2010 IEEE Fourth International Conference on Semantic Computing

Ontology-Driven Event Detection and Indexing in Smart Spaces

Zang Li, Chao-Hsien Chu, Wen Yao
College of Information Sciences and Technology
The Pennsylvania State University
University Park PA, USA
Zul110@psu.edu, chu@ist.psu.edu, wxy119@psu.edu

Richard A. Behr
Smart Spaces Center
The Pennsylvania State University
University Park PA, USA
BEHR@engr.psu.edu

***Abstract*—By riding the tide of sensor technologies, smart space services collects information about its surroundings and internal objects. However, sensor readings are usually at primitive stage and difficult to be viewed and further retrieved since they have little meaning for naive users, who usually prefer to identify activities or state changes using high-level semantics or concepts. To bridge this gap, we propose an ontology-driven event processing framework as part of the middleware for smart spaces. Smart space event ontology (SSEO) is developed to enable semantic indexing, detect machine-processable events and exchange event data between different processes. A model named Smart Space Event Processors (SSEP) maintains and coordinates various event processes (e.g., event patterns, ontology reasoning rules, and machine-learning algorithms) for semantic events in a smart space. An implementation of SSEP—OntoCEP is introduced. The model elaborates on event composition and semantic labeling by combining event composition technology and semantic reasoning into one coherent system. It has been applied in a smart-space infrastructure named SENSIP.**

Keywords-ontology-driven; smart space; event processing

I. INTRODUCTION

Smart spaces, or called smart environment, represent the next evolutionary development in buildings, homes, hospitals, transportation systems, industries and government automation. By riding the tide of sensor technologies, the smart environment collects data about its surroundings and internal objects to support context awareness and intelligent inference. Event logs generated from multiple sensors of different modalities, such as ambient sensors, biosensors and RFID technologies are fed to various information viewing and analysis services. Users are notified with the events and perform different types of measurements. The logs also give a way of keeping track of how the smart-space (including surroundings and internal objects) are behaving. However, the event notifications and logs derived directly from sensor readings are usually at primitive stage and difficult to be viewed and further retrieved since they have little meaning for naive users, who prefer to identify activities or state changes using high-level semantics or concepts. For example, a doctor wants to retrieve or be notified with the events that a patient forgets to take medicine before sleep, rather than the events that the RFID tag on the patient has been detected by the reader nearby the bed. That is, users always want to get application views about the high-level information in smart spaces. Motivated by this, many research efforts have been made on exploring more efficient event detection and indexing technologies. An idea is to partition continuous sensor data streams by detecting the qualitatively significant changes based on which low-level events are extracted [1]. However, these change-detection approaches which do not consider the context and causal relationships among streams have less capability of discovering semantics and therefore, the detected events still do not make much sense to human perception. Usually, users would like the monitoring or retrieving tool to provide:

1) Real-time information on activity and status changes. An user would like to get a visual feel for the activities and status of the smart spaces. For example, a doctor wants to get a real-time view about a patient's activities, such as eating and sleep, and the health status.

2) Notifications about abnormal behaviors. Whether the behavior of patient conforms to certain instructions or regulations are important for doctors. For example, a doctor wants to be notified whether a patient is leaving her designated area.

3) Multiple-level indexes and summarization on activities and status changes. An user would like to log the activities at different levels for further retrieval and statistical analysis. For example, it is useful to present a pie graph of the percentage of different types of activities made by a patient in a day, or a bar graph describing the number of times forgetting taking medicines in each month of the year.

4) Activity Patterns and causal relationships among high-level events. Based on the event indexes, tools for pattern analysis might discover the patterns and the causal relationships among activities and status changes. For example, a doctor may find that a patient has a quality sleep after one hour of reading.

As we see, comparing with event logs from sensors, events about activities and status changes are of more semantics and can easily be identified by users. To retrieve this object, we proposed an ontology-driven event processing framework as part of the SENSIP middleware for smart spaces. Based on ontology information retrieval, the framework aims to extract high-level information about the activities and status changes happened in a smart space. Previous research has used ontologies to retrieve textual

978-0-7695-4154-9/10 $26.00 © 2010 IEEE
DOI 10.1109/ICSC.2010.63

resources [2], but applying ontology learning to sensor information is a totally new area. Especially this paper presents many new ideas:

First, we propose to develop event ontology (SSEO) and inference graph for semantic events in smart spaces. Several advantages can be achieved: (1) semantic indexing and detecting events with respect to a certain ontology makes it machine-processable and allows for exchanging this data between different processes since conceptualized events improves semantic interoperability for a wide variety of data sources and the heterogeneous event inference engines. (2) Detecting and indexing these semantic events allows a user to subscribe, search and mine information by specifying a request in terms of a limited vocabulary (keywords) of semantic concepts. And (3) The inference graph of the event ontology presents the tree of events in the causal ancestry of any event. By reviewing the update log of the inference graph, we can conduct a statistical analysis on the causal relations (e.g., taking a certain medicine always causes oversleep.

