Agent-oriented techniques for network supervision

Babak Esfandiari (LIRMM)
Gilles Deflandre (France Telecom - CNET)
Joël Quinqueton (INRIA-LIRMM)
Christophe Dony (LIRMM)
LIRMM - 161 rue Ada F-34392 Montpellier Cedex 5
France Telecom - CNET
2 avenue Pierre Marzin Technopole Anticipa F-22307 Lannion Cedex
esfandia@lune.lirmm.fr, deflandr@lannion.cnet.fr
jq@lirmm.fr, dony@lirmm.fr

Abstract

This paper presents the results of the "Réseau futé (Smart net)" project, the goal of which is to use distributed AI and multi-Agent techniques for network management and supervision. More precisely, theses techniques have been applied to the partial automation of the dynamic (what is known about a network is always uncomplete and can change at any time) processing of alarms and of various event notifications received by network management platforms. The system that we propose is able for example to automatically handle some alarms or to filter events of no-interest for a given operator.

To achieve this goal, an assistant, or "interface agent" according to the model proposed by Patti Maes [MK93] has been realized. The goal of the assistant is firstly to learn, by observation, the behavior of network supervision operator and secondly to reproduce such a behavior when the conditions in which the behavior has been learnt are detected again. The learnt information are stored using chronicles [Gha94]. A chronicle is a data-structure allowing programers to represent sequences of events while taking into account temporal knowledge.

Our assistant has been implemented and tested within Magenta which is a program, written in Smalltalk, that simulates (in a simplified way) a network management platform. This program respects the GDMO and CMIS standards.

Keywords:
Interface Agents, Machine Learning, Network Management, Chronicles.

Titre français :

Techniques à base d'agents pour la supervision de réseaux.

Résumé français :

Cet article présente les résultats du projet "Réseau futé" dont le but est d'utiliser des techniques d'intelligence artificielle distribuée et de systèmes multi-agents pour la gestion et la supervision de réseaux. Plus précisément, ces techniques ont été appliquées à l'automatisation partielle du traitement dynamique (on ne sait pas tout sur le réseau et ce que l'on sait peut changer à tout moment) des nombreuses alarmes et notifications d'événements reçues par les plateformes de gestion de réseau. Le système réalisé est capable par exemple de traiter automatiquement certains problèmes ou de réaliser un filtrage des événements non intéressants pour un opérateur particulier.

Pour ce faire, un assistant, ou "agent d'interface" selon le modèle proposé par Pattie Maes [MK93], a été réalisé. Le rôle de cet agent est premièrement d'apprendre, en l'observant, le comportement d'un opérateur de supervision de réseau et secondement de reproduire ce comportement lorsque les conditions ayant donné lieu à l'apprentissage sont détectées. Les informations apprises sont stockées sous forme de chroniques. Une chronique [Gha94] est une structure de donnée permettant de représenter des suites d'événements en prenant en compte les informations temporelles.

Cet assistant a été mis en oeuvre et intégré dans un programme (MAGENTA) de simulation (simplifiée) d'une plateforme de gestion de réseau que nous avons réalisé à cet effet. Ce programme respecte les normes GDMO et CMIS.

Mots clés français :

Agents d'Interface, Apprentissage Automatique, Supervision de Réseaux, Chroniques.

1 Introduction

The aim of the "Réseau Futé (Smart Net)" project is to introduce agent-oriented techniques in network management. But the term "agent" has many different meanings, and not all of them seemed useful to our concerns. We have privileged the following directions:

- Interface Agents: in order to help the processing of the large volume of alarms and various event notifications received by network management platforms, we have implemented an interface agent to assist the network supervision operator. An interface agent is "a computer program that employs Artificial Intelligence techniques in order to provide assistance to a user dealing with a particular computer application. Such agents learn by 'watching over the shoulder' of the user and detecting patterns and regularities in the user's behaviour" [MK93]. We had to deal with temporal knowledge: events often become meaningful when associated to their context, which partly consists in the previous and following events. Furthermore the dates of detection of those events may vary depending on the traffic. Therefore we chose a well-adapted formalism to deal with these aspects: the patterns to be learnt by our agent will be "chronicles" [Gha94]. In section 2, we describe this formalism, and how our assistant has been designed to learn chronicles.

- "Agentification": ascribing "mental states" (beliefs, goals, capabilities, commitments,...) to a computer program has been considered useful by some authors [Sho90] when it helps the understanding of its behaviour by the final user. We are considering this approach to improve the cooperation between the operator and the machine assistant.

- Multi-agent systems: decentralizing artificial intelligence is often practical for design purposes, useful for fault-tolerance issues, and sometimes mandatory when the problem is already decentralized by nature. But the main problem with these systems is to control the collective behaviour. Following [DBK94], we adopt "interaction protocols", not only to make several interface agents cooperate to improve their performance, but also to design MAGENTA, our simple simulation of a generic network management platform.

