Temporal Aspects in Diagnosis Validation

Manuela Popescu*, Pascal Lorenz**, Marc Gilg**, Jean Marc Nicod*
*) University of Besançon, France
**) University of Haute Alsace, France
manuela.popescu@univ-fcomte.fr, lorenz@ieee.org, marc.gilg@uha.fr, jean-marc.nicod@lifc.univ-fcomte.fr

Abstract— **In this article, we introduce temporal aspects related to diagnosis validation. Event correlation and action triggering are essential for an accurate diagnosis decision. There are several time-related challenges referring to event timestamps, timely event correlations, and timely corrective actions, in both absolute time (precise moment), or relative time (between events, actions, and events and actions). We propose here a new timestamp approach and we consider a series of temporal operators defining the event relative temporal position that allows a more fine grain interpretation of the system behavior. A combination of proposed mechanisms is used to complete the main functions of a diagnosis engine.**

Keywords- diagnosis validation; timestamps ; temporal features; temporal actions; temporal logics.

I. INTRODUCTION

The complexity of networks and distributed systems gives rise to management challenges when unexpected situations occur. There is an overwhelming number of feedback events coming from the system in the form of status reports towards the monitoring and management applications and human operators. Actually, very few of these events, less than 10%, can be considered for potential status understanding and remedy. Given the numbers, it is inevitable that many relevant events are dropped. The remedy actions can come too late (and sometimes be useless). There are numerous management applications in commercial use. However, the variety of the systems to be managed, their complexity, and the fact that most of the successful decisions are rarely recorded, rise serious challenges in the ability to accurately handle unexpected situations.

Some of the multiple causes leading to the current state are (i) lack of successful validation of corrective actions, (ii) heterogeneity of the events to be handled, and (iii) incomplete correlation and time synchronization between status reports, decision processing and corrective actions.

To address the lack of successful validation of corrective actions, two loops of the diagnosis process were identified in [1]: (a) one loop deals with measuring the system parameters (system state, events, *i.e.*, pre-conditions) and takes the most suitable actions; this was referred to as the *diagnosis loop* (b) a second loop deals with validating that the corrective actions were indeed successful; this was referred to as the *validation loop*. The main goals of the validation loop were (a) to establish the new state of the system, *i.e.*, post-conditions and (b) to gather knowledge on how to solve future similar situations, in case the actions taken were considered successful. In addition, through the concept of *Quality of Diagnosis (QoD)* introduced into the validation loop, the accuracy of the corrective actions and their use in similar situations were enhanced.

A step towards automated diagnosis was introduced in [2], where an event ontology and a progressive diagnosis ontology were proposed. Event dependencies captured by ontology and specific event relations have been formalized. Probable cause and recommended actions were associated with events. Additionally, an augmented specification for actions was proposed to help the validation loop. Both proposals had as a target the reuse of knowledge for problem fixing, identification of recommended diagnosis actions, and validation of successful actions.

The third identified challenge is time-related; this refers to event timestamps, timely event correlations, and timely corrective actions, in both absolute time (precise moment), or relative time (between events, actions, and events and actions). This aspect is more difficult, as many events issued at different timestamps might be processed for event compression/aggregation. The correct adoption of temporal aspects can solve potential conflicts among the post-conditions of the actions already validated as “successful” and helps evaluate the accuracy of the diagnosis actions (preciseness versus permanent damage).

In this paper, we highlight the relevance of temporal aspects, identify the challenging issues, and propose a new timestamp approach. We consider a series of temporal operators defining event relative temporal position that allows a more fine grain interpretation of the system behavior. A combination of proposed mechanisms is used to complete the main functions of a diagnosis engine.

The article has the following structure: Section II presents the state of the art with respect to temporal considerations. In Section III, we talk about approaching temporal aspects. Section IV describes the use of temporal aspects for diagnosis. Section V presents the conclusion and future work.

 ISBN: 978-1-61208-104-5

II. STATE OF THE ART

Temporal features are related to several generic aspects concerning (i) inaccurate (wrong, un-synchronized, or missing) clocks, (ii) loss of events, and (iii) hierarchical event processing at layers exposing different clocks. These are somehow related to event propagation skew but also to different syntactic and semantic implementation decisions of the timestamps (including time zones). One approach in dealing with real-time measurements of propagation skew uses a statistical evaluation to update the timer values [6].

Some diagnostic constraints might be temporal. In [2], temporal constraints are used for event tags to define the event ontology and to detect the relative temporal constraints. Walzer *et al.* use specific operators for time-intervals with quantitative constraints in rule-based systems to trigger certain actions [7]. In the following sections, we present the main approaches used to specify temporal aspects on events and actions.