Secondly, a model named Smart Space Event Processors (SSEP) is developed to maintain and coordinate various concept detectors (e.g., event patterns, ontology reasoning rules, and machine-learning algorithms) for semantic events in a smart space. In this model, any process implements an interface which is explicitly labeled with the input and output events. By doing this, different types of concept detectors can interface with others on shared vocabulary (SSEO) and run in a chaining manner. This model provides a universally compliant framework to fit multiple usage environments. Additionally, the paper introduces an implementation of SSEP—OntoCEP, in which event composition processes based on Complex Event Processing [3] and semantic labeling processes based on ontology reasoning are integrated into a coherent system.

II. Related Works and Scenario

A. Related Works

Before the emergence of the idea of smart space, many researchers have studied to use sensor network technologies to monitor environment. In [4], a generic framework that integrates data acquisition network and data distribution network is proposed. A wireless sensor network is built in to meet different application needs. With the help of a behavior model, reference [5] used both ambient and wearable sensing for inferring changes in patient behavior patterns. These researches demonstrate the benefits of sensor networks in environmental monitoring; however, they lack the ability to fulfill some important needs (e.g., context-aware, complex event detection) of a smart space and thus cannot be evolved as a framework to solve the above challenges we mentioned.

In recent years, many infrastructures using the concept Smart Space are proposed. In [6], an extension to Java language, namely Metaglue, is developed to build agent systems for intelligent environments. The language aims to address the specific requirements for interactive, distributed computations. In [1], a software infrastructure for Smart Space (SISS) is proposed, based on a loose-coupling structure and a Publish/Subscribe coordination model. Reference [7] presents a Smart Space Lab (SSLab) environment, in which an Extended Transport Layer middleware is proposed to enable communication between heterogeneous networks. However, these frameworks do not provide a mechanism to process the sensor data and infer high-level information in a smart space.

Research has also been done on using Semantic Web technologies for context inference in a smart space. In [8], a context infrastructure is proposed. Using an ontology-based context model, the infrastructure successfully handles some context-aware issues in a smart space, such as explicit representation and information querying reasoning. However, this context model does not involve the stream characteristics of sensor data; moreover, it has only considered the facts derived from ontology knowledge, while ignoring various relationships among data streams and the complex events inferred from low-level events.

B. Scenario

Let's consider the following scenario: the Greens live in a house. The elder, Bob Green, has heart disease and has retired for several years. From Monday to Friday, the other family members go out to work, whereas Bob Green stays at home. In order to prevent the occurrence of any emergency, the house is installed with various sensors (e.g., temperature, humidity, light sensor). Additionally, Bob wears a wearable blood pressure sensor and an RFID wristband with a temperature and motion sensor so her identity, temperature, location and motion status can be captured in time. The sensor middleware is running, collecting sensor data and conducting data processing. To monitor the status of Bob, Alice, Bob's daughter, subscribes Bob's activity and status information. By doing this, Bob's information on Alice's laptop or cell phone can be updated in a real-time manner. Moreover, Alice will be alerted if Bob is likely to have any emergency. For example, Bob is still for a while in a spot, which is not a rest area (e.g., bed, sofa, etc.) or his blood pressure is greater than 140/90 mmHg. On the other hand, Bob's doctor, John retrieves and analyzes Bob's activity and status data every month, summarizing Bob's health condition and discovering any patterns, for example, how a new medicine affects Bob's sleep quality.

III. Ontology-driven Event Processing Framework

A. Semantic Events

As we discussed, smart-space technologies are enabled by context-aware applications which adapt to the changes of environment and human activities or statuses. These changes are usually named as events. The followings are formal definition of event and the categories.

Definition. *Event, sensor event, semantic event*. A *event e* is defined as a record of an activity in a system for the processing purpose [9]. In our case, an event *e* can be

defined as a triple, e = (*ID*, *tp*, *t*), where *ID* means the event ID, *tp* is the event type and *t* is the timestamp of the event. According to the sources of events, an event can be categorized as *sensor event* or *semantic event*. A *sensor event* is the atomic record created by one sensor device or logical device. A *semantic event* is an easily human-understandable event which was deduced by semantic inference techniques, such as pattern-based and machine-learning-based approaches.

In a smart space, we can employ event hierarchy which is believed to be a powerful way of organizing how we view an event processing system [9]. The hierarchy consists of a sequence of levels of activities and each level defines a set of event types which have the same degree of abstraction. The event at higher level is aggregated and inferred by the events at levels below it. For the previous scenario, a three-level event abstraction hierarchy is shown in Table 1.

