PAPER

Context-Aware Dynamic Event Processing Using Event Pattern Templates

Pablo Rosales Tejada[†], ***Nonmember*** **and Jae-Yoon Jung**[††a)], ***Member***

SUMMARY A variety of ubiquitous computing devices, such as radio frequency identification (RFID) and wireless sensor network (WSN), are generating huge and significant events that should be rapidly processed for business excellence. In this paper, we describe how complex event processing (CEP) technology can be applied to ubiquitous process management based on context-awareness. To address the issue, we propose a method for context-aware event processing using event processing language (EPL) statement. Specifically, the semantics of a situation drive the transformation of EPL statement templates into executable EPL statements. The proposed method is implemented in the domain of ubiquitous cold chain logistics management. With the proposed method, context-aware event processing can be realized to enhance business performance and excellence in ubiquitous computing environments.

key words: *Context-awareness, complex event processing, ubiquitous logistics, RFID, wireless sensor network*

1. Introduction

Information technologies have played a key role in logistics by providing tools that have greatly enhanced their effectiveness and efficiency; bar codes and electronic data interchange can be highlighted [1,2]. Recent technological developments make available new components and combinations of them at the disposal of logistics companies. The first are sensors technologies, key enablers of the vision of ubiquitous computing [3,4]. In the context of ubiquitous logistics the following sensor technologies deserve to be mentioned: a) radio frequency identification (RFID), which provides identification and data storage to facilitate the location of objects through supply chain [5,6]; b) temperature and humidity sensors, which allow monitoring and logging temperatures and humidity of products and environments along their transportation and storage [7,8]; c) security devices such as door locks that allow recognizing and controlling the state (open/closed) of the door at a particular location [9]. These technologies have in common two advantages that make them especially attractive for logistics: automatic data capture and real-time data collection [10]. The second family of technologies is complex event processing (CEP). CEP enables proactive action by real-time event processing as follows: filtering out irrelevant information, aggregating events, monitoring events at every level in information systems, detecting patterns of events, tracing causal relationships between events in real-time and taking appropriate actions when patterns of events of interest are detected [11]. And the third family of technologies is context-aware systems, which provide significant functionalities of surveying the environment, reasoning about its information and taking the corresponding action [12].

Although three kinds of technologies provide many solutions for logistics management, they still have limitations of directly applying to practical complicated logistics environment. For example, in the cold chain logistics, which we are focusing in this paper, it is required to monitor and control various physical sensors (e.g. temperature, humidity, CO_2) for the purpose of preserving the best quality of foods such as fruit, vegetable, meat, and diary. However, it is not so easy to integrate three kinds of technologies to one another while configuring and maintaining a variety of food types, logistics activities, facilities like vehicles and refrigerators, etc. For instance, since the suitable temperature and humidity are dependent on the food type, we have to configure the best condition to every vehicle and refrigerated storage according to the food types. To address the issue, we present a new approach to intelligent ubiquitous event processing by using CEP and context-aware technologies. It is the combined issue of context-awareness and ubiquitous complex event processing.

In this paper we propose an approach to context-aware dynamic event processing. It is performed by a context-aware system to generate semantic event processing language (EPL) statements to be executed in CEP engines. Through the proposed approach we address questions that arise in the context of ubiquitous logistics environment: What are the monitoring activities associated to a situation? What are the EPL statements used to monitor that situation? What are the appropriate actions to the situation based on the context?

Ubiquitous cold chain logistics is a good example environment which the context-aware complex event

Manuscript received July xx, 2012
Manuscript revised xx xx, 20xx.
†The author is with Universidad de San Carlos de Guatemala, Zona 12, Ciudad de Guatemala, Guatemala.
††The author is with Kyung Hee University, Yongin, Gyeonggi-do, 446-701, Republic of Korea.
a) Corresponding author, jyjung@khu.ac.kr

processing is required to apply. In such logistics environment, it is necessary to control many kinds of ubiquitous sensing devices (e.g. temperature, humidity, CO_2, etc.) according to a variety of food types such as diary, meat, fish, and fruit. CEP technology can filter and detect event patterns for the huge amount of sensor event streams, while context-aware technology can support intelligent control according to the food types detected by RFID data. Although main functions of the two technologies could be implemented by conventional technologies, the complexity of the system will increase seriously because of event processing many devices and food types.

On the contrary, in this paper we can edit flexibly the event processing patterns and rules by adopting CEP technology, of which core components include CEP engine and EPL language. The two components enhance flexibility and scalability of event processing in integrated information systems like logistics. And we also introduced context-awareness to reflect a variety of food models to lessen the complexity of the integrated systems. These can be said to be advantages of the context-aware complex event processing proposed in this research. In particular, EPL language plays a critical role of flexibility and scalability by enabling automatic generation of EPL statements in this research. Suppose that we do not use EPL language which can be executed in the corresponding CEP engine. We have to declare so many similar event processing rules for each type of foods, and each type of refrigerators. But, because we adopt EPL language and CEP engine, we can register some EPL statement templates shared to food types, and the remainder of the complex environments (e.g. temperature thresholds for each food types, managers of each refrigerator, drivers of each vehicle, etc.).

By applying the proposed method, we foresee benefits from two perspectives. From the technical point of view, executable EPL statements are automatically generated and enabled when the associated situations arise in the logistics environment. And, from the business point of view, the logistics process management can be enhanced since sensor readings and domain knowledge drive effective activity monitoring.

2. Related Work

A great number of previous studies have developed their stand-alone system of context-aware event processing for the last decade until CEP technology was developed. SOCAM (Service-Oriented Context-Aware Middleware) was a service-oriented middleware that had been developed for building context-aware services [13]. They also proposed a formal context model based on OWL ontology language. In [14], SOCAM was also adopted for developing RFID middleware to support the RFID context-aware service model. Gao et al. [15] analyzed sequential event patterns to create context-aware adaptive application although they did not adopt CEP technology (e.g. CEP engine or EPL language). Such studies tried to develop concrete and robust context-aware event processing systems, but they cannot be regarded as flexible and scalable systems because their event processing technique is dependent on their stand-alone systems.