Our interface agent has been implemented as the management application part of MAGENTA in order to test our ideas. MAGENTA roughly respects network management standards such as GDMO and CMIS. In section 3 we describe MAGENTA.

2 Intelligent assistance to a network supervision operator

This section presents our interface agent which assists the network supervision operator by observing and learning his behavior. More precisely, what is to be learnt is: what action should be triggered after any given sequence of events.

The first issue to achieve is to choose an adequate formalism to represent the sequence of events, the corresponding action, and time. After having described the chronicle model [Gha94], chosen to fulfill that goal, we present the assistant's structure and the learning algorithm.

2.1 The need for an explicit temporal reasoning: the chronicle model

Temporal knowledge is required for reasoning on events, actions and change [Gha94], in order to model facts such as : precedence, overlapping, simultaneity between events. While "numerical" approaches based on Operations Research are not adequate for symbolic reasoning, classical and modal logic approaches do not provide a good balance between expressiveness and algorithmic complexity.

The *chronicle model* proposed by Ghallab is based on two elementary types of formulae taken from the reified temporal logic : *events* and *holds*.

- a "hold" expresses that some ground domain attribute holds over some interval, for instance :
  Hold (position (robot1, docking-site), (t5, t6))
- an "event" specifies a discrete change of the value of an attribute, for instance :
  Event (state (switch): (off, on), t8)

A chronicle model is a set of *event patterns* and temporal constraints between them and with respect to a context specified by *hold* assertions. If some observed events match the event patterns, and if their occurence dates meet the specified constraints within its context, then an instance of this chronicle occurs. Here is an example of a chronicle taken from [Gha94] :

```
Chronicle RobotLoadMachine {
event (Robot: (outRoom, inRoom), e1);
event (Robot: (inRoom, outRoom), e4);
event (MachineInput: (UnLoaded, Loaded), e2);
event (Machine: (Stopped, Running), e3);
e1 < e2;
1' ≤ e3 - e2 ≤ 6';
3' ≤ e4 - e2 ≤ 5';
hold (Machine: Running, (e2, e2));
hold (SafetyConditions: True, (e1, e4));
when recognized {report ''Successful load''; }}
```

Low-complexity real-time chronicle recognition [Dou94] is processed in several steps :

- Transform "holds" into "forbidden events", since an event should not change the value of the "held" attribute within the duration of the "hold".
- Create, if needed, a new instance of a possible chronicle and update (in fact this is always a restriction as time goes by) the *window of relevance* (acceptable time intervals for the expected events in order to complete a chronicle pattern) of all possible chronicles when a new event has been observed.
- Detect "deadlines" and occurence of "forbidden events". In such cases, the corresponding chronicle will be removed from the list of the possible chronicles.
- Trigger off the action corresponding to the completed chronicle.

2.2 The assistant's structure

Our machine assistant is mainly inspired by Pattie Maes's Interface Agents [MK93]. Since we had to deal with real-time aspects and time was a very important parameter, we chose to handle chronicles.

As shown in figure 1, our assistant has two main components :

- The Chronicle Recognition System (RS) is inspired from [Dou94]. Its goal is to receive dated event notifications (such as alarms) and try to match them with chronicles stored in the Confirmed Chronicle Base. Whenever a chronicle has

