Alarm Management System applied to SCADA environments in Power Utilities

Carneiro, V.M. [1], Coego, J. & Guerrero, C.[1]
Departament of Electronics and Systems
University of La Coruña, SPAIN
E-mail:{victor,javier,clopez}@des.fi.udc.es

Abstract

Nowadays several management protocols coexist, offering heterogeneous information about events which occur in managed objects. Some information is superfluous and in some cases unnecessary. Operators can be satured with irrelevant fault information. Alarm correlation and filtering processes are necessary to simplify the events analysis and identification of their main cause. Fault management increases its significance in complex environments like power utilities. This paper analyzes the standardization necessary to maintain an alarm distributed management environment, applied to a specific field, alarms in SCADA (Supervisory Control And Data Acquisition) environments. We have designed an alarm MIB. This MIB refers to both operating system and SCADA alarms. Through this MIB we can manage the most important faults arisen in the operating system. We also propose an architecture for a management system, consisting of both agent and manager. The agent can use either the SNMP, CMIP, or proprietary protocols. Manager supports alarm composition, polling, alarm correlation, protocol and information mapping, historics management and filtering in a metamanagement distributed environment. The management of each SCADA environment is made by a MIS (Management Information System) and there is a continuous flow of information among all the MIS at the Power Utility, distributing alarm management. An experimental prototype, which implements the proposed architecture, is presented.

Hoje em dia diversos protocolos da gerência coexistem, oferecendo a informação heterogênea sobre os eventos que ocorrem em objetos gerenciados. Alguma informação é suplerfua e em alguns casos desnecessários. Os operadores podem recebar moita informação sem importancia da falha. A correlação do alarme e os processos filtrando são necessários para simplificar a análise dos eventos e identificação de sua causa principal. A gerência de falha aumenta seu significado em ambientes complexos como companhias elétricas. Este papel analisa a estandardização necessária para manter um ambiente distribuído da gerência de alarmes, aplicado a um campo específico, alarmes em ambientes de SCADA (controle e a aquisição de dados de supervição). Nós projetamos um MIB do alarme. Este MIB consulta o sistema operando-se e os alarmes de SCADA. Através deste MIB nós podemos controlar as falhas mais importantes levantadas no sistema operando-se. Nós propomos também uma arquitetura para um sistema de gerência, dado forma por um agente e um gerente. O agente pode usar o SNMP, CMIP, ou protocolos proprietários. O gerente suporta a composição do alarme, a consulta, a correlação do alarme, traçar do protocolo e da informação, a gerência do históricos e filtrar em um ambiente distribuído de metagerência. A gerência de cada ambiente de SCADA é feita por um MIS (sistema de informação da gerência) e há um fluxo contínuo da informação entre todo o MIS na companhia elétrica, distribuindo a gerência do alarme. Um protótipo experimental, que execute a arquitetura proposta, é apresentado.

1. Adjunct Teacher at Facultad de Informática, Campus de Elviña, S/N. 15071 - A Coruña. SPAIN.

1 Introduction

Fault detection is vital in current systems, which guarantee by contract a predefined QoS. The main problem in traditional fault management systems is that, in spite of displaying fault ocurrency, do not provide information that assists to identify fault source. Many proprietary protocols (and even standard protocols like SNMP) have not been designed for carrying this information in their PDU formats. In this case, it is necessary to make use of translation functions that give us a more complete format.

Normally, alarm information generated in heterogeneous networks does not support an integrated model (SNMP traps, CMIP events, proprietary alarms). An operator will require an uniform set of notification types, with normalized parameters and technology independent definition. This normalization of the information mapped in the PDUs will need predefined and implicit data that complete the normalized alarm format. Management performance is improved with robust and object-oriented databases that simplify and improve access to events management information and allow the storage of specific data about the managed equipments and the identification of the main cause of the fault.

Other problems inherent to fault management are communications between network function managers, alarm redirection, polling and efficient storage of management information generated by the managed service.

All these problems are increased when we are working with SCADA fault management. SCADA systems perform real-time remote monitoring and remote control on transformer substations and stations (high, medium and low voltage). Many of these faults taking place on SCADA environments are detected by abnormal conditions on several parameters of the operating systems.

This paper addresses problems about fault management. First of all, we show the general architecture of the manager making the suitable conversions that allow the integration of multiprotocol alarms. This architecture is showed in section 2 and a prototype description is explained in section 3. Section 4 presents the Power Utility environment and in section 5, we explain the modelling of the Alarm MIB used by management (SNMP and CMIP) agents to register all arised errors. In section 6 the integration of AMS and SCADA services will be showed. Finally, conclusions, lessons learned and future work are discussed.