On the other hand, recent studies on event processing started to adopt CEP engine and EPL language, which are core component of CEP technology, even though these engines and languages are not yet standardized in the CEP application domain. CEP technology was created to support flexible and rapid processing of huge event streams. If EPL language is used to define the even patterns and rule processing, it is very easy to configure the event patterns and integrate the event processing system to many legacy systems such as RFID middleware, sensor system, and warehouse management systems. For the flexibility and scalability of CEP technology, we also apply it for context-aware dynamic event processing in ubiquitous event processing.

CEP is an emerging technology of filtering, aggregating, and detecting event patterns from a variety of event streams such as sensor data, network speed, stock ticks, and web search queries. The technology can be combined to ubiquitous technologies in many domains of application, which vary from logistics, manufacturing, oil industry to entertainment. In this section, we especially summarized the related works on ubiquitous CEP and context-aware CEP, and then we compared our approach and the previous research.

There are a few relevant work of applying CEP to ubiquitous event processing such as sensor and RFID data. Dunkel [16] proposes a reference architecture to process and analyze complex event streams in real-time. The architecture is built upon several concepts and techniques. First, event-driven architectures provide the architectural concept to deal with streams of events. Second, ontologies are used to model the structure of events and their relationships. Third, CEP is used as the process model for event-driven decision support. And fourth, sensor networks are used as the elements that capture data from the environment and push it into the event processing system. The processing of RFID events is approached by using CEP in [17]. Some of the issues related to the nature of RFID events, including aggregation of events, high volumes of data and on-the-fly processing, are addressed with a framework. The framework includes the formalization of the specification and semantics of RFID events and rules; it also provides an RFID detection engine to process complex RFID events. This framework can be used in applications such as object tracking and real-time monitoring. Sensing technologies and CEP have also been integrated in the domain of cross-reality environments in the DejaVu system [18]. However, the practical sensor event processing systems such as cold-chain logistics environment required more complicated context-aware information processing. For example, they involve a variety of food types, many kinds of vehicles, and lots of refrigerators and storages, which increased the complexity of event processing. They result in the necessity of adopting context-awareness into the ubiquitous event processing environment [19, 20], which is focused in this paper. To deal with the issue, we proposed a novel approach to context-aware complex event processing by using EPL statement templates and automatic generation of executable EPL statements.

3 Context-Aware Event Processing for Ubiquitous Process Management

In this section we present the method for context-aware event processing. We first present the framework for context-aware event processing. We then introduce semantic annotation to EPL statements to offer context-awareness. We finally show the main components of the method, their relationships and a high-level view of the dynamics of the processing.

3.1. Framework

The framework for our approach to context-aware event processing is composed of three layers as shown in Fig. 1. The *Event Cloud Layer* is where logical and sensor events are created. The ubiquitous logistics environment has equipped ubiquitous devices such as RFID and WSN that generate physical events like temperature and humidity. Enterprise information systems such as Warehouse Management Systems (WMS), Supply Chain Management (SCM), Business Process Management (BPM) systems generate logical events such as the states of activities and artifacts. The simple events generated in the cloud layer are forwarded to the *Event Processing Layer*, where the Event Middleware integrates events and stores them for its later processing. Simple events are then processed in the Complex Event Processing module that creates high-level events representing contextual information. Each piece of contextual information is delivered to the Context Manager in the *Ontology Layer*. The Context Manager plays a main role in semantic annotation to EPL statements, which will be explained in detail in the next subsection. The Context Manager makes use of the Query Engine and the Reasoning Engine to interact with the ontology. Whenever a situation associated with an EPL statement template is detected, the Context Manager finds the appropriate values to create a context-aware EPL and delivers it to the Complex Event Processing module.

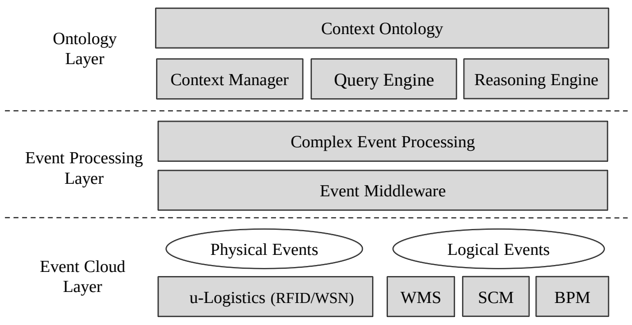


Fig. 1 Framework for context-aware event processing.

3.2. Semantic Annotation to EPL Statements

Semantic annotation to an EPL statement is the process of adding context to an EPL statement template, with the purpose of generating an EPL statement ready to be executed in a CEP engine. A panoramic view of the idea is illustrated in Fig. 2.

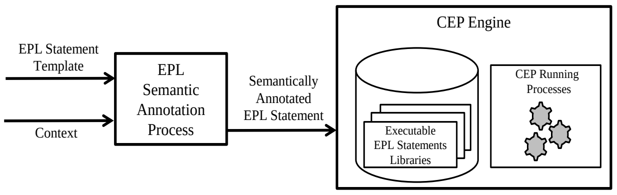

EPL Statement Template
Context
EPL Semantic Annotation Process
Semantically Annotated EPL Statement
CEP Engine
Executable EPL Statements Libraries
CEP Running Processes

Fig. 2 Panoramic view of the process for semantic annotation of EPL statements.

An EPL statement template is an abstraction of EPL statements which contains parameters in its clauses. The parameters can be substituted with specific values derived from context. Context in this case is the information that characterizes a situation that is associated to the template. The output of the process is an executable EPL statement, which contains contextual information on the parameters of the EPL statement template. The executable EPL statement will be registered in a CEP engine to process events.

We conceived semantic annotation to EPL statements to be used in ubiquitous cold chain logistics environments. EPL statements specify how the underlying events in a system should be monitored. However, the situations monitored with a particular EPL statement running on a CEP engine may change dynamically due to the dynamics of the business environment.