A. Temporal aspects for events

Timestamps are usually carried by the events themselves; basic events possess special timestamp fields that are instantiated when an event instance occurs. Timestamps are storing time in the native format of the platform in which the event processing runs. There are two standard ways to represent the time: (i) using the universal time, or (ii) using time zones. Since one still needs to preserve the zone indication for a device for hourly performance reports, the representation in the universal time is only for the computational point of view. Another standard way to represent the time is the UNIX-format time as a four-byte integer that represents the seconds elapsed since January 1, 1970. For the same reasons, the time zone of the source device should be stored.

An event might have multiple timestamps; the source timestamp (not always present), the logging host timestamp, the console timestamp, and the processing timestamp. Temporal correlation and event aggregation should consider all these timestamps.

Event processing and correlation need a time-based logic to express the relative position of start / end /duration of the events [3]. While attempts were identified for classifying the relative position of the events, no particular commercial solutions are known where a full range of temporal situations are used.

B. Temporal aspects for actions

An enhanced action model was proposed in [2]. One temporal aspect is related to the triggering condition (guard). Others temporal aspects are related to the temporal dependencies between actions, *i.e.,* some action must start at a given period after one action was triggered or was deemed successfully finished.

A diagnosis-oriented augmented action definition was introduced in [2], as follows.

action::= <<guard><ID><post-conditions>
<mode><conflicting>,
where

ID::= READ | WRITE | DELETE | CHANGE | , etc.

mode ::= <potential | recommended | successful <context>>,
with
potential: any diagnosis action that is designated as being related to a potential domain
recommended: any potential action that is perceived as solving a given problem, eventually based on a diagnoses history
successful: when post-conditions were validated as true
context: <d:D, c:C>
d:D is d instance of Domain
c:C is c instance of Cloud

Also in [2], we associated the notion of "conflicting" with a given *action*, which designates the actions a potential action is in conflict with, in a given domain:

conflicting ::= <a_1, a_2,… a_k | a_i:A>

A <guard> is acting as pre-conditions and igniter (initial timestamp), and the <post-conditions> are expected to be true (after the action is considered successfully performed). In general, actions are applied following a simple rule:

IF <pre-conditions>
THEN <action> WITH <post-conditions>

Post-conditions are assumed to hold. A composition of actions, a plan, is a set of related actions and it is used to specify dependencies between actions. This is schematically represented in Figure 1. The model can be summarized as follows, where a plan is introduced as a temporal combination of atomic actions (see ID above) [8].

policy::= IF <pre-cond> THEN {<> 1<action> 1<plan>}
[ELSE {<> 1<action> 1<plan>} <action> 1<plan>}]
«post-cond>]

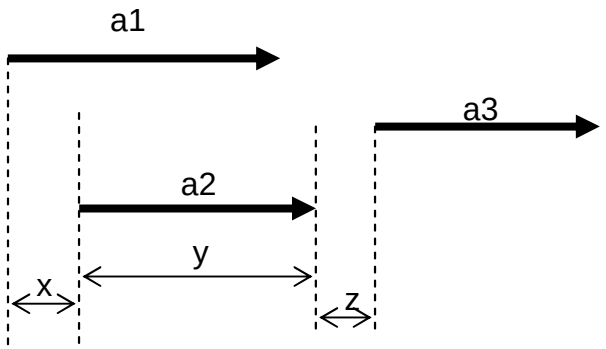

Figure 1. A plan – actions: a_1, a_2, a_3; time durations: x, y, z

Based on the analysis of the state of the art, we conclude that there is a need for a unified timestamps approach and a set of operators that must be used in synchronism to express the dependency between events, between actions, or between events and actions [4][5].

In the following sections, we propose a representation of temporal features allowing various semantics used to correlate the events and the actions.

III. APPROACHING TEMPORAL ASPECTS

This section describes aspects related to timestamps, event correlation with temporal operators and gives an example of use of temporal operators.

A. Timestamps

In a hierarchical model, an event model should allow multiple timestamps, depending on the event hosting and processing. In an XML-like specification, we introduce for the device (source), host (server), and processing application (management application or console), the timestamp and the time zone a source, host or processing application belongs to.

TABLE I: Timestamp specification

<time>
<device_time> device_time></device_time>
<device_zone> device_time_zone</device_zone>
<server_time> server_time</server_time>
<server_zone> server_time_zone</server_zone>
< processor_time> event_processor_time</ processor_time>
<processor_zone> processor_time_zone</processor_zone>
</time>

The timestamp of the event is best set by the event producer (*device_time*). The timestamp representing the moment of event registration on the server, *server_time* is of relevance for correlation. Finally, the timestamp of the entity performing correlation or event processing is relevant for synchronization among multiple such event processing systems.

