26

Real-time telecommunication network management: extending event correlation with temporal constraints

G. Jakobson, M. Weissman
GTE Laboratories Incorporated
40 Sylvan Rd, Waltham, MA 02254
tel: 1-617-466-2325, fax: 1-617-466-2960, email: gj00@gte.com

Abstract

Event correlation is becoming one of the most central techniques in managing the high volume of event messages. Practically, no network management system can ignore network surveillance and control procedures which are based on event correlation. The majority of existing network management systems use relatively simple *ad hoc* additions to their software to perform alarm correlation. In these systems, alarm correlation is handled as an aggregation procedure over sets of alarms exhibiting similar attributes. In recent years, several more sophisticated alarm correlation models have been proposed. In this paper, we will expand our knowledge-based event correlation model to capture temporal constraints.

Keywords

Real-time telecommunication network surveillance, temporal reasoning, event correlation, network fault propagation, knowledge-based systems

1 INTRODUCTION

Modern telecommunication networks may produce large numbers of alarms. It is not unusual that a burst of alarms during a major network failure may exhibit 40–50 alarms per second. This leads to serious difficulties in the network management process, particularly as follows:

- The inability to follow the stream of incoming events: alarms may pass unnoticed, or be noticed too late.
- The incorrect interpretation of groups of alarms: decision making and application of network controls is based on a single event rather on a macroscopic, generalized event level.
- The concentration of the operations staff on less important events.

A. S. Sethi et al. (eds.), *Integrated Network Management IV*

Event correlation is becoming one of the most central techniques in managing the high volume of event messages. Practically, no network management system can ignore network surveillance and control procedures which are based on event correlation. The majority of existing network management systems use relatively simple *ad hoc* additions to their software to perform alarm correlation. In these systems, alarm correlation is handled as an aggregation procedure over sets of alarms exhibiting similar attributes.

In recent years, several more sophisticated alarm correlation models have been proposed. In this paper, we will expand our knowledge-based event correlation model (Jakobson and Weissman, 1993) to capture temporal constraints.

2 EVENT CORRELATION DOMAIN

2.1 Definition

We define the task of event correlation as a conceptual interpretation procedure in the sense that a new meaning is assigned to a set of events that happen within a predefined time interval. As discussed in Section 2.3, the conceptual interpretation procedure could stretch from a trivial task of alarm compression to a complex pattern matching task. A typical event correlation is determined to be a dynamic pattern matching over a stream of network events. In addition, the correlation pattern may include network connectivity information, diagnostic test data, data from external databases, and any other information.

Events in the managed network are very diverse in nature. These events include raw event, status, and clear messages from network elements (NEs); events from mediation devices, subnetwork management systems, test systems and other equipment; user action messages from network operator terminals; and system interrupts. The origin of an event could be easily expanded beyond the "real" real-time events. For example, searching through records in a stored event log file, or any other file, could trigger the creation of an "alarm" if the value of some field in the record does not satisfy predefined constraints.

Applying event correlation rules may yield several results. First, a new event (message) may be sent to the operator's terminal. Second, an action may clear or resolve existing events (see Section 4.3). Third, a diagnostic message may be sent about faults occurring in the network. Fourth, a procedure may be called to access a database, run a diagnostic test procedure, generate a trouble ticket, or perform any other executable external procedure. Fifth, an internal system action to store data, change the system mode of operation, or perform any other internal system procedure may be taken. Any event generated as a result of event correlation will be considered a "regular" event coming from the network and, as such, is a subject for futher event correlation. Formally, nothing prevents us from considering the event correlation procedure as a network element which produces events (messages). The process of building correlations from correlations allows the formation of complex multilevel correlations

Networks function in discrete space and time. Even if some process, e.g., temperature in the network management center, takes its physical value in a continuum, it could be, in this particular domain, quantified and thresholded. The time model that we are using will be discrete time, with two modifications: point time and interval time. In point time, the events take place at time moments represented as integers at a predefined time scale (seconds, minutes, etc.). For example, the use of Universal Time (UT) requires that the date/time stamps attached to the event message be reduced to a numeric value in seconds or minutes. Point time is applied to most actions, such as user commands and system interrupts. In interval time, events are described by two time moments, the time of origination and the time of termination. Network events corresponding to NE faults, changes in the system behavior or changes in the state of the sensor or other control equipment, are usually described in interval time.

