# Context Aware Computing and its utilization in event-based systems

Opher Etzion opher@il.ibm.com
Yonit Magid yonit@il.ibm.com
Ella Rabinovich ellak@il.ibm.com
Inna Skarbovsky inna@il.ibm.com
Nir Zolotorevsky nirz@il.ibm.com
IBM Haifa Research Lab

## ABSTRACT

*Recently, the notion of context aware computing has attracted attention in various domains, and was mentioned as emerging direction in Gartner's 2009 hype cycle for enterprise architecture. The paper provides a brief report of a tutorial given in DEBS 2010 that discusses the notion of context, generally in computing, in event processing modeling, giving examples from the current state-of-the-practice that supports some types of context, but still do not view it as a first class citizen in the model.*

## Keywords

Context, context-aware computing, context-driven architecture, temporal processing, spatial processing, event processing modeling, event processing products.

## 1. INTRODUCTION: THE ROLE OF CONTEXT IN COMPUTING

People usually act within a context, we dress differently at home than in work; we are carrying money in hidden pockets in certain countries, and in a plain wallet in other countries , we might chose different transportation means in snowing conditions, in rainy conditions, and in regular conditions. Context may relate to various external conditions, like the time of the day, location, gender, participants history, experience etc.

While context is a major factor in human's behavior, most software systems do not support such a notion directly and typically hard-code this functionality.

Gartner [24] defines "context-aware computing" as the concept of leveraging information about the end user to improve the quality of the interaction. In the world of software, the utilization of context in processing is a major step towards modeling the real world functional behavior. Most of the software only utilizes direct relevant information for processing (direct input). In contrast, context-aware software acts also upon indirect relevant information, which represents the conditions under which the processing is done.



*DEBS'10*, July 12–15, 2010, Cambridge, UK.


Location context coupled with user's identity can be very valuable for various service applications. Consider, for instance, a call routing application, routing phone calls based on the user's location and pre-defined preferences: at home – wire line phone, at office – IP phone or instant messaging application, at the movie theater – SMS only (speech-to-text) or call forward to an answering service. Another example is an automatic tow service ordering system: if your car breaks down on the side of the road, instant information about nearby tow, pushed automatically to your mobile device, will be more helpful than making phone calls or surfing the net.

Stimulated by the opportunities provided by GPS systems, context-aware applications pioneers mainly focused on mobile computing, wearable computing devices and person related context. A number of location-aware guides have been designed for city tours [1] [7] and for museums [6] [10] [15]. Early exploration of the usage of context in domain of event processing was reported in [2]. Event processing middleware as context extractor for medical devices was presented by [12].

Additional context related research efforts exist in domain of business process modeling. A project investigating how context awareness can become an integral part of business process modeling is reported in [13] [14]. Along the way, several approaches were also investigated for formalization of notion of context [4] [5] [11].

Although context-aware computing is usually associated with ubiquitous computing, it also becomes a key concept in other areas, such as social analysis, healthcare and information retrieval. The same medical query should be answered differently, depending on the requester medical history (usually healthy or sick), the source of the query within the system (family doctor or a doctor in an emergency room) or the conditions external to the system (swine flue epidemic announced). A search engine should offer the users different search results based on their searching history, as well as the "news" web sites.

This paper summarizes a tutorial that surveys the concept of context in general and concentrates on context in event processing. Section 2 discusses context delivery applications and some practical implementation in computing, section 3 discusses the notion of context in event processing modeling, section 4 discusses various context dimensions, section 5 discusses several examples of context implementations in event processing products and section 6 summarizes the paper.

# 2. CONTEXT ARCHITECTURES AND IMPLEMENTATIONS

In the field of engineering, several approaches are investigated for building context-aware services and applications; most of them introduce the concept of *context service* – a component that maintains contextual information, and provides services to other system components upon request, thus leveraging them with additional information, not provided directly as part of their invocation. According to the Service Oriented Architecture (SOA) paradigm, services are required to apply their business logic consistently across all invocations, using only the direct input as the argument. However, most services do retrieve additional information from databases and other sources that directly influence the output of the service. Some of this additional information is, in fact, contextual information.

Figure 1 presents the context-aware computing model, as described by Gartner.

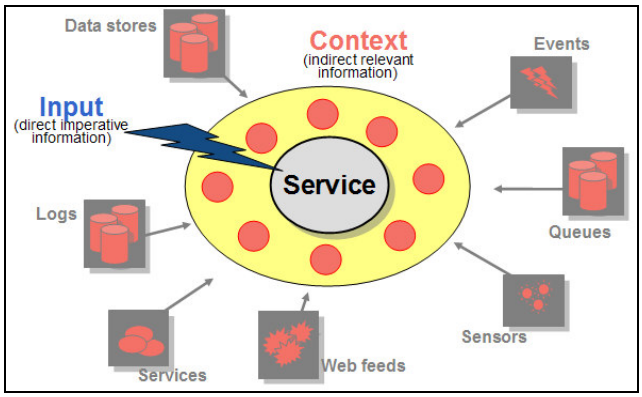


**Figure 1. Context-aware Computing Model**

Context is indirectly relevant information that is useful to functioning of the service, but not provided to the service as part of its invocation. In Figure 1, context is depicted by ellipse with circles enclosing the service, emphasizing the fact that this information belongs to the service environment, rather than to the service itself. Context information can come from various sources such as: sensors, event brokers, social interaction logs, and web feeds. Systems capable of dynamically adding new types and sources of context possess greater intelligence and are the most valuable to users.

An architecture that is aware of the end user's context and delivers the information that is most suited for it, was first presented by Gartner as Context Delivery Architecture (CoDA). This architecture is further introduced in the next subsection.

## 2.1 Context Delivery Architecture

Context Delivery Architecture (CoDA) [24] is aimed for service-oriented or event-driven applications, where the functioning of the software elements (services or event handlers) is determined not only by the input to the element, but also by the secondary sources of information – the context. Two invocations of the same service with the same parameters may yield different results in different circumstances.

