SEMANTIC EVENT MODEL
AND
ITS IMPLICATION ON SITUATION DETECTION

A. Adi, D. Botzer, O. Etzion
IBM Research Laboratory in Haifa, Israel
(Adi / Botzer / Etzion)@il.ibm.com

Abstract - Events are at the core of reactive applications, which have become popular in many domains. Contemporary modeling tools lack the capability express the event semantics and relationships to other entities. This research is aimed at providing the system designer a tool to define and describe events and their relationships to other events, object and tasks. It follows the semantic data modeling approach, and applies it to events, by using the classification, aggregation, generalization and association abstractions in the event world. The model employs conditional generalizations that are specific to the event domain, and determine conditions in which an event that is classified to lower level class, is considered as a member of a higher-level event class, for the sake of reaction to the event.

The paper describes the event model, its knowledge representation scheme and its properties, and demonstrates these properties through a comprehensive example.

I. INTRODUCTION AND MOTIVATION

Reactive applications [8], [11] are those who react to the occurrence of events by the activation of alerts or actions.

In contemporary systems there is an increasing number of reactive components that are used for many application types such as: E-Commerce applications (auctions, stocks trading alerts), system management, customer relationship monitoring control systems and publish/subscribe systems.

Current tools for application modeling do not provide any capability to focus on the event semantics and relationships to other entities. While the concept of event exists in modeling tools such as UML [7], the modeling power is limited to interactions of event and state changes. There were references to events as first class modeling entities in the workflow modeling area [4], [5], but these works did not supply a complete event modeling system.

This research is aimed at providing the system designer a tool to define and describe events and their relationships to other events, object and tasks.

The work is based on application of semantic data modeling [9] to the domain of event modeling.

This paper describes an event model, which is part of the Amit (Active Middleware Technology) framework, an application development framework for reactive systems that may react to complex situations, such as composite events.

An event is an instantaneous occurrence that has an active role in the application's flow of control.

A situation is an event that is detected when a predicate over the event history becomes true. This concept is intended to bridge the gap between the detected events and the situations to which the system should react. A situation is an extension of the term composite event [2], [3], [6]

Example:
- Events: printer-is-offline, printer-is-online.
- The situation that requires reaction: at least three printers in the same building are offline simultaneously.

In this paper we present a semantic event model and its implications to the events and situations framework.

This paper is structured in the following way: Section II describes the semantic abstractions and the implied knowledge representation scheme. Section III describes the model's properties and demonstrates the usage of these properties in a simple example. Section IV concludes the paper.

II. THE SEMANTIC MODEL

A. Semantic Abstractions

The area of semantic data modeling [7] has provided a substantial contribution to the conceptualization and to the functionality of data models. In this section we briefly survey the semantic abstractions, and show their relevance to the event management domain.

Classification: This abstraction classifies an event to an event class [1], i.e. a set of events with similar characteristics. This is a fundamental abstraction, which allows defining operations on events in the class level. A single event is considered to be a member of a single class, denoted by: classified-to \((e, E)\). instance-of is a predicate with two arguments: an event-class and an event. If instance-of \((E, e)\) is evaluated to true, then all the functionality associated with \(E\) applies for \(e\). The classification rule states:

\[
\text{classified-to (e, E) } \rightarrow \text{instance-of (E, e)}.
\]

According to the classification rule, instance-of \((E, e)\) is evaluated to true, for each event \(e\) that has been classified to \(E\). In the sequel we discuss additional ways in which an event
e can behave as if it is an instance of E, besides direct classification.

**Aggregation:** Most of the existing event models view event as an atomic unstructured entity [3]. We employ the aggregation abstraction that views an event as a set of attributes, each of them with a value specific to this event [10]. The attributes are defined at the class level, where the instances of these attributes are considered as a part of the event values. Attributes may have different types such as: numeric, string and references to objects and events. The supported data structure should be flexible and include atomic attributes, sequence and set attributes, and tuple attributes.

**Additional Classification:** The classified-to function classifies an event into a unique event class. There are several cases in which it is desirable to act in certain contexts as if an event is classified into additional classes. These cases are strict generalization, conditional generalization and association.

**Strict Generalization:** A generalization is a subset hierarchy relationship among two event classes [10], denoted by a function \( \text{generalized-to}(E, \{E_1, \ldots, E_n\}) \). It means that each of the elements in the set \( \{E_1, \ldots, E_n\} \) is a generalization of E, i.e. multiple generalization is permitted.

**Specialization-of (E, E')** is a predicate that is evaluated to true if both E, E' are event classes, and

\[ \text{generalized-to}(E, ES) \land E' \in ES. \]

Specialization is the opposite relationship of generalization.