2.2 The role of event correlation in network management

Event correlation supports the following network management tasks:

- Reduction of the information load presented to the network operations staff by dynamic focus monitoring and context-sensitive event suppression (filtering).
- Increasing the semantic content of information presented to the network operations staff by aggregation and generalization of events.
- Real-time network fault isolation, causal fault diagnosis, and suggestion of corrective actions.
- Analysis of the ramification of events, prediction of network behavior, and trend analysis.
- Long-term correlation of historic event log files and network behavior trend analysis.

2.3 Event correlation types

Depending on the nature of the operations performed on events, we will consider the following types of event correlation:

1. $[a, a, ..., a] \Rightarrow a$	Compression
2. $[a, p(a) < H] \Rightarrow \varnothing$	Filtering
3. $[a, C)] \Rightarrow \varnothing$	Suppression
4. $[n \times a] \Rightarrow b$	Count
5. $[n \times a, p(a)] \Rightarrow a, p'(a), p' > p$	Escalation
6. $[a, a \subset b] \Rightarrow b$	Generalization
7. $a, a \supset b] \Rightarrow b$	Specialization
8. $[a\ T\ b] \Rightarrow c$	Temporal Relation
9. $[a, b, ...T, \wedge, \vee, \neg] \Rightarrow c$	Clustering

Event compression (1) is the task of reducing multiple occurrences of identical events into a single representative of the events. No number of occurrences of the event is taken into account. The meaning of the compression correlation is almost identical to the single event *a*, except that additional contextual information is assigned to the event to indicate that this event happened more than once.

Event (alarm) filtering (2) is the most widely used operation to reduce the number of alarms presented to the operator. If some parameter *p(a)* of alarm *a*, e.g., priority, type, location of the NE, time stamp, etc., does not fall into the set of predefined legitimate values H, then alarm *a* is simply discarded or sent into a log file. The decision to filter alarm *a* out or not is based solely on the specific characteristics of alarm *a*. In more sophisticated cases, set H could be dynamic and depend on user-specified criterion or criterion calculated by the system.

Event suppression (3) is a context-sensitive process in which event *a* is temporarily inhibited depending on the dynamic operational context C of the network management process. The context C is determined by the presence of other event(s), network management resources, management priorities, or other external requirements. The change in the operational context could lead to exhibition of the suppressed event. Temporary suppression of multiple events and control of the order of their exhibition is a basis for dynamic focus monitoring of the network management process.

Another type of correlation (4) results from counting and thresholding the number of repeated arrivals of identical events. Event escalation (5) assigns a higher value to some parameter *p'(a)* of event *a*, usually the severity, depending on the operational context, e.g., the number of occurrences of the event.

Event generalization (6) is a correlation in which event *a* is replaced by its super class *b*. Event generalization has a potentially high utility for network management. It allows one to

deviate from a low-level perspective of network events and view situations from a higher level. Event specialization (7) is an opposite procedure to event generalization. It substitutes an event with a more specific subclass of this event.

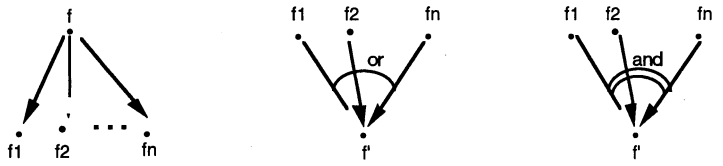
Temporal relations (8) T between events *a* and *b* allow them to be correlated depending on the order and time of their arrival. Different temporal relations for event correlation will be described in Section 5.