CoDA is presented by Gartner as the next step in the evolution of Service Oriented Architecture (SOA). Like SOA, it introduces the concept of creating composite applications through reusable services, and in addition aims to enhance user's experience using the knowledge of context and adapt the application behavior to it, delivering the right information at the right time. Another helpful background for CoDA is Event Driven Architecture (EDA). Events can be viewed as a source of context, since they are snippets of the past activities; event processing can be viewed, in certain environment, as context detecting technology. Therefore, event processing results can be carried to other applications, injecting context related information into services and processes.

Next we provide some insights on the current state of the practice of context-aware applications, in different domains.

## 2.2 The current state of the practice

We discuss some applications in telecommunication and social platforms which constitute the main application areas in which context aware computing currently exists.

### *2.2.1 Telecommunication*

Networks, hardware and software of mobile devices constantly advance, making it possible to assess the location and activity of the user – i.e. the user's personal context, for example whether he is at home, at the office, driving his car, sitting at a restaurant, and more. This ability is the basis on which context-aware communications systems have emerged.

One example for a context aware mobile communications system is GeoVectors applications on smart phones in the field of mobile commerce [25]. Mobile commerce is the ability to purchase products, contents or services via a mobile device. GeoVector applications allow these smart phones users to find out when the museum opens, find an ATM on the way home, ask for a coupon, make a purchase, see a movie trailer, see an historic site as it was 200 years ago, and more, all by simply pointing the mobile device at the desired direction or object. When the user points the phone at a target object or objects, the GeoVector applications combines the location information, direction and a geographical object database in order to identify the target object. The user's location information, his personal identity and the object or direction he is pointing at comprise the context for the purchase (of either products or services) he is conducting.

Another example is NTT DoCoMo's mobile payment system which illustrates the emergence of handsets using multiple radio protocols that intelligently select the context on which a consumer checks out of retail stores [26].

### *2.2.2 Web 2.0 Social Platforms*

While in Web 1.0 content was the primary element, in Web 2.0 context is the primary element [16].

Web 2.0 applications, such as Google, Amazon, and Facebook, have been very innovative in being context aware when interacting with the users. Their context analysis algorithms have strategic importance to their business models and are therefore held as top secret.

In Amazon, the history of a user's activity, together with his location (extracted from the IP address), current activities by other relevant users, political and social news environment form the user's context, are used to offer the user the most-relevant experience in e-commerce.

In Google, sophisticated context analysis is performed in order to improve search relevance, and for placement of advertisements according to context.

Facebook collects and analyzes event-driven interactions and relationships between its members in order to add unique social-context-based relevance to its services. The content a user creates, seeks and finds depends on his social context, i.e. his identity, memberships, and the power of his connections to other members.

We move from the general computing to the utilization of contexts in event processing.

## 3. Context in event processing modeling

Context [8] plays the same role in event processing that it plays in real life; a particular event can be processed differently depending on the context in which it occurs, and it may be ignored entirely in some contexts.

There are three main uses of context by event processing applications:

- An event stream is defined [3] as open-ended set of events. If you want to perform an operation on the stream you cannot wait until all these events have been received. Instead you have to divide the stream up into a sequence of context partitions, or *windows*, each of which contains a set of consecutive events. You can then define the operation in terms of its effect on the events in a window. The rule that determines which event instances are admitted into which window is something we call a *temporal context*.
- A collection of events, arriving in one or multiple streams, may contain events that are not particularly connected to one another, even though they might occur close together in a temporal sense. They might, for example, refer to occurrences in different locations, or to occurrences involving different entities in the real world. Suppose you were to do some processing of an event stream, such as a simple *aggregate* agent that counts the number of events. By default this would count all the events in the stream, but what if you want to see separate totals for each location where the events occurred? To do this you need to have a separate agent, or at least a separate *instance* of the agent, processing the events for each location. *Spatial* contexts and *Segmentation-oriented* contexts let you assign related events to separate context partitions. You can then have each partition processed by a distinct instance of the event processing agent, so that events in one context partition are processed in isolation from the events in other partitions.
- Context also allows event processing agents to be context sensitive, so that an agent that is active in some contexts may be inactive in others. We refer to this as *state-oriented* context.

In essence, in event processing modeling, EPA (Event Processing Agent) serves as the basic unit of event processing computation. EPA may be of three types: filter EPA, transform EPA, and pattern matching EPA. Context may apply to all of these types; Figure 2 taken from [8] shows the role of context in the EPA processing

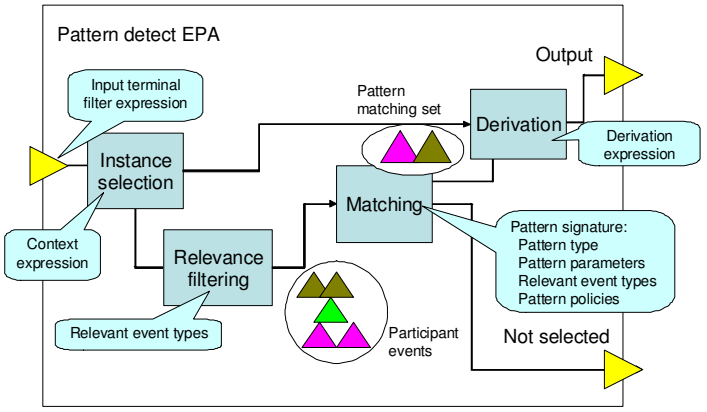


**Figure 2. The Event Processing Agent functionality model**

As seen in Figure 2, there is a context (simple of composite) associated with each EPA. This context determines:

- Whether an input event is relevant at all for a specific EPA
- If the EPA has several instances, to which of these instances, an event is relevant for.

In order to understand the role, let's look at a simple example of segmentation oriented context; events that relate to transactions may be partitioned by customer, since we would like to process the events related to each customer separately.

Within event processing languages, contexts may be explicit, implicit, or partially explicit. Explicit context means that context primitives are first class primitives in the language. Some languages do not support any notion of context, and some support partial notion of contexts. Survey of contexts in various languages will be discussed in section 5, dealing with contexts in practice.

## 4. Context dimensions

Figure 3 taken from [8] shows the various context dimensions: temporal, spatial, state oriented, and segmentation oriented that we describe in this section.