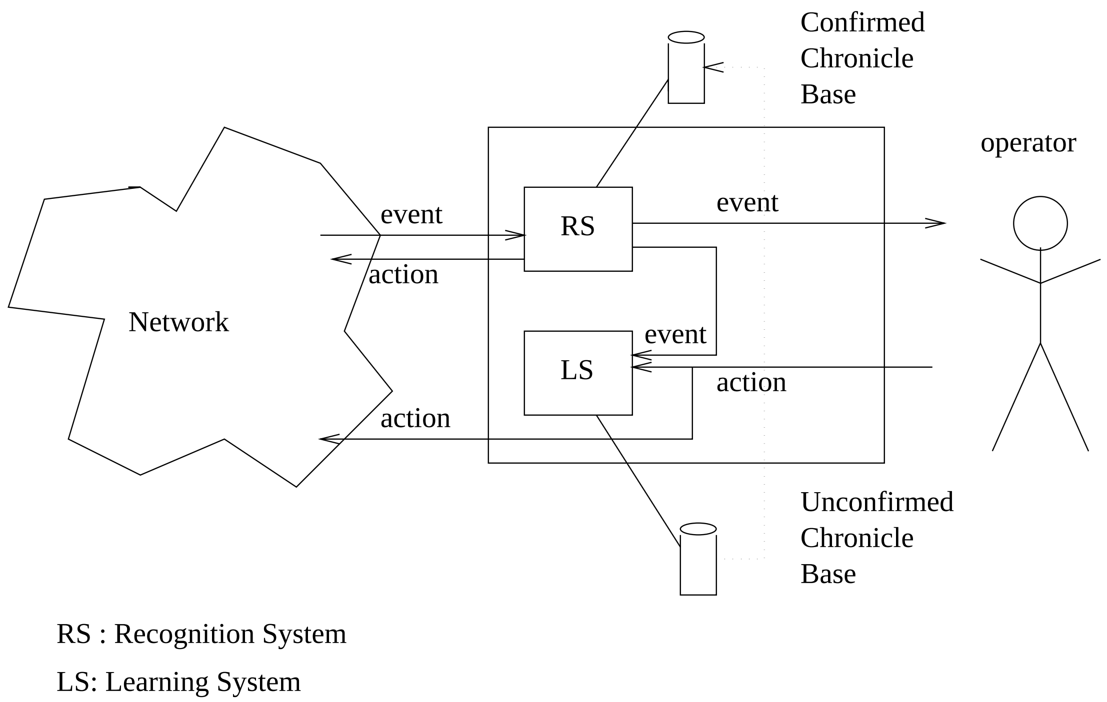

Figure 1: The assistant's structure

been recognized, the RS processes the corresponding action (such as filtering, fault diagnosis...). If the received events do not match any chronicle, it is up to the supervision operator to take a decision. For more details about the Recognition System, see [Dou94].

- The Learning System (LS), whose goal is to watch "over the shoulder" of the supervision operator when he takes a decision, in order to feed the Recognition System with new chronicles. New chronicles are stored in an Unconfirmed Chronicle Base before being "mature" enough in order to be transferred to the Confirmed Chronicle Base (i.e. confirmed). In the following subsections, we describe more precisely the Learning System.

2.3 Learnability issues

In [EN96], we show that given certain assumptions (no "overlaps" between the received sequences of events, an action appearing only in one chronicle), chronicle learning can be viewed as learning monotonous monomials, and therefore is PAC-learnable (i.e. learnable according to Valiant's [Val84]learnability model). But when releasing those assumptions, there is a need for interaction with an oracle (the provider of the examples): the user or any other expert agent. The more complex and numerous are the interactions and the oracles, the more efficient can be the learning task. Of course, a certain degree of cooperation can be expected from the supervision operator, but since we do not want to disturb him too much (do not forget that our goal is to simplify his task !), we have to restrict ourselves to heuristics, and try to interact with other oracles, for instance other assistants. In the following subsections, we present two algorithms that can both learn chronicles under the above mentioned assumptions, but with the first one also dealing with actions that appear in more than one chronicle, and the other one dealing with "overlapping" chronicles - since practically, trying to satisfy both constraints revealed to be contradictory.

2.4 The learning algorithms

This section describes our learning algorithms.

Let us specify learnt chronicles as consisting in a set of events terminated by an action. We will see later how we deal with temporal constraints and "holds".

As constraints, let us assume that the set of events and actions is finite and "relatively small", in order to generate a reasonable amount of chronicles with enough genericity. We will furthermore consider a particular action called "silence", which does not correspond to any actions of the supervision operator, but which is useful to generate particular chronicles : filters.

As the supervision operator has to interact with the Learning System (for instance to confirm chronicles), it is important for him to understand how our system learns. We believe that incremental learning is easier to follow by a human operator. Furthermore, "on-line" learning enables better cooperation with the Recognition System.

Chronicles are (1) created, (2) evaluated and (3) confirmed. The following section describes these steps.

2.4.1 The chronicle creation

A chronicle is created in two cases :

- When the supervision operator triggers an action : the created chronicle is the set of the received events before the action, plus the action itself.
- When nothing happens during a given time interval : neither received events nor actions. The created chronicle (which will be used as a filter) is the set of

the received events since the previous action, followed by the action “silence”.

2.4.2 The chronicle evaluation

The role of the evaluation step is to determine whether the created chronicle is worth being added to the Unconfirmed Chronicle Base.

Evaluation requires two operators : the inclusion ($\subset$) and the subtraction ($\backslash$) between two chronicles. A chronicle A is included in chronicle B iff their actions are equal and A’s sequence of events is a “subword” of B’s sequence of events. Dates of occurence of the events are not compared. If A $\subset$ B, then we can subtract A from B by removing A’s events from B’s sequence of events and replacing B’s action with the action “silence”. The result is what we call a “filter”. Of course the result of a subtraction can depend on how to select the subword in B. An easy way is to select the first occurence of each event. For more “realism”, it can be interesting to minimize the date differences, but this can be too costly, except if we do not want to obtain necessarily the “best” subword.

Another operation is the creation of temporal constraints. By increasing time intervals, we can somehow “generalize” chronicles by accepting chronicles with the same sequence of events and with different time points.