Event clustering (9) allows the creation of complex correlation patterns using logical operators $\wedge$ (and), $\vee$ (or), and $\neg$ (not) over component terms. The terms in the pattern could be primary network events, previously defined correlations, or tests of network connectivity.

3 REAL-TIME FAULT DIAGNOSIS USING EVENT CORRELATION

3.1 Network fault propagation

The original sources for most network events are physical faults occurring in the managed NEs. These faults can be causally related or not, i.e., they can be independent. Causal relations between faults can be represented by fault propagation rules (Figure 1).



(i) r :f -> f1, f2, ... fn (ii) r′ :f1∨ f2 ∨...∨ fn -> f′ (iii) r″:f1∧ f2 ∧...∧ fn -> f″

"->" - causal implication; "∨"-logical or; "∧" - logical and

Figure 1 Fault propagation rules.

Rule (i) defines fault f as a root cause for multiple faults f1, f2,..., fn. In rule (ii), fault f′ could be caused by any of the faults f1, f2,..., fn; while in rule (iii), all faults f1, f2,..., fn should be present in order to cause fault f′.

Composition of fault propagation rules forms an acyclic fault propagation graph where f′ from rule (ii) corresponds to the or-node, and f″ from (iii) corresponds to the and-node. A set of independent fault propagation graphs form the fault propagation model of the network.

A fuzzy fault propagation model could be constructed by supplying fault likelihood distribution for faults f1,..., fn in initial rules (i), and defining likelihood calculation algorithms for the logical or (ii) and the logical and (iii) nodes. Different fuzzy reasoning models could be used here, however this topic is beyond the scope of this paper.

Determination of fault propagation rules is a subject of domain knowledge acquisition. It is based on general principles of telecommunication systems, physical construction of NEs, and the behavior of the individual NEs and the whole network. Many fault propagation rules can be derived by examining the network configuration (connectivity) model. For example, knowing the nature of the faults f1–f4, and the fact that NEs NE1 and NE3, and NE2 and NE3 are connected (Figure 2), one may derive causal propagation rules f1 –> f3, f4 and f1 ∨ f2 –> f4.

3.2 Heuristics of fault detection

Not all faults have events associated with them. Such faults can be recognized indirectly by correlating available events. Let's consider a simple fault propagation model, shown in Figure 3.

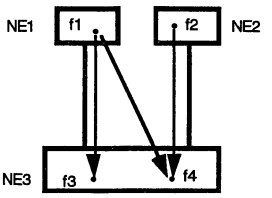


Figure 2 Network fault propagation.

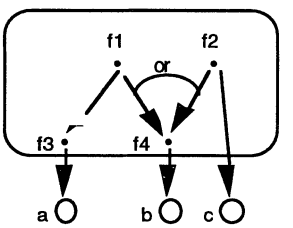


Figure 3 Network diagnostic using alarm correlation.

Fault f3 is caused by fault f1, while f4 could be caused by f1 or f2 or by both of them. Fault f3 is exhibited by alarm a, f4 by alarm b, and f2 by alarm c. Faults f3 and f4 may also happen independently of faults f1 and f2. For example, if alarm *a* was generated, the reason could be directly fault f3 (no presence of the fault f1), or the reason could be fault f1, which consequently caused fault f3. By correlating alarms into simple Boolean patterns, one can construct the following fault diagnostic rules:

Rule 1: if (not a) and b and c then	f1	— definitely no
	f2	— definitely yes
	f3	— definitely no
	f4	— unlikely a root cause

The fact that alarm *a* is not present allows us to conclude that fault f3 and, consequently, fault f1 didn't happen. Obviously, the fault should happen, because it is the sole reason for alarm c. Generally, alarm b could be caused either by fault f4 as a consequence of faults f1 or f2, or by fault f4 as an independent root cause. In our example, fault f1 didn't happen, so alarm c could be potentially caused by fault f4 as the root cause or as a fault caused by f2. The presence of fault f2 definitely caused f4, and it is unlikely that f4 happened simultaneously as a root cause and as a fault caused by f2.