TABLE I. SEMANTIC EVENTS IN SMART SPACES

L	Category	Event Types
3	Activities	*Studying, FallSleep, PeopleInEmergency ForgetMedicine, HeartAttack, TooCold*
2	Actions	*OpenDoor,SitDown,TouchBooks,OpenMedicineCabinet, GoToBed, LieDown, TurnOnLight StillFor1Min,ShortOfBreath,BloodPressureHigh, ChestDiscomfort*
1	Primitive Semantic Events	*EnvTemp, EnvHumidity, Brightness, AirQuality, BodyTemp,BloodPressure,ObjectMotion,ObjectLocation, Vibration, RFScanning, EKG*

Table 1 shows the hierarchy of semantic events. At the lowest level, there are events signifying the attribute information of objects and environment at different time points. Most of these events are directly generated from the raw data from sensors or other monitoring devices (e.g., RFID, cameras, etc.), by referring to the relationships among entities and devices in ontology knowledge base. At level 2, we see semantic events that aggregate sets of other events for level 1 or 2 based on the predefined aggregation rules and inference rules, which specify the patterns of level 1 events and how they should be related by time and causality. Level 3 events are more sophisticated events inferred by other events, for example, *studying*, which involves many level 2 events such as *TurnOnLight*, TouchBooks and *StillFor1Min*. Levels 2 and 3 events are usually subscribed by users and applications that need to know the context information in real time. These events are also named as *Subscribed Events*.

B. Semantic Event Detection Framework

In this section, we describe our proposed semantic event detection framework (SEDF) for smart spaces. The framework is shown in Figure 1. Major components in the framework are introduced as follows.

Sensor Cloud consists the sensor readings collected from various sensors, including wired/wireless sensors, RFID devices and even virtual data generators such as GUI for data entry and sensor simulator. The sensor cloud also deals with various formats of sensor readings supplied by different vendors. For example, two types of RFID devices, Xtive SYRD245 and Alien ALR-9800, are both used in our platform to identify assets or people. However, the readings from Alien reader are normally in XML or text format, whereas those from Xtive reader outputs are binary readings. The sensor cloud decodes these readings and translates them into unified event objects, which are sent to the smoothing aggregation processor and semantic adaptor.

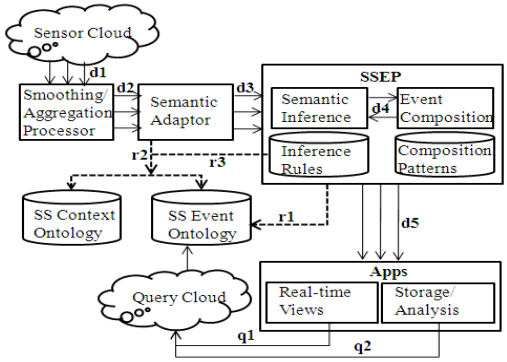

Figure 1. Semantic Event Detection Framework

Query Cloud includes a variety of queries from *applications* (*Apps*). One major application in smart spaces is to monitor the activities or state changes of the objects in smart spaces in real-time. With Publish-Subscribe messaging paradigm, the SEDF accepts subscription requests of some semantic events from applications and pushes back the corresponding event instances if they are detected. For storage and purpose of later-time analysis, the SEDF also indexes all semantic events defined in event hierarchy and store them to the database. These subscriptions and indexing requests are specified in terms of a limited vocabulary of keywords and then pooled into the query cloud. The query cloud maps the keywords to the event concepts defined in SSEO. The event concepts, also called subscribed events, will be used as goals when backward chaining paradigm is applied by SEDF.

Suppose we observe the following sensor streams $\mathcal{S}1$, $\mathcal{S}2$ provided by sensor cloud and the subscription $\mathcal{Q}1$ queried by query cloud:

$\mathcal{S}1$(*blood pressure*) = {…,(t1)97/82 *mmHg*, (t2)99/85 *mmHg*,…}; $\mathcal{S}2$(*motion*) = {…, (t3)*still*, (t4)*still*, …}

$\mathcal{Q}1$(*Heart Attack*) = {(*blood pressure*>= 140/90*mmHg*) ∧ (*motion* =*still* for at least 30s)}

As we can see, users always query multiple, independent data streams and expect meaningful answers. It is difficult for human to capture a heart attack event by viewing the streams $\mathcal{S}1$, $\mathcal{S}2$. Bridging the gap between what we are given in sensor cloud and what we are expect to see in query cloud is the major mission of SEDF.