Examples of such situation changes in cold chain logistics environments are: a new driver is assigned to pilot a truck; a new product is to be delivered to clients; updated environmental conditions to transport a product are required. These changes in situations imply modifications of how a situation should be monitored. That is, the values on EPL statements need to be adapted in accordance to the situation at hand. This means that: a) appropriate EPL statements should be selected for the situation in hand and b) the selected EPL statement should reflect the appropriate values. Selecting the appropriate EPL statement and setting the suitable values on the statement can be handled manually or automatically. The manual approach is a daunting task, considering the complexity in logistics environments with many variables to handle. To implement the automatic manner, we designed and implemented dynamic event processing based on context-awareness.

3.3. EPL Statements for Process Monitoring

In this subsection we set the semantic annotation to EPL statements in the context of business process. First, we use a meta-model to explain the concepts and how they relate to each other. And second, we explain the dynamics of event processing by relating the concepts in an algorithm.

As illustrated in Fig. 3, the meta-model for context-aware dynamic event processing is composed of three parts: business modeling, context-aware event processing, and event sources. The top part is business monitoring which is conducted in executing business processes under business policy. A business process is regarded as "a way for an organizational entity to organize work and resources (people, equipment, information, and so forth) to accomplish its aims." A business policy is a directive or guideline that shapes the design of business process and monitoring activities. Monitoring activities are verification tasks of a particular aspect of a process performed by some agent, with the goal of complying with the business policy. In this approach, the agent that processes the events generated in the monitoring activity is a CEP engine, which is provided with executable EPL statements. An executable EPL statement is built from two components: an EPL statement template and EPL clause parameter values.

In the middle part, several situations of interest can arise from the monitoring activity. It means that the information used to reason context-awareness is generated in business activities in business process. When a situation arises, two things can happen: a) an EPL statement template is required to be semantically annotated and started or b) an executing EPL statement is required to be stopped. Situations and EPL clause parameter values are closely related to context, which plays two main roles: a) it allows determining the situation at hand by aggregating contextual information and b) it provides the information to be used to fill out the parameters in the EPL statement templates.

Finally, the bottom part contains largely two types of event sources: sensors and enterprise information systems. Events are objects that signify something that occurs in the real world (captured with sensors) or something that happens in an enterprise information system (e.g. enterprise resource planning, customer relationship management, workflow management). Such enterprise information systems play a critical role of providing information of activities in which many kinds of events are generating. The information is important because the same types of sensor events are differently recognized if they are generated during different kinds of tasks in enterprise information systems (e.g. activities in enterprise information systems).

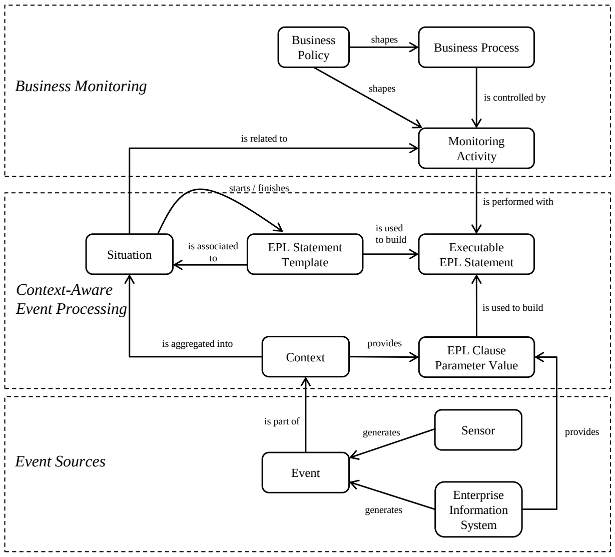


Fig. 3 Meta-model for context-aware dynamic event processing.

The context-aware dynamic event processing algorithm, described in Fig. 4, is aimed to transform EPL statement templates into executable EPL statements in a particular business process. The outputs of the algorithm can be: a) a semantically annotated EPL statement is registered in a CEP engine or b) a semantically annotated EPL statement is finished in the CEP engine.

In the algorithm, the ontology with the knowledge of interest for the food logistics domain is retrieved and loaded into the data structure, logistics_ontology[] (line 1). An infinite loop is started to constantly perform the subsequent instructions (line 2). The context is read and stored in the logistics_context[] (line 3). Reading the context means gathering the information that describes the situation of the entities of interest to be monitored, including the retrieval events from the environment.

The next step is to update the knowledge in the ontology (logistics_ontology[]) with the new information gathered from context (logistics_context[]) (line 4); updating may also generate new knowledge by inference methods activated with the contextual knowledge. The active situations are loaded into the data structure active_situations[]by querying the ontology for active situations (line 5). An active situation means that the situation is occurring at the moment of executing the algorithm.

```
1:  logistics_ontology[] := get_ontology();
2:  while TRUE do
3:      logistics_context[] := read_context();
4:      update_ontology(logistics_ontology[], logistics_context[]);
5:      active_situations[] := ask_for_active_situations(logistics_ontology[]);
6:      for each active_situation in active_situations[] do
7:         epl_stmt_name := get_epl_stmt_name (active_situation);
8:         action := get_action(active_situation);
9:         if action = "start" then
10:          epl_stmt_template := get_epl_stmt_template (active_situation);
11:          template_parameters[] := get_template_parameters(epl_stmt_template);
12:            for each parameter in template_parameters[] do
13:              parameter_value := get_parameter_value(parameter);
14:              annotate_epl_stmt(annotated_epl_stmt, parameter, parameter_value);
15:              register_epl_stmt(epl_stmt_name, annotated_epl_stmt);
16:        else unregister_epl_stmt(epl_stmt_name);
```

Fig. 4 Algorithm for context-aware dynamic event processing.

For each active situation in the active_situations[] list, its associated EPL statement name (line 7) and corresponding action (line 8) are retrieved. If the action to be taken is "start", the EPL statement must be semantically annotated and further registered into the CEP engine (lines 10 to 15). The first step is to load the template (line 10) and the parameters to be annotated in the template (line 11). For each of parameter, the values are queried (line 13) and annotated in the EPL statement (line 14). When the EPL statement is semantically annotated, it is registered in the CEP engine (line 15). If the action to be taken is "finish", the statement was already running in the CEP engine and has to be stopped (line 16).