Any of these three entities can belong to different time zones that should be considered when temporal priorities count.

The values of these parameters are set by various entities. Some protocols provide the capability to supply the time in the occurred event, or the time when the event producer sent the event. With the Network Time Protocol (NTP) the time from event producers will be the most accurate. Alternatively, the time registered by the event processing system might be considered.

We advocate the following representation, similar to Syslog protocol, *e.g., device_time: Jan 1 14:22:45* represents the local time on the device at the time the message is signed. For devices with no clocks, *device_time: Jan 1 00:00:00* should be the representation.

B. Event correlation with temporal operators

Temporal relations are used to build time-dependent event correlations between events. For instance, we may correlate the alarms that happened within the same 10-minutes period, which means the correlation window is 10 minutes. We abstract an event and consider only the temporal aspects.

Let ***e1*** and ***e2*** be two events defined on a time interval:

$T_1 = [t_1, t_1']$
$T_2 = [t_2, t_2']$
and e_1 within T_1
e_2 within T_2

two events occurring within the time intervals T_1 and T_2, respectively.

The following temporal relations R(t) or R are identified:

R(t):: = {**after**(t), **follows**(t), **before**(t), **precedes(**t)}

R ::= {**during**, **starts**, **finishes**, **coincides**, **overlaps**}

The following deductions hold:

after: e_2 ***after(t)*** $e_1 \Leftrightarrow t_2 > t_1 + t$

follows: e_2 ***follows(t)*** $e_1 \Leftrightarrow t_2 \geq t_1' + t$

before: e_2 ***before(t)*** $e_1 \Leftrightarrow t_1' \geq t_2' + t$

 ISBN: 978-1-61208-104-5

precedes: e_2 **precedes(t)** $e_1 \Leftrightarrow t_1 \geq t_2'+t$

during: e_2 ***during*** $e_1 \Leftrightarrow t_2 \geq t_1$ *and* $t_1' \geq t_2''$

starts: e_1 ***starts*** $e_2 \Leftrightarrow t_1 = t_2$

finishes: e_1 ***finishes*** $e_2 \Leftrightarrow t_1' = t_2'$

coincides : e_2 ***coincides*** *with* $e_1 \Leftrightarrow t_2 = t_1$ *and* $t_1' = t_2'$

overlaps: e_1 ***overlaps(ε)*** $e_2 \Leftrightarrow t_2' \geq t_1' \pm \varepsilon > t_2 \geq t_1 \pm \varepsilon$
where ε is the accepted threshold for measurement variation.

With respect to the algebraic properties of the temporal relations,

- all are transitive, except **overlaps**,
- **starts**, **finishes**, **conincides** are also symmetric relations.

C. Example of using temporal operators

In [1], time-oriented diagnosis was defined as

$$[e_1, e_2, e_3 \ldots e_n]t_1 \rightarrow \{p_i\}t_1 \rightarrow \{d_i\}t_1,$$

where

p_i, d_i, and e_i represent a given instance of a problem, diagnosis, and event, respectively.

As an example, let us consider the instantiation:

$$\{[e_1, e_2, e_3] \mid e_2 \textbf{ follows}(x)\ e_1 \ \&\ e_2 \textbf{ overlaps}(\varepsilon)\ e_3\} \rightarrow p_{123} \rightarrow d_{123}$$

where x is the time duration between e_1 and e_2.

As a note,

$$\{[e_1, e_2, e_3] \mid e_2 \textbf{ precedes}(x)\ e_1 \ \&\ e_2 \textbf{ overlaps}(\varepsilon)\ e_3\} \rightarrow p'_{123} \rightarrow d'_{123}$$

represents a different problem and therefore, a different diagnosis.

In the case that the above specification designates a given diagnosis and it is determined that e_1 did not follow e_2 after time x, a diagnosis engine issues an anomaly (no concrete diagnosis is derived).

An event has a series of event attributes, which we represent as:

$e = (f_1, f_2, f_3 \ldots, f_n)$
where f: $(value:V)$,
where V is the type of the attribute

Examples of event attributes we consider are:

f_1: *ID*
f_2: *source*
f_3: *timestamp*
f_4: *timezone*
f_5:*English text defining the potential cause*
etc

$e.f_3$ represents the value of attribute f_3 in event e.

The operators on relative event position (**follows**, **overlaps**, etc.) are related to the attributes f_3 and f_4.