Here is the evaluation of a new chronicle C, created following the previous step, given his action α :

Algorithm 1: EVALUATE(C)

```
% C is a chronicle with an action α
begin
    Trust(C) := 1
    for each C' ∈ the CB so that C'(action) = α do
        switch C do
            case C ⊆ C' : exit
            case C ⊃ C' : create-and-evaluate-filter C'' := C \ C'
            case C ≠ C' : continue
        endswitch
    endfor
    for each C' ∈ the UCB so that C'(action) = α do
        switch C do
            case C ⊂ C'
                add C to the UCB; Trust(C) := Trust(C') + 1
                widen C time intervals given C'
                remove C' from the UCB;
                create-and-evaluate-filter C'' := C' \ C
            case C ⊇ C'
                create-and-evaluate-filter C'' := C \ C';
                widen C' time intervals given C
                Trust(C')++
            case C ≠ C' : continue
        endswitch
    endfor
    add (C) to the UCB
end
```

Let us point out that filters, like C'', created during the evaluation algorithm must also be evaluated. However, their evaluation is easier, as less cases are possible.

2.4.3 The chronicle confirmation

Chronicles initially stored in the Unconfirmed Chronicle Base (UCB), can be confirmed, i.e. transferred to the Confirmed Chronicle Base (CCB) in two ways:

- When the "trust" in a chronicle reaches a certain threshold, it can be automatically confirmed.
- The operator can confirm manually a chronicle by simply "clicking" on it.

Note that there is a need for a coherence maintenance system in the CCB. Indeed, if the system meets cases where the same sequences of events lead to different actions, it has to detect them. This is done by comparing each new chronicle to those stored in the CCB and trying to find mutual inclusions (this time with different actions). When incoherence is detected, the operator can either cancel the confirmation, or confirm the chronicle anyway (sometimes events can trigger several chronicles). Coherence maintenance is not important concerning the UCB, since those chronicles are not used yet in the recognition system. But this could also help creating "holds": the events obtained by the subtraction of two chronicles with different actions would correspond to "holds".

2.4.4 Dealing with "overlapping"

"Overlapping" is a problem that often occurs in network supervision: since several faults can appear simultaneously and that some event notifications are received with delay, events get mixed. Therefore the learning system has to "disentangle" them and attribute the right action to the right events.

To do this, we have to modify the way we create and evaluate our chronicles. First of all we have to always bufferize the k last events received by the system, even if they are actions and that they still trigger the creation of the chronicles. Thus we will not lose events that have appeared before a given action but that belong to another action that will be triggered later. Another reason is that by doing so, an action can also be taken in account as an event for a following action.

The evaluation of the chronicle still uses the same principles as before, but this time, instead of searching a "subword" relationship between sequences of events, we calculate (and store) the longest common subsequence *(LCS)* between two chronicles that trigger the same action. For this we use the classical LCS algorithm presented in [HJ77] with an $O(nm)$ complexity, n and m being the sizes of the two chronicles.

Therefore here is the modified version of the chronicle evaluation algorithm:

Algorithm 2: EVALUATE-2(C)

% C is a chronicle with an action α
begin
  $Trust(C) := 1$
  if $\exists\ C' \in$ *the* CB *so that* $C'(action) = \alpha$ **then** exit;
  if $\exists\ C' \in$ *the* UCB *so that* $C'(action) = \alpha$ **then**
    $C'' := LCS(C, C')$;
    $Action(C'') := Action(C)$;
    $Trust(C'') := Trust(C') + 1$
    remove C' from the UCB;
    add C'' to the UCB;
  endif
  else add C to the UCB;
end

Note that the counterpart of this algorithm is that we cannot store more than one chronicle with a given action, since calculating the longest common subsequence could then often lead to "empty" or truncated chronicles.

The next section explains how our assistant has been integrated into a simple network management platform.

3 An agent-oriented approach to design our assistant and our network management platform

In order to obtain all the needed event-reports, we had to integrate our assistant in a network management system.

To do so we have built our assistant as an agent interacting both with the network management system and with the operator.

We first built a generic multi-agent platform, called MAGENTA, that allows the simulation of interacting agents in the network management domain. This domain already uses established communication protocol standards to allow multi-agent interoperability in an open system such as network management systems.

The MAGENTA platform allows connection of our assistant to the appropriate network management system. It will also allow us to interconnect multiple operator assistants in order to improve learning and chronicle recognition efficiency.

In this section, we will see how we implement our assistant as a particular application of this platform.

3.1 MAGENTA's structure and implementation

MAGENTA's first goal was to be a simplified simulation of a Network Management Platform which respects network management standards such as GDMO [ISO90b] and CMIS [ISO90a]. But we found out that MAGENTA could also be used as a true multi-agent generic platform. MAGENTA implements the OSI Manager-Agent model : MAGENTA is a Smalltalk program which can play either the manager or the agent role, and that communicates with other Smalltalk programs (which can run on distant machines) via a TCP/IP protocol. Smalltalk has the advantage of being object-oriented, which makes GDMO very easy to apply. The genericity of MAGENTA is also mostly due to the choice of this language.

