2000

Semantic Event Model and Its Implication on Situation Detection

A. Adi
IBM Research Laboratory, adi@il.ibm.com

D. Botzer
IBM Research Laboratory, botzer@il.ibm.com

O. Etzion
IBM Research Laboratory, etzion@il.ibm.com

search

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 Middlewere 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:
classified-to (e, E) $\rightarrow$ *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 ***generalized-to*** *(E, {E1…, En}).* It means that each of the elements in the set {E1,…,En} 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

generalized-to (E, ES) Λ 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:

instance-of (E, e) Λ *specialization-of (E, E')* $\rightarrow$ *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:

Instance-of (E, e) Λ *specialization-of (E, E')* Λ COND $\rightarrow$ *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:

Instance-of (E, e) Λ *association-of (E, E', COND)* Λ COND $\rightarrow$ *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.

- *Gener_Cond*: 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: Events: 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)

“sequence” is an operator that is satisfied if instances of its arguments occur in the specified order.
The events e2∈ E2, e3∈ E3 have already occurred, and a new event occurs now.

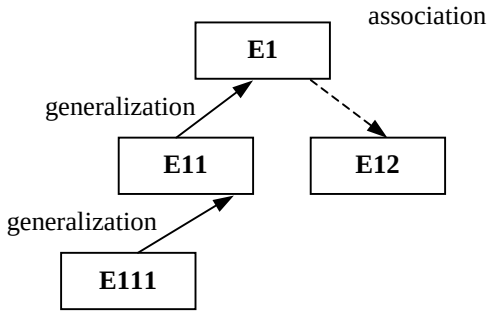

Fig. 1: The events' relationships

The following tables (Tables 1, 2, 3) demonstrate dome 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.*Associaition_Set*[E12].*Assoc_Cond* = TRUE, for all the scenarios in Table 1.

TABLE 1
RELATIONSHIP DETECTION

		E111. *Generalization_ Set*[E11]. *Gener_Cond*	E11. *Generalization_ Set*[E1]. *Gener_Cond*	The relevant events	E1.*Situation_ Set*[S1]. *Special_ SC_Cond*	E11.*Situation_ Set*[S11]. *Special_ SC_Cond*	**the detected situation/s**
Case no.	Occurs	Property #2			Property #4		
1	e1	-	-	E1, E12	TRUE	-	**S1,S12**
2	e1	-	-	E1, E12	FALSE	-	**S1**
3	e11	-	FALSE	E11	-	-	**S11**
(*) 4	e11	-	TRUE	E11, E1, E12	TRUE	-	**S1, S11, S12**
5	e11	-	TRUE	E11, E1, E12	FALSE	-	**S11**
6	e111	FALSE	-	E111	-	-	**S111**
7	e111	TRUE	FALSE	E111, E11	-	TRUE	**S111, S11**
8	e111	TRUE	FALSE	E111, E11	-	FALSE	**S111**
9	e111	TRUE	TRUE	E111, E11, E1, E12	TRUE	TRUE	**S1, S11, S111, S12**
10	e111	TRUE	TRUE	E111, E11, E1, E12	TRUE	FALSE	**S1, S111, S12**
11	e111	TRUE	TRUE	E111, E11, E1, E12	FALSE	TRUE	**S11, S111**
12	e111	TRUE	TRUE	E111, E11, E1, E12	FALSE	FALSE	**S111**

(*) - 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.**

search

Situation **S11** will be detected because s11 is a strict function of E11 (with no conditions).
S12 will be detected because E12 participate in S12.
E1 's Special_SC_Cond (specialized situation context) is TRUE respecting to situation S1, so situation **S1** will be detected.
As a result, the detected situations will be: **S1, S11, S12.**

In Table 2:

The "TRUE/FALSE" can be replaced by conditions.
Notice that not all the possible cases are described in Table 2.