Smart Space Event Ontology (*SSEO*) *Knowledge base* defines the event concepts in smart spaces. As discussed, SmartSpace services require tools for detecting, indexing, retrieving and analyzing events. One possible approach is to use the index-by-source structure in which system detects and indexes the events and users retrieve and analyze events

based on the event source or event identifier, such as sensor ID or event composition pattern. As we pointed out, such mechanisms have obvious limitations since users wish to subscribe or retrieve in terms of semantics or concepts. However, automatically detecting and indexing high-level events bring several challenges, for example, i) the events should be machine understandable; ii) the events are not rely on certain detection techniques, but can be processed through various algorithms; and iii) constraints and event relationships can be clearly defined. All of these motivate the design of SSEO to model the various event types signifying activities or state changes in smart spaces. More importantly, the inference relationships among events are presented in SSEO in an execution explicit. For example, suppose we use a "<owl:Class rdf:ID="FallSleep">" entity to defines an event type "FallSleep", and the object property <owl:inferredBy> markups inference dependencies in an OWL ontology, then we could use the following syntax:

```
<owl:Class rdf:ID="FallSleep">
<rdfs:subClassOf><owl:Restriction>
    <owl:someValuesFrom><owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#People_Location"/>
        <owl:Class rdf:ID="StillFor1Min"/>
      </owl:intersectionOf>
    </owl:Class> </owl:someValuesFrom>
    <owl:onProperty>
      <owl:ObjectProperty rdf:ID="inferredBy"/>
    </owl:onProperty>
  ……
```

As we see, every event concept has a set of <rdfs: inferredBy> properties reference to the event concepts that potentially inferred the event. Placing these causal references in every event concept gives us a practical way to trace causal relationships among events. Especially, we can define:

Definition SSEO Inference Graph. An *Inference Graph* is a collection of instantiated event concepts and can be denoted as a 2-tuple $\mathbb{G}=(\mathbb{V},\mathbb{E})$, where $\mathbb{V}$ is set of nodes, representing the set of event concepts and their related dynamics, and $\mathbb{E}$ is a set of directed edges, representing the i*nferredBy* relationship set among concepts.

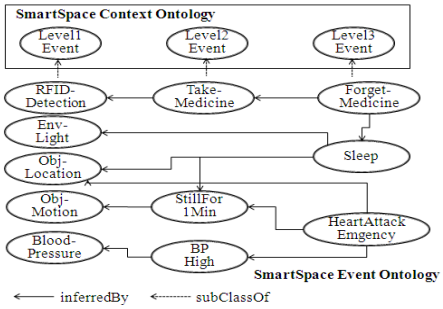

Figure 2. Inference Graph

Figure 2 shows an inference graph of a simplified SSEO. The processing framework reads the SSEO into memory and stores it as an inference graph. Each node has information including the availability of the corresponding stream and the related processes which use the event as inputs or outputs. The purpose of inference graph is to facilitate the chaining process of the event processing framework.

Smart Space Context Ontology (*SSCO*). The SSCO is designed to support knowledge sharing and context-aware services. The context ontology model describes semantic relationships among concepts and the properties of each concept. Figure 3 depicts a context ontology used in the current implementation. We define *Entity*, *Device* and *Event* as top-level concepts of a smart space. The concept *Entity* defines the general features of the monitored objects in a smart space. Every *entity* instance is uniquely identified by a *URI* in the system. An entity instance can be a *People*, or *Asset*, or *Zone* instance. Class *People*, *Asset* and *Zone* can be further divided according to experience and requirements. For example, for indoor applications, *Zone* can be divided into *LivingRoom* and *BedRoom*, which includes *SleepArea* and *Non-SleepArea*. An entity *People* can be classified into *PeopleWithWatcher* and *PeopleBeingMonitored* according to the roles a people plays in the system. Each entity has data properties. A data property of an entity can be either non-temporal information, e.g., *name*, *gender*, *salary*, etc., or temporal information, e.g., *body temperature*, *blood pressure*, *motion status*, etc. The temporal properties can be updated by corresponding event instances. The *Device* class defines the general features of a device, which collects or presents information in the system. Two sub-classes are defined: *Watcher* and *Sensor*. A watcher device is used to present Smart Space information. It can be a laptop, PDA or phone, etc. *Sensor* is a concept generalizing devices which collects sensor data in a smart space, for example, temperature sensor. Any device instance is always assigned to an entity in smart space; that is, given a device *d*, $\exists(d, assignedTo, et)$, where $et \in Entity$.

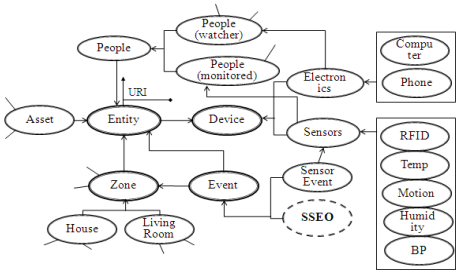

Figure 3. Semantic Event Detection Framework

Smoothing Aggregation Processor removes noisy readings, gathers missing readings, and reduces redundant data to ensure data quality. Based on temporal and spatial information for time series data, different algorithms (e.g., sliding window[10], particle filter[11]) can be applied to process different types of sensor data.