2 AMS (Alarm Management System) Architecture

Nowadays, networks are characterized by multiprotocol management: proprietary, SNMP [8] and OSI based management equipments and services. Both SNMP and CMIP [3] protocols give very little information about the fault (only that the alarm raised and its type). In case of SNMP, also generic type (*coldStart*, *warmStart*, *linkDown*, *linkUp*, *authentification Failure* and *egpNeighborLoss*) and equipment specific (*enterpriseSpecific*) are indicated.

In OSI management, more exactly in X.721 information recommendation [4], five types of notifications with a more complete information are defined: *Communications*, *QoS*, *Processing*, *Equipment* and *Environmental*. The parameters of OSI event service primitive, defined in the X.710 recommendation [2] give information about the request type, class and object that generates the notification, type, timestamp and additional information defined in X.733 recommendation in which new parameters are incorporated as probable causes, severity, backup situation and so on.

In the proposed architecture there is an ICF (*Information Conversion Function*), concept taken from TMN standard [1] and applied to the prototype. The goal of this function is the mapping of a great variety of fault information formats to the standard designed in X.733 recommendation. This translation is developed at semantic and syntactic level with the appropriate filtering, alarm

correlation [7] and historic management processes.

Figure 1 displays the architecture of the AMS, see [9]. It is a function-centralized model. The information received by the management system can be in legacy format, SNMP trap or CMIP event. The Event Handling module manages an event queue that is accessible by other management systems. This issue facilitates a management distributed environment, because AMS supports communications with other AMSs through Manager-to-Manager Communication module. Every event has an associated callback that performs the functions associated to the event services, that is to say, it is an event-oriented application. The use of timing routines to service events can generate the interruption of some action by another one of higher priority. In order to avoid undesirable situations, the implementation of critical region control mechanism is necessary.

The *Translator* module is responsible for the completion and reelaboration of the information received within the event. In order to achieve this purpose, it uses the information of the resources data base (network configuration, equipment identifications, ...) and correlation algorithms to identify the notifications (log files) that are related to the one generated at this instant. Another fields need to be completed by the interpolation of values received within the implicit information of this notification. For example, a severity field is mapped from the generic or specific trap field of the SNMP format. Another information that the translator must fill in is the probable cause of the fault (it is part of the information defined in managed object class), specific problems, severity, support operation, an indication of the severity trend in force in the managed object, threshold information, notification identification, correlated alarms, definition of the state change or transition and the actions that can be developed in order to solve the situation. All this information has been taken from ITU-T Recommendations so as to achieve a fault information standard format.

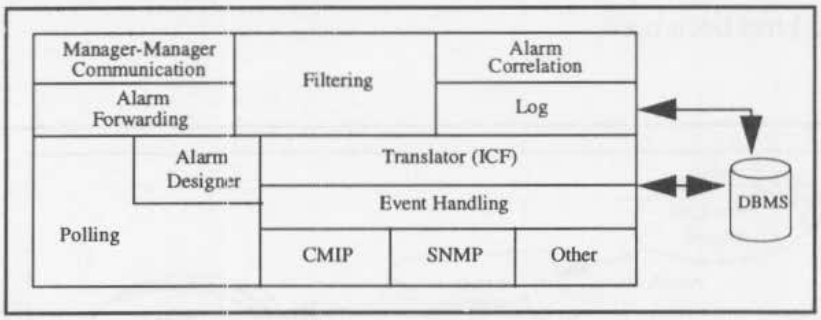

Figure 1 *Architecture of Alarm Management System*

The management system stores this information in a common repository (log) where both the Translator module and the alarm correlation process (dedicated to detect the main cause of a series of sequenced events) can access. The management of this database has some additional difficulties caused by the architecture of the fault management system. However, the use of an object-oriented model, with properties such as abstraction and inheritance, in order to represent managed objects, simplifies to a great extent the development of management applications.

Some equipments may not emit alarms spontaneously. In this situation, it is necessary to program polling routines which periodically allow to request the equipments and identify their current state. These polling routines are made of state diagrams that relate state changes and generate notifications in certain situations. These alarms can be designed previously by an alarm design module that indicates the value of the PDU fields.