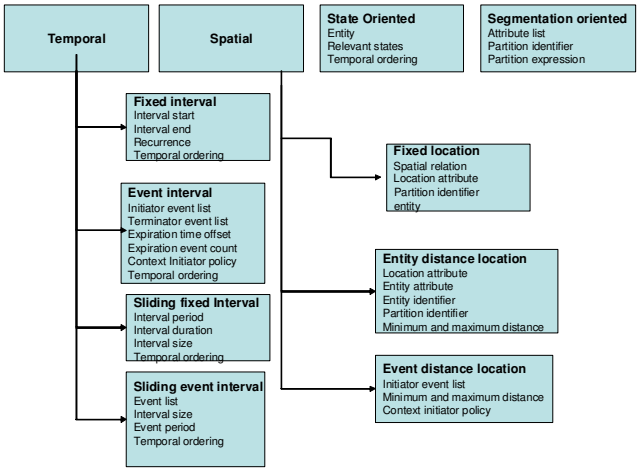


**Figure 3. Context dimensions**

## 4.1 Temporal context

Temporal context is aimed to partition the temporal space into time intervals. It is used for two purposes:

- **Events grouping**: the typical use of temporal contexts is to group events to process them together based on the fact they have occurred during the same interval
- **EPA applicability**: Another use of temporal contexts is to indicate that a certain event processing agent is applicable within certain intervals and is not applicable in other intervals. The temporal interval convention is having half-open interval [Ts, Te), such that Ts is the interval starting point, included in the interval, and Te is the interval end-point, not included in the interval.

An example of event grouping is: *count all the bids within the auction period.* In this case the *auction period* stands for a temporal interval, and all the events of type *bid* that occur within this interval are grouped together for calculating the aggregation function (count).

An example of EPA applicability is: *detect if all printers at the same floor are off-line during working hours.* In this case there is a pattern that is used to check whether all printers in the same floor are off-line at the same time. However monitoring for this pattern is relevant only within working hours, since it is assumed that after working hours there is no technician on-site.

One of the design decisions in associating events to temporal context is to determine whether a specific event falls within the interval. In some cases this is being determined by the *detection time*, which is the timestamp created by the system [9] when the event enters the system, or by *occurrence time* which is the time in which the event source specifies as the time in which the event occurred in reality. This decision is determined by using the *temporal ordering* parameter in the various temporal context definitions.

Note that temporal context can be long running, single context partition can span over days, months or even years. The implementation of short range context and long range context is quite different, but conceptually there are no differences.

There are four types of temporal contexts: fixed interval, event interval, sliding fixed interval and sliding event interval. We'll survey briefly each of these types.

### *4.1.1 Fixed temporal interval*

A fixed temporal interval consists of one or more temporal intervals whose boundaries are pre-defined timestamp constants. This can be a one time interval: [July 12 2010 13:30, July 12 2010 17:00) which designates a DEBS 2010 tutorial session, it also can be stated as [July 12 2010 10:30, + 3.5 hours), where Te is an offset relative to Ts. Fixed temporal interval can also be recurrent, and in this case the frequency should be specified. Some example of that is [7:00, 9:00) daily, or [Monday 13:30, 14:30) weekly which determines event processing operations that are applicable periodically, for example: some traffic monitoring is done during morning rush hour only, or that during a weekly meeting some monitoring is disabled.

### *4.1.2 Event interval*

Event interval is a temporal interval that is being opened or closed when one or more events occur; the meaning of "event occurs" is determined by the *temporal ordering* parameter and can be interpreted either as the detection time or the occurrence time as explained earlier. The collection of events that open such an interval are called *initiator events,* and the collection of events that close the temporal interval is called *terminator events.* An interval may also expire after a certain time offset is reached.

Some examples:

A temporal interval starts with a patient's admission to a hospital, and ends with the release of the same patient from the hospital. Note that here the temporal context is combined with segmentation context, since the temporal context refers to a single patient.

A temporal interval that starts with a shipment of a package and ends with the delivery of the same package; the temporal interval expires after 3 days, which is the delivery designated time, providing time-out handling.

### *4.1.3 Sliding fixed interval*

In a sliding fixed interval context, new windows are opened at regular intervals. Unlike the non-sliding fixed interval context these windows are not tied to particular times, instead each window is opened at a specified time after its predecessor. Each window has a fixed size, specified either as a time interval or a count of event instances.

A sliding fixed interval is specified by interval period which designates the frequency in which new windows are opened, interval duration which specifies how long this interval spans, and interval size, which determines how many events are included in a

single window. The specification must include an *interval period* parameter and either an *interval duration* or *interval size* (or both, in which case it may end earlier than the duration if the event count is exceeded). Sliding intervals may be overlapping or non-overlapping, they are overlapping if and only if the *interval period* < *interval duration*.

Some examples:

A temporal interval starts every hour, with the duration of one hour. In this case we partition the time to time windows of one hour each.

A temporal interval starts every day, and ends when 50 orders have been placed in the system, or at the end of the day. In this case the interval partitions the temporal space in time windows of one day; however, the daily interval can terminate earlier if there are 50 orders.

A temporal interval starts every 10 minutes, and lasts for hour. In this case, each event (starting from the second hour of operation) is associated with 6 different windows that are active in parallel. This type of context can be used for trend seeking windows in time series' events.

### *4.1.4 Sliding event interval*

The *sliding event interval* context is similar to the sliding fixed interval context. The difference is that the criterion for opening a new window is specified as a count of events, rather than as a time period. A sliding event interval is specified by list of events (possibly with predicate for each event), interval size (in event count), and event period. The event list designates event types whose instances can start the temporal window, the interval size determines the event count that closes the interval, while the event period (defaults to the interval size) determines the event count to open an additional window.

An example:

A sensor measures the temperature; an interval consists of five consecutive measurements and starts with every single measurement. Each measurement is associated with five different intervals.

It should be noted that the use of temporal context is orthogonal to applying real time constraints on the system's latency. Temporal context, like any context, determine the outcome of the event processing functions, while real-time constraints determine the scheduling and optimization characteristics of the system.

## 4.2 Spatial context

A spatial context groups event instances according to their geospatial characteristics. This type of context assumes that that an event has a spatial attribute designates its location. Location can be represented in three ways [8]: point in space, line or polyline, and area (polygon). As shown in Figure 4, there are three types of spatial contexts: fixed location, entity distance location and event distance location.

