return(FSCE)$
```

FSCE elements: the designer can filter FSCEs based on its fuzzy or semantic properties. The *semantization* step is skipped as FSCE are already enhanced. With fuzzy events, the *fuzzyfication* can rely on their fuzzy properties (on top of semantic properties) in order to compute the weights. The $MF$ function is different: in our work, we use a similar weighted mean as proposed in Sect. 5.2 that additionally takes into account the fuzzyness of events.

## 6 Implementation and Evaluation

We have implemented the described FSCEP and evaluated it through simulation.

Our prototype combined a well known CEP, ESPER [7], and ontologies through the Jena framework. It is developed in Java and it is agent oriented (Jade). The knowledge is carried by a fuzzy context ontology based on the one presented in [10]. This ontology was used for user identification. We have improved this ontology by adding the needed notions. The fuzzy logic is integrated through annotations and the ontology is compatible with fuzzy reasoner such as FuzzyDL. Our prototype uses the ontology as a T-box [6] for its process. The fuzzy result of our FSCEP is saved in the A-box [6]: the fuzzy ontology is then instantiated and can be used for other needs, such as activity recognition.

In order to simulate our environment, we used Freedomotic[4], a development framework used for managing smart spaces. Freedomotic allows us to control virtual and physical devices. It carries a basic event processing unit, a knowledge of the rooms and some plugins. One of the goal of Freedomotic is to simulate the environment before setting it up for real. In fact, sensor values can be easily simulated. For our experimentation, we have implemented our scenario and added the presented sensors as plugins, namely motion sensors, pressure sensors, beacons and cameras (cf. Fig. 8). These newly added devices generate events and are able to communicate with our FSCEP.

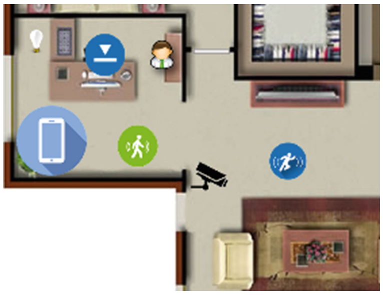


**Fig. 8.** Freedomotic map with set of sensors

**Table 1.** Example of fuzzy weight provided according to the triggered events

| [Bec, Mot1, Cam, Mot2] | FSCEP | | CEP | |
|---|---|---|---|---|
| | Office | LR | Office | LR |
| [1, 0, 1, 1] | 0,44 | 0,56 | 0 | 1 |
| [1, 0, 1, 0] | 0,57 | 0,43 | 1 | 0 |
| [0, 1, 1, 1] | 0,28 | 0,72 | 0 | 1 |
| [1, 1, 1, 1] | 0,54 | 0,46 | 1 | 0 |
| [0, 1, 1, 0] | 0,4 | 0,6 | 0 | 1 |
| [1, 1, 0, 0] | 1 | 0 | 1 | 0 |

To evaluate our model, we set up a simple activity recognition scenario close to the one described in Sect. 4. In this scenario, the user, Monika, is working in the office and we aim to identify she is doing so. To do so, we need to acquire context data about her position and her current stance. The office is equipped with a beacon (to detect phones), a motion sensor and the chair has a pressure sensor, while the living room has a camera and a motion sensor. Sensor event uncertainty has been modeled for each one. In this experiment, an activity is recognized by applying simple rules in a context ontology (A-Box) fed by FSCEP or classical CEP: the objective of the experiment is to evaluate the gain of using a FSCEP over CEP for a smart home application. Thus, we measure the activity recognition rates with and without FSCEP.

For each case, 500 runs were executed. One run consists in a salve of events, the event processing (FSCEP or CEP) and the activity recognition. To model uncertainty (accuracy), each sensor has a probability of failure. An example of output of our FSCEP can be found in Table 1. The table shows the computed fuzzy weights according to devices triggering. The first column displays which event is triggered from, respectively, office's beacon and motion sensor, and living room's camera and motion sensor. Taking the first example, the FSCEP prototype generates a fuzzy value for user's location. Indeed, the user is located in office with 0.44 **AND** in living room with 0.56. According to the classic CEP, the user is located **ONLY** in the living room. We can see that FSCEP prevents

[4] http://freedomotic.com/.

the disqualification of some events, that may be actually true. These results are then used for recognizing the current user activity.