Rule 2: if *a* and b and not c then	f1	— likely
	f2	— definitely no
	f3	— unlikely a root cause
	f4	— unlikely a root cause
Rule 3: if *a* and b and c then	f1	— likely
	f2	— definitely yes
	f3	— likely
	f4	— unlikely a root cause
	(f1, f3)	— unlikely together
Rule 4: if *a* and not b and c then	"Error in alarm message processing"	

4 TIME-DEPENDENT EVENT MANAGEMENT

4.1 Events

Formally, an event is a pair (preposition, time quantifier), in which preposition describes the content of the event, and time quantifier is a moment in the point time, or a time interval of duration of the event in the interval time. Without losing generality, we will refer to prepositions as messages. (Strictly speaking, a preposition is a formal representation of a message obtained after parsing the message.) Further in the paper, we will use the following notation:

event = (message, time quantifier)

time quantifier = [t1, t2]
t1 — time of origination
t2 — time of termination

The origination time of the event is issued by the NE or its management system. The event message sent to the event list for display at an operator terminal stays there until it is cleared by the network management system or by the operator. The event will be ultimately eliminated from the event list either by clearing or expiration of the lifespan, whichever comes first.

In addition to the event clearing procedures, an event can "die by a natural cause," i.e., when the event expiration time is over. Event expiration time is determined by the lifespan of the event, *a* potential maximum duration of the event. The lifespan is assigned duration based on event class, and depends on the practices and policies of the particular network management domain.

For many NEs, the events (alarms) are issued pair-wise — the original event message manifesting a beginning of some physical phenomenon, e.g., a fault, and a complimentary clear message manifesting the end of the phenomenon. After origination, these two logically inverse messages may exist together, unless a clear command to remove the first message is issued by the network operator. In network management systems that support logical reasoning and event correlation, the logically inverse messages should be detected and resolved automatically.

4.2 Correlation window

Each event correlation process has an assigned correlation time window (Figure 4), a maximum time interval during which the component events should happen. The correlation process will be started at the time of arrival of the first component event (event *a*) and stopped as the last component event (event *c*) arrives. As any other event, correlation has its time of origination, time of termination, and lifespan. By definition, the time of origination of the correlation is equal to the time of origination of the last component event.

Event correlation is a dynamic process so that the arrival of any component event instantiates a new correlation time window for some correlation. This means that the correlation time window for some correlation slides in time to capture new options to instantiate a correlation. However, if temporal constraints are assigned to the component events, e.g., event *b* should be always after event *a*, no correlation time window is started when event b arrives.

Determining the length of the correlation window and the lifespan of an event (correlation) directly affects the potentials of creating correlations. Widening the correlation window and increasing the lifespans increases the chance of creating a correlation. For very fast processes, e.g., a burst of alarms during T3 trunk failure, the width of the correlation window could be seconds, while for slow processes, such as analyzing a trend of failures from an alarm log file, the correlation window may be several hours, or even several days, long. The same is true for the lifespan: informative events could last several seconds, while the lifespan of critical events should be indefinite, i.e., these events should be always cleared by the operator or by the system.

The right value for the correlation window and the lifespan will emerge from the practice of managing a specific network.

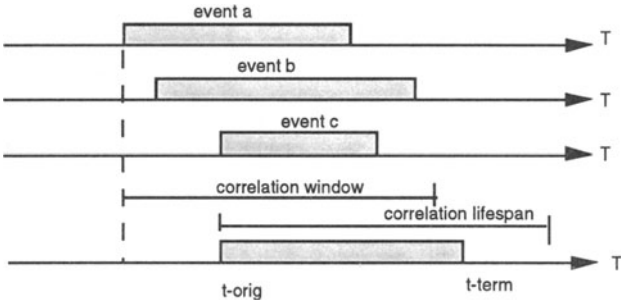


Figure 4 Correlation time window and lifespan.