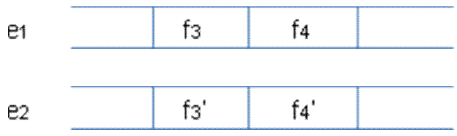

Figure 2.Timestamp and timezone event fields

In this example, $e_1.f_4$ and $e_2.f_4$' are known, since they represent the timezones of the sources of the two events. Only $e_1.f_3$ and $e_2.f_3$' need to be set by the local clocks. Let us assume that:

clk_1 sets $e_1.f_3$ and clk_2 sets $e_2.f_3$',
where clk is the local clock of the event source.

$|clk_1 - clk_2| \leq \varepsilon_{12}$,
where ε_{12} is the clock skew between the two local clocks for two domains represented by two semantic clouds [2].

e_2 ***follows***(x) e_1 is computed as follows:

$$(e_1.f_3 + \varepsilon_{12}) + x < e_2.f_3 \quad \text{(for the same time zone)} \qquad (1)$$

For different time zones, this becomes:

$$[(e_1.f_3 + \varepsilon_{12}) \blacksquare Abs(e_1.f_4)] + x < (e_2.f_3) \blacksquare Abs(e_2.f_4), \qquad (2)$$

where $\blacksquare$ $Abs(e.f_4)$ represents the operator for normalizing the time between timezones.

Following the same logic, e_2 ***overlaps(ε)*** e_3 for different time zones is computed as follows:

$$|(e_2.f_3) \blacksquare Abs(e_2.f_4) - (e_3.f_3) \blacksquare Abs(e_3.f_4)| < \varepsilon_{23} \qquad (3)$$

where
|x| is the absolute value of x
and
ε_{23} represents an acceptable error.

These event-based computations are performed each time a diagnosis is triggered and validated.

In the next section we will use this example in the diagnosis scenario.

 ISBN: 978-1-61208-104-5

IV. USING TEMPORAL FEATURES FOR DIAGNOSIS

This section presents a formal specification of the ontology-based diagnosis, considering temporal relations. Let us assume that the diagnosis engine and the Quality of Diagnosis (QoD) engine introduced in [1] have to trigger the following operations: INTERPRET, APPLY, VALIDATE and MARK.

- Diagnosis engine: INTERPRET events from the system.
- Diagnosis engine: APPLY the diagnosis actions.
- Quality of Diagnosis engine:
  VALIDATE the diagnosis actions.
  and
  MARK successful actions.

The APPLY, VALIDATE and MARK functions were shown in [2]. We reconsider the example with INTERPRET functionality as well.

As discussed in [2], there is a semantic tag hierarchy within each domain, with special dependency relations between semantic tags. Within a domain, semantic tags and their relations form a semantic tag cloud; a domain might have multiple semantic tag clouds associated with it. Let us assume that a system is represented by two semantic tag clouds (Figure 3). Semantic cloud #1 defines the tags and their relationships for a fault related to a power supply while Semantic cloud #2 relates to a potentially real-time and latent fault.

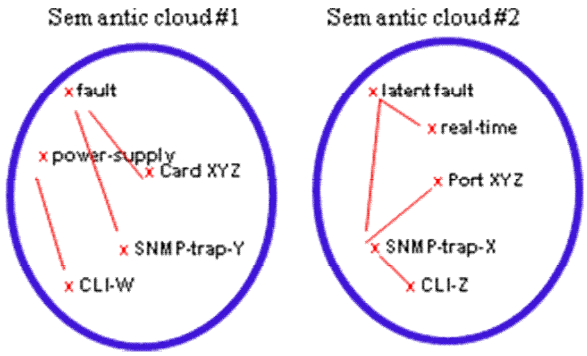

Figure 3. Two Semantic Tag Clouds [2]

When some event patterns occur and diagnosis actions must be triggered (and validated), the Diagnosis Engine interprets the events from the system and applies the diagnosis actions. Next, the Quality of Diagnosis engine validates the actions and marks the successful actions.

The following algorithm is used by the engines to perform the required actions for a given occurrence of combinations of events. A particular series of events occurs as shown in the INTERPRET part of the following algorithm (we use the '.' Notation, i.e., *a.b* means the property '*b*' of the instance '*a*'). When the conditions (2) and (3) explained in Section III hold, the necessary condition to enter the rest of the algorithm is met.

