A multi-agent perspective for assistance to a network supervision operator

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

Abstract

This paper describes the "Réseau Futé (Smart Net)" project whose aim is to introduce DAI and Multi-Agent techniques in network management and supervision, in order to help the processing of the large volume of alarms and various event notifications received by network management platforms. Actually, many of these alarms prove to have a user-depending utility and have to be filtered. We have chosen a Chronicle model in order to incorporate temporal reasoning in our experimental platform. Thus some of the tasks (alarm filtering, log recording, fault detection...) can be automated via a chronicle recognition system, letting the supervision operator focus on more important tasks. Although it is possible to have a model-based approach, we will assume that we do not have a complete knowledge of the network, and that the model can evolve quickly.

Therefore "on-line" knowledge acquisition seems to be a good solution. We have been mainly inspired by Pattie Maes's Interface Agents, where an agent learns by "looking over the shoulder" of the human operator. The association of the chronicle recognition system and the chronicle acquisition system can eventually provide us a true intelligent assistant to network management and supervision.

In order to test our ideas, we have implemented a simplified simulation of a network management platform which respects network management standards (GDMO and CMIS) and encloses our assistant.

Keywords: Network management, interface agent, GDMO, CMIS, KQML, Chronicle, Machine Learning

This paper contains original and unpublished work and will not be submitted to any other conference before the notification of acceptance/rejection from MAAMAW.

1 Introduction

The aim of the "Réseau Futé (Smart Net)" project is to introduce DAI and Multi-Agent techniques in network management and supervision in order to help the processing of the large volume of alarms and various event notifications received by network management platforms. Actually, many of these alarms prove to have a user-depending utility and have to be filtered. Other events become meaningful when associated to their context, which partly consists in the previous and following events, which dates of detection may vary depending on the traffic.

Therefore time has to be explicitly taken in account. We have chosen the Chronicle model [Gha94] in order to incorporate temporal reasoning in our experimental platform. Thus some of the tasks (alarm filtering, log recording, fault detection...) can be automated via a chronicle recognition system [Dou94], letting the supervision operator focus on more important tasks. Although it is possible to have a model-based approach (such as in [GH92]), we will assume that we do not have a complete knowledge of the network, and that the model can evolve quickly.

Therefore "on-line" knowledge acquisition seems to be a good solution. We have been mainly inspired by Pattie Maes's Interface Agents [MK93], where an agent learns by "looking over the shoulder" of the human operator. The association of the chronicle recognition system and the chronicle acquisition system can eventually provide us a true intelligent assistant to network management and supervision.

In order to test our ideas, we have implemented a simplified simulation of a network management platform which respects network management standards (GDMO and CMIS) and encloses our assistant. The correct use of CMIS-primitives could eventually lead to a true cooperation between network management entities and assistants.

2 Intelligent assistance to a network supervision operator

2.1 The need for an explicit temporal reasoning

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 have problems in finding a good balance between expressiveness and algorithmic complexity.