Semantic Adaptor translates sensor readings to primitive semantic events based on SSEO and context ontology. For example, Xtive SYRD245 and Alien ALR-9800 use different name and format for the attributes. Some of the attributes are supposed to have the same semantic meanings. For instance, Alien devices use *TagID* as the name of identifiers that conform to the EPCglobal standard, whereas Xtive uses *UID*

that conform to their own identifier format. By enabling agreement on the meaning of specific vocabulary terms, semantic adaptor facilitates the correlation and composition across multiple event streams. Now we use the following procedure to illustrate the execution of smoothing aggregation process and semantic adapting process.

Procedure 1. Semantic_Adaptor_Processing()	
Step 1:	*r*=**SmoothingAggregation**(**getNextReading**()), where *r* is a cleaned sensor event;
Step 2:	*et*=**SearchEntity**(*condition*), where *condition*=(*cid*, *assignedTo*, *et*). Search the *SSCO* to determine which entity the sensor is attached to;
Step 3:	E_1=**GenerateEvents**(*r, et.URI*), where E_1 is the set of new events. Referring to the *SSEO*, generate a set of primitive semantic event related to *et*;

For illustration, if an Xtive sensor identified by *cid* is detected by the reader identified by *rid*. Processed by query cloud and smoothing aggregation processor, the sensor event, r_1 = (*cid*, *rid*, *t*, *d*) is outputted, where *t* is the timestamp, *d* is the data of the reading, for example, *motion*=*still*. In step 2, the semantic adaptor executes step 2:

et_1=**SearchEntity**(*condition1*), where *condition1*= (*cid*, *assignedTo*, et_1); z_1=**SearchEntity**(*condition2*), where *condition2*= (*rid*, *assignedTo*, z_1).

In step 3, referring to *condition1* and *condition2*, E_1=**GenerateEvents**(*x*, et_1.*SIP_URI*) = {e_1, e_2}, where e_1, e_2 are defined by *ObjectMotion* and *ObjectLocation* respectively in *SSEO*, e_1=(et_1.*URI*, *t*, *still*) and e_2=(et_1.*URI*, *t*, z_1)

Smart Space Event Processors (*SSEP*) are a set of event learning processes operating on streams of semantic events. Each process fulfills one task to infer one or more implicit events from a set of available events. The processes call various event processing engines, which can be either based on patterns or machine learning models. As will be introduced later in this paper, the event composition engine and context inference engine are combined to generate semantic events.

Normally these processes, usually developed by system administrator, are designed to support interoperable process-to-process interaction in SSIP framework. To fulfill this goal, each process has an interface describing the input and output event concepts, which are strictly defined in SSEO. One example of inferring event stream *C* from *A* and *B* is shown in Figure 4. By doing this, multiple processes can be combined to obtain subscribed events on pre-defined plan templates.

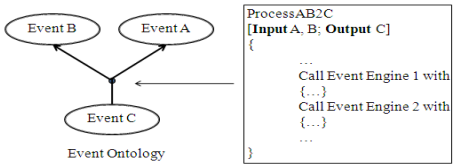

Figure 4. Infer Events in SSEP

IV. ONTOCEP: A SEMANTIC EVENT DISCOVERING MODEL

In this section, we introduce the details of a semantic event discovering model-OntoCEP, which is developed on the ontology-driven event processing framework. OntoCEP elaborates on event composition and semantic labeling by combining event composition technology and semantic reasoning into one coherent system. The OntoCEP has been applied in our smart-space infrastructure named SENSIP.

A. Abstraction Model

In OntoCEP, a *semantic event* is inferred by matching one or more events in any special pattern using a set of event operators such as conjunction and sequence, and then assigning context semantics to the event. Two event inference engines are employed as the core of the SSEP: Complex Event Processing (CEP) Engine and Context Inference Processing (CIP). The major task of CEP engine is to identify event patterns which signify the correlations between events in a huge event cloud using pattern rules and thereby determine corresponding actions. The CEP engine uses event processing languages (EPL) based on event algebras [12] to define event streams and event operators that aggregate events into complex events. On the other hand, by linking the context knowledge defined in the smart-space context ontology, the CIP receives the available events, infers implicit events by combining the context information defined in SSCO. Guided by SSEO, the CEP model and CIP model are called by the various event inference processes in SSEP.

B. Implicit Events Based on Event Composition

To composite semantic events from low-level events, complex event processing is adopted in our system. CEP includes a set of techniques and tools to assist the framework understand and control the event-driven information system. The major task for CEP is to derive higher level information using event patterns that are executed upon all available events. In OntoCEP, CEP engine conducts the detection of complex patterns of events, event correlation and abstraction based on the relationships defined in SSOE, such as causality, membership, and timing, and generates meaningful events.