Another important key of the AMS is the Event Forward Discriminator (EFD), described in Figure 2. The EFD is used to determine the event reports that must be sent to a fixed destination during specified time periods. For further description of EFDs, see [6]. Every event discriminator contains a builder that indicates the characteristics that an event report must satisfy to be sent. The management of event signalling provides to the management system with the necessary

mechanisms to communicate with other management systems and to carry out the initiation, termination, suspension and resumption of event processes.
All the received events stored at the log database can be filtered. This filtering process expedites the operator task. It can configure the system to receive only the critical alarms as well as allowing an automatic preprocessing that improves the network performance of the data network dedicated to send the required events to the user.
The proposed fault management architecture centralizes all management system functions. This approach has the lacks and problems associated with centralized environments. However, the simplicity of development and the low use of resources makes this approach an efficient front distribution of the functionality in different operation systems, situation in which the management system has only the information translation functions among the distributed management modules.

3 Prototype Description

Based on the previous proposed architecture, our research laboratory has developed an Alarm Management System (AMS). Its main goal is fault management in a heterogeneous open framework where diverse platforms and management protocols coexist. AMS has a modular architecture, where every module provides with easy access to the user thanks to a graphical interface, hiding the details of the underlined platform. Alarms are presented in a X.733 standard format according to the configuration designed by the network operator. Other functionalities have been enabled: communications with other management systems, configuration of alarm redirection, polling of equipments that do not have the capacity to generate spontaneous alarms, efficient access to alarm registers in logs and specific equipment data in a relational database.
For AMS implementation, API of Solstice Enterprise Manager (SEM) platform (Sun Microsystems Inc.) has been used.

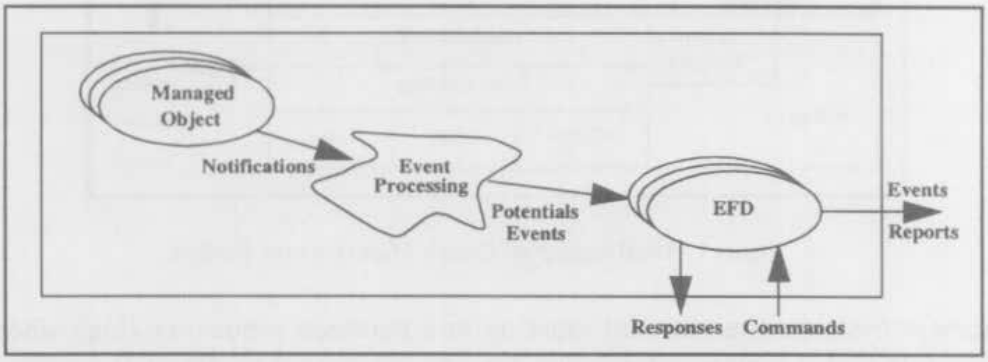

Figure 2 *Event Report Management Model*

The API provided by the SEM platform allows to use higher level functionalities that facilitate the development of management applications.

Figure 3 describes the software architecture of the developed prototype. The development of management applications over existing platforms expedites the implementation task. In our particular case, the use of services provided by the Management Information Service (MIS) in the SEM platform has provided to us with a communication infrastructure and fast prototyping of an object oriented high level application, completely integrated in the management system. This platform has an Event Management Module (EMM) that facilitates a list of all the active EFDs of the managed object represented in the Management Information Tree (MIT). The platform also supports the storage in and access to a relational database by the Relational Database Logging Daemon (RDL). Besides, the platform allows the interconnection of two or more MIS. This fact

facilitates management and information distribution. These communications between managers are transparent to management applications. The platform allows the integration of several types of agents located at different hosts, building a multiprotocol management system, easy to configure and able to incorporate pre-existing agents. In this case, the management system is implemented as an application over the management platform.

Also a trap redirector module has been developed. This module can be executed over other platforms with the objective of receiving and re-sending events to other managers. The received events of a manager are forwarded, independently of the evidence of their arrival, to another manager. This allows the management of other subdomains, depending on the different criteria (timetable, geography, ...). This module is the first step to the *metamanagement*, the management of managers.

The information shown to the user is based on the event information parameters compiled in the X.733 recommendation. This information can be filtered according to different operator criteria. In order to configure these filters, the updating of a database that provides feasible values in every field is needed.