The *chronicle model* proposed by Ghallab is based on two elementary types of formulaes 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''; }}
```

Realtime (and therefore with low complexity) chronicle recognition [Dou94] is processed in several steps :

- Transform "holds" into "forbidden events", i.e. an event should not change the value of the "held" attribute within the duration of the "hold"
- Possibly create 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 these cases, the corresponding chronicle will be removed from the list of the possible chronicles

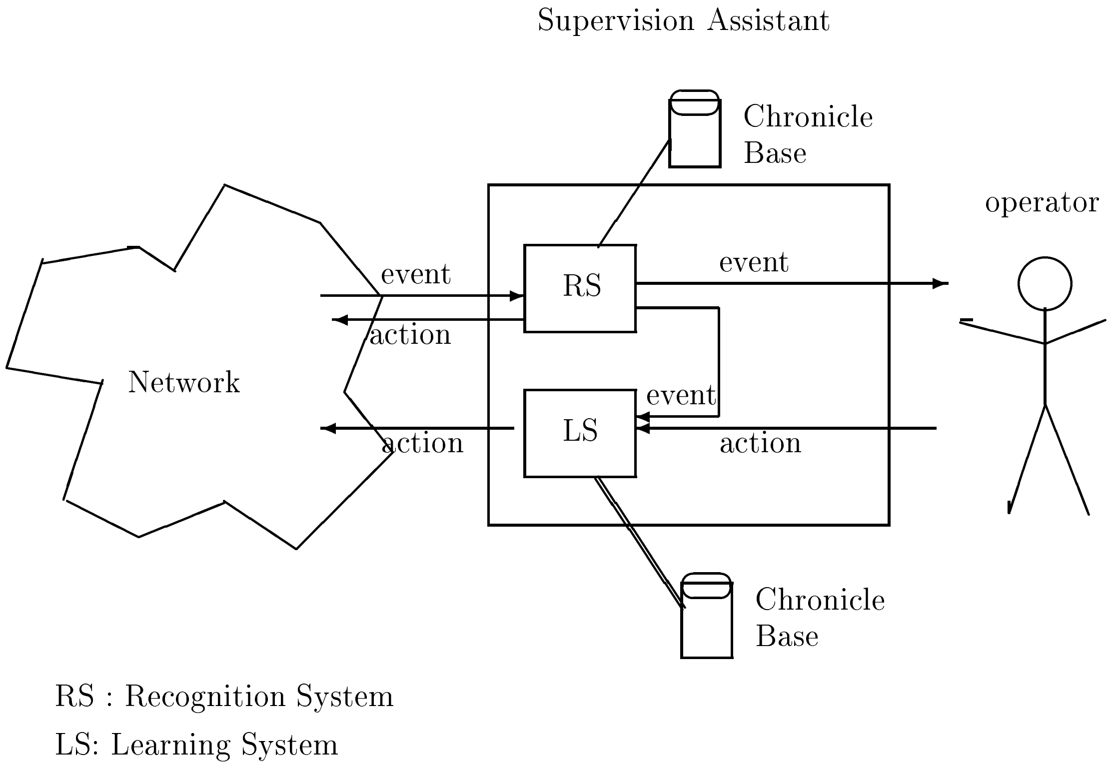


Figure 1: The assistant's structure

2.2 The assistant's structure

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

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

- The Chronicle Recognition System (RS), which goal is to receive dated event notifications (such as alarms) and try to match them with chronicles stored in the so-called Confirmed Chronicle Base. Whenever a chronicle has 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), which 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 a so-called Unconfirmed Chronicle Base before being "mature" enough in order to be transferred to the Confirmed Chronicle Base (i.e. confirmed). In the following subsection, we describe more precisely the Learning System.

2.3 The learning algorithm

Before beginning the description of our system, 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.

Learnt chronicles will consist in a set of events terminated by an action. We will see later how we deal with temporal constraints and "holds".

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, if the learning algorithm is fast enough, it can be used in realtime, just like the Recognition System, thus enabling better cooperation.

The chronicle learning is processed in three steps :

- The chronicle creation
- The chronicle evaluation
- The chronicle confirmation

2.3.1 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".

Obviously, when the action is triggered by the Recognition System, the corresponding events are deleted, since the corresponding chronicle has already been learnt.

2.3.2 The chronicle evaluation

Now we have to know whether the created chronicle is worth being added to the Unconfirmed Chronicle Base.

First we need to define 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 a :

Algorithm 1: EVALUATE(C)

% C is a chronicle with an action a
begin
  for *each* $C' \in$ *the* CB *so that* $C'(action) = a$ **do**
    switch C **do**
      case $C \subseteq C'$: exit
      case $C \supset C'$: create $C'' := C \setminus C'$; exit
      case $C \neq C'$: continue
    endswitch
  endfor
  for *each* $C' \in$ *the* UCB *so that* $C'(action) = a$ **do**
    switch C **do**
      case $C \subset C'$
        add C to the UCB; $Trust(C) := Trust(C') + 1$
        remove C' from the UCB; create $C'' := C' \setminus C$; exit
      case $C \supseteq C'$
        create $C'' := C \setminus C'$; widen C' time interval; $Trust(C')$++; exit
      case $C \neq C'$: continue
    endswitch
  endfor
  add (C) to the UCB
end

Filters, like C'', created during the evaluation algorithm must also be evaluated. However, their evaluation is easier.

Note that there is a need for a coherence maintenance system in the Chronicle bases. Indeed, if the system encounters cases where the same sequence of events leads to different actions, it has to detect them. This can be done by comparing each new chronicle to those stored in the Chronicle Base and trying to find inclusions (this time with different actions). This can also help creating "holds" : the events obtained by the subtraction of two chronicles with different actions would correspond to "holds".