TABLE II. OPERATORS FOR EVENT COMPOSITION

Operator	Expression	Meaning
AND(∧)	$e_1 \wedge e_2$	Conjunction of two events disregarding their occurrence orders
OR(∨)	$e_1 \vee e_2$	Disjunction of two events disregarding their occurrence orders
NOT(¬)	$\neg e_1$	Negation of an event
Every(*)	e_1*	Event occurrence of an event
Occurrence(n)	e_1(n)	The event e_1 occurs *n* times
SEQ (;)	SEQ(e_1; e_1)	Event e_1 occurs followed by event e_2
At ()	At(e_1, *t*)	Event e_1 occurs at time *t*
Within()	Within(e_1,*t*) , Within(e_1, t_1, t_2)	Event e_1 occurs within less than *t* seconds Event e_1 occurs within interval t_1 and t_2.

Event operators are designed to express the relationships among events and thereby define any event patterns to detect variation of the context. In our paper, we adopt the set of event operators introduced in [3] to express the event patterns.

As we see, event composition patterns are able to depict timing relationship among events, which cannot be fulfilled by normal knowledge inference engine. A CEP engine identifies the meaningful events, analyzes their impact and then notifies users or system to take subsequent actions in real time.

C. Implicit Events Based on Context inference

Smart-space services are expected to be context-aware; that is, semantic events must be inferred by perceiving the current context, and accordingly, the inference engine adapt their processing behavior to different context clues, e.g., time, location, subject profile and environmental information.

In OntoCEP, the CIP model inferred the implicit events which cannot be discovered by pattern-based event composition. CIP links to the expertise knowledge and static information defined in SSCO, translates an event object into a higher level event with more semantic meaning, and thereby supports semantic interoperability and context-aware services.

In semantic web, ontology-based reasoning [13] can infer implicit information from explicit data based on class relationships and property characteristics. Inference rules entailed semantics can be defined by conducting analysis on relationships such as *subClassOf*, *propertyOf*, *containedBy*. In OntoCEP, guided by the relationships among concepts and defined rules, CIP infers implicit events by combining logical reasoning techniques such as conditional test. For example, given an event "a person p is still in a spot z for a while t", if the system refers to the *SSCO* and identifies the facts that (p, *InstanceOf*, *patient*), (z, *instanceOf*, *NotSleepArea*) and (t > 2 mins) , then the system infers an semantic event implying the patient has lost consciousness and fallen on the ground on sudden; otherwise, if spot z is identified as a bed then the systems infers that the patient has fallen into sleep.

D. How OntoCEP Works

In this section, the mechanism of how OntoCEP integrates event composition with context inference is illustrated. Figure 5 explains how the events are processed through the data processing model. As we see, primitive events generated from semantic adaptor are sent to the OntoCEP to be pooled as available events. An SSEO inference graph has been built in the memory. A node in the graph maintains the dynamics about the corresponding event concept, for example, the availability, when the last time the event is seen, and so on. Referring to the graph, OntoCEP runs in a forward-chaining (note that with the inference graph in SSEO, it is easy to execute backward-chaining). The OntoCEP starts with these primitive events and select the processes of which the input events are known to be available. A listener process receives new events from the selected processes and marks their corresponding node in SSEO as available for other processes. By iterating through the process, OntoCEP can detect and index as many semantic events as possible.

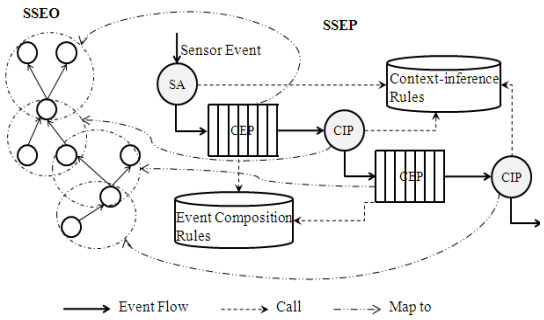

Figure 5. How OntoCEP Works

For current implementation, the event composition processes and context inference processes are called in interleaved manner. By "inserting" CIP into CEP engine, it is possible that the events on different levels can be processed by CIP and thereby the system can infer implicit events as many as possible. In particular, the CIP is called in when the CEP engine posts events to its listener process. The process in output listener can be illustrated in **Procedure 2**, where is e_o is an output event posted from CEP.

Procedure 2. Listener_Processing(e_o)	

Step 1:	IF e_o in QueryCloud
	THEN **Emit**(e_o); //output e_o
	ENDIF;

Step 2:	SP = **SelectProcess**(e_o,*SSEO*);
	E_1=∅;
	FOREACH *process* in SP
	E_1=E_1∪*process*.**Run**(e_o, SSCO) ;// E_1 is a set of implicit events inferred by CIP
	ENDFOR

Step 3:	FOREACH e IN E_1
	IF e in QueryCloud
	THEN **Emit**(e); //output e
	ENDIF;
	SendEventToCEP(e) //send events back to CEP
	ENDFOR;

In step1, the system outputs the e_o if it is subscribed or needs to be stored; otherwise, the procedure **SelectProcess** in step 2 is called, which selects the related processes by searching in *SSEO* (matching input events). Each process inputs e_o and generates a set of implicit events by linking to the SSCO. These implicit events are then labeled as available. In step 3, the new events are outputted and sent back to the event composition processes. Insider of the CEP engine, the procedure **SelectProcess** is called again to decide the related event composition processes. Their implicit events are posted back to the listener process.