In our context, a generalization relationship is a way to create additional instances to a class, according to the generalization rule:

\[ \text{instance-of}(E, e) \land \text{specialization-of}(E, E') \rightarrow \text{instance-of}(E', e). \]

This rule can be applied in a recursive way, e.g. if e is classified to E, that is generalized to E', while E' is generalized to E'', then according to the generalization rule, e is an instance of E' and E'' in addition to E. This strict generalization abstraction is common in systems, and is equivalent to strict inheritance in object-oriented models.

**Conditional Generalization:** A conditional generalization is a generalization that is contingent upon additional conditions. Data models support only strict generalization; conditional generalization is a novelty of our work. The types of conditions can be attribute values and context variables. The conditional generalization rule is:

\[ \text{Instance-of}(E, e) \land \text{specialization-of}(E, E') \land \text{COND} \rightarrow \text{Instance-of}(E', e). \]

Examples:

1. The event-class “HP Printer Failure” is generalized to “HP Product Problem”, only if the attribute failure-type = “unrecoverable”. In this case an event e which is classified to “HP Printer Failure” is considered as an instance of the event-class “HP Products Problem” only if its failure-type = “unrecoverable”.

2. The event-class “HP Printer Failure” is generalized to “HP Product problem”, only in the context of the situation “alert on more than 3 HP related problems per day”, and not in any other context. A condition can also be a conjunction or disjunction of both types.

Note that conditional generalization does not exist in data models, because generalization of objects denotes a strict subset hierarchy. In the event world, the classification to higher order does not necessarily implies subset hierarchy, it can imply events that have the same operational semantics in certain cases. This additional classification may be context-sensitive and not strict.

**Association:** An association is a relationship between two classes and a conditional expression [1], denoted as Association-of (E, E’, COND). This relationship denotes that E’ is an association of E, under the condition COND. The association relationship creates an additional virtual event class, for which there are no events that are directly classified to.

Example: the event class is “Printer Failure”, the condition is “Time between 8:00am and 5:00pm”, and the association is: “Printer Failure during working hours”. The classification is defined by the association rule:

\[ \text{Instance-of}(E, e) \land \text{association-of}(E, E', \text{COND}) \land \text{COND} \rightarrow \text{Instance-of}(E', e). \]

Note that association is distinct from specialization. In the specialization case, the event is classified to a specialized class, and is inferred to be classified to generalized class as well, while in the association case, the event is classified to the more general class, and it is classified to the associated class if a predicate is satisfied.

**Uncertain Generalization/association:** A generalization or an association relationship may have some certainty value, which designates the strength of this relationship. The exact interpretation of this value is out of this paper scope.

### B. Knowledge Representation Scheme

Each event has the following scheme:

**The Basic Schema:**

- **Attributes set:** A set of attributes that are aggregated to each event instance.
- **Temporal dimensions:** Event occurrence time, event detection time.

**The Semantic Event Model attributes:**

- **Generalization_set:** Set of generalized events.
  
  An element in this set is composed of:
  
  - **Gener_event:** An event-class id.
The conditional expression that if evaluated to “true”, the event is generalized (the default is “true” to denote strict generalization).
- **Gener_Certainty**: The certainty value associated with the generalization operation.

**Association_Set**: Set of associated events.

An element in this set is composed of:
- **Assoc_event**: An event-class id.
- **Assoc_Cond**: The conditional expression that if evaluated to true, the event is associated.
- **Assoc_Certainty**: The certainty value of the association operation.

The Situation Manager attributes:

- **Situations_Set**: Set of related events (situations) that an instance of ei participates in their composition (detection).

An element in this set is composed of:
- **Situation event**: An event-class id.
- **Special_SC_Cond**: (SC = Situation Context) the conditional expression that determines whether specialized events can play the role of this event.
- **Special_SC_Certainty**: The certainty value associated with the "specialization" attribute.

**Operands_list**: List of operands used to define the composition properties of ei.

The reactive attributes:

- **Reaction_Set**: set of conditions and actions.

III. THE MODEL PROPERTIES

C. Properties

1. **Generalization and Association**

The attributes: **Generalization_Set** and **Association_Set** include all the events directly generalized from an event and all the events associated from the event, respectively.

**Example:**
- Laser-printer-error.**Generalization_Set.Gener_event** = {printer-error}
- Printer-error.**Generalization_Set.Gener_event** = {peripheral-device-error, device-error, floor4-device-error}
- Laser-printer-error.**Association_Set.Assoc_event** = {HP-Laser-printer-error, Canon-Laser-printer-error}.

2. **Condition**

The attributes **Gener_Cond** and **Assoc_Cond** represent the conditional expression in which this event is generalized or associated to the **Gener_event** and **Assoc_event**, respectively.