START

INTERPRET
IF {[e_1, e_2, e_3] | e_2 precedes(x) e_1 & e_2 overlaps e_3}
  CLOCK = t_0
  AND e_1 belongs to $cloud_1$
  AND e_2 belongs to $cloud_2$
  AND e_3 belongs to $cloud_2$
  AND x < t_0
THEN
  ERROR
ELSE
  ASSUME
    e_2 precedes(x) e_1 & e_2 overlaps e_3 == TRUE
  AND
IF there is exist r_c < $cloud_1$, $cloud_2$>
  AND $cloud_1$.state = active
  AND $cloud_2$.state = active
  AND
  IF there is r_{dto} <e_1, $domain_1$>
    AND tag_1 belongs to $domain_1$
    AND tag_1 belongs to $cloud_1$
    AND tag_2 belongs to $domain_2$
    AND there is r_T <tag_1, tag_2>
    AND there is r_{CA1} <$cloud_1$, {$action_1$}>
    AND there is r_{CA2} <$cloud_2$, {$action_1$}>
  WITH
    $action_1$ = {a_1, a_3, a_6}
    AND
    $action_2$ = {a_1, a_5, a_7}
THEN

APPLY {{a_1, a_3, a_5, a_6, a_7} – {
    a_1.conflicting ∪
    a_3.conflicting ∪
    a_5.conflicting ∪
    a_6.conflicting ∪
    a_7.conflicting}
VALIDATE
    a_1.post-conditions = TRUE
    a_3.post-conditions = TRUE
    a_5.post-conditions = TRUE
    a_6.post-conditions = TRUE
    a_7.post-conditions = TRUE
MARK
    a_1.mode = successful
    a_3.mode = successful
    a_5.mode = successful
    a_6.mode = successful
    a_7.mode = successful
END

 ISBN: 978-1-61208-104-5

Legend (for details, see [2]):
r_C: R_C | r_C ::= <c_1: C, c_2: C>, cloud to cloud relation
r_T: R_T | r_t ::= <t_1: T, t_2: T>, tag to tag relation
r_{CA}: R_{CA} | r_{CA} ::= <c :C, {a_i : A | p_i: P}>, cloud to action relation
r_{dto}: R_{Dto} | r_{dto} ::= <e:E, d:D>, event to domain relation.

As a result, the successfully marked actions can be re-used as recommended actions when similar event patterns occur. When an event pattern inventory exists, a similar algorithm is associated with each pattern. In this case, the Diagnosis Engine behavior is a combination of all these algorithms.

V. CONCLUSION AND FUTURE WORK

In this article, we proposed a new timestamp approach and considered a series of temporal operators defining event relative temporal position that allows a more fine grain interpretation of system behavior. Based on these concepts, we provided examples on diagnosis interpretations considering temporal dependencies between events and a more complete behavior specification of a diagnosis engine.

As future work, an event dependency pattern repository based on temporal relationships is the target. This will allow a semantic interpretation of different situations and support validations of the actions timely triggered based on probable-cause.

VI. REFERENCES

[1] M. Popescu, P. Lorenz, and J.M. Nicod, An Adaptive Framework for Diagnosis Validation, The Proceedings of The Third International Conference on Advanced Engineering Computing and Applications in Sciences, ADVCOMP 2009, Sliema, Malta, pp. 123-129, IEEE Press

[2] M. Popescu, P. Lorenz, M. Gilg, and J.M. Nicod, Event Management Ontology: Mechanisms and Semantic-driven Ontology, The Proceedigns of The Sixth International confernces on Networking and Servcies, ICNS 2010, Cancun, Mexico, pp. 129 - 136, IEEE Press

[3] W. Stallings, SNMP, SNMPv2, and CMIP: The Practical Guide to Network-Management Standards, Addison-Wesley Publishing Company, 1993, ISBN 0-201-63331-0

[4] M. Popescu et al., US Patent 7275017, Method and apparatus for generating diagnoses of network problems

[5] L. Lamport, TLA: Temporal logic of Actions, http://research.microsoft.com/en-us/um/people/lamport/tla/tla.html
Retrieved: August 12, 2010

[6] R. Griffith, J.L. Helelrstein, G. Kaiser, and Y. Diao, Dynamic Adaptation of Temporal Event Correlation for QoS Management in Distributed Systems, 2006
www.cs.columbia.edu/techreports/cucs-055-05.pdf
Retrieved: August 2, 2010

[7] K. Walzer, T. Breddin, and M. Groch, Relative temporal constraints in the RETE algorithm for complex event detection, Proceedigns of the Second International Conference on Distributed Event-based Systems, 2008, pp. 147-155

[8] M. Popescu, Temporal-oriented policy-driven network management, Master Thesis, Mcgill University, Canada 2000, p. 140

 ISBN: 978-1-61208-104-5