Ontology-based Business Activity Monitoring Agent

Duco N. Ferro
Almende B.V.,
The Netherlands
duco@almende.org

Mark Hoogendoorn
Department of Artificial Intelligence,
Vrije Universiteit Amsterdam,
The Netherlands
mhoogen@cs.vu.nl

Catholijn M. Jonker
Man-Machine Interaction group,
Delft University of Technology,
The Netherlands
C.M. Jonker@tudelft.nl

Abstract

Business Activity Monitoring (BAM) and Business Intelligence (BI) solutions are both intended to provide insight into the activities and performance of the enterprise. Deployment of such systems requires extensive tailoring to the enterprise, best left to experts. The dynamics of the enterprise demands a solution to the maintenance of BAM/BI solutions. This paper presents an Ontology-based BAM-Agent, called OBAMA that supports the maintenance of the system in light of changing business processes. Furthermore, for the formulation of aspects and properties to be monitored, it combines the expressive power of SQL, and TTL (a temporal trace language of first order logic). OBAMA helps in the preparation of regular assessment reports on the enterprise, taking into account key performance indicators as set by its operation manager. The paper describes the architecture, the combination of SQL, and TTL techniques for monitoring, and provides description of its kernel processes. OBAMA's performance in a surveillance company is presented.

1. Introduction

Business activity monitoring (BAM) is software that supports the monitoring of those activities that are implemented in computer systems. It is intended to provide real-time summaries of business activities to operations managers and upper management, and to detect and warn of impending problems [10]. This technology is a competitive differentiator for companies. Since the emphasis is on analysis, analysis tools are of the utmost importance [8]. The need for more complex forms of analysis on the huge relational data sets companies collect is growing along with the technology to support the analytical process. These more complex forms of analysis can be eloquently represented in constraints, and extended forms of first order logic. However, in current practice, solutions to the problem are implemented in ad-hoc and difficult to maintain procedural code that accesses the data through embedded SQL programming. Lohfert et al., [6] propose to use more elegant solutions that involve the use of declarative languages that integrate constraint modeling with database access in transparent ways.

In this paper we present a way of validating complex temporal properties formulated in TTL against log data of the system. The TTL-techniques assist in the monitoring process of the complex temporal properties. A preprocessing step is performed on the original dataset, to significantly reduce the size of the search space.

Furthermore, the paper advocates an ontology-based approach to assist the users in formulating their properties in terms of TTL. The advantage of using ontologies is a reduction in the number of errors made in specifying the intended properties and constraints for analyzing the business processes.

The approach is demonstrated for the example domain of Mobile Human Surveillance in private security, for which an ontology and decision support system is presented in [4]. In this domain a security company plans frequent visits by security guards to their client premises to deter and observe illegal activities, such as theft and vandalism. In addition, personnel of the security company act upon alarms (e.g., burglar alarms) by sending a guard for further inspection. The requirements with respect to all of these tasks (e.g., the maximum response time) are specified as part of a contractual agreement. Monitoring the performance of these processes is crucial to safeguard overall performance of the security company.

This paper is organized as follows. First of all, in Section 2 the techniques used within the agent to enable monitoring of the performance indicators are introduced. Thereafter, Section 3 presents the overall architecture of the agent. The evaluation of the architecture in the domain of the case study (private security) is addressed in Section 4, and Section 5 is a discussion.

2. Formulating Properties

The core of OBAMA consists of the specification and monitoring of key properties within the business processes of the company. In order to formulate these, two approaches are used. First, the logical format to be used is specified, thereafter the more common SQL format is briefly addressed.

2.1. Temporal Trace Language

The first language used to specify properties to be verified upon an organization is TTL (for Temporal Trace Language, cf. [1]) that features an automated checker. This predicate logical temporal language supports formal specification and analysis of dynamic properties, covering both qualitative and quantitative aspects. TTL is built on atoms referring to *states* of the world, *time points* and *traces*, i.e. trajectories of states over time. In addition, *dynamic properties* are temporal statements that can be formulated with respect to traces based on the state ontology Ont in the following manner. Given a trace γ over state ontology Ont, the state in γ at time point t is denoted by state(γ, t). These states can be related to state properties via the formally defined satisfaction relation denoted by the infix predicate |=, comparable to the Holds-predicate in the Situation Calculus: state(γ, t) |= p denotes that state property p holds in trace γ at time t. Based on these statements, dynamic properties can be formulated in a formal manner in a sorted first-order predicate logic, using quantifiers over time and traces and the usual first-order logical connectives such as $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\forall$, $\exists$. Below, the properties and the results of the verification upon the representative traces are shown.