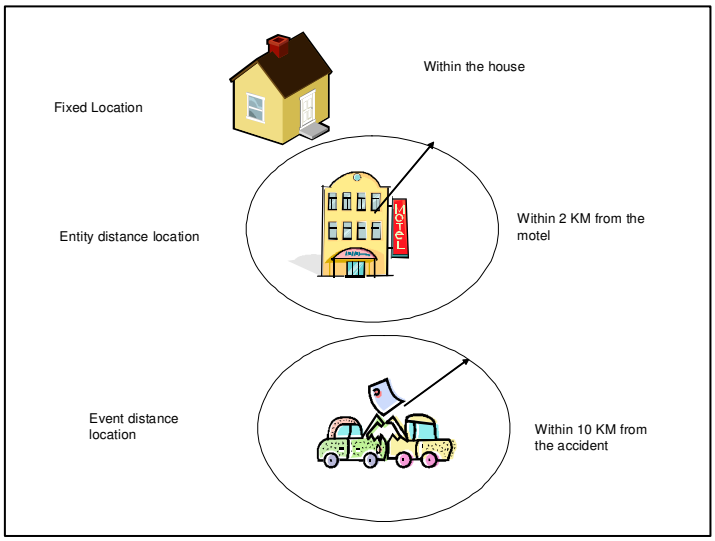


**Figure 4. Spatial contexts**

### *4.2.1 Fixed Location*

A *fixed location* context has one or more context partitions, each of which are associated with the location of a reference entity, sometimes referred to as a *geofence*. An event instance is classified into a context partition if its location attribute correlates with the spatial entity in some way. Like the event location, the reference location can be of any of the three types: point, line, area, thus there can be several relations between the event's location and the reference location as shown in Figure 5.

The *contained in* relation is true if the entity location is completely enclosed within the event location.

The *contains* relation is true if the event location is completely enclosed within the entity location.

The *overlaps* relation is true if there is some overlap between the entity location and event location, but neither of them is contained within the other.

The *disjoint* relation is true if there is no overlap between the entity location and event location.

The *equals* relation is true if the entity location and event location are identical.

The *touches* relation is true if the event location borders the entity location.

Not every combination of entity location type and event location type is valid, as seen in the figure, for example: a point cannot be contained in another point, an area cannot be equal to a line, and a point cannot overlap an area.

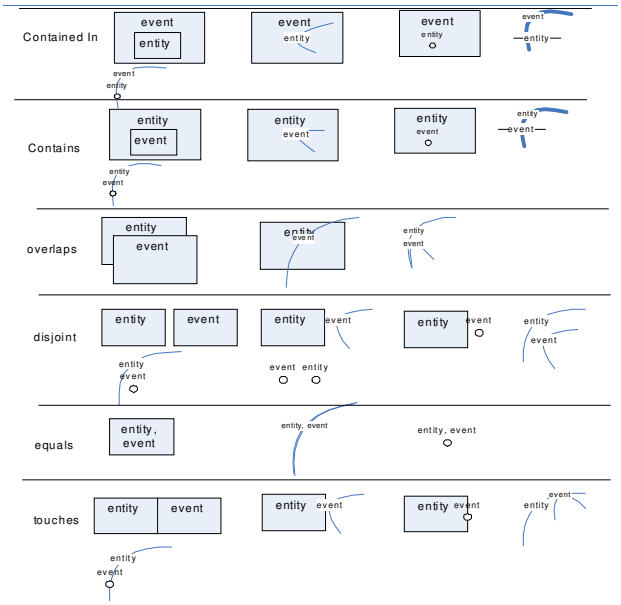


**Figure 5. fixed location relations**

Here are some examples:

A car fleet management application follows cars that are driving on a certain highway. The car location, obtained by the GPS, is represented as a point, the highway is represented by a polyline, and the relationship is "contained in".

A plague is detected within a geographical area that overlaps the borders of a certain city, both the event's location and the entity location are areas, and the relationship is "overlaps."

### *4.2.2 Entity distance location*

An *entity distance location* context gives rise to one or more context partitions, based on the distance between the event's location attribute and some other entity. This entity may be either stationary or moving. If the entity is moving, the distance relates to the location of the entity at the time that the event occurred (its *occurrence time*). The entity may either be one that is specified by another attribute of the event, or one that is specified in the context definition.

Some examples:

The example shown in Figure 4, where the entity is a motel and the context is used to track fire alarms within 2 km of the motel, in order to alert the motel manager of possible danger. This context has a single partition, and both the event location and the entity location are given as points.

Vehicle breakdown events partitioned according to their distance from a particular service center. They are grouped by distance as follows: less than 10 km, between 10 km and 30 km, between 30 km and 60 km and more than 60 km. In this case the service center is a fixed entity specified in the context definition, and the partition specifications are <10 km; ≥ 10 km and < 30 km; ≥ 30 km and < 60 km; ≥60 km.

An alerting service at a big conference, which attendees can use to receive alerts when they are in the proximity of another person. Attendees send periodic events giving their own location and these events include the name of the person they are interested in meeting. This example shows the use of an entity distance location context where the entity is itself moving. The context specification has an entity attribute which tells the context which attribute from the event gives the name of the person being tracked and an explicit partition with a maximum distance of 100m. This means that the alert generation event processing agent is only invoked if the two people are less that 100m apart.

### *4.2.3 Event distance location*

This type of context specifies an event type and a matching expression predicate. A new partition is created if an event occurrence is detected that matches this predicate. Subsequent events are then included in the context partition if they occurred within a specific distance of the initiating event. Distance is defined as in the entity distance location context.

Some examples:

Detecting the presence of an aircraft within a 10 km radius of a volcanic ash

Detecting a case of scarlet fever within a distance of 100km from a previous outbreak of this disease (*event type* is disease report, predicate is disease="scarlet fever", *maximum distance* is 100 km).

## 4.3 Segmentation oriented context

A *segmentation-oriented* context is used to group event instances into context partitions based on the value of some attribute or collection of attributes in the instances themselves. As a simple example consider an event processing agent that takes a single stream of input events, in which each event contains a customer identifier attribute. The value of this attribute can be used to group events so that there's a separate context partition for each customer. Each context partition only contains events related to that customer, so that the behavior of each customer can be tracked independently of the other customers.