The platform can be divided into 4 distinct components (see figure 2) :

- The communication manager, which receives and sends CMIS-like messages to other agents through Unix sockets
- The object manager, which processes the CMIS-like queries (GET, SET, ACTION, CREATE, DELETE) and updates data (a sorted collection of managed objects)
- The event manager, which determines if the objects modifications are worth sending event notifications (EVENT-REPORT) to other agents
- The management application, which decides, depending on received events, how to behave. It is able, for instance, to compose queries and send them to other agents via the communication manager. This part is the only one specific to each problem. It can simply consist in an interface with a human operator.

Thus, depending on the type of the received messages, we use different "interaction protocols" to coordinate our components, i.e. we add a set of protocols that comes to *restrict* the different interactions an agent can link with other agents with regards to some problem to solve. Classical Interaction Protocols in the DAI litterature include the Contract Net Protocol[Smi80], the Hierarchical Protocol[ED] and the Introduction Protocol[SB].

In figure 2, arrows represent the data flow. We can see how, depending on the type of the received message, different interaction protocols are triggered between the components.

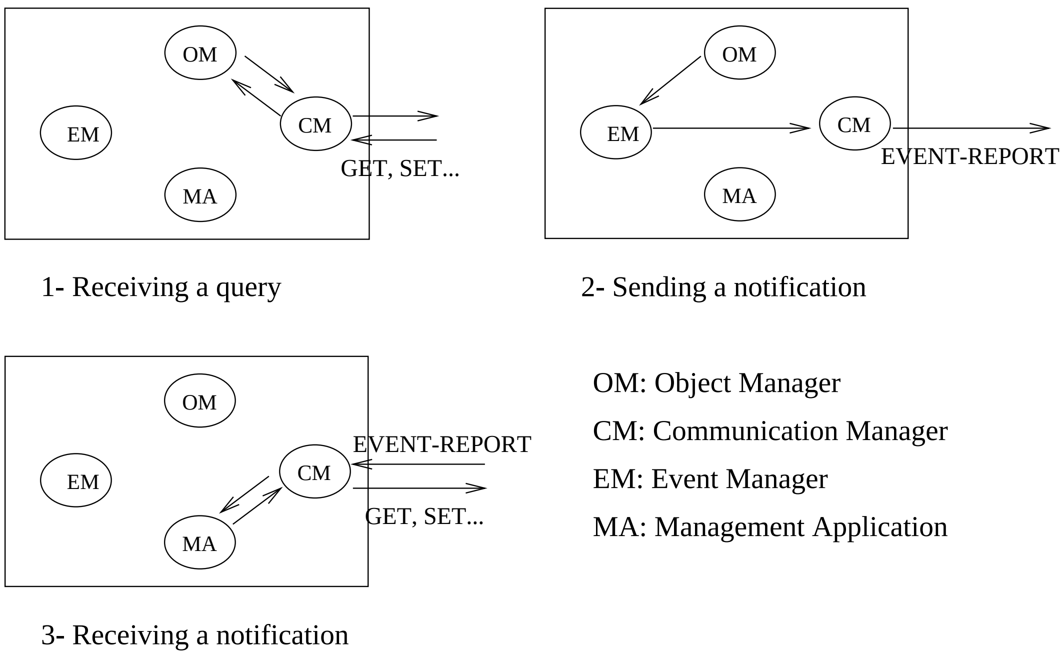

Figure 2: MAGENTA's structure

3.2 Building our assistant in Magenta

MAGENTA facilitates the implementation of multi-agent applications by offering ready-to-use communication tools, a minimal set of speach acts represented by the CMIS primitives, and an adequate mechanism to generate and interpret these primitives. Therefore here are the steps to follow for each agent to implement an application :

1. Creation of the classes and their initial instances. The default "behaviour" of the classes will derive from the "GDMO-Object" class.

2. Determination of the conditions of sending an event report for each class.

3. Implementation of the behaviour of the "Management Application" when receiving EVENT-REPORTs. The assistant has been implemented as being a part of the "Management Application". In this case a chronicle is a set of EVENT-REPORTs terminated by a GET / SET / ACTION / CREATE / DELETE primitive.

   This part should be augmented in the future to enclose cognitive notions of "Mental State".

Mental states have been defined in a proposition of an agent-oriented programming approach [Sho90].

An agent state is defined by its capabilities, the information it believes to be true or false at a precise time point, and commitments it has toward itself or other agents. Commitments between agents can be seen as contracts between them.

Conditions under which capabilities are available, or commitments can be undertaken are precisely defined respectively in agent private action description and agent commitment rules. The time parameter is also an important feature in this agent model. It is included in:

- the state and behaviour description,
- commitments to act (an agent commits to do something or to have something done at a precise time point),
- messages (emission and reception dates) and information included in messages.