4.3 Dynamic event memory

Each event, either originated in the managed network or produced by the event correlation process, will be placed into Dynamic Event Memory. Events residing in Dynamic Event Memory are available for the correlation processes. An event is removed from Dynamic Event Memory if one of the following happens:

- Command Resolve is issued by the operator or by the system.
- The lifespan of the event is over.

Command Resolve effectively "kills" the event, preventing its future use. This command should be handled carefully by the operator and the system.

As events are originated, they are also placed into the Event List. The Event List keeps events only for display purposes. An event is removed from the Event List under the following circumstances:

- Command Clear is issued by the operator or by the system.
- Command Resolve is issued by the operator or by the system.
- A new correlation is originated which contains this event as its component.
- The lifespan of the event is over.

The pragmatics of the Clear action is to reduce the number of displayed event messages, while at the same time leaving the events in Dynamic Event Memory for potential future correlations.

4.4 Predictable run-time behavior

Predicting the worst-case run-time behavior is a prerequisite for real-time network event surveillance and fault management systems. The system must guarantee network event collection, parsing, correlation, fault diagnosis, display, and execution of corrective actions within the time limits set by the specific network operational standards. Slow transmission and processing of network messages may result in distortion (masking) of the real sequence of actual physical events and result in incorrect correlations. Two definitions of the predictable execution time exist:

- Event processing must complete before the new event, i.e., $\partial < t$, where ∂ is the maximum event processing time and t is the minimum time interval between synchronous or asynchronous events.
- Event processing must complete during a predefined time interval, i.e., $\partial < D < t$, where D — predefined event processing time.

The value of D could be in the order of tenth or hundredth of seconds during the peak of alarm bursts. For example, a cellular switch supporting a region with 30–40 cell sites produces normally 2–3 alarms per minute. A medium-size wireline network with 3–4 large Class 5 switches and 8–10 digital cross-connects may produce during a major T1/T3 trunk failure tens of alarms per second. Collecting and parsing these alarms should be very fast. It is not unusual that even very fast network management platforms with clock speed of 150 MHz or higher need event buffering for correlating bursts of alarms.

5 TEMPORAL REASONING FOR EVENT CORRELATION

5.1 Issues

Temporal reasoning, reasoning about time, plays a critical role in monitoring network events. The system should be able to reason about the relative and absolute times of occurrence of events, duration of events (or duration of the absence of events), and sequence of events. The time interval between events can be defined on a quantitative time scale or on a qualitative time scale.

5.2 Temporal relations

In this section we will examine temporal relations that will be used to build time-dependent event correlations between events. Temporal relations together with temporal reasoning rules form temporal event calculus (Allen, 1983).

For the specific tasks of network event correlation, we will define the following set of temporal relations. Let e1 and e2 be two events defined on an interval time: e1 = (msg1, [t1, t1′]) and e2 = (msg2, [t2, t2′).

1. Event e2 by an interval h starts after event e1.
$$e2\ \text{AFTER}(h)\ e1 <=> t2 > t1 + h$$
2. Event e2 by an interval h follows event e1.
$$e2\ \text{FOLLOWS}(h)\ e1 <=> t2 \geq t1' + h$$

From the definitions (1) and (2) follows that
$$\text{if } e2\ \text{FOLLOWS}(h)\ e1 \text{ then } e2\ \text{AFTER}(d + h)\ e1,$$
where d is the duration of the event e1.

3. Event e2 by an interval h ends before event e1 (ends).
$$e2\ \text{BEFORE}(h)\ e1 <=> t1' \geq t2' + h$$

Note that relations (starts) AFTER and (ends) BEFORE are not logically inverse.