The strict generalization case is a special case, in which **Gener_Cond** = “TRUE”, another trivial (and useless) case is that that the value of **Gener_Cond** = “FALSE”.

In a similar way, we can assign “TRUE” or “FALSE” values to **Assoc_Cond**.

3. **Certainty**

The attributes **Gener_Certainty** and **Assoc_Certainty**, represent the certainties in which this event is generalized or associated to the **Generalization_event** and **Association_event**, respectively.

Special cases are: **Gener_Certainty** = 1 (the default case), and **Gener_Certainty** = 0 (again, useless case).

In a similar way, we can give 1 or 0 values to **Assoc_Certainty**.

4. **Situations**

The attributes **Special_SC_Cond/Certainty** (SC = Situation Context).

Let E1 be a generalization of E11, and S1 be a situation for which E1 is a participating event. Let e be an event that is classified to E11. The condition/certainty that are defined on (E1, S1) determines whether e participates in the evaluation of S1 in the role of E1.

In another words, this attribute enables the situation manager user/designer to control (or to filter) the generalization and the association relationships for any specific situation context’s point of view. Note that the condition may refer to the original event class of an event using the predicate instance-of.

A situation is a special case of an event; thus, we can define generalization and association relationships among situations, as well. A situation that participates in another situation (like an event) is handled the same as an event handling. This is an additional way to define relationships among situations even if there are no relationships among their events.

D. **Example**

There are three events that are defined as follows:

E1, E11, E111, E12 – classes.

Events: e1 ∈ E1, e11 ∈ E11, e111 ∈ E111, e12 ∈ E12
Example:
e1 = “printer-error”
e11 = “Laser-printer-error”
e111 = “HP- Laser-printer-error”.
e12 = “Printer Failure during working hours”

Property #1:
E11. Generalization_Set = \{E1\}
E111. Generalization_Set = \{E11\}
E1. Association_Set = \{E12\}.

Suppose E2, E3 are classes too, when the appropriate events are: e2 ∈ E2, e3 ∈ E3

Example:
e2 = “printer had some trouble”
e3 = “printer was fixed ”

There are the following situations:
S1 = sequence (E2, E3, E1)
S11 = sequence(E2, E3, E11)
S111 = sequence(E2, E3, E111)
S12 = sequence(E2, E3, E12)

The following tables (Tables 1, 2, 3) demonstrate some scenarios (and their detected situations) related with the example.
The notation (-) in all the Tables indicates that this column is irrelevant.

In Table 1:
Property #2: E1. Association_Set[E12]. Assoc_Cond = TRUE, for all the scenarios in Table 1.

<table>
<thead>
<tr>
<th>Case no.</th>
<th>Occurs</th>
<th>Property #2</th>
<th>Property #4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>e1</td>
<td>-</td>
<td>E1, E12</td>
</tr>
<tr>
<td>2</td>
<td>e1</td>
<td>-</td>
<td>E1, E12</td>
</tr>
<tr>
<td>3</td>
<td>e11</td>
<td>-</td>
<td>FALSE</td>
</tr>
<tr>
<td>(*) 4</td>
<td>e11</td>
<td>TRUE</td>
<td>E11, E1, E12</td>
</tr>
<tr>
<td>5</td>
<td>e11</td>
<td>-</td>
<td>TRUE</td>
</tr>
<tr>
<td>6</td>
<td>e11</td>
<td>FALSE</td>
<td>E111</td>
</tr>
<tr>
<td>7</td>
<td>e11</td>
<td>TRUE</td>
<td>FALSE</td>
</tr>
<tr>
<td>8</td>
<td>e11</td>
<td>TRUE</td>
<td>TRUE</td>
</tr>
<tr>
<td>9</td>
<td>e11</td>
<td>TRUE</td>
<td>TRUE</td>
</tr>
<tr>
<td>10</td>
<td>e11</td>
<td>TRUE</td>
<td>TRUE</td>
</tr>
<tr>
<td>11</td>
<td>e11</td>
<td>TRUE</td>
<td>TRUE</td>
</tr>
<tr>
<td>12</td>
<td>e11</td>
<td>TRUE</td>
<td>TRUE</td>
</tr>
</tbody>
</table>

(* ) - For example let us examine case no. 4. Event e11 occurs. The generalization condition of E11 respect to E1 is TRUE. As a result E1 is also a relevant (class) event. The association condition of E1 respect to E12 is TRUE (for all the cases). As a result E12 is also a relevant (class) event.

So, the relevant events are: **E11, E1, E12**.
Situation \textbf{S11} will be detected because \( s11 \) is a strict function of \( E11 \) (with no conditions).
\textbf{S12} will be detected because \( E12 \) participate in \( S12 \).
\( E1 \)'s \text{Special\_SC\_Cond} (specialized situation context) is TRUE respecting to situation \( S1 \), so situation \textbf{S1} will be detected.
As a result, the detected situations will be: \textbf{S1, S11, S12}.