4. Context-Aware Event Processing in Cold Chain Logistics Environment

To illustrate context-aware event processing, cold chain logistics is adopted as an example scenario. First, the entities in ubiquitous logistics environment are represented in ontology. Then, transformation of EPL statement templates to executable EPL statements is illustrated through the context-awareness based on the ontology. Finally, how the EPL statements can be applied in the cold chain logistics is exemplified with event-driven rule processing.

4.1. Ontology in Cold Chain Logistics

In this example scenario, the situation of interest is the "Product Shipment Activity" because the possible event such as door lock and temperature can occur in that activity. The classes and relationships in the ontology designed for the scenario is shown in Fig. 5. In the ontology, the "Product Shipment Activity" is a specialized class of the "Warehousing Activity" class in that the activity is also included in many kinds of warehousing activities. In executing the logistics process, the activity is scheduled to perform and it is also assigned to vehicle for transportation. Those representations are depicted in the relationships between "Product Shipment Activity" class and other two classes, "Scheduled Product Shipment Task" and "Vehicle". In addition, "Warehousing Activity" class is assumed to be able to occur in "Atomic Location" and transport some "Product" as illustrated in the relationships of "Warehousing Activities".

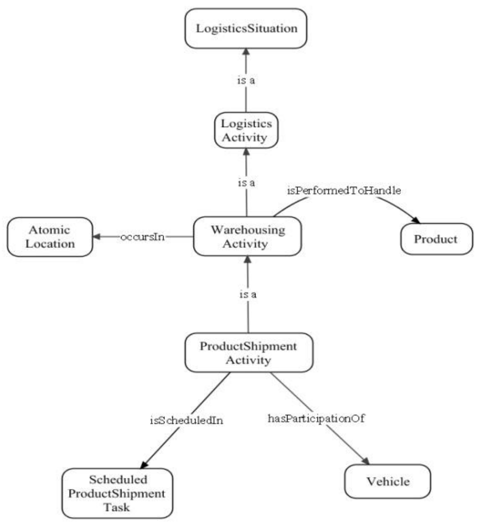
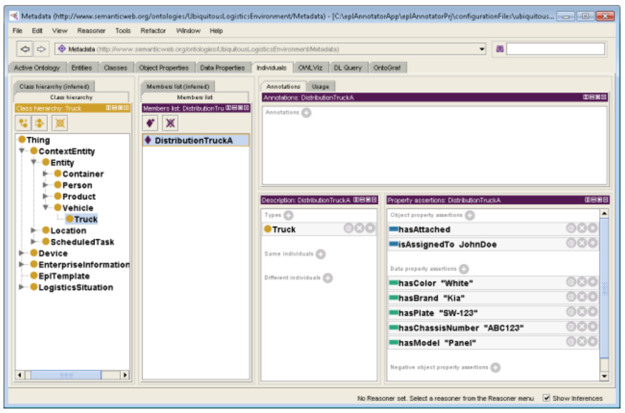


Fig. 5 Entities and relationships for the product shipment activity.


Metadata (http://www.semanticweb.org/ontologies/UbiquitousLogisticsEnvironment/Metadata) - [C:\eplAnnotatorApp\eplAnnotatorPrj\configurationFiles\ubiquitous...

Class hierarchy: Truck
Thing
ContextEntity
Entity
Container
Person
Product
Vehicle
Truck
Location
ScheduledTask
Device
EnterpriseInformation
EplTemplate
LogisticsSituation

Members list: DistributionTruckA
DistributionTruckA

Annotations: DistributionTruckA

Description: DistributionTruckA
Types: Truck

Property assertions: DistributionTruckA
Object property assertions
hasAttached
isAssignedTo JohnDoe
Data property assertions
hasColor "White"
hasBrand "Kia"
hasPlate "SW-123"
hasChassisNumber "ABC123"
hasModel "Panel"

No Reasoner set. Select a reasoner from the Reasoner menu ✓ Show Inferences

Fig. 6 Protégé screenshot for the DistributionTruckA individual.

The initial configuration of the scenario was defined by using Protégé [19]. The necessary individuals for each class involved in the scenario were created; as an example, the individual for the distribution truck and its data and object properties are shown in Fig. 6.

4.2. Dynamic Generation of EPL Statements

To illustrate a situation where our approach can be of benefit, we assume a refrigerated container in the truck. The container has located temperature, humidity and door lock sensors; it also has RFID readers that detect the presence of RFID tags attached to products. Two types of products can be loaded into the container, milk and fish, whose environmental requirements inside the container may vary.

In order to monitor the conditions for any product inside the container, an EPL statement template is related to a product type in a specific activity; the template defined in Esper's EPL [18] is shown in Fig. 7.

```
INSERT INTO SHIPAKERT
SELECT * FROM PATTERN [
  EVERY a=Door(state=open) -> (
    NOT Door(state=close) AND
     (timer:interval(ctx:monitorInterval min) OR
     TEMPMON(tempDiff>ctx:tempThreshold))
  )]
```

Fig. 7 EPL statement template to monitor conditions in container.

The statement indicates to the CEP engine that a SHIPALERT event must be created when the door of a container has been open for ctx:monitorInterval minutes or the temperature difference is more than ctx:tempThreshold degrees while door is open. ctx:monitorInterval and ctx:tempThreshold are filter expressions; their specific values depend on the product that is being loaded into the container (e.g. milk, fish). To assign the appropriate values to ctx:monitorInterval and ctx:tempThreshold, the context of the container indicates what product is being loaded (as detected by the sensors) and what are the appropriate values for its requirements. In this way, when milk is being loaded, the EPL statement expressions would be as shown in Fig. 8. On the other hand, when fish is loaded, the EPL statement expressions would be as shown in Fig. 9.

```
INSERT INTO SHIPAKERT
SELECT * FROM PATTERN [
  EVERY a=Door(state=open) -> (
    NOT Door(state=close) AND (timer:interval(3 min)
     OR TEMPMON(tempDiff>5))
  )]
```

Fig. 8 Executable EPL statement for milk.

```
INSERT INTO SHIPAKERT
SELECT * FROM PATTERN [
  EVERY a=Door(state=open) -> (
    NOT Door(state=close) AND (timer:interval(2 min)
     OR TEMPMON(tempDiff>3))
  )]
```

Fig. 9 Executable EPL statement for fish.