TABLE 2
CONDITIONS

		E11. *Generalization_Set*[E1]. *Gener_Cond*	E1. *Association_Set*[E12]. *Assoc_Cond*	The relevant events	E1.*Situation_Set*[S1]. *Special_SC_Cond*	**the detected situation/s**
Case no.	Occurs	Property #2			Property #4	
1	e1	-	FALSE	E1	-	**S1**
2	e1	-	Cond1	If Cond1: E1, E12 Else: E1	TRUE	**If Cond1: S1, S12 Else: S1**
3	e1	-	TRUE	E1, E12	FALSE	**S1**
4	e11	FALSE	-	E11	-	**S11**
5	e11	TRUE	FALSE	E11, E1	-	**S1, S11**
6	e11	Cond2	TRUE	If Cond2: E11, E1, E12 Else: E11	FALSE	**If Cond2: S1, S11 Else: S11**
7	e11	TRUE	Cond3	If Cond3: E11, E1, E12 Else: E11, E1	Cond4	**If Cond3 and Cond4: S1, S11, S12 Else: S1, S11**

Cond1 = (e1.*time* > 8).

Cond2 = (e11.*error_code* > 10).

Cond3 = (e11.*type* = "fault").

Cond4 = (e11.*type*="warning" and e11.*validtime*="true").

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)

search

TABLE 3
CERTAINTIES

Case no.	Occurs	E11. *Generalization_ Set*[E1]. *Gener_Cond*	E1. *Association_ Set*[E12]. *Assoc_Cond*	The relevant events	E1.*Situation_ Set*[S1]. *Special_ SC_Cond*	**the detected situation/s**
		Property #2			Property #4	
1	e1	-	FALSE	E1	-	**S1**
2	e1	-	Certainty 0.6	With certainty 0.6: E1, E12 Else: E1	TRUE	**With certainty 0.6: S1, S12 Else: S1**
3	e1	-	TRUE	E1, E12	FALSE	**S1**
4	E11	FALSE	-	E11	-	**S11**
5	E11	TRUE	FALSE	E11, E1	-	**S1, S11**
6	E11	Certainty 0.3	TRUE	With certainty 0.3: E11, E1, E12 Else: E11	FALSE	**With certainty 0.3: S1, S11 Else: S11**
7	E11	TRUE	Certainty 0.8	With certainty 0.8: E11, E1, E12 Else: E11, E1	Certainty 0.7	**With certainty function(0.8, 0.7): S1, S11, S12 Else: S1, S11**

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

[1] M. L. Brodie and D. Ridjanovic: On the Design and Specification Database Transactions. On Conceptual Modeling, Springer-Verlag 1984: 277-312

[2] C. Collet, T. Coupaye and T. Svenson - NAOS - Efficient and modular reactive capabilities in an object-oriented database system. VLDB'94

[3] S. Chakravarthy and D. Mishra - Snoop: an expressive event specification language for active databases. Data & Knowledge Engineering, 13(3), Oct 1994.

[4] O. Etzion – Kerem – Reasoning about partially cooperative systems. In Dogac et al (eds) – Workflow Management Systems and Interoperability, Springer-Verlag, November 1998.

[5] A. Gal, O. Etzion - CODES - a design tool for computerized systems. Proceed 2nd International Workshop on Next Generation Information Technologies and Systems, Naharia, June 1995, pp. 116-123.

[6] S. Gatziu, K. Dittrich - Detecting composite events in active database systems using Petri Nets. IEEE RIDE'94.

[7] C. Kobryn: UML 2001: A Standardization Odyssey. CACM 42(10): 29-37 (1999)

[8] P. Osmon and P. Sleat: IDRIS: Interactive Design of Reactive Information Systems. CAiSE 1992: 494-506.

[9] J. Peckham and F. J. Maryanski: Semantic Data Models. Computing Surveys 20(3): 153-189 (1988).

[10] J. M. Smith and D. C. P. Smith: Database Abstractions: Aggregation and Generalization. TODS 2(2): 105-133 (1977).

[11] D. Tombros, A. Geppert, and K. R. Dittrich: Semantics of Reactive Components in Event-Driven Workflow Execution. CAiSE 1997: 409-422.