Specifying such behavior in TTL is not a trivial matter, often documents describing the goals of the company as well as procedural specifications can be used as a basis for such a behavioral description. However, such a specification typically lacks sufficient detail to obtain a complete behavioral description. Therefore, a three step process is specified to formalize these properties.

First of all, informal behavior descriptions are translated into a semi-formal format. This is a necessary step to structure the precise property more easily. For instance the following property can be specified for the mobile surveillance case:

> "in case an alarm is received, this alarm should eventually be accepted by someone."

When looking at the informal rule, it can be seen that such a rule can easily be translated into a semi formal format of the if-then form:

> if an alarm A for object O is received at time t
> then there should be a time point t' greater than t and smaller than t + MAX_TIME at which this alarm is accepted

As can be seen, variables have now been introduced into the rules.

Table 1. Sort definition

Sort	Explanation
ALARM	An identifier of an alarm
OBJECT	An identifier of an object

Table 2. Predicate definition

Predicate	Explanation
alarm_received: ALARM x OBJECT	An alarm with particular id has been received for a particular object.
alarm_accepted: ALARM x OBJECT	An alarm at a particular object has been accepted.

The second step in the formalization process is to define an *ontology* suitable for this particular organization which is based upon the semi-formal rules that have been distinguished, and the terms that occur in such rules. For example from the rule as presented above the ontology elements presented in Table 1 and Table 2 can be extracted. For more information on the ontology, the reader is referred to [4].

The final step in the process is to translate the semi-formal rules into formal ones using the ontology which has been created. Take for example the semi-formal rule which was specified previously. In formal format using TTL this rule can be expressed as follows:

$$\forall t{:}\mathrm{TIME}, A{:}\mathrm{ALARM}, O{:}\mathrm{OBJECT}$$
$$[\ \mathrm{state}(\gamma, t) \models \mathrm{alarm\_received}(A, O)\ \&$$
$$\Rightarrow \exists t' > t\ \ [t' < t + \mathrm{MAX\_DUR}\ \&$$
$$\mathrm{state}(\gamma, t') \models \mathrm{alarm\_accepted}(A, O)\ ]\ ]$$

2.2. Structured Query Language

The second language used for specifying properties to be monitored at an organization is the structured query language (SQL), a widely accepted standard programming language for querying and manipulating databases often used by users with no or little formal training in informatics [3]. Its applications range from simple retrieval in web-interfaces to complex functions that aggregate the stored data into useful information. These are important reasons to support the formulation of properties by SQL in OBAMA.agent.

2.3. Pre-Processing SQL

In business settings many data storage solutions, including the storage of logging data, are based on SQL. In order to verify properties using TTL upon the information stored within the company databases, certain pre-processing is required. Hereby, the pre-processing part takes care of generating a set of traces upon which the properties specified in TTL can be verified. Consider, for instance, a SQL table holding logged data with a

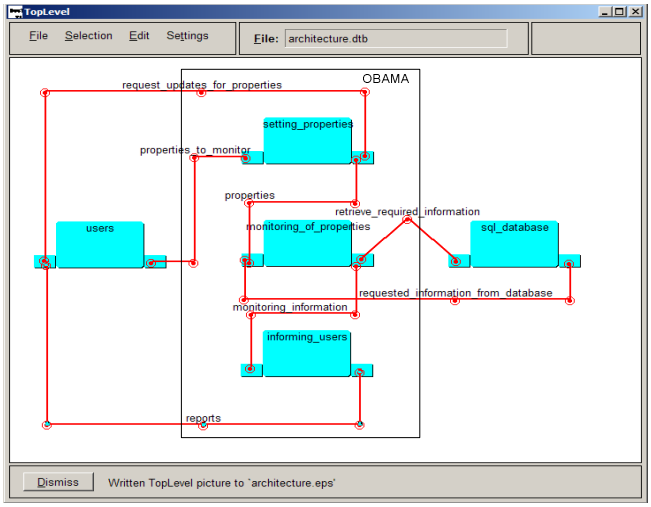

Figure 1. Agent Architecture

column representing the timestamp at which each event occurred. Thus, for each table holding a relation ship $R \subseteq A_1 \times \ldots \times A_n \times \text{TIME}$ the following observation could be used to produce a corresponding temporal trace:

$$\text{state}(\gamma, \text{TIME}) \models R(A_{1,\ldots,} A_n) \Leftrightarrow \pi_{A1,\ldots, An,\text{TIME}}(R)$$

The mechanism used here is to execute simple SQL queries that deliver the desired information for the trace, and thereafter convert the results to the appropriate TTL trace format.