This example shows that the same EPL statement template is configured dynamically according to the context of the container.

The specific events and rules to handle these event patterns and trigger specific business actions can be specified by using event-driven rules in a rule engine. In the example scenario of cold chain logistics environment, the Event-Condition-Action (ECA) rule can be described in Jess rule language [21]. The rule indicates that when a product shipment activity is executing and a SHIPALERT event is generated, a sendSMS action should be executed; the meaning of this action can be interpreted as the sending of a mobile phone message to the operator whenever the conditions detected in the CEP stage are satisfied [22].

```
(defrule SHIP_TEMP_ALERT
(and ?evt <- (SHIPALERT)
     (Activity {name == 'Product Shipment Activity'
          && state == 'executing'}))
=> (assert (InvokeService (name 'sendSMS')
          (to evt.containerID)))
```

Fig. 10 Event-driven Rules for Dynamic Event Processing [22].

4.3. Discussion

In practice, a third-party logistics generally transports many kinds of food types such as meat, fish, and diary food, and they also run many kinds of vehicles and refrigerated storages in logistics process. In such environment, the ubiquitous logistics devices generate a variety of huge event streams including sensor data (e.g. temperature, humidity, CO_2), GPS, and RFID. If we do not apply the approach of using EPL statement templates, we have to describe a great number of event processing rules considering many food types, vehicles, and refrigerated storages. Moreover, if a new food type or vehicle is added in the logistics environment, we have to configure the corresponding rules in the system. Contrarily, in the proposed method, the complicated information of logistics components, such as food types, vehicles, storages, and managers, are delegated to the context-aware systems.

5. Prototype System

We introduce a prototype system to illustrate the proposed context-ware event processing for ubiquitous process management. The system was extended from the event-driven ubiquitous flow management system, called edUFlow [22].

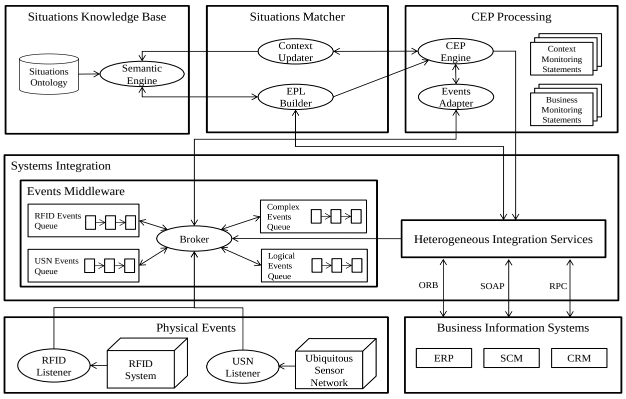


Fig. 11 System architecture for context-aware event processing.

The system architecture is drawn in Fig. 11. First of all, event processing systems in the ubiquitous logistics environment gather physical events from RFID and USN located throughout the logistics environment. And, Business Information Systems such as ERP, SCM, and CRM also provide logical events, which are used to understand business activities. To consolidate and distribute physical and logical events it exists the Events Middleware that is orchestrated by one or more brokers. Brokers receive events from the WSN Listener, the RFID Listener and Business Information Systems and allocate them in special purpose queues: WSN Events Queue, RFID Events Queue and Logical Events Queue, respectively.

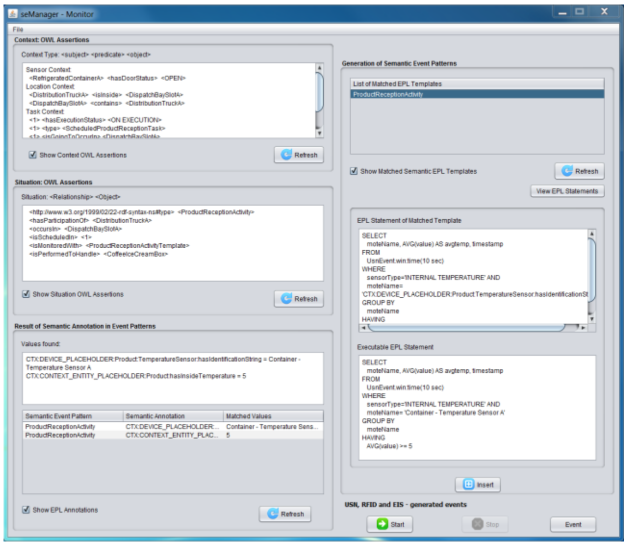
The CEP Processing is performed by a CEP engine; it receives events, physical, logical and complex through the Events Adapter, a process that interacts with the Broker to get and send events from and to queues.

The reasoning function is performed in the Situations Knowledge Base (SKB), built of the Situations Ontology and Semantic Engine. The Situations Ontology is a representation of the domain knowledge, ubiquitous food logistics in this particular case. It models the concepts of interest and relationships that are necessary to detect situations and execute the semantic annotation of EPL statements. The Semantic Engine interacts with the Situations Ontology and allows to infer new knowledge and to query semantic data.

Finally, the acting function is executed by the Situations Matcher. Within the Situations Matcher, the Context Updater gets the relevant events detected with the Context Monitoring Statements in the CEP engine. The Context Updater uses those events to make assertions in the Semantic Engine. The EPL Builder queries the Semantic Engine for the EPL Statement Template, the EPL statement parameters and the specific values for parameters. When constructed, the annotated EPL statement is registered and started in the pool of Business Monitoring Statements. On the other hand if the requirement is to stop a statement, its name is retrieved with the Semantic Engine and further stopped in the CEP engine.

The EPL editor and CEP monitor have been developed for the proposed framework. RFID and WSN systems were respectively implemented using two subsystems: Alien 9800 Development Kit and Ubee430-AP-Kit. These two kits provide software libraries written in Java that allow the transmission of sensor readings data to the Events Middleware.