We use the following example to explain the whole process of the data processing model. In this example, we try to provide a solution for the introduced scenario. For the purpose of explicitly, for CIP processing we use first-order logic with operators (e.g., *AND*(⋀), *OR*(⋁), *IMPLY*(→), etc.)

to represent the inference rules. They can be easily converted to OWL-DL statements for implementation. For CEP processing, we use a simplified event composition language based on the operators in Table 2 to define the aggregation rules among events,

According to the description of the scenario, what we have in sensor cloud includes sensor readings from the environmental sensors (i.e., temperature, humidity sensor, light) as well as the body sensors (i.e., motion, blood pressure). RFID scanning records can be used to infer the location of objects attached with RFID tags. In the query cloud, we can extract a set of subscribed events including *peopleInEmergency*, *peopleInSleep*, *ForgetMedicine* from the subscription from Alice and the potential indexes defined by the doctor.

After cleaned and aggregated by smoothing aggregation processor, the semantic adaptor translates the sensor events to primitive semantic events. For example, the following two ontology reasoning rules can be applied:

SA Rule1	SA Rule2
(?People, equippedWith, ?RFTag)	(?People,
∧ (?RFTag, detectedBy, ?Reader)	equippedWith, ?MotionSensor)
∧ (?Reader, covers, ?Zone)	∧(?MotionSensor,detectedAs,?Motion)
→ (?People, hasLocation, Zone)	→ (?People, hasMotion, Motion)

These primitive events (e.g., *ZoneLight, ObjectMotion*, *ObjectLocation*) are sent to the OntoCEP and the corresponding nodes in inference graph are updated. Referring to the graph, OntoCEP selects the processes of which all input events are known to be available. For example, the system uses the following process to detect the composited event *stillFor1Min*, representing that one object has been still for one minute in a certain place.

Process1(**Input** objMotion, **Output** stillFor1Min)
{…
Call CEP with Pattern
{
Within((objMotion(motion=still)) (10), 60 seconds)
∧ objLocation
→ stillFor1Min
}
…}

When an event *stillFor1Min* is posted to the listener process, the processes with input *stillFor1Min* are selected. For example, the following **process2** is triggered. The process calls the CIP to infer whether the people is in sleep or emergency based on two defined reasoning rules related to the context. Based on the inference results, an implicit event, either *peopleInSleep* or *peopleInEmergency* is outputted and stored.

Process2(**Input** stillFor1Min; **Output** peopleInSleep, peopleInEmergency)
{…
Call CIP with Rule
{
(?People, stillFor1Min, ?Zone)
∧ (?location, subClassOf, ?SleepArea)
→ (?People, sleepIn, ?Zone)
(?People, stillStatus1Min, ?Zone)
∧ (?location, subClassOf, ?NotSleepArea)
→ (?People, EmergencyIn, ?Zone)
}
…
}

Since the event *peopleInSleep* or *peopleInEmergency* now is known be true, semantic events in higher level can be inferred. For example, the event *forgetMedicine* can be detected by calling process3:

Process3(**Input** takeMedicine, peopleInSleep; **Output** forgetMedicine)
{…
Call CEP with Pattern
{
SEQ(¬ takeMedicine; peopleInSleep)
→ forgetMedicine
}
…
}

E. *Apply OntoCEP to SENSIP*

The OntoCEP is adopted in our smart space infrastructure, SENSIP. The SENSIP aims to i) make use of various sensor devices; ii) process semantic events in real-time by providing context awareness and intelligent inference of surroundings as well as internal settings of the smart environment; and iii) share smart space information over a broad range of network environment. The current implementation of SENSIP is developed with Visual C# and employs the event composition engine of ESPER.

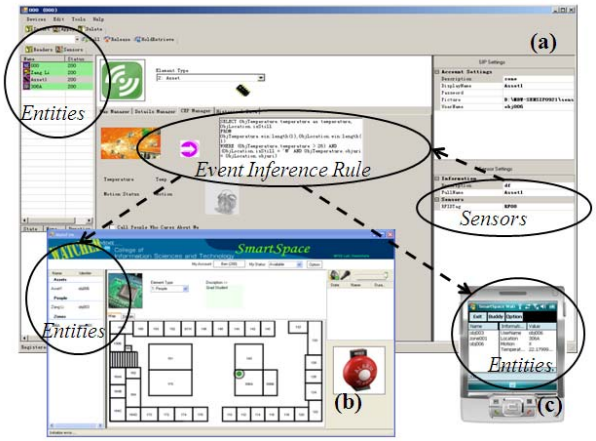

Figure 6. SENSIP Components: (a) SENSIP Middleware; (b) Watcher for PC; (c) Mobile Watcher

Figure 6 presents some snapshots of the SENSIP components. As shown, people are allowed to specify queries using PC or mobile watchers, which communicate with SENSIP middleware through SIP event communication infrastructure [14]. The SENSIP middleware manages various sensor devices in a smart space and maintains entities and context cues in context knowledge base (SSCO).