In this kind of segmentation-oriented context, the context specifies just the attribute (or attributes) to be used, and this implicitly defines a context partition for every possible value of that attribute. Alternatively, a segmentation-oriented context definition can list its context partitions explicitly. Each partition specification includes one or more predicate expressions involving one or more of the event attributes; an event is assigned to a context partition if one of its predicates evaluates to true.

Partitions can be specified:

- By attribute value: there is a context partition for each customer, for all events related to this customer
- By combination of attributes: all car accidents are grouped by car type and driver license type.
- By explicit predicate: there is a context partition by an employee's band. One partition for band < 5, one partition for band = 6 or 7, one partition for band 8, one partition for band 9, and one partition for band > 9.

## 4.4 State-oriented context

*State-oriented* context differs from the other dimensions in that the context is determined by the state of some entity that is

external to the event processing system. This is best illustrated with some examples:

- An airport security system could have a threat level status taking values green, blue, yellow, orange, or red. Some events may need to be monitored only when the threat level is orange or above, while other events may be processed differently in different threat levels.
- Traffic in a certain highway has several status values: traffic flowing, traffic slow, traffic stationary.

Each of these states issues a context partition.

## 4.5 Context composition

Event processing applications often use combinations of two or more of the simple context types that we have discussed so far. In particular a temporal context type is frequently used in combination with a segmentation-oriented context, and we show and example of this in figure 6

Figure 6 is showing composition of segmentation-oriented context and temporal context. Each square is a separate context partition and designates the combination of an hour-long interval and a specific customer

Composition of context is typically an intersection between two context dimensions, but can also consist of other set operation, such as: union and set difference, where the member contexts are both of the same dimension.

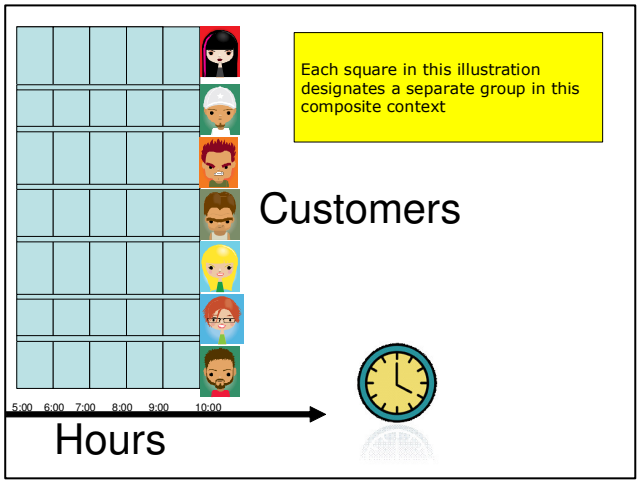


**Figure 6. composite context**

These context dimensions generalize the functionality that exists within various products and models of event processing; next we are moving to context implementation in five event processing products that issue a sample of various products.

# 5. PRODUCTS OVERVIEW

In the following sections we introduce the context support within a sample of existing event processing commercial platforms. We classify this context support according to the four context dimensions defined in section 4: temporal context, segmentation context, spatial context and state-oriented context. The five languages we survey are: StreamBase StreamSQL, Oracle EPL, Rulecore Reakt, Sybase's CCL (formerly Aleri/Coral8) and IBM's Websphere Business Events.

## 5.1 StreamBase StreamSQL

StreamSQL[17] is a SQL based EPL language focusing on event stream processing, extending the SQL semantics and adds stream-oriented operators as well as windowing constructs for subdividing a potentially infinite stream of data into analyzable segments.

StreamSQL supports the following context dimensions as defined in section 4: temporal context, segmentation context.

### *5.1.1 Temporal context*

The windowing constructs correspond to the temporal context dimension, where the windows are defined based on time, or based on number of events, or based on an attribute value within an event.

The temporal context supported is the sliding fixed interval called in StreamSQL **sliding fixed window** and sliding event interval called **sliding event window**. Default evaluation policy for the operators is deferred upon reaching the specified window size, however this can be changed in the operator specifications to immediate or semi-immediate (performing the operation once every number of units even if window size is not reached).

#### *5.1.1.1 Sliding fixed window*

A window construct of type "time" supports the notion of sliding fixed interval. New windows are opened at regular intervals relative to each other, as specified in the ADVANCE parameter, the windows remain open for a specified period of time, as stated in the SIZE parameter. Additional parameters farther influence the behavior of the window construct, such as OFFSET which indicates a value by which to offset the start of the window.

For example, Figure 7 create sliding overlapping windows which are opened every second and remain open for 10 seconds

```
CREATE WINDOW tenSecondsInterval(
        SIZE 10 ADVANCE 1 {TIME}
);
```

**Figure 7. StreamSQL sliding fixed window example**

#### *5.1.1.2 Event interval and sliding event interval*

A window construct of type "tuple" supports the notion of sliding event intervals, where the window size is defined based on the events number, and the repeating intervals are defined in terms of events as well. The repetition is optional, a policy determines whether it is necessary to open new windows on event arrival or not, therefore this supports the notion of event interval.

For example, Figure 8 defines a context where a new window is opened upon arrival of a new tuple (event).

```
CREATE WINDOW eventsInterval (
  SIZE 10 ADVANCE 1 {TUPLE} TIMEOUT 20
);
```

**Figure 8. StreamSQL sliding event window example**

The windows are closed either after accumulating 5 events or when a timeout of 20 seconds from window opening expires.

### 5.1.2 Segmentation Context

Additional constructs in StreamSQL, such as GroupBy and PartitionBy clauses correspond to segmentation context dimension.

### 5.1.3 Composite context

Options exist to use the temporal and segmentation context dimensions in conjunction thus introducing a composite context consisting of multiple dimensions.

## 5.2 Oracle EPL

Oracle CEP [27] offering is a Java-based solution targeting event stream processing applications.

Oracle EPL supports both the segmentation and temporal contexts dimensions [18].

### 5.2.1 Temporal context

The temporal context supports four types of sliding windows: row-based, time-based, batched row and time-based, which correspond to sliding event and sliding fixed interval in this dimension with different initiation policies.