2.3.3 The chronicle confirmation

Now that we have put chronicles in the Unconfirmed Chronicle Base, the question is : when can we put those in the Confirmed Chronicle Base ? We have basically two ways of doing this :

- 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.

3 A multi-agent agent-oriented approach to build our assistant system

To operate effectively on real network resources and with real operators, our assistant has to be integrated with multiple real network management systems to obtain all the needed event-reports. To do so we build our assistant system as an agent system interacting both with the network management system and with the user.

We first build a generic multi-agent platform, called MAGENTA, that allows the simulation of interacting agents in the network management domain. This domain has previously established communication protocol standards to allow multi-agent interoperability in an open system such as network management systems. We will see that these communication protocols embed the basis of a generic Agent Communication Language as promoted by the ARPA KSE [GK94].

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, after a description of some network management concepts and standards used by MAGENTA, we will see how we implement our assistant as a particuler application of this platform.

3.1 Some Network Management Standards and Concepts : a Simplified Presentation

3.1.1 A Guideline for the Definition of Managed Objects (GDMO)

GDMO [ISO90b] is an object-oriented formalism for the description of network resources. Each resource is considered as a *managed object*. Objects having the same properties are considered as instances of an object class, which belongs to an inheritance tree.

An object class is a template, mainly defined by its :

- attributes
- actions : operations using the above attributes
- notifications : messages sent by the object to signal an event,
- informal description of behaviour

In order to make queries upon a set of objects possible, the identifiers of the managed objects are defined within a *naming tree*. The hierarchy expresses that an object can contain, physically (components) or conceptually (directories, files, records, fields...), other objects, which can belong to different classes.

The conceptual set of the objects of a Managed System is the Management Information Base (MIB).

3.1.2 The Common Management Information Service Element(CMISE)

CMISE [ISO90a] enables the dialog between two network management entities by offering a set of primitives, which are used to send queries to a Management Information Base or to report events. The main primitives are :

- M-GET : gets the attribute value of an object
- M-SET : modifies the attribute value of an object
- M-CREATE : creates an object with its default values
- M-DELETE : deletes an object
- M-ACTION : triggers an action on the named object
- M-EVENT-REPORT : reports an event

The query primitives, besides the designation of the object, can have the following parameters, which allow the selection of a set of managed objects :

- The scope : determines the depth of the naming subtree on which the queries apply
- The filter : selects a subset of the above subtree by setting conditions on attributes

3.1.3 The Manager-Agent Model

In network management, the entity that sends queries and receives the answers and event notifications is called the *manager*. The symmetric entity, which receives queries, answers them and sends event notifications is called the *agent*. Both are shown on figure 2. Note that a same entity can alternatively play both roles, which means that a manager can be managed too.

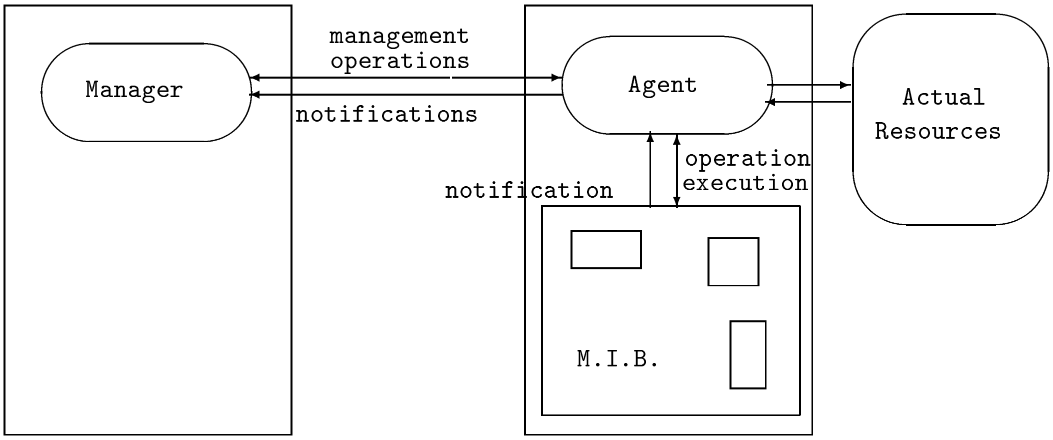


Figure 2: The Manager-Agent model