3. OBAMA Agent Design

The languages for the specification of properties to be monitored and validated form the basis of the component-based design of the OBAMA agent shown in Figure 1. Note that the design approach followed is the DESIRE approach (cf. [2]). Hereby, the components are represented as boxes, which each have a small square on the left side indicating the component input, and a small square in the right side indicating the output. The lines between the components indicate information links, and finally, the solid lined box indicates the boundaries of the OBAMA agent. In the figure, it can be seen that the agent consists of three main components. Each of these components is discussed in a separate subsection.

3.1. Setting Properties

The first component concerns the setting of the properties that ought to be monitored. This component becomes active when the agent is first used to fill the initial set of properties. Hereby, the user needs to specify a number of elements for each property:

- The property itself using one of the approaches specified in Section 2. This is performed using a dedicated user interface. An example of the user interface for the specification of TTL properties is shown in Figure 2 (i.e., the property expressed in Section 2.1). Note that a new approach to specify these properties can be easily inserted into the component.
- The report interval (how frequently reports should be generated for the property).
- The responsible user.
- Whether this property should be continuously monitored and users should be informed immediately in case of deviations.
- How frequent the agent should request updates of the property specification.

In case all of this information is provided, the property is forwarded to the *monitoring of properties* component.

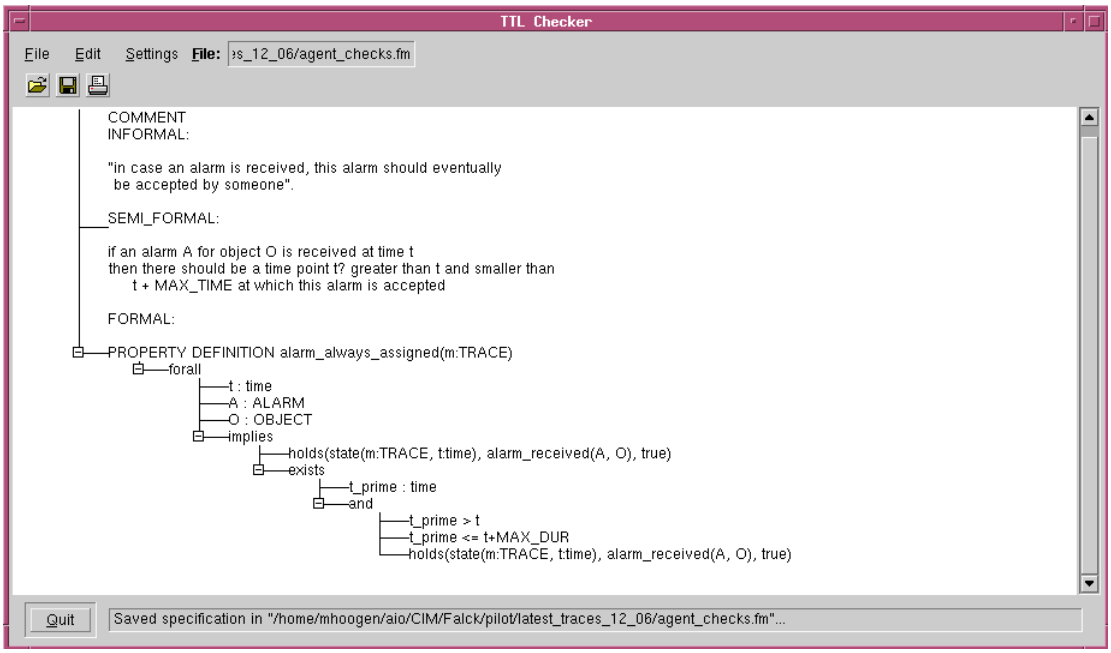

Figure 2. Example of property specification screen of the agent

3.2. Monitoring of Properties

After the properties and the required information accompanying these properties are known, the component *monitoring of properties* starts to reason. In case of properties that need to be monitored continuously, the component does so. In case of properties that only need to be considered after a particular interval these are only evaluated at the appropriate time points. Hereby, the component retrieves the necessary information from the SQL database, and, in the case of TTL properties, does some pre-processing (see Section 2.3). In case information is discovered that needs to be communicated to the user, the information is forwarded to the component *informing user*.