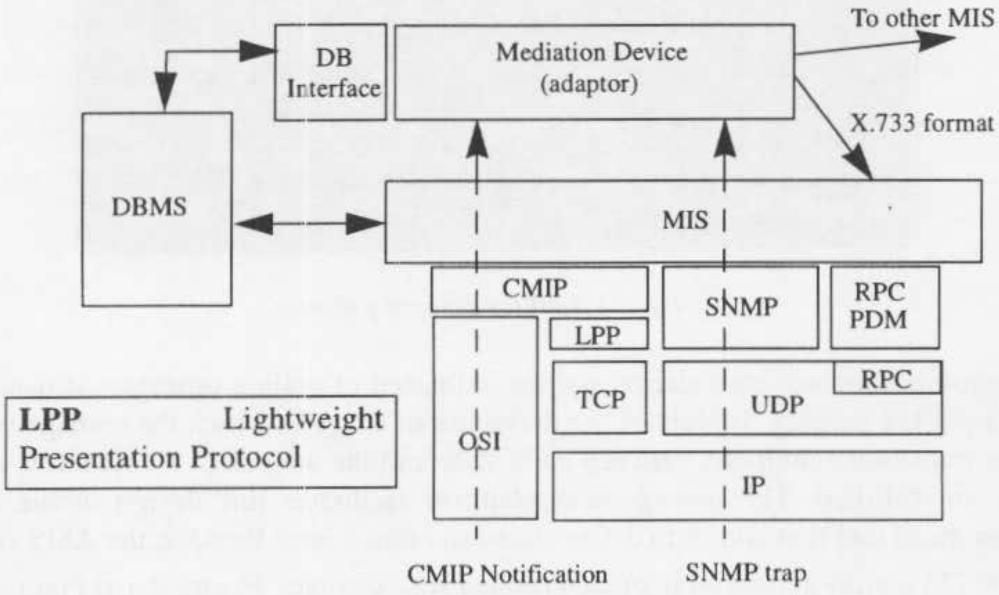

Figure 3 *Prototype software architecture.*

Due to the great amount of fields in X.733 format and its little clear utility in some cases, the alarm format is configured previously to the final presentation to the operator. The graphical application displays a shining indication with variable colour per alarm, according to the severity.

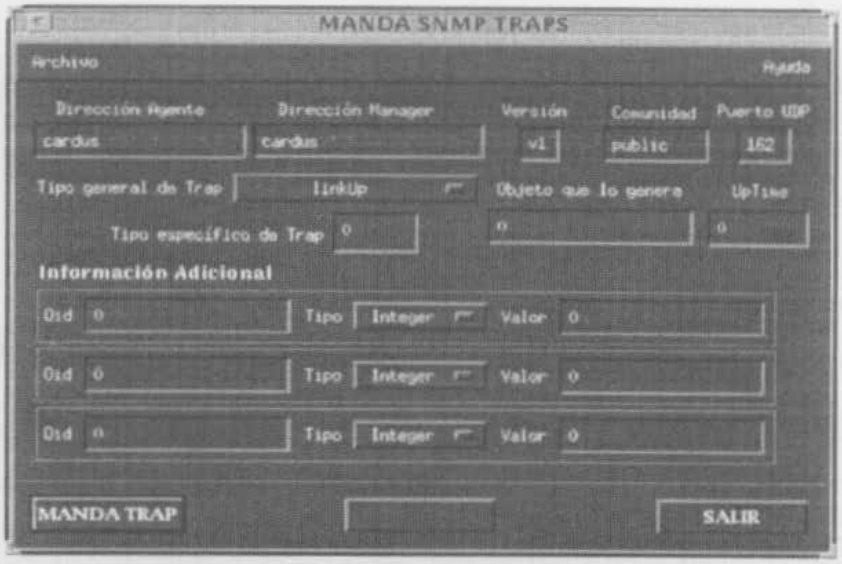

Figure 4 *Tool for designing alarms.*

Passive equipments do not emit alarms and the definition of polling processes is necessary. The design of a polling process consists of the definition of states in which the managed equipment can be, the transition conditions between each state and the actions to be achieved when some conditions are fulfilled. The management platform facilitates this design within an object-oriented graphical tool that allows to define states and transitions. Besides, the AMS supports the design of X.733 alarms generated in predetermined state changes. Figure 4 and Figure 5 describe the graphical interface of these two tools. The polling tool is an application provided by the SEM platform, while the alarm design tool has been developed from scratch, independently of the SEM platform.

The mediation module is configured to avoid report events in X.733 format to local or remote MIS. The management platform partially delegates the alarm handling to the management system. This approach provides distributed fault management among the applications executed in both platforms and the sharing of the management information located in both managers. The management database access by a mediation device is made by an object-oriented DBMS interface. This fact simplifies the storage and treatment of the management information. The use of the information provided by this database allows an efficient network management by the management system.