3.1.4 Event Control

The X.721 [CCI92] and X.734 [CCI91] ITU-T standards define an 'event forwarding' control process that enables the operator to dynamically control the flow of network events through the manipulation of GDMO objects called 'discriminators' via CMIS. These event forwarding discriminators can be set so that the reporting of a network event depends on the result of a test on some fields of the notification information. These tests are logical combinations of eight possible assertions :

- equality, greater or equal, less or equal,
- present (there can be optional fields in a notification information),
- substrings, subset, superset and non-null set intersection.

Here is an example of the use of such an object :

- first, the managing system creates an event forwarding discriminator by sending a CMIS CREATE request specifying :
  - the discriminator id
  - the discriminator construct, i.e. the test condition that will be used by the discriminator in testing potential event reports
  - a destination, i.e. a list of Application Entity Titles that determine the managing systems to which event reports should be sent when the test succeeds
  - an administrative state that can be Locked or Unlocked (default value)
- later on, the managing system can toggle this forwarding process on or off by sending a CMIS SET request on the administrative state attribute of the discriminator, setting it Unlocked or Locked

This feature may be very useful in a multi-agent environment, for it enables agents to inform each other with what notification information interests them.

3.2 A short detour : some comparison with an Agent Communication Language

Some similarities between those network management protocols and the famous agent-oriented approach of the ARPA KSE works on agent interoperability can be noticed.

In short they are promoting the use of an Agent Communication Language (ACL) [GK94] to deal with open system information and computing entities heterogeneity. Heterogeneous computing entities can be various classical softwares like the mathematical Fortran softwares, but also expert system applications, or databases, each being written with their own programming language, with different development teams, different development tools and in different computing environments.

Information heterogeneity may be syntaxical with multiple incompatible information formatting rules, but also semantical with meaning mismatch of a concept used by different softwares, or pragmatical like inconsistencies in hypothesis made by interoperating agents. Common standards of information format, like GIF or JPEG for graphics, MPEG for animated pictures, Postcript for complex images, mainly deal with the syntaxical aspect of information heterogeneity. An agent-based approach of interoperability has also to deal with all the other aspects.

With an ACL, agents are communicating with a common language which has an agent-independent semantic. Three components of an ACL have been defined:

- KQML (Knowledge Querry and Manipulation Language), a request language,
- KIF (knowledge interchange Format), a knowledge representation language,
- and Ontologies, sort of open-ended dictionary that defines the used vocabulary.

Messages sent using this ACL are KQML requests on KIF expressions using concepts defined in ontologies.

Almost semantically less clear, GDMO assumes the same role as KIF in an ACL. Studies on the formal semantic of GDMO are currently undertaken [KD94].

Multiple standard information models are already defined in GDMO. Those standard models are defined for specific management services according to :

- the nature of the managed network ; for example the model defined in the ETSI-NA5 recommendation is specific to ATM networks ;