4. Event e2 by an interval h precedes event e1.
$$e2\ \text{PRECEDES}(h)\ e1 <=> t2 \geq t1' + h$$

The following statement is true:
$$e2\ \text{FOLLOWS}(h)\ e1 <=> e1\ \text{PRECEDES}(h)\ e2$$

5. Event e2 happens during event e1.

$$e2 \text{ DURING } e1 <=> t2 \geq t1 \text{ and } t1' \geq t2'$$

The following derivation rule holds between DURING, BEFORE, and AFTER:

If e2 DURING e1, then e2 AFTER e1 and e2 BEFORE e1 (and *vice versa*).

6. Event e1 starts at the same time as event e2.

$$e1 \text{ STARTS } e2 <=> t1 = t2$$

Obviously the following rule holds:

If e2 AFTER(h) e1 and e1 AFTER(h) e2, then e1 STARTS e2 (and *vice versa*).

7. Event e1 finishes at the same time as event e2.

$$e1 \text{ FINISHES } e2 <=> t1' = t2'$$

Similarly, as for the previous case, the following rule holds:

If e2 BEFORE(h) e1 and e1 BEFORE(h) e2, then e1 FINISHES e2 (and *vice versa*).

8. Event e1 coincides with event e2.

$$e2 \text{ COINCIDES with } e1 <=> t2 = t1 \text{ and } t1' = t2'$$

As a consequence of the definition of the coinciding events, the following is true:

If e2 COINCIDES with e1, then e2 STARTS e1 and e2 FINISHES e1 (and *vice versa*).

If e2 DURING e1 and e1 DURING e2, then e2 COINCIDES with e1 (and *vice versa*).

9. Event e1 overlaps with event e2.

$$e1 \text{ OVERLAPS } e2 <=> t2' \geq t1' > t2 \geq t1$$

From the definition OVERLAPS, it follows that

If e1 OVERLAPS e2, then e2 AFTER(h) e1 and e1 BEFORE(h) e2.

Regarding the algebraic properties of the temporal relations, we can say that all of them are transitive, except OVERLAPS, while STARTS, FINISHES, and COINCIDES are also symmetric relations.

6 KNOWLEDGE FRAMEWORK FOR EVENT CORRELATION

6.1 Model-based approach

Our approach to event correlation uses the principles on model-based reasoning originally proposed in (Davis, Shrobe, and Hamscher, 1982) for troubleshooting electronic circuit boards. The idea of the model-based approach is to reason about the system from representation of its structure and functional behavior. We will extend this model into real-time event correlation.

The structural representation means the description of the NEs and the topology of the network. Under the topology, we understand not only the connectivity between NEs but also the containment relations between the elements.

The behavioral representation describes the dynamic processes of event propagation and correlation. These processes are described using correlation rules. Each rule activates a new event (correlation), which in its turn may be used in the firing condition of the next correlation rule. In the following sections, we describe the components of the overall event correlation model: the network configuration model, event correlations, and correlation rules.

6.2 Network configuration model

Networks are composed of NEs. Traditional examples of NEs are switches, digital cross-connect systems, channel service units, trunks, routers, bridges, etc.

In a broader sense, a NE is any real or virtual hardware or software entity that composes the telecommunication network or the surrounding environment. The network itself can be considered a NE, e.g., at a certain level of abstraction, a local area network could be considered a NE of a regional network. Following the given definition of the NE, a virtual private network

overlaid on a physical public network could be considered a NE, or a cell site of a cellular network is a NE, or an amplifier in a power supply unit is a NE, etc. All NEs working together (whether physically connected or not, contained one in another or not) form the network configuration model.

Each particular NE is described by its model, which is instantiated from the corresponding NE class model. Network element classes (models) form a class-subclass hierarchy. All NE classes, except the terminal classes, are mathematical abstractions of existing "real" NEs, while the terminal classes describe the types of existing NEs.

Following the inheritance paths in the class hierarchy, the constraints, attribute values, and default values of a class (parent) will be passed to its subclasses (children). There are two types of built-in constraint types in the classes: connectivity constraints and containment constraints. On the NE class level, the connectivity constraints will determine the possible connections between the NEs, while the containment constraints define the possible containment relations between the NEs. These constraints, originally defined by the domain expert, will be passed to the terminal classes of the hierarchy, and then enforced during instantiation of a NE model corresponding to the physical NE. For example, if a switch type A can be connected only to a digital cross-connect type B, then this constraint is enforced when a particular network connectivity model is constructed.