Obtained results can be found in Fig. 9. It depicts the success recognition rate with and without FSCEP.

Although it is a very simple scenario, FSCEP allows to be more efficient in activity recognition by providing a frank increase of 10 %. By providing FSCE and not excluding values believed distrustful, the activity recognition can achieve better result by overlapping multiple information, in this case for instance, location and stance. This proves the potential of our model for smart home applications.

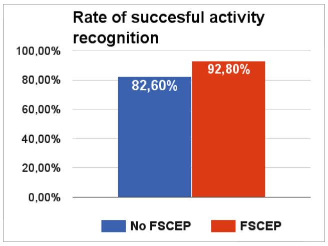


**Fig. 9.** Experiments results

In order to provide a strong validation of FSCEP, we have started integrating it with real devices. Our objectives is to evaluate FSCEP facing 'real' uncertainty encountered by devices available on the market. We used Estimotes beacons[5] (Fig. 10a) and Netatmo Welcome[6] (Fig. 10b) to localize a user. Beacons are simple, but quite erroneous, in particular when the user is in range of multiples beacons. Netatmo Welcome is more reliable, but can be mislead in some situation (see Fig. 10c). Our FSCEP is able to handle these issues to provide proper FSCE about location. In future works, we aim to experiment it over a complete living scenario in a smart apartment with various devices and multiple situations.

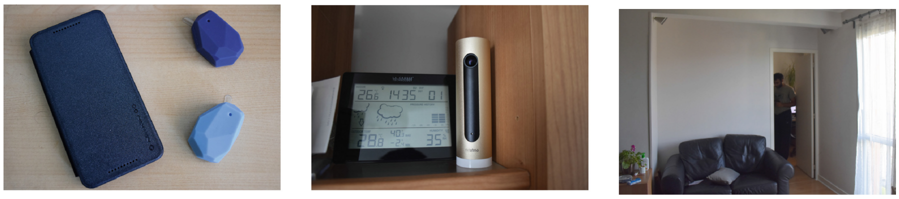


**Fig. 10.** Integration with physical devices. (a) Beacons used to locate the phone. (b) Indoor camera with face recognition. (c) In this configuration (Please note that due to technical issues, the picture was not taken from the Welcome camera.), the camera is placed to detect users watching TV, it can however see users in the other room if the door is opened, leading to miscalculation of location

[5] http://estimote.com/.
[6] https://www.netatmo.com/en-GB/product/camera.

## 7 Synthesis and Discussion

In this section, we discuss the fourth requirements addressed in the *Introduction* section. Our FSCEP approach handles different dimensions of uncertainty by proposing different techniques.

In fact, our contributions resolve the requirements as follow:

1. Freshness Requirement (FR): To keep context data fresh, we used a technique based on event detection *Complex Event Processing*. It's designed to process low level event notifications to identify higher level complex events according to a set of user defined queries and in a time sliding window. Taking an example of our scenario, the user's location is determined immediately through the position of motion sensors, phone's beacon and camera when they are triggered.
2. Precision Requirement (PR): Event notifications for the observed context data, are enriched with semantic knowledge according to *Event semantization* process. For example, event notifications from motion sensors, phone's beacon and camera are enriched with trust degree and location of the corresponding sensor. Then, they are fuzzifed by applying our membership function to generate a fuzzy value for the observed context data. According to the example, each semantic event will have a fuzzy value for the observed user's location.
3. Accuracy Requirement (AR): We assign a trust degree for each sensor on the fuzzy context ontology. It defines the confidence level to which a context data is correct. Taking sensors of our example, motion sensors, phone's beacon and camera have respectively trust degrees 0.2, 0.4 and 0.8.
   Semantic events contain semantic knowledge context including the trust degree of the corresponding sensor. Therefore, semantic events with a trust value under a defined threshold, are considered as wrong.
4. Contradiction Requirement (CR): Thanks to the *fuzzification process* of semantic events, the fuzzy value generated by the defined membership function allows us to provide multiple interpretations for the observed context data which resolve ambiguities and contradictions. In Example 2 (Sect. 5.2), the user is located in living room with fuzzy value 0.33 and in office with fuzzy value 0.67.