- the abstraction level of these services ; for example the ETSI-NA5 recommendation defines network-level management services whereas the ETSI-NA4 recommendation defines an equipment-level management service [CCI] ;
- the nature of the user ; for example the X-USER interface information model is designed for customer network management systems whereas the X-COOP interface information model is designed for multiple telecom operator management systems.

Those standard information models are supposed to be ontologies in the network management domain.

Although CMISE defines less query types than KQML, its purpose is nearly the same. Furthermore [CL95] notices some redundance in KQML "performatives".

Table 1 gives some sort of "equivalence" between KQML "performatives" and CMISE queries.

KQML	CMISE
tell	M-SET, or M-EVENT-REPORT
ask-one, ask-all	M-GET + scope + filter,
discard	M-CANCEL-GET
evaluate, achieve	M-ACTION
subscribe, monitor	M-CREATE Discriminator, M-SET filter attribute

TABLE 1

ACL is also used to facilitate the matching of informational concepts of two agents. Although less sophisticated than the Mediator or Facilitators functionalities developped in the KSE with the ACL language, some negociation process has been also designed between Managers and Agents to facilitate the efficiency of agent interoperation.

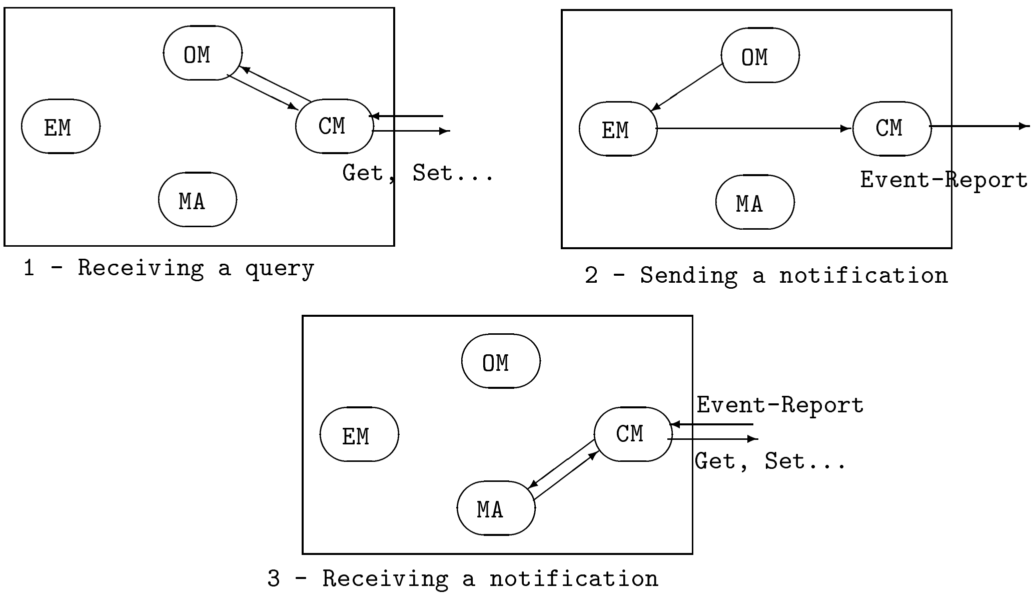
To understand each other, the Manager and the Agent negociate a shared information model on wich queries can be applied. This negociation process consists in computing the intersection between the Manager's and the Agent's information models. Object definition using the standard information models guaranties some intersection. A more sophisticated matching process between objects has been also designed. Compatibility rules between objects have been specified to guaranty that an object is functionaly equivalent to another one. This compatibility property is called allomorphism. It allows for example a Manager to send a query to an object which is not directly known by the agent, but can be replaced by another one known by the agent for doing the same purpose. To improve interoperation between agents' standard information models, standard communication functions called System Management Functions (SMF), have been also defined. Some examples are event-report functions , event-logging functions , threshold computing functions, acces control functions, etc. Each of those standards functions are defined with GDMO objects. For example, the event report function defines discriminator objects. During the association establishment process, the SMFs applicable on each part of the Agent information model are also negociated. This negocation mainly depends on the Manager requirements and the functions implemented and available in the Agent. The Shared Information Model and SMF negociation largely increase open system interconnection flexibility.

3.3 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 and CMIS. 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 parts (see figure 3) :

- 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 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.



OM : Object Manager
CM : Communication Manager
EM : Event Manager
MA : Management Application

Figure 3: MAGENTA's structure

3.4 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. First create the classes and their initial instances. The default "behaviour" of the classes will derivate from the "GDMO-Object" class.

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