6.3 Correlations and correlation rules

On a phenomenological level, a correlation is a statement about the events happening on the network, e.g., Bad-Card-Correlation states that some port contains a faulty circuit card. On a system internal level, a correlation is an object-oriented data structure that contains its component objects and attributes. All correlations are organized into a correlation class hierarchy. The root node of the correlation class hierarchy describes the basic correlation class. The terminal nodes represent correlation classes, which will be instantiated each time particular conditions are met in the stream of incoming events.

A correlation rule defines the conditions under which correlations are asserted, e.g., if there is a red carrier group event (CGA) from a digital cross-connect (DCS), and there is a yellow CGA from another DCS, and these DCSs are connected, then Bad-Card-Correlation should be asserted. Different correlation rules can lead to the assertion of one and the same correlation.

The conditional, or so-called left-hand side (LHS), part of a correlation rule uses NEs, messages, and correlations as arguments to form the rule-firing condition. The condition can contain Boolean patterns, sequences of events based on time relations, as well as event counters. The arguments for the Boolean patterns could be the following entities:

- Parameters of event messages, e.g., alarm severity level
- Event message class types, e.g., DS0 class alarm message
- Parameters of NEs, e.g., location code
- NE class types, e.g., Class 5 switch
- Connectivity and containment statements between NEs
- Temporal relations between events
- Correlations

The subsequent application of correlation rules, instantiation of correlations, and consumption of the produced correlations by the next rule describes the event propagation process.

Figure 5 illustrates how correlations and correlation rules could be described. Let's consider the following sample situation which should be detected and reported at the operator's terminal:

A carrier group alarm type "A" happened at a time ?t1 on some NE named ?ne, and during the following 1-minute interval an expected carrier group alarm type "B" did not occur at the same NE.

The events to be correlated are alarm A (?msg1) and not alarm B (?msg2). The fact that event B did not happen is formally also an event. The additional constraints are that (1) a simple network configuration constraint that both messages are coming from the same network element ?ne, and (2) a temporal constraint that the event "not alarm B" came 60 seconds later than alarm A. The first constraint is achieved by using the same reference to the network element ?ne in both messages, while the second constraint is implemented using temporal relation AFTER.

```
Rule Name: EXPECTED-EVENT-RULE
Conditions
     MSG:    ALARM-TYPE -A              ?msg1
             NE                         ?ne
     not
             MSG:ALARM-TYPE-B           ?msg2
             NE                         ?ne
     after   TIMESENT ?t                ?msg1 ?msg2 60
Actions
     Assert: EXPECTED-EVENT-CORR
                  MSG: ALARM    ?msg1

Correlation Name: EXPECTED-EVENT-CORRELATION
Lifespan 120 minutes
Requires
     MSG:    ALARM
             INSTANCE                   ?ne
             TIMESENT                   ?t
Parents: BASIC-CORRELATION
Children:
Template:    ?t  ?ne Expected event type "B" did not happen
             during 1 minute after the alarm type "A"
Slots
     Slot: INSTANCE                     Value: ?ne
     Slot: TIMESENT                     Value: ?t
```

Figure 5 Correlation and correlation rule for an expected event situation.

If the logical condition of the rule is true for certain events in the Dynamic Event Memory, the correlation EXPECTED-EVENT-CORRELATION is asserted to the memory and a message is sent to the operator terminal. Variable ?msg1, binding all information about the ALARM-TYPE-A message, is sent from the rule condition part to the correlation asserted in the action part. The correlation has built-in slots (parameters) to store information that could be passed to the higher level correlations that use this correlation as a component. As with NEs, correlations are organized into class hierarchies. The class references are implemented through Parents and Children relationships. The EXPECTED-EVENT-CORRELATION has one parent, BASIC-CORRELATION, and no child correlations.