The middleware maintains the sensor cloud and query cloud (queries are extracted from SIP queries) and uses OntoCEP to fulfill the inference from sensor readings to high-level information (see the arrows in the graph). Once these high-level events are detected, the middleware notifies the corresponding watchers and the events can then be presented in various forms (e.g., map, alarm).

V. CONCLUSION

In this paper we have presented a novel framework for semantic indexing and detecting of events in smart spaces. The proposed framework aims to bridge the gap between what to get from sensors and what the user hopes to know about the surroundings and internal objects in a smart space. The framework mainly focuses on processing of semantic events, which is human-understandable and easy to specify for users to use keywords. Smart space event ontology (SSEO) is developed to enable semantic indexing and detecting of events machine-processable and exchanging event data between different processes. A model named Smart Space Event Processors (SSEP) maintains and coordinates various event processes (e.g., event patterns, ontology reasoning rules, and machine-learning algorithms) for semantic events in a smart space. Moreover, an implementation of SSEP—OntoCEP is introduced. The model elaborates on event composition and semantic labeling by combining event composition technology and semantic reasoning into one coherent system. Its deployment in SENSIP smart-space infrastructure has verified its effectiveness.

We envision some enhancements to the framework. The first one is to introduce statistical processes to the SSEP since the original events from sensor readings are still of uncertainties even though processed by smoothing aggregation processor. We need to assign probabilistic labels to the nodes of inference graph and plan to employ machine learning algorithms (e.g., particle filter, Hidden Markov Model) for time series to join and infer event streams.

ACKNOWLEDGEMENT

This work was supported in part by the College of Information Sciences and Technology and the Smart Spaces Center at Pennsylvania State University, University Park, PA USA.

REFERENCES

[1] V. Guralnik and J. Srivastava, "Event detection from time series data," in Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, 1999, pp. 42.

[2] S. Handschuh and S. Staab, Annotation for the semantic web: IOS Pr Inc, 2003.

[3] F. Wang, S. Liu, P. Liu, and Y. Bai, "Bridging physical and virtual worlds: Complex event processing for RFID data streams," Lecture Notes in Computer Science, vol. 3896, pp. 588, 2006.

[4] F. L. Lewis, "Wireless sensor networks," Smart environments: technologies, protocols, and applications, pp. 1–18, 2004.

[5] L. Atallah, M. Elhelw, J. Pansiot, D. Stoyanov, L. Wang, B. Lo, and G. Z. Yang, "Behaviour profiling with ambient and wearable sensing," in IFMBE PROCEEDINGS, vol. 13, 2007, pp. 133.

[6] M. Coen, B. Phillips, N. Warshawsky, L. Weisman, S. Peters, and P. Finin, "Meeting the computational needs of intelligent environments: The metaglue system," in Proceedings of MANSE'99, 1999, pp. 201–212.

[7] T. Okoshi, S. Wakayama, Y. Sugita, S. Aoki, T. Iwamoto, J. Nakazawa, T. Nagata, D. Furusaka, M. Iwai, A. Kusumoto, and others, "Smart space laboratory project: Toward the next generation computing environment," in IEEE Third Workshop on Networked Appliances (IWNA 2001), 2001.

[8] X. Wang, J. S. Dong, C. Y. Chin, S. R. Hettiarachchi, and D. Zhang, "Semantic Space: an infrastructure for smart spaces," IEEE Pervasive Computing, pp. 32–39, 2004.

[9] D. Luckham, The power of events: an introduction to complex event processing in distributed enterprise systems: Springer, 2002.

[10] S. R. Jeffery, M. Garofalakis, and M. J. Franklin, "Adaptive cleaning for RFID data streams," in Proceedings of the 32nd international conference on Very large data bases: VLDB Endowment, 2006, pp. 174.

[11] L. Reznik, G. Von Pless, and T. Al Karim, "Embedding intelligent sensor signal change detection into sensor network protocols," in IEEE SECON, 2005.

[12] J. Schiefer, S. Rozsnyai, C. Rauscher, and G. Saurer, "Event-driven rules for sensing and responding to business situations," in Proceedings of the 2007 inaugural international conference on Distributed event-based systems, 2007, pp. 205.

[13] X. H. Wang, D. Q. Zhang, T. Gu, and H. K. Pung, "Ontology based context modeling and reasoning using OWL," in Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, vol. 18: Citeseer, 2004.

[14] A. B. Roach, "RFC3265: Session Initiation Protocol (SIP)-Specific Event Notification," RFC Editor United States, 2002.