GlassFish Message Queue was also selected to implement the Events Middleware to integrate the event processing subsystems. There, a broker receives data coming from sensors readings and stores each reading as a message in specially designated queues, namely the WSN Events Queue and the RFID Events Queue. The broker also retrieves messages from RFID and WSN queues when the CEP processing subsystem requests them. Besides the RFID and WSN messages, the broker also handles messages coming from the CEP Processing subsystem storing and retrieving them in and from the Complex Event Queue.



Context: OWL Assertions

Context Type: <subject> <predicate> <object>

```
Sensor Context
  <RefrigeratedContainerA>  <hasDoorStatus>  <OPEN>
Location Context
  <DistributionTruckA>  <isInside>  <DispatchBaySlotA>
  <DispatchBaySlotA>  <contains>  <DistributionTruckA>
Task Context
  <1> <hasExecutionStatus> <ON EXECUTION>
  <1> <type> <ScheduledProductReceptionTask>
  <1> <isGoingToOccurIn> <DispatchBaySlotA>
```

☑ Show Context OWL Assertions

Situation: OWL Assertions

Situation: <Relationship> <Object>

```
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>  <ProductReceptionActivity>
  <hasParticipationOf>  <DistributionTruckA>
  <occursIn>  <DispatchBaySlotA>
  <isScheduledIn>  <1>
  <isMonitoredWith>  <ProductReceptionActivityTemplate>
  <isPerformedToHandle>  <CoffeeIceCreamBox>
```

☑ Show Situation OWL Assertions

Result of Semantic Annotation in Event Patterns

Values found:

```
CTX:DEVICE_PLACEHOLDER:Product:TemperatureSensor:hasIdentificationString = Container -
Temperature Sensor A
CTX:CONTEXT_ENTITY_PLACEHOLDER:Product:hasInsideTemperature = 5
```

Semantic Event Pattern	Semantic Annotation	Matched Values
ProductReceptionActivity	CTX:DEVICE_PLACEHOLDER:...	Container - Temperature Sens...
ProductReceptionActivity	CTX:CONTEXT_ENTITY_PLAC...	5

☑ Show EPL Annotations

Generation of Semantic Event Patterns

List of Matched EPL Templates
ProductReceptionActivity

☑ Show Matched Semantic EPL Templates

EPL Statement of Matched Template

```
SELECT
  moteName, AVG(value) AS avgtemp, timestamp
FROM
  UsnEvent.win:time(10 sec)
WHERE
  sensorType='INTERNAL TEMPERATURE' AND
  moteName=
'CTX:DEVICE_PLACEHOLDER:Product:TemperatureSensor:hasIdentificationSt
GROUP BY
  moteName
HAVING
```

Executable EPL Statement

```
SELECT
  moteName, AVG(value) AS avgtemp, timestamp
FROM
  UsnEvent.win:time(10 sec)
WHERE
  sensorType='INTERNAL TEMPERATURE' AND
  moteName= 'Container - Temperature Sensor A'
GROUP BY
  moteName
HAVING
  AVG(value) >= 5
```

USN, RFID and EIS - generated events

Fig. 12 User interface of executable EPL statement generation

In shown in Fig. 12, the module for the activation, deactivation and visualization of the flows of events is presented. The capture of complex events of the two basic types, namely WSN and RFID, can be started or stopped from the middleware's queues and later processed with the CEP engine. If the event processing can be matched to the EPL statement templates based on the context, the executable EPL statement is generated.

The executable EPL statements which have been generated based on the EPL statement templates are registered to the CEP engine for the CEP processing as shown in Fig. 10. Finally, the CEP engine will monitor the raw event pattern and detect the registered event patterns in newly generated EPL statements.

6. Example Scenario

To illustrate our approach, we explain how the generation of EPL statements would work in the delivery process of a perishable product, ice cream. The scenario takes place in the delivery zone in a retailer’s warehouse, where jars of ice cream are dispatched from the container attached to a truck. The case for monitoring this process is that safety and quality of perishable goods have to be guaranteed. The ubiquitous logistics environment is configured as follows. The delivery zone in the retailer’s warehouse is delimited by the interrogation zones of RFID antennas and readers, so that when an RFID tag attached to an object is read, it means that the object is within the delivery zone. RFID tags are attached to: a) the roof of delivery trucks and b) ice cream jars. RFID antennas are located in strategic positions at the delivery zone; there is one antenna at the entrance of the parking zone (intended to read RFID tags attached to trucks) and another one at the entrance of the warehouse building (intended to read RFID tags attached to jars). The door of the truck’s containers, where ice cream jars are delivered, is locked with an electronic door lock. It is important to note that prior to attaching RFID tags to objects, complementary information is associated to them and stored in Enterprise Information Systems. In the case of trucks, plate, driver and supplier information is associated to the RFID tag number; and in the case of ice cream jars the EPC standard may be used, where information as the serial number, category and manufacturer is associated to the RFID tag id. In this model, the ubiquitous logistics environment senses the presence or absence of objects (trucks and jars) in the delivery zone and also the state of the door lock, open or closed. These environmental readings can be modeled as three types of events, shown in Fig. 13.

```
EpcTagReading {
  epc_code:String, antenna_id:Byte,
  discovery_time:Datetime, last_seen_time:Datetime,
  reader_id:String
}

VehicleTagReading {

  vehicle_tag_id:String, antenna_id:Byte,
  discovery_time:Datetime, last_seen_time:Datetime,
  reader_id:String
}

ContainerDoorEvent {
   vehicle_tag_id:String, event_type:String,
   event_timestamp:Datetime
}
```

Fig. 13 Data objects for event sources in cold chain logistics

The first event signifies that an ice cream jar has been delivered; the second signifies that a vehicle has entered the delivery zone and the third signifies an event associated to the door lock, for example the opening or closing of the door.

As previously explained in Section 3.3, we assume that business policies shape how business processes and its associated monitoring activities are designed and implemented. In this particular example, the business policy that drives the monitoring activity is related to expired products: all perishable foods whose expiration date is previous to current (system) date on delivery at a warehouse have to be rejected. So, the monitoring activity to implement that policy in the perishable products delivery process consists in detecting the products that have already expired or are going to expire soon.