#### 5.2.1.1 Sliding event interval

The sliding row-based window supports retention of last N event entries. For example, the clause shown in Figure 9 retains the last five events of the *Withdrawal* stream, when the oldest event is "pushed out" of the sliding window, providing a form of sliding event interval

```
SELECT * FROM Withdrawal RETAIN 5 EVENTS
```

**Figure 9. Oracle EPL sliding raw-based window example**

Figure 10 demonstrates the flow of events throw the event stream and the content of the window for this statement.

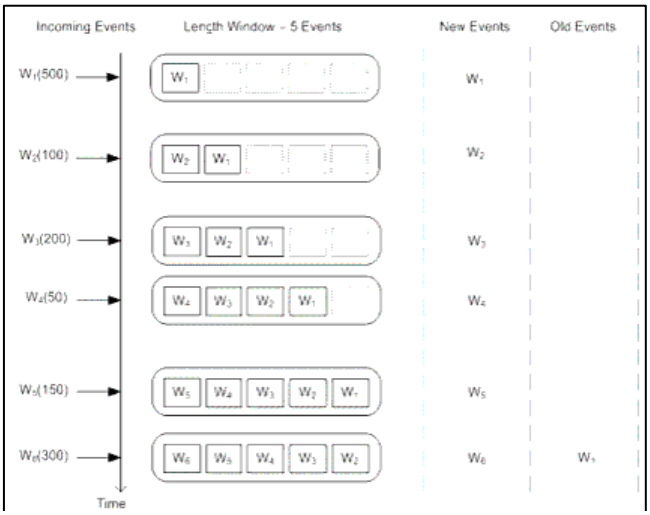


**Figure 10. Oracle EPL Sliding row-based window retaining 5**

#### 5.2.1.2 Sliding fixed interval

The time based sliding window is a moving window aggregating events for a specified time, implementing a case of sliding fixed temporal interval. For example, the clause shown in Figure 11 defines a window holding a view into event stream of all events received in the last 4 seconds.

```
SELECT * FROM Withdrawal RETAIN 4 SECONDS
```

**Figure 11. Oracle EPL sliding time-based window example**

Figure 12 demonstrates the time-based window behavior over a period of time

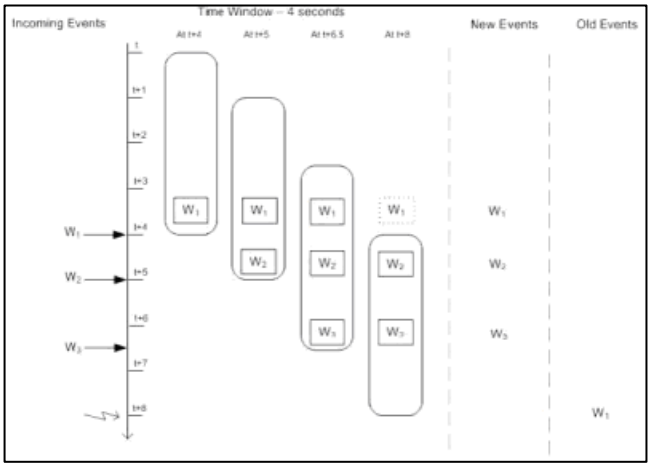


**Figure 12. Oracle EPL Sliding time-based window**

An example of a batched time windows is shown in the clause presented in Figure 13.

```
SELECT * FROM Withdrawal RETAIN BATCH OF 4
SECONDS
```

**Figure 13. Oracle EPL batched time window example**

The windows are non overlapping windows, and a new window is opened only when its predecessor window is closed.

### 5.2.2 Segmentation context

Additionally Oracle EPL supports the GROUP BY clause which enables further segmentation of the event stream.

### 5.2.3 Context composition

The EPL language supports the composition of the temporal and segmentation context dimensions together thus supporting the notion of composite context. Example of syntax supporting the composite context: is shown in Figure 14.

```
INSERT INTO TicksPerSecond
SELECT feed, COUNT(*) AS cnt FROM MarketDataEvent
RETAIN BATCH OF 1 SECOND
GROUP BY feed
```

**Figure 14. Oracle EPL composite context example**

## 5.3 RuleCore Reakt

RuleCore[20] executes ECA rules defined using the Reakt language.. The main language element in Reakt is the rule. RuleCore uses a notion of "views" as execution context for the rule. Each rule instance maintains its own virtual view into the incoming event stream. The view provides a window into the incoming event stream providing context for rule evaluation.

Reakt supports segmentation, temporal, spatial and state contexts.

### *5.3.1 Temporal context*

The temporal context is implemented using **MaxAge** and MaxCount view properties

#### *5.3.1.1 Sliding fixed interval*

The **MaxAge** view property which defines the maximum age for each event in the view.

Example: Figure 15 shows a definition that limits every rule instance to view only events from the past 10 minutes.

```
<MaxAge>00:10:00</MaxAge>
```

**Figure 15. ruleCore Reakt fixed window example**

#### *5.3.1.2 Sliding event interval*

The sliding event-based **MaxCount** view supports retention of last N event entries, for example the statement guarantees that an event view never contains more than 100 events.

```
<MaxCount>100</MaxCount>
```

**Figure 16. ruleCore Reakt - sliding event interval example**

### *5.3.2 Segmentation context*

Segmentation context is implemented using **match** view property which is very flexible and can select values from the event bodies using the full expressive power of XPath. This allows for powerful semantic matching even of complex event structures. An example is shown in Figure 17.

```
<Match>
  <Value>
<Event><base:XPath>EventDef[eventType="Warning]</bas
e:XPath></Even>
<Field><base:XPath xmlns:base="base">base: ServerIP
base:XPath></Field>
  </Value>
<Match>
```

**Figure 17. ruleCore Reakt match view**

### *5.3.3 Spatial context*

Spatial context is implemented using **Type** view property by defining new geographic zone entity type defined by its coordinates and **Match** view property. An event is then associated specific entity which may have a set of properties such as position or a Zone. Spatial Evaluation is done by applying **Match** property on specific Entity type with geographic location and to specific Zone that specifies the reference geo area, all events of specific entity type and which match Zone property will be classified to this specific context partition.