\textbf{In Table 2:}
The “TRUE/\text{FALSE}” can be replaced by conditions.
Notice that not all the possible cases are described in Table 2.

<table>
<thead>
<tr>
<th>Case no.</th>
<th>Occurs</th>
<th>\text{Property #2}</th>
<th>\text{Property #4}</th>
<th>The relevant events</th>
<th>\text{E1.\text{Situation_Set}[S1].\text{Special_SC_Cond}}</th>
<th>the detected situation/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( e1 )</td>
<td>- \text{FALSE}</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>\text{S1}</td>
</tr>
<tr>
<td>2</td>
<td>( e1 )</td>
<td>- \text{Cond1}</td>
<td>\text{E1, E12}</td>
<td>E1 \text{TRUE}</td>
<td>\text{E1, E12} \text{TRUE}</td>
<td>\text{S1, S12}</td>
</tr>
<tr>
<td>3</td>
<td>( e1 )</td>
<td>- \text{TRUE}</td>
<td>\text{E1, E12}</td>
<td>E1 \text{FALSE}</td>
<td>\text{E1, E12} \text{FALSE}</td>
<td>\text{S1}</td>
</tr>
<tr>
<td>4</td>
<td>( e11 )</td>
<td>FALSE</td>
<td>-</td>
<td>\text{E11}</td>
<td>-</td>
<td>\text{S11}</td>
</tr>
<tr>
<td>5</td>
<td>( e11 )</td>
<td>TRUE</td>
<td>FALSE</td>
<td>\text{E11, E1}</td>
<td>-</td>
<td>\text{S1, S11}</td>
</tr>
<tr>
<td>6</td>
<td>( e11 )</td>
<td>\text{Cond2}</td>
<td>TRUE</td>
<td>\text{E11, E1, E12}</td>
<td>-</td>
<td>\text{S2, S11, S12}</td>
</tr>
<tr>
<td>7</td>
<td>( e11 )</td>
<td>TRUE</td>
<td>\text{Cond3}</td>
<td>\text{E11, E1, E12}</td>
<td>\text{Cond4}</td>
<td>\text{S1, S11, S12}</td>
</tr>
</tbody>
</table>

\text{Cond1} = (\( e1.time > 8 \)).
\text{Cond2} = (\( e11.error\_code > 10 \)).
\text{Cond3} = (\( e11.type = \text{"fault"} \)).
\text{Cond4} = (\( e11.type = \text{"warning"} \) and \( e11.validtime = \text{"true"} \)).

\textbf{In Table 3:} The certainty in property \#3 (certainty) is treated in similar way:
Certainty function is a function that maps \( n \) certainty values to one representative value.
Examples:
1. \( f(c1, c2) = c1 + c2 - (c1*c2) \)
2. \( f(c1, c2) = \min(c1, c2) \)
3. \( f(c1, c2) = \max(c1, c2) \)
TABLE 3
CERTAINTIES

<table>
<thead>
<tr>
<th>Case no.</th>
<th>Occurs</th>
<th>Property #2</th>
<th>Property #4</th>
<th>the detected situation/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>e₁</td>
<td>FALSE</td>
<td>E₁</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>e₁</td>
<td>Certainty 0.6</td>
<td>With certainty 0.6: E₁, E₁₂ Else: E₁</td>
<td>TRUE With certainty 0.6: S₁, S₁₂ Else: S₁</td>
</tr>
<tr>
<td>3</td>
<td>E₁₁</td>
<td>TRUE</td>
<td>E₁₁</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>E₁₁</td>
<td>FALSE</td>
<td>E₁₁</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>E₁₁</td>
<td>TRUE</td>
<td>E₁₁, E₁</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>E₁₁</td>
<td>Certainty 0.3</td>
<td>TRUE</td>
<td>With certainty 0.3: E₁₁, E₁, E₁₂ Else: E₁₁</td>
</tr>
<tr>
<td>7</td>
<td>E₁₁</td>
<td>TRUE</td>
<td>Certainty 0.8</td>
<td>With certainty 0.8: E₁₁, E₁, E₁₂ Else: E₁₁, E₁</td>
</tr>
</tbody>
</table>

IV. CONCLUSION

This research provides the system designer a tool to define and describe events and their relationships to other events, object and tasks. The relationships to other events are defined through the generalization and association abstractions and through attributes that may reference events. The relationships to other objects are defined through attributes' values; the relationships with tasks are defined through the reactive attributes.

This paper's contribution is the capability to describe a comprehensive event model, and refer to event as a first class citizen in the modeling world. This is an important feature in the modeling and design of reactive components or applications.

REFERENCES