The ontology that we use to represent the above mentioned knowledge is shown in Fig. 14. The “perishable products delivery process” of ice cream is controlled by the “check expiration monitor”. The tasks to be performed by the monitor are defined in an EPL statement template, which pertains to the Business Monitoring Statements group in our framework. The EPL statement that is appropriate for this case is shown in Fig. 15.

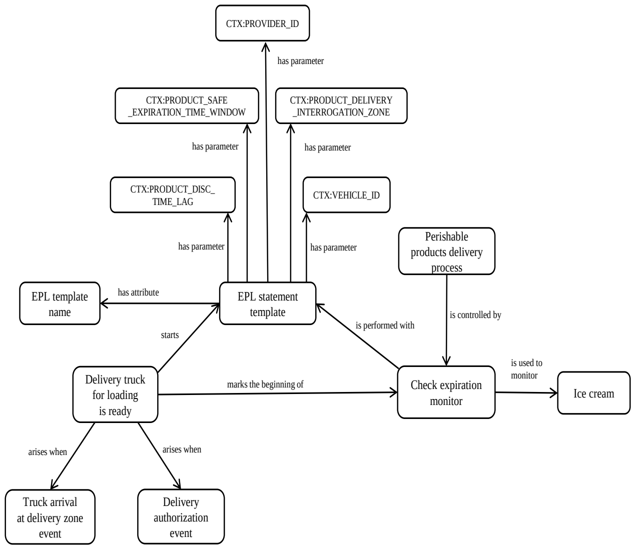


Fig. 14 Ontology for context-aware chain logistics

```
(1)  INSERT INTO rejected_expired_products (epc_code,
       discovery_time, provider_id)
(2)  SELECT p.epc_code, p.discovery_time, CTX:PROVIDER_ID
(3)  FROM PATTERN [
(4)    d = ContainerDoorEvent (event_type = "OPEN"
          AND  vehicle_tag_id = CTX:VEHICLE_ID) ->
(5)       EVERY p = EpcTagReading(p.antenna_id =
          CTX:PRODUCT_DELIVERY_INTERROGATION_ZONE)]
(6)  WHERE ( get_expiration_date (p.epc_code) <
       get_system_date()  OR
(7)    get_expiration_date(p.epc_code)< get_system_date()
       + CTX:PRODUCT_SAFE_EXPIRATION_TIME_WINDOW )
(8)    AND (p.discovery_time - d.event_timestamp) <
       CTX:PRODUCT_DISC_TIME_LAG
```

Fig. 15. EPL statement template to monitor expiration dates of products in a product delivery process.

This template is the basis for generating a stream of products (rejected_expired_products) that were rejected because they are already expired or are going to expire soon (line 1). That stream is fed by a SELECT statement (lines 2 to 10) that detects a pattern of events in which the door of a container is opened (line 4) followed by one or more readings of RFID tags attached to ice cream jars (line 5), in the delivery zone. To detect which of those products that cross the delivery zone have already expired or will expire soon, two comparisons are made: the first one verifies if the expiration date of the product has already occurred (line 6) and the second one checks if the expiration date will occur before a safe period (line 7). Additionally, it is checked that the time occurred between the discovery time of the product and the door opening does not exceed a certain amount of time (line 8). The EPL statement has associated five semantic parameters: CTX:PROVIDER_ID, CTX:VEHICLE_ID, CTX:PRODUCT_DELIVERY_INTERROGATION_ZONE, CTX:PRODUCT_ SAFE_EXPIRATION_TIME_ WINDOW and CTX:PRODUCT_DISC_TIME_LAG. These values are to be annotated with particular values determined by contextual information that is gathered from several sources: the ontology itself, enterprise information systems and events generated by sensors.

```
(1)   SELECT v.vehicle_tag_id, v.provider_id,
        a.timestamp
(2)   FROM PATTERN [
(3)     EVERY v = VehicleTagReading (antenna_id=123)
          -> a = DeliveryAutorizationEvent
(4)     WHERE timer:within(2 min)]
(5)   WHERE v.vehicle_tag_id = a.vehicle_tag_id
```

Fig. 16. EPL statement to monitor contextual events.

```
(1)  INSERT INTO rejected_expired_products (epc_code,
       discovery_time, provider_id)
(2)  SELECT p.epc_code, p.discovery_time, "12A8-Z789"
(3)  FROM PATTERN [
(4)    d = ContainerDoorEvent (event_type = "OPEN"
           AND vehicle_tag_id = "TR-589") ->
(5)        EVERY p = EpcTagReading(p.antenna_id=123)]
(6)  WHERE ( get_expiration_date (p.epc_code)  <
       get_system_date()  OR
(7)    get_expiration_date(p.epc_code)<
       get_system_date() + 10 days )
(8)  AND (p.discovery_time - d.event_timestamp) < 5
     min
```

Fig. 17. Executable EPL statement to monitor the expiration dates of products in a product delivery process.

In our framework, the EPL statement template has to be annotated and further started in a CEP Engine, when the situation "Delivery truck for loading is ready" of the ontology in Fig. 14 occurs. This situation becomes active when two events occur: first, a truck arrives at the delivery zone and second, an authorization is received from an ERP system. These events are monitored with an EPL statement intended to monitor context (the Context Monitoring Statements group), as shown in Fig. 16.

This statement detects every VehicleTagReading event that occurs within the delivery zone (read by antenna with id equal to 123), followed by DeliveryAutorizationEvent, with a difference of 2 minutes between them (lines 2 to 4). Additionally, the vehicle detected must correspond to the vehicle authorized (line 6). These two events can be then asserted in the Semantic Engine, giving as a result the readiness of the delivery truck for loading. This new state drives the starting of the annotating of the EPL statement, which retrieves the EPL statement template, the template's name, the associated action (in this case "start") and the contextual values. After annotating the template, an executable EPL statement is generate as presented in Fig. 17.

And finally, the last step in the process is to register and start the generated EPL statement, shown in Fig. 17, to perform the monitoring tasks in the process.