Example: Figure 18 illustrates the association of specific events that are contained in a location Zone. For this rule we define ZoneEntry view, which will contain events from "vehicle" entity type (Match/vehicle) and only from specific zone location (Match/Zone).

```
<ViewDef name="ZoneEntry">
<Properties><!– All events in the view must be of types InsideZone or OutsideZone –>
  <Type><Event><XPath>EventDef[@eventType="InsideZone"]</XPath></Event>
   <Event><XPath>EventDef[@eventType="OutsideZone"]</XPath> </Event>
</Type>
  <!– All events in the view must be from the same vehicle –>
  <Match name="vehicle">
   <Value><Event><XPath>EventDef[@eventType="OutsideZone"]</XPath></Event>
    <Property name="Vehicle"/></Value>
   <Value> <Event> <XPath>EventDef[@eventType="InsideZone"]</XPath></Event>
 <Property name="Vehicle"/> </Value>
  </Match><!– All events in the view must be from the same zone –>
  <Match name="zone">
   <Value> <Event> <XPath>EventDef[@eventType="InsideZone"]</XPath> </Event>
    <Property name="Zone"/> </Value>
   <Value> <Event><XPath>EventDef[@eventType="OutsideZone"]</XPath> </Event>
    <Property name="Zone"/> </Value>
  </Match>
 </Properties>
</ViewDef>
```

**Figure 18. ruleCore Reakt - spatial context example**

### *5.3.4 State context*

State contexts are implemented in a similar way to the way in which spatial contexts are implemented, using Type view property where each event entity has specific properties. State evaluation is done by applying Match property on specific Entity type property, all events of specific entity type and which match specific property will be in the context.

### *5.3.5 Composite context*

The Reakt language allows the composition of the temporal, segmentation, spatial and state context dimensions together thus supporting the notion of composite context.

## 5.4 Sybase Aleri -Coral8 and CCL

Sybase Aleri CEP offering consists of Aleri Streaming Platform and Coral8 engine. Coral8's authoring language[22] [23] CCL (Continuous Computation Language) is a SQL-based event processing language extending SQL in a different way relative to the Streambase and Oracle examples presented previously. CCL supports different combinations of windows types: sliding and jumping windows, either row-based (event-based) or time-based.

Examples include:

Time intervals ("keep one hour's worth of data").

Row counts ("keep the last 1000 rows").

Value groups ("keep the last row for each stock symbol").

Relative to the context dimension classification defined in section 4, CCL supports both the temporal and segmentation context dimensions.

### *5.4.1 Temporal context*

In the temporal dimension, the sliding fixed interval, sliding event interval and event interval are supported. The difference in CCL terminology between sliding and jumping windows is the way they treat the intervals periods and the window initiation policies.

#### *5.4.1.1 Sliding fixed interval*

Figure 19 illustrates a sliding fixed interval representing a 15 minute window into *Trades* event stream, and the aggregation is determining the highest trade in the last 15 minutes. The interval period is once every minute, and the interval duration is 15 minutes. The initiation policy is "refresh" – the previous window is closed once the new one is created.

```
INSERT INTO MaxPrice
SELECT Trades.Symbol, MAX(Trades.Price)
FROM Trades KEEP 15 MINUTES
WHERE Trades.Symbol = 'IBM';
```

**Figure 19 Sybase CCL - Sliding fixed interval example**

#### *5.4.1.2 Event interval and sliding event interval*

A similar example illustrating the sliding event interval is deducing the highest price in the last 15 trades instead of last 15 minutes. This example is shown in Figure 20

```
INSERT INTO MaxPrice
SELECT Trades.Symbol, MAX(Trades.Price)
FROM Trades KEEP 15 ROWS
WHERE Trades.Symbol = 'IBM';
```

**Figure 20. Sybase CCL - Sliding event interval**

The window can be also defined implicitly in pattern matching queries, such as the following sliding event interval definition, in Figure 21, describing a period which starts with each event A arrival, and ends 10 seconds later .

```
[10 SECONDS: A, B, !C]
```

**Figure 21. Sybase CCL - pattern embedded context example**

### *5.4.2 Segmentation context*

The GROUP BY clause supports the segmentation context implementation.

### *5.4.3 Composite context*

A combination of the temporal and segmentation context enables the expression of composite contexts. In the example shown in Figure 22, the sliding fixed interval window is further segmented according the stock's symbol. This query determines the maximum price for each stock type in the last 15 minutes.

```
INSERT INTO MaxPrice
SELECT Trades.Symbol, MAX(Trades.Price)
FROM Trades KEEP 15 MINUTES
GROUP BY Trades.Symbol;
```

**Figure 22. Sybase CCL - composite context example**

## 5.5 IBM Websphere Business Events

IBM WBE [21] is a business event processing system, focusing on the business user and his needs. WBE supports both the temporal and segmentation context dimensions.

### *5.5.1 Temporal context*

The temporal context capability is supported using deferred evaluation mode (Figure 23). Events can be evaluated using time delays from triggering event. These time delays can be for a fixed amount of time or until a certain amount of time has elapsed after another event has occurred, or until a specific date.

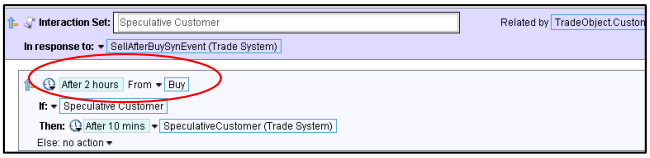


**Figure 23. WBE definition of temporal context**

WBE also supports temporal event evaluation within context applying filtering conditions to the rule. This is done using temporal functions such as **: Follows After, Follows before, Follows Within** . These functions determine if the specified event in the associated scope will follow the event or action defined in the filter after/before/within certain amount of time. Functions such as**: Is Present** before, **Is Present After, Is Present Within**, determine if the event specified in the associated scope has previously occurred in the same context after/before/within the date/time period defined in the filter. An example of the Follows By construct is shown in Figure 24.