Programming our assistant in the management application part with such an agent model would offer the operator the knowledge of its assistant's current state. This observation should help him understand for example what the assistant is doing, what he can expect from it, when the assistant needs to do a specific task, why it does not behave as expected.

Let us have a look at the Recognition System part of the assistant.

Information processed by the assistant are actions executed and event-reports sent or received at precise dates. This temporal property also implies that the information may be true, false or unknown depending on a given time point. Such information are the beliefs of the assistant. These beliefs can also model the assistant's interpretation of received event-reports. This interpretation can be formalized differently depending on the operator's tasks. An abstract event report EV1 about a connection can be represented as a connection failure for one operator whereas it can be represented as a counter update for another one. By browsing its assistant's beliefs, an operator obtains all the needed information about the state of the network represented in such a way that this information is relevant to his job.

The assistant's tasks are the execution of actions like displaying a message to the user or sending CMISE commands to other MAGENTA agents (which are used to gather and send event-reports, and to reply to CMISE queries about the network resources), at a precise date after a chronicle recognition. Chronicle recognition can be modelled as a set of temporal information believed by the assistant at a precise time. Thus, the assistant's commitments would be to execute an associated action when some conditions on its internal states are satisfied. Creating at time t0 a commitment of the assistant toward its user to execute a private action STATE-CHECK 5 minutes after having received an event report EV1 at time t1 about a connection and an event-report EV2 at t1 + 2 minutes about the same connection can be defined as follows:

```
(assistant user t0
   (If t1
       (Bel (t1 (RECEIVED-EV-REPORT EV1 ?connection)))
       (If (+ t1 2)
           (Bel ((+ t1 2) (RECEIVED-EV-REPORT EV2 ?connection)))
           (Do  (+ t1 7) (STATE-CHECK ?connection))
       )
   )
)
```

Once such commitment has been dynamically defined in the current state of an agent, an operator can know, by looking at his assistant state, at any time what his assistant is automatically going to do considering the occurence of specific events. To be user readable such a commitment formalization could be automatically rewritten in natural language. This commitment modelling is only useful for programming the assistant and for the user. Other more efficient models are used for the chronicle recognition system.

The assistant capabilities description would define the conditions under which it is able to warn its user about a chronicle recognition. For example chronicle recognition capabilities depend on the availability of the network connection that transmits event-reports. Traffic overload on this connection can also affect the delay between chronicle event-report sequences. With such a capability modelling, by observing that it is no more able to commit to chronicle recognition, the assistant could alert its user.

Finally agent commitment rules define the conditions under which a commitment can be created. Those conditions can be useful to define the context under which a chronicle is useful to be recognized. It can also be used to define the various thresholds that determine when a chronicle can be considered as learned and the associated action automatically triggered on the next occurence of the chronicle. Once again such a model is only useful for the user's presentation of the assistant's behaviour. More efficient representations are used to execute the assistant.

Picture 3 shows MAGENTA's temporary interface for the assistant, which is composed of the following parts:

1. The "notification" window displays the received events;
2. The "query" window enables the sending of queries and actions to MAGENTA agents by typing the query or the action and designating the agent;
3. The "chronicle bases" window displays both the UCB and the CB and therefore shows the Learning System's activity. It is possible to "confirm" a chronicle of the UCB and "edit" any chronicle. The interface should be more ergonomic in the future in order to be easily understood and accepted by the user. For instance, chronicles could be displayed in a graphical way.
4. The lower right part of the interface is dedicated to the Chronicle Recognition System. We have not connected it to our Learning system yet.

3.3 Multi-assistant cooperation

Once the assistant being programmed, the Magenta platform is used to make it interoperate with simulated network management systems with real network management network protocols. This interconnection capability is also used for multi-assistant cooperation.

Indeed, in order to increase the efficiency of our assistant, it could be useful to pair it with other assistants throughout the network. Thus assistants could exchange learnt chronicles to increase learning speed, and recognized chronicles to increase diagnosis speed. Here again, our main inspiration is Pattie Maes's collaborative interface agents.

To do so, we are using the following interaction protocols:

- An introduction protocol, when inserting a new assistant in the network. To do so, the new assistant broadcasts its address to other assistants. Those assistants update their list of active assistants, and reply to him, giving him their address. Thus, the assistant will only send its requests to active assistants.