3.3. Informing User

The final component within the agent is the component for informing the user. Hereby, the appropriate user is selected, and informed using the same interface as used in Section 3.1. In case the user is not responding, the OBAMA agent will inform the superior of the user.

4. Experiments

In order to test the agent designed, we have conducted an extensive case study within the private security domain. First, a selection of the properties that have been inserted into the OBAMA agent by users will be shown. Thereafter, the results of the OBAMA monitoring component that are forwarded to the user are shown.

```
SELECT AVG(service_times.time_to_service)
FROM (
    SELECT
        SUBSTRING(a.log_text,16,6) AS alarm_request,
        SUBSTRING(c.log_text,19,6) AS alarm_service,
        a.log_time AS alarm_request_time,
        c.log_time AS alarm_service_time,
        MIN( (   UNIX_TIMESTAMP(c.log_time) -
                 UNIX_TIMESTAMP(a.log_time))/60)
                     AS time_to_service
    FROM v_logbook a
    LEFT JOIN v_logbook c
    ON SUBSTRING(c.log_text,19,6)=SUBSTRING(a.log_text,16,6)
    WHERE a.log_type=18
        AND a.obj_type=9
        AND a.log_text like 'alarm aanvraag %'
        AND a.log_time   BETWEEN   '2007-02-20T12:00:00.000'
        AND                        '2007-02-24T12:00:00.000'
        AND c.log_type=18
        AND c.obj_type=9
        AND c.log_text like 'ter plaatse alarm%'
        AND c.log_time   BETWEEN   '2007-02-20T12:00:00.000'
        AND                        '2007-02-24T12:00:00.000'
        AND c.log_time > a.log_time
        AND ((   UNIX_TIMESTAMP(c.log_time) -
                 UNIX_TIMESTAMP(a.log_time))/60) < 4*60
    GROUP BY alarm_request,alarm_request_time
    ORDER BY alarm_request_time,alarm_service_time
) service_times
```

Figure 3. SQL specification of property P2

4.1. Properties

Below, a number of properties that have been identified for the case study are shown. The first property (**P1**) specifies the most important performance indicator for the company, namely the average response time for alarms. The user decided to specify the property in SQL which is expressed in Figure 3.

The second performance indicator, P2, expresses the property concerning the presence of patrollers at an object for which an alarm has been received. This has been specified in TTL because the query became too complex to oversee well enough when using SQL.

P2: Patroller leaving before contact person
A patroller leaves before the contact person arrives if an alarm is received at time point t, and the alarm is accepted at t1, and the patroller leaves the alarm at t2, whereas there is no intermediate time point at which the patroller has left the alarm, and there exists t3 which is within a certain duration MIN_TIME from leaving the alarm. Hereby MIN_TIME indicates the minimum time required to perform a task.

Formally:

P2(m:TRACE, t:TIME) ≡
∃A:ALARM, O:OBJECT, P:PERSON, t2:TIME > t:TIME, t1:TIME ≥ t:TIME
[state(m, t) |= alarm_received(A, O) &
  state(m, t1) |= alarm_accepted(A, O) &
  state(m, t2) |= leaves_alarm (P, O) &
  ¬∃t0:TIME < t2
    [t0 > t & state(m, t0) |= leaves_alarm(P, O)] &
  ∃t3:TIME > t2, O2:OBJECT_CODE ≠ O
    [t2 + MIN_TIME > t3 & [state(m, t3) |= leaves_alarm(P, O2) |
                            state(m, t3) |= leaves_object(P, O2)]

A third example of a property which has been verified is the response time to status checks. The status check basically is a request from a patroller whether he/she has set the alarm of an object correctly. The property counting the response time to status checks has been defined in TTL and is expressed as follows:

P3: Status check response time
The average status check response time is the sum of the response times of the individual status check response times, divided by the total number of status checks in the trace.

P3(m:TRACE) ≡
$\sum_{t:TIME}$, case(status_check_received(m, t),
  $\sum_{t2:TIME}$ case(no_check_response_yet (m, t, t2), 1, 0)), 0) /
    number_of_status_checks

Here, the case(x,y,z) is a special TTL construct for counting and indicates that if x is the case, y is added to the sum, and otherwise z is added.

4.2. Monitoring Results

After having identified the properties, these were inserted into the OBAMA agent that started the monitoring process. All properties were set to regular report properties for which a weekly report should be generated. The following results were found by OBAMA:

P1: Average response time
The average response time was 23.6 minutes

P2: Patroller leaving object before contact person
With a minimum task time of 5 minutes, 7.9% of the cases the security guards left the alarm early.