7 IMPACT

The event correlation model described in the previous sections is implemented in IMPACT, a general-purpose telecommunication network alarm correlation system (Jakobson and Weissman, 1993; Jakobson, Weihmayer, and Weissman, 1994). As an example of a specific implementation of the correlations discussed in Section 2.3, we will refer to the event counting correlation. There are two operators in IMPACT that are used for counting events: Timespan and Count. The operator Timespan takes as an input an event correlation pattern and a time interval and returns the count of how many times the event pattern happened during the time interval. The function of the Count operator is opposite to Timespan: It takes as an input an event correlation pattern and a given number of event counts, and returns the time interval needed to count the pattern.

IMPACT contains three major components: Application Run-Time Component, Application Development Environment, and the Network Knowledge Base. The Application Run-Time Component monitors the network events in real-time. It performs the following functions: (1) alarm message collection and parsing, (2) event correlation, and (3) execution of external procedures (test, database access, message logging, etc.). The Application Development Environment provides powerful tools for building the Network Knowledge Base. The core of the environement consists of eight editors, with a common look and feel, which are grouped into three sets of tools: Network Configuration Tools, Alarm Correlation Tools, and Network Graphics Tools.

IMPACT has been implemented using the CLIPS expert system shell (Giarratano, 1993). The graphical user interface is programmed in Tcl/TK (Ousterhout, 1990). Many time-critical functions are written in C. The system runs on various UNIX workstations, and it is integrated with two GTE network alarm management systems, SmartAlert and ISM2000. IMPACT is currently used for a land-based telecommunication network and for cellular network alarm correlation, fault diagnostics, and calling card abuse monitoring.

TRADEMARKS

UNIX is a trademark of UNIX Systems Laboratories
SmartAlert is a trademark of GTE TSI, ISM/2000 is a trademark of GTE NMO

ACKNOWLEDGEMENTS

We thank network management personnel from GTE Mobilnet, GTE NMO, and GTE TSI for valuable domain knowledge and feedback, and Dr. S. Goyal for his continuous encouragement and support. Our thanks go also to an anonynous reviewer for many useful comments and suggestions.

REFERENCES

Allen, J.F. (1983) Maintaining knowledge about temporal intervals. *Communications of the ACM*, pp. 832-853

Davis, R., Shrobe, H., and Hamscher, W. (1982) Diagnosis based on description of structure and function. Proceedings of the 1982 National Conference on Artificial Intelligence, Pittsburgh, PA, pp. 137-142

Giarratano, J. (1993) CLIPS user's guide. NASA LBJ Space Center, Software Technology Branch.

Jakobson, G., M. and Weissman (1003) Alarm Correlation. *IEEE Network*, 7 (6), pp. 52-59.

Jakobson, G., Weihmayer, R., and Weissman, M. (1994) A domain-oriented expert system shell for telecommunication network alarm correlation. In *Network Management and Control*, Volume II, (editor M. Malek), Plenum Press, New York, NY.

Ousterhout, J. (1990) Tcl: An embeddable command language, Proceedings of the Winter USENIX Conference, pp. 133-146.

BIOGRAPHIES

Gabriel Jakobson is a Principal Member of Technical Staff at GTE Laboratories, where he has been project leader of several expert systems, intelligent database, and telecommunication network management systems development projects. He received M.S.E.E. from the Tallinn Polytechnic Institute and Ph.D. in CS from Estonian Academy of Sciences in 1964 and 1971, respectively. Dr. Jakobson is the author or co-author of more than 40 technical papers in the areas of databases, man-machine interfaces, expert systems, and telecommunication network management.

Mark D. Weissman received his BS in Chemical Engineering and his BA in Computer Science from the State University of New York at Buffalo in 1983 and 1984, respectively. He is a Senior Member of Technical Staff at GTE Laboratories, where he has been a major contributor to the development of several expert systems for network management applications.