## 8 Conclusion and Future Works

In this work, we have presented an extension of CEP to handle multiple dimension of uncertainty namely freshness, accuracy, precision and contradiction. We have proposed a Fuzzy Semantic Complex Event Processing (FSCEP) model that operates in three steps. First step aims to gather events from devices using filters and to batch them using a classical CEP. Events are then semantized and enriched thanks to a background knowledge (ontology). This step (event semantization) includes the addition of a trust value on events. Finally, the events are analyzed to compute one fuzzy semantic complex event. The output of our

FSCEP can then be used by other applications, including activity recognition or situation identification.

This paper has only presented a first step and multiple perspectives of improvements are considered. Firstly, as performance and scalability are a serious matter in CEP, we aim to optimize and to measure the performance of our FSCEP with more complex scenarios. Secondly, an objective is to make our approach compatible with open smart environments where devices can be added or removed at runtime and the content of the environment is unknown at design time. Thus, we want to be able to discover new devices and to take them into account in the FSCEP without having to re-parameterize the system. Finally, as privacy is an important issue for final users, a possible perspective would be to add restrictions and permissions to user query in order to control the usage of our output event.

## References

1. Anicic, D., Rudolph, S., Fodor, P., Stojanovic, N.: Stream reasoning and complex event processing in etalis. Semant. Web **3**(4), 397–407 (2012)
2. Artikis, A., Etzion, O., Feldman, Z., Fournier, F.: Event processing under uncertainty. In: Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems, pp. 32–43. ACM (2012)
3. Brenna, L., Demers, A., Gehrke, J., Hong, M., Ossher, J., Panda, B., Riedewald, M., Thatte, M., White, W.: Cayuga: a high-performance event processing engine. In: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, pp. 1100–1102. ACM (2007)
4. Cugola, G., Margara, A., Matteucci, M., Tamburrelli, G.: Introducing uncertainty in complex event processing: model, implementation, and validation. Computing **97**(2), 103–144 (2015)
5. DAniello, G., Loia, V., Orciuoli, F.: A multi-agent fuzzy consensus model ina situation awareness framework. Appl. Soft Comput. **30**, 430–440 (2015)
6. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook, Theory, Implementation, and Applications. The DescriptionLogic Handbook, 2nd edn. Cambridge University Press, Cambridge (2010)
7. Event stream intelligence, E.: Espercomplex event processing (2010)
8. Lee, O.J., Jung, J.E.: Sequence clustering-based automated rule generation for adaptive complex event processing. Future Gener. Comput. Syst. **66**, 100–109 (2016)
9. Morrell, J., Vidich, S.: Complex event processing with coral8 (2007)
10. Rodríguez, N.D., Cuéllar, M.P., Lilius, J., Calvo-Flores, M.D.: A fuzzy ontology for semantic modelling and recognition of human behaviour. Knowl. Based Syst. **66**, 46–60 (2014)
11. StreamBase, I.: Streambase: Real-time, low latency data processing with a stream processing engine (2006)
12. Wang, F., Liu, S., Liu, P., Bai, Y.: Bridging physical and virtual worlds: complex event processing for RFID data streams. In: Ioannidis, Y., Scholl, M.H., Schmidt, J.W., Matthes, F., Hatzopoulos, M., Boehm, K., Kemper, A., Grust, T., Boehm, C. (eds.) EDBT 2006. LNCS, vol. 3896, pp. 588–607. Springer, Heidelberg (2006). doi:10.1007/11687238_36

13. Wasserkrug, S., Gal, A., Etzion, O.: A model for reasoning with uncertain rules in event composition systems. arXiv preprint (2012). arXiv:1207.1427
14. Wu, E., Diao, Y., Rizvi, S.: High-performance complex event processing over streams. In: Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data, pp. 407–418. ACM (2006)
15. Ye, J., Dobson, S., McKeever, S.: Situation identification techniques in pervasive computing: A review. Pervasive Mobile Comput. **8**(1), 36–66 (2012)
16. Zhang, H., Diao, Y., Immerman, N.: On complexity and optimization of expensive queries in complex event processing. In: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, pp. 217–228. ACM (2014)
17. Zhou, Q., Simmhan, Y., Prasanna, V.: Scepter: Semantic complex event processing over end-to-end data flows. Technical Report 12–926. Computer Science Department, University of Southern California (2012)