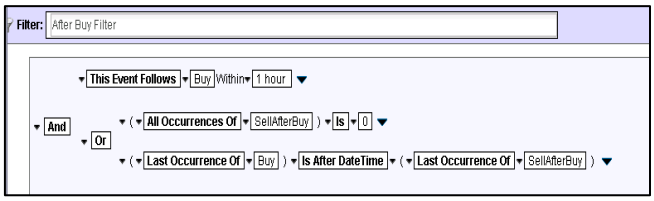


**Figure 24. WBE definition of segmentation context**

### *5.5.2 Segmentation context*

A segmentation context is easily expressed in WBE using the "Related By" construct, shown in Figure 25, which support the segmentation of group of events based on event field value or a composition of field values [19].

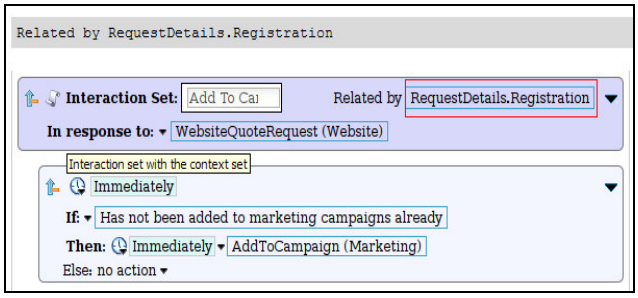


**Figure 25. WBE definition of segmentation context**

To conclude, most CEP offerings today support one or another variation on temporal and segmentation context dimensions. While in most cases state-oriented context can be implemented in those solutions indirectly, using event data and filtering on this data, out-of-the-box geospatial capabilities are lacking in most of the mentioned products.

## 6. CONCLUSION

This paper summarized a tutorial that discusses various aspects of contexts in event processing; it presented the general notion of context, a general view of context in event processing, and a sample of implementations of this notion within various products. We believe that the next generation of event processing as well as computing will view context as a major term, and that standardization of the context semantics will be useful to the community in large. There are various challenges in the areas of modeling, implementation, architecture and optimization of contexts. Moreover, going back to the introduction, while context is a key concept in event processing, we expect that it will get into the infrastructure of enterprise computing infrastructure as a main enabler for context driven computing.

## REFERENCES

[1] Abowd, G.D., Atkeson, C.G., Hong, J., Long, S., Kooper, R., Pinkerton, M.: Cyberguide: A mobile context-aware tour guide. Wireless Networks: special issue on mobile computing and networking: selected papers from MobiCom.96, Vol. 3, No 5. (1997) 421-433.

[2] Adi A., Biger A., Botzer D., Etzion O., Sommer Z. Context Awareness in Amit. Active Middleware Services 2003: pp. 160-167.

[3] Arvind Arasu, Brian Babcock, Shivnath Babu, Mayur Datar, Keith Ito, Itaru Nishizawa, Justin Rosenstein, Jennifer Widom: STREAM: The Stanford Stream Data Manager. SIGMOD Conference 2003: 666

[4] Buvac S. and Mason I.A., Propositional logic of context. In Proc. of the 11th National Conference on Artificial Intelligence, 1993.

[5] BuvacS. Quantificational Logic of Context. In Proc. of the 13th National Conference on Artificial Intelligence, 1996.

[6] Broadbent, J., Marti, P.: Location-Aware Mobile Interactive Guides: Usability Issues. Proc. Inter. Cultural Heritage Informatics Meeting, Paris, France, 1997.

[7] Heverst, K., Davies, N., Mitchell, K., Friday, A., Efstratiou, C.: Developing a Context-Aware Electronic Tourist Guide: some issues and experiences, Proc. CHI, The Hague, Netherlands, 2000.

[8] Etzion O, Niblett P. – Event Processing in Action, Manning Publications, 2010.

[9] Jensen C. et al: The Consensus Glossary of Temporal Database Concepts - February 1998 Version. In: Opher Etzion, Sushil Jajodia and Suryanarayana Sripada: Temporal Databases Research and Practice, Springer, 1998: 367-405

[10] Kindberg, T. et al.: People, Places, Things: Web Presence for the Real World. Proc. 3rd Annual Wireless and Mobile Computer Systems and Applications, Monterey, CA, 2000.

[11] Luciano S., Fausto G. ML Systems: A Proof Theory for Contexts, Journal of Logic, Language and Information, vol. 11, pp 471-518 2002.

[12] Mohomed I., Misra A., Ebling M., Jerome W. Context-aware and personalized event filtering for low-overhead continuous remote health monitoring. International Symposium on a World of Wireless, Mobile and Multimedia Networks, 2008.

[13] Rosemann M., Recker J., Flender C., and Ansell P., "Context-Awareness in Business Process Design," in 17th Australasian Conference on Information Systems, Adelaide, Australia, 2006.

[14] Rosemann M., Recker J., and Flender C., Contextualization of Business Processes, International Journal of Business Process Integration and Management, vol. 3, pp. 47-60, 2008.

[15] Woodruff, A. Aoki, P.M., Hurst, A., Szymanski, M.H.: Electronic Guidebooks and Visitor Attention. Proc. 6th Int. Cultural Heritage Informatics Meeting, Milan, Italy, 2001, pp. 437- 454.

[16] http://blog.roundarch.com/2010/03/25/in-the-realm-of-web-2-0-context-is-king-part-one/

[17] http://dev.streambase.com/developers/docs/sb37/streamsql/index.html

[18] http://download.oracle.com/docs/cd/E13157_01/wlevs/docs30/epl_guide/overview.html

[19] http://publib.boulder.ibm.com/infocenter/wbevents/v7r0m0/index.jsp?topic=/com.ibm.wbe.install.doc/doc/configuringtheruntime.html

[20] http://rulecore.com/content/view/35/52/

[21] http://www-01.ibm.com/software/integration/wbe/features/?S_CMP=wspace

[22] http://www.aleri.com/products/aleri-cep/coral8-engine/ccl

[23] http://www.aleri.com/WebHelp/programming/programmers/c8pg_using_windows.html

[24] http://www.gartner.com/DisplayDocument?id=535313

[25] http://www.geovector.com/

[26] http://www.nttdocomo.com/

[27] http://www.oracle.com/technologies/soa/complex-event-processing.html

[28] http://www.sigmod.org/sigmod03/eproceedings/papers/dem0pdf