Although the database is centralized, the use of a distributed one in the future is foreseen. However, the complexity associated to a distributed database framework can dissuade its use in certain situations.

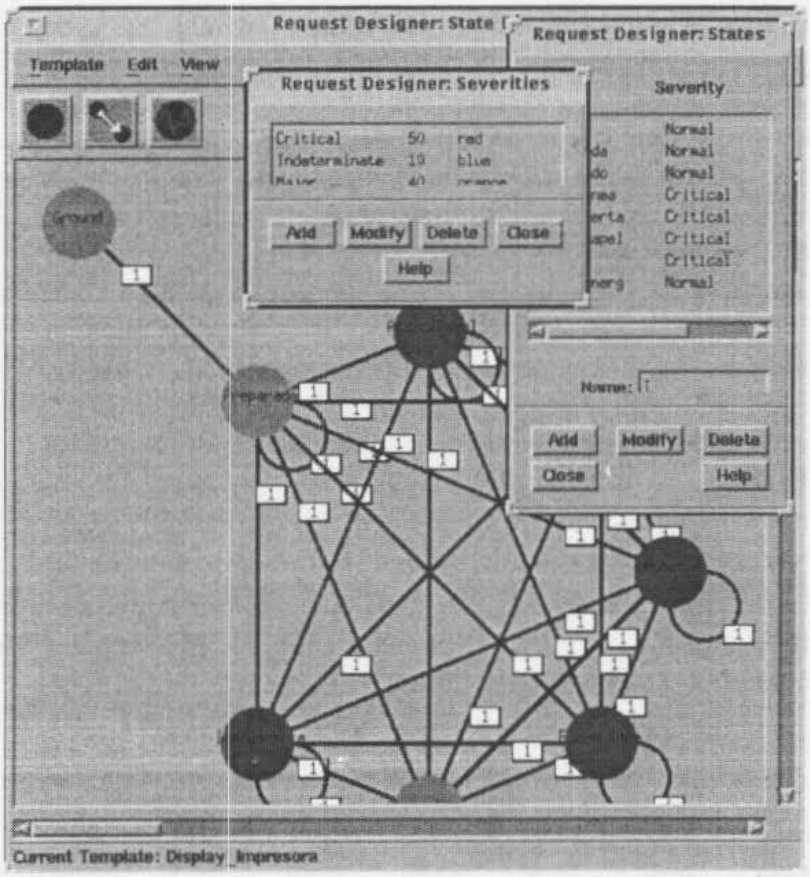

Figure 5 *Tool for designing polling process.*

4 SCADA/Operation environment

In this environment there are three main functional areas : SCADA, Operation and BDI. The standard architecture is showed in Figure 6. SCADA performs real-time remote monitoring and remote control on transformer substations and stations (high, medium and low voltage), enabling real-time control of the electricity network. The use of SCADA for operating elements of the network (remote control) and taking readings at a distance (remote measuring) is very extended. Most of the applications running on this environment are proprietary and running on SunOS 4.1.3, though at the moment systems are being migrated to Sun Solaris 2.5. This matter makes development more complex, forcing us to implement a OS-independent management application. SCADA environment consist of several machines. First of all, there is one administrator that supports most of the system workload. Because of the significance of its operations, there is also a second administrator, the spare one, which is in standby mode waiting for severe problems in the main administrator. At this moment, the spare/backup machine replace the bad-functioned one, maintaining the system with no problems in a transparent way to the users.

To obtain all kind of information from power stations, there are several types of machines : communicators, gateways and CUs (Control Units). Communicators gather information from high voltage stations whereas gateways and CUs gather information from medium and low voltage stations. Gateways and CUs coexist in SCADA environments, but they does not work at the same time. Environments with functional gateways don't have CUs, and environments with functional CUs have gateways, but their functions are disabled. Nowadays, the power utility is replacing CUs with gateways. Meanwhile, both classes of computers coexist, complicating management issues. Both communicators and gateways are replicated because of matters of fault tolerance. Usually, there is a machine in PC state, that is, running with no problems to appear in