P3: Status check response time
The average duration of status check responses was found to be 41.6 minutes.

5. Discussion

In this paper we introduce the power of TTL [1] and its tools to handle complex spatio-temporal properties in the context of Business Activity Monitoring and Business Intelligence. The OBAMA agent we created for this purpose has been developed in the domain of mobile surveillance security in which spatio-temporal information is essential for the business.

The simple, but effective design of the OBAMA agent makes it easy to maintain, and extend with future functionality. Its ontology-based nature enabled the support of the user in formulating properties by a incremental refinement method, as presented in [5]. This approach has been fully integrated in the TTL parts of OBAMA.

Finally, the rich tools for SQL proved essential not only for the more standard properties to be monitored, but also to do the necessary preprocessing for the more complex properties for which the user needs to use the TTL components of OBAMA.

More research has been conducted within the area of monitoring business processes. In [7] an approach is presented whereby web services are used to monitor the business process, and measure performance indicators. The disadvantage of such an approach is however that the whole information system of the company needs to be based upon a web-service, which is certainly not always the case. Our agent architecture works based upon existing techniques within companies, and can in the future easily be extended with for instance the ability to utilize a web-service architecture. A language to specify organizational performance indicators is presented in [9]. Such a language can be used as an input ontology for our OBAMA agent to allow users to start off with generic language constructs already.

In our future work we intend to further investigate the nature of spatio-temporal properties relevant in different domains and organizations. We wonder if it would be possible to represent/compute the dynamic properties in one Temporal SQL-like language. This would bring us a step closer to having a single SQL-like language for OBAMA agents.

Acknowledgment
The authors would like to thank James J. Lu for his comments on an early version of this paper.

6. References

[1] Bosse, T., Jonker, C.M., Meij, L. van der, Sharpanskykh, A., and Treur, J. (2008). Specification and Verification of Dynamics in Agent Models. *International Journal of Cooperative Information Systems*. In press, 2008.

[2] Brazier, F.M.T., Dunin-Keplicz, B.M., Jennings, N.R. and Treur, J.: 1997, Formal Specification of Multi-Agent Systems: a Real World Case, *in* Lesser, V. (ed.), *Proceedings First International Conference on Multi-Agent Systems, ICMAS'95*, MIT Press pp. 25-32; extended version *in*: M. Huhns and M. Singh (eds.), *International Journal of Co-operative Information Systems*, *IJCIS* vol. **6**(1), 67-94, (1997), special issue on Formal Methods in Co-operative Information Systems: Multi-Agent Systems.

[3] Elmasri, R. and Navathe, S. B.,(1994). *Fundamentals of Database Systems (2nd Ed.)*. Benjamin-Cummings Publishing Co., Inc.

[4] Ferro, D.N., and Jonker, C.M., (2008). Filtering Algorithm for Agent-Based Incident Communication Support in Mobile Human Surveillance. In: Proceedings of MATES 2008, to appear.

[5] Herlea Damian, D.E., Jonker, C.M., Treur, J., and Wijngaards, N.J.E., Integration of Behavioural Requirements Specification within Compositional Knowledge Engineering. *Knowledge-Based Systems Journal*. Vol. 18, 2005, pp. 353 – 365. Kochar, H., (2005/12/25), Business Activity Monitoring and Business Intelligence. In: ebiz, The Insider's Guide to Business and IT Agility.

[6] Lohfert, R., Lu, J.J., and Zhao, D, (2008). Solving SQL Constraints by Incremental Translation to SAT. In: N.T. Nguyen et al. (Eds.): proceedings of IEA/AIE 2008, LNAI 5027, pp. 669–676.

[7] McGregor, C. and Kumaran, S., (2002) Business Process Monitoring Using Web Services in B2B e-Commerce. In: Proceedings of the IPDPS 2002 Workshops, pp. 0219b

[8] Negash, S., and Gray, P., (2008). Business Intelligence. In: Burstein, F., and Holsapple, C.W. (eds.) Handbook on Decision Support Systems 2. Springer Berlin Heidelberg, pp. 175—193.

[9] Popova, V.N., and Treur, J., A Specification Language for Organisational Performance Indicators. *Journal of Applied Intelligence*, vol. 27, 2007, pp. 291-301.

[10] Wikipedia. Business Activity Monitoring.