7. Conclusion

In this paper, the main approach to context-aware event processing is the process of adding contextual values to EPL statement templates for the purpose of adapting them dynamically to the situation that occurs in a particular moment based on the predefined ontology. To realize the approach, we discussed the concepts, algorithms, models and a framework to perform context-aware complex event processing in ubiquitous logistics environment. We also provided an example scenario to show how we can apply the proposed method to cold chain logistics.

The proposed method can support more precise and automated monitoring tasks of business process by dynamically adapting the values of EPL statements based on the physical condition captured by RFID and sensors and the ontology designed for various logistics environment, as illustrated the presented example in Section 6.

The future work of this research include as follows. First, we can extend our approach to context-ware complex event processing by applying it not only to the values of EPL parameters, but also to the structure of EPL clauses. Second, we can strengthen the robustness of the proposed method by supporting the analysis function of conflict detection and resolution of event-driven rules [23, 24]. The issue is more significant as the number of events and rules increases in practical business environment.

Acknowledgement

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (No. 2012003505).

References

[1] M. Kang and S. Kim, "A Study on the Military Logistics System in Ubiquitous Environment", *Int. J. Mult. Ubiq. Eng.*, vol.1, no,1, pp.1-4 , 2006.

[2] D.M. Lambert, J.R. Stock, and L.M. Ellram, *Fundamentals of Logistics Management*, McGraw-Hill Book Co., Singapore, 1998.

[3] M. Weiser, "The Computer for the 21st Century", *Scientific American*, vol.265, no.3, pp.94-104, 1991.

[4] G. Yoo and E. Lee, "Self-Healing Methodology in Ubiquitous Sensor Network", *Int. J. Adv. Sci. Tech.*, vol.3, pp.9-18, 2009.

[5] E. Bajic, "A Service-Based Methodology for RFID-Smart Objects Interactions in Supply Chain", *Int. J. Mult. Ubiq. Eng.*, vol.4, no.3, pp.37-56, 2009.

[6] C. Heinrich, *RFID and Beyond*, Wiley Publishing, Inc., Indianapolis, 2005.

[7] M. Ju and S. Kim, "Logistic Services Using RFID and Mobile Sensor Network", *Int. J. Fut. Gen. Comm. Netw.*, vol.1, no.2, pp.25-29, 2008.

[8] V. Mattoli, B. Mazzolai, A. Mondini, S. Zampolli, and P. Dario, "Flexible tag datalogger for food logistics", *Sensors and Actuators A: Physical*, vol.162, no.2, pp.316-323, 2010.

[9] I. Hwang and J. Baek, "Wireless Access Monitoring and Control System based on Digital Door Lock", *IEEE Trans. on Cons. Elec.*, vol.53, no.4, pp.1724-1730, 2007.

[10] J. Yoo and Y. Park, "An Intelligent Middleware Platform and Framework for RFID Reverse Logistics", *Int. J. Fut. Gen. Comm. Netw.*, vol.1, no.1, pp.75-82, 2008.

[11] D. Luckham, *The Power of Events*, Pearson Education Inc., Boston, 2002.

[12] S. Loke, *Context-Aware Pervasive Systems*, Auerbach Publications, Boca Raton, 2007.

[13] T. Gua, H.K. Pung, D.Q. Zhang, "A service-oriented middleware for building context-aware services", *J. Netw. Comp. App.*, vol.28, pp.1-18, 2005.

[14] J. Sheng, W. Zou, L. Yang, B. Wang, "A RFID-based Context-Aware Service Model", Proc. *Int'l Conf. IEEE TrustCom-11/IEEE ICESS-11/FCST-11*, 2011.

[15] C. Gao, J. Wei, C. Xu, S.C. Cheung, "Sequential Event Pattern Based Design of Context-Aware Adaptive Application", *Int J Softw. Infor.*, vol.4, no.4, pp. 419-436, 2010.

[16] J. Dunkel, "On Complex Event Processing for Sensor Networks", Proc. *Int'l Symp. on Auto. Decentr. Syst.*, Athens, Greece, 2009.

[17] M. Suntinger, H. Obweger, J. Schiefer, M.E. Gröller, "Event Tunnel: Exploring Event-Driven Business Processes", *IEEE Comp. Grap. App.*, vol.28, no.5, pp.46-55, 2008.

[18] N. Dindar, C. Balkesen, K. Kromwijk, N. Tatbul, "Event Processing Support for Cross-Reality Environments", *IEEE Perv. Comp.*, vol.8, no.3, pp.34-41, 2009.

[19] The Protégé Ontology Editor and Knowledge Acquisition System. http://protege.stanford.edu/

[20] Esper Homepage. http://www.espertech.com/

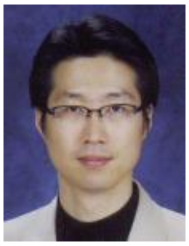
[21] Jess Rule Engine Homepage. http://www.jessrules.com/

[22] P. Rosales and J.-Y. Jung, "Complex Sensor Event Processing for Business Process Integration", IEICE Trans. Comm., vol.93-B, no.11, pp.2976-2979, 2010.

[23] K. Teymourian, "Enabling Knowledge-Based Complex Event Processing", Proc. VLDB, 2011.

[24] D Riemer, L. Stojanovic, N. Stojanovic, "Using Complex Event Processing for Semantic Requests in Real-Time Social Media Monitoring", AAAI Technical Report WS-12-12, 2012.

Pablo Rosales Tejada received BS from the Department of System Engineering at Universidad de San Carlos de Guatemala, Guatemala, in 2006. He also received MBA from Universidad Mesoamericana, Guatemala in 2008, and MS from Kyung Hee University, Korea in 2011. He has been a Business Intelligence consultant for more than 5 years. He is currently an assistant professor in System Engineering at Universidad de San Carlos de Guatemala, Guatemala. His research interests include business intelligence, business analytics, and systems engineering.



Jae-Yoon Jung received BS, MS, and PhD from the Department of Industrial Engineering at Seoul National University, Seoul, Rep. of Korea, in 1999, 2001, and 2005, respectively. He is an assistant professor in the Department of Industrial and Management Systems Engineering at Kyung Hee University, Seoul, Rep. of Korea. His research interests include ubiquitous service computing, Internet business, and business process management.