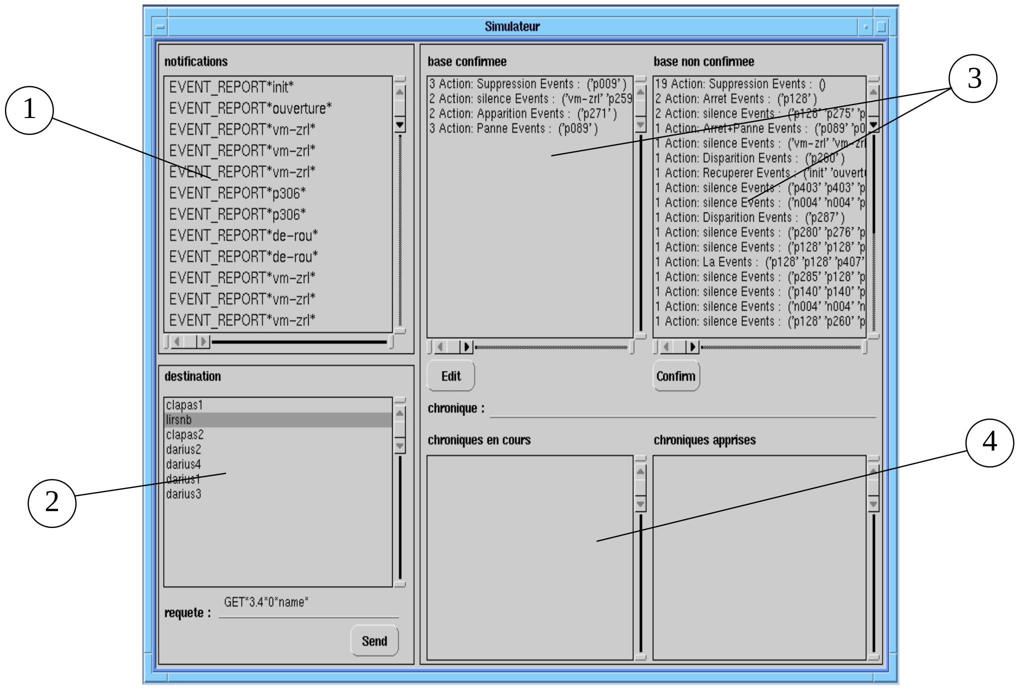

Figure 3: The assistant's interface

In particular, when an assistant leaves the network, it only has to notify it to active assistants ;

- An "exploration" protocol, sort of a contract-net protocol, to find assistants dealing with similar patterns. This can be done by comparing confirmed chronicles that trigger the same action from different chronicle bases. The result can be the attribution of a confidence rate to other assistants, depending how "close" are their chronicles ;

- A query protocol, to ask for some advice when facing a new case: when the trust value of a chronicle reaches a certain "consultation threshold", the assistant sends the sequence of events to other assistants. The latters try to match this sequence with their own chronicles, and their answer consists in three parts: (1) whether the received sequence includes or is included in a chronicle, (2) whether this chronicle belongs to the confirmed or unconfirmed chronicle base, (3) the action that this chronicle triggers. Depending on the received answers and its confidence in the assistants, the sender of the query can increase or decrease its own trust in the chronicle.

It is important to note that the above mentioned protocols have also been implemented with CMIS queries. We are so far testing the efficiency of this cooperation in accelerating the learning task.

Chronicle recognition could also migrate from one assistant to another depending on the network load and the proximity to events, in order to increase the reliability of the system by guarantying a certain stability of the time intervals through the network, since the events are dated only when they are received.

3.4 Experimentation

The assistant has been tested with supervisory data of a french packet-switched data network. Its components send a large volume of notifications dealing with their state, the state of their environment and of the components they are linked to. Thus, whenever a failure occurs, a large volume of notifications are brought to the supervisor. We have used these alarms as 'events' for the assistant. Moreover, an expert system has been developped to perform real-time fault diagnosis on these alarms. For our purpose, this system has played the role of the operator and its conclusions are considered as 'actions' by the assistant. The experiment has consisted in feeding the assistant with thirteen hours of log files in order to let appear interesting chronicles. The behaviour of the assistant revealed promising for it proved to learn quite well chronicles containing up to two events and an action, with a good relative trust rate. However, real network data include many overlapping chronicles (due to the size of the network), and this phenomenon causes an over-reduction of long chronicles. Solutions to this problem are yet under study, using, for example, bufferized events to build chronicles.

4 Related works

Building interface agents (sometimes also called software agents) is a very active research field, mostly due to the development of the World Wide Web. The reader shall read for instance [MK93] or [SH93]. However, so far we have not found papers dealing explicitly with temporal knowledge. Learnability issues are also seldom dealt with.

In network management and supervision, many intelligent architectures, such as expert systems, have been designed to help the operator's tasks, taking in account realtime aspects [Gai91] [GH92]. The authors only mention the addition of learning mechanisms as a promising perspective.

5 Conclusion and Perspectives

This paper has presented an agent-oriented architecture of an assistant to network management and supervision, which progressively learns the behaviour of the network management operator in order to trigger some of the usual actions, letting the operator focus on more important tasks, and giving him a simpler view of the network state. The learning system which has been presented in this paper has been implemented in the MAGENTA simulation platform, and the results seem promising although some inconsistencies have to be solved, mostly when faults occur nearly at the same time. To solve this, we are currently testing cooperation between several assistants using generic interaction protocols. New algorithms, based on the bufferization of the k-last events independently of the triggered actions, are also being tested. Besides, we would like our learning system to generate more "abstract" chronicles, but we understand that this could be a difficult task, considering that first-order logic raises complexity problems.