3. Implement the behaviour of the "Management Application" when receiving EVENT-REPORTs. Our 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 ifself 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 ability to look at its assistant current state. This observation should help him to understand for example what his assistant is doing, what he can expect from his assistant, when his 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 according the time point being considered. Such information are the believes of the assistant. These believes can also model the assistant interpretation of event-reports reception. This interpretation can be formalized differently according to the operator tasks. An abstract event report EV1 about a connexion can be represented as a connection failure for one operator whereas it can be represented as a counter update for another one. By looking at his assistant's believes, an operator obtains all the needed information about the state of the network repensented in such a way that this information is relevant to his job.

The assistant tasks are the execution of actions like displaying a message to the user or sending CMISE commands to other MAGENTA agents, 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 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 according the occurence of specific events. To be user readable such a commitment formalization can be automatically rewritten into a natural language formulation. 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 can be automatically launched on the next occurence of the recognition 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.

3.5 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 going to be 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.

Thus, 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.

4 Conclusion and perspectives : towards a recursive agent conception

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.

This decreasing granularity may be a step towards a recursive agent conception, agents being built by other agents, once a common, unified and simple enough formalism could be found for them.

A first step could consist in the adoption of interaction protocols [EQKD95] at each layer.

5 Acknowledgements

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

[CCI] CCITT ETSI-NA4 (draft version H3). *Telecommunication Management Network (TMN) Generic Managed Object Class Library for the Network Level View.*

[CCI91] CCITT Recommendation X.734 — ISO/IEC IS 10164-5. *Information technology - Open Systems Interconnection - Systems Management - Part 5 : Event Report Management Function*, Juin 1991.

[CCI92] CCITT Recommendation X.721 — ISO/IEC IS 10165-2. *Information technology - Open Systems Interconnection - Systems Management - Part 2 : definition of Management Information*, Juin 1992.

[CL95] Philip R. Cohen and Hector J. Levesque. Communicative actions for artificial intelligence. In *1st International Conference on Multi-Agent Systems*, 1995.

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

[EQKD95] Babak Esfandiari, Joël Quinqueton, Jean-Luc Koning, and Yves Demazeau. Systèmes multi-agents et gestion de réseaux. In *Actes des JN du PRC-GDR Intelligence Artificielle*, Nancy, 1995. Teknea.

[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.

[GK94] M. R. Genesereth and S. P. Ketchpel. Software agents. *ACM*, Juillet 1994.

[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.

[JW93] Gabriel Jakobson and Mark D. Weissman. Alarm correlation. *IEEE Network*, pages 52–59, 1993.

[KD94] J. Keller and O. Dubuisson. Formal description of osi management information structure as a prerequisite for formal specifications of tmn. In *proceedings of IS&N'94 ; Lecture Note in Computer Science N.851*, pages 433–442. Springer Verlag, Septembre 1994.

[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.

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