So far we have seen several multi-agent perspectives in network management assistance :

- the assistant itself is an agent that interacts with network management entities, other assistants, and the operator. The mental state and the communication language are more or less sophisticated depending on the "interlocutor" ;
- the assistant's implementation in MAGENTA can be seen as a multi-agent one, since the distinct parts communicate using a communication protocol, and that they are autonomous enough so that one can add or remove parts without causing great harm to the system
- when an alarm is recognized, must it be affected to only one hypothesis or to several [Kle91]? The problem here is that we are not guaranteed that a given chronicle will lead to a unique possible action. Then, we must manage possible conflicts between several hypothetical decisions. Therefore the chronicle recognition part of the system can also have a multi-agent structure. This could allow a concurrent maintenance of multiple hypotheses in order to take a decision when it has a maximum likelihood, or to propose several decisions

otherwise. The sequence of events is read concurrently by all agents representing the knowledge, and an agent becomes idle when it recognizes no more events. By the way, each active agent represents a possible action. When such a possible action is likely enough, or when only one remains possible, it is proposed to the operator.

Acknowledgements
Many thanks to Michel Plu (CNET Lannion A/EIA/AIA) for his very accurate remarks.
This work is part of the "Réseau Futé" project, which is supported by France Telecom - CNET under contract no 93 1B 141/142/143 5115.

References

[DBK94] Yves Demazeau, Olivier Boissier, and Jean Luc Koning. Using interaction protocols to control vision systems. In *International Conference on Systems, Man and Cybernetics*, San Antonio, Octobre 1994. IEEE.

[Dou94] Christophe Dousson. *Suivi d'évolutions et reconnaissance de chroniques.* Thèse d'université, Université Paul Sabatier, Toulouse, LAAS, Septembre 1994.

[ED] T. Montgomery E. Durfee. A hierarchical protocol for coordinating multi-agent behavior. In *Proceedings of the AAAI '90 Conference, AAAI, 1990.*

[EN96] Babak Esfandiari and Richard Nock. Chronicle learning for assistance to a network supervision operator. Research Report 96028, Lirmm, Janvier 1996.

[Gai91] D. Gaiti. *L'utilisation des techniques de l'intelligence artificielle pour la gestion des réseaux.* Thèse d'université, Université Paris 6, 1991.

[GH92] Francisco J. Garijo and Donn Hoffman. A multi-agent architecture for operation and maintenance of telecommunication networks. In Jean Paul Haton, editor, *Proceedings of the 12th Int. Avignon Conference*, pages 427–436. EC2 and AFIA, EC2, Paris, Juin 1992.

[Gha94] Malik Ghallab. Past and future chronicles for supervision and planning. In Jean Paul Haton, editor, *Proceedings of the 14th Int. Avignon Conference*, pages 23–34. EC2 and AFIA, EC2, Paris, Juin 1994.

[HJ77] Szymanski T.G. Hunt J.W. A fast algorithm for computing longest common subsequences. In *Commu. ACM, 20, 5*, pages 350–353, 1977.

[ISO90a] ISO/DIS 9595. *Information Technology - Open Systems Interconnection - Common Management Information Service Definition*, Janvier 1990. 2nd DP N3070.

[ISO90b] ISO/DP 10165-4. *Information Technology - Structure of Management Information - Part 4. Guidelines for the Definition of Managed Objects*, Juin 1990.

[Kle91] K. Klein. Supporting conflicts resolution in cooperative design system. *IEEE Transactions on Systems, Man and Cybernetics*, 21(6):1379–1390, 1991.

[MK93] Pattie Maes and Robyn Kozierok. Learning interface agents. In *Proceedings of the 11th Nat Conf on Artificial Intelligence*. AAAI, MIT-Press/AAAI-Press, 1993.

[SB] O. Boissier S. Berthet, Y. Demazeau. Knowing each other better. In *Proceedings of the 11th International Workshop on Distributed Artificial Intelligence, IWDAI '92, Glen Arbor, MI, February 1992.*

[SH93] J.C. Schlimmer and L.A. Hermens. Software agents: Completing patterns and constructing user interfaces. *Journal of Applied Intelligence Research*, 1:61–89, 1993.

[Sho90] Yoav Shoham. Agent oriented programming. Research Report STAN-CS-90-1335, Stanford University, Octobre 1990.

[Smi80] R. Smith. The contract net protocol: High level communication and control in distributed problem solver. *IEEE Transactions on Computers*, 29, 1980.

[Val84] L.G. Valiant. A theory of the learnable. In *Com. of the ACM*, pages 1134–1142. AAAI, 1984.