the operating communicator or gateway. The other one is in SB (standby) state, waiting for possible problems. If the operating machine has problems and reaches NC (inoperative) state, the backup one starts running automatically. Sometimes, this switching has to be made by hand.
Another functional area is Operation, a distributed environment on UNIX platforms. In Operation area we have a GIS (Geographic Information System) running. GIS support the applications requiring integrated handling of graphic and alphanumeric data. The exchange information between SCADA and Operation environments are made through the interface machine placed on SCADA. Operation environment has a client-server architecture, through the incorporation of RPC technology for all queries. Its architecture consists of one server and two client machines. Operation area also has an active connection with a mainframe host through a SNA network. This connection and its respective emulation causes frequent problems too.
All graphical applications from both environments are displayed on two MMIs (Man Machine Interface). These applications also have several bugs, and because of this, faults in applications running on a MMI can alter the good performance of the other MMI.
The third functional area, BDI, is a facilities data base, containing several levels of alphanumeric, graphically integrated and organized information, which can be interrelated. BDI is out of the scope of this project and it will be probably included in following phases.
All these areas (SCADA, Operation and BDI) are called Distribution Handling Center. This structure, shown in Figure 6, is replicated in each of the company's seven distribution areas, varying only in the number of operating machines.

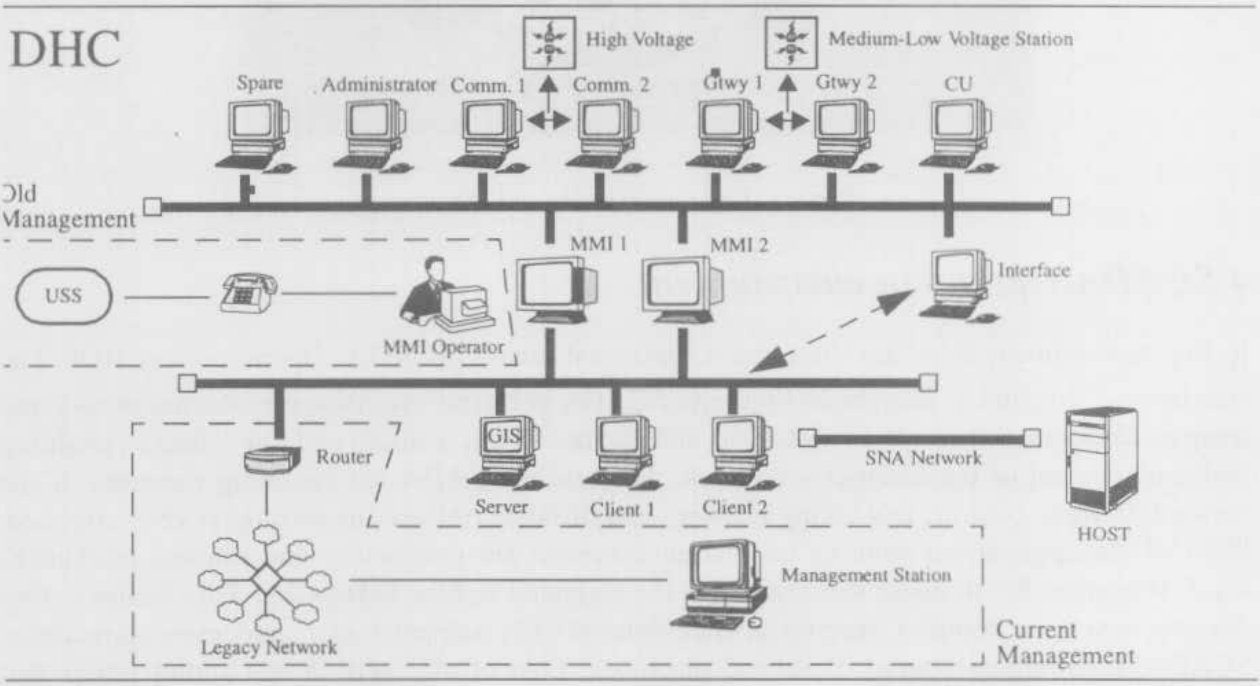

Figure 6 *Technical Architecture : SCADA-Operation generic environment*

The current *modus operandi* is that MMI s operators phone to USS (User Support System) when problems appear. The USS operator has a manual of procedures with sequences of actions to solve different faults. This fact results in severe lacks of effciency and late recover from faults. It is very usual that faults which occur in one MMI have efects on the other MMI, hence the operator has to investigate the alarm cause, checking by hand both MMIs. So it can be said that there is no real current management system for this environment.

5 Alarm MIB

So as to completely manage our SCADA-Operation environment, we have made good use of

several ITU-T Recommendations about alarm reporting function. First of all, we are going to explain the general structure of the MIB and then we will explain several tables from this MIB in detail.

We have a table containing all machines running on our work environment. This system has exactly the structure shown in Figure 6, except for old management schema, which has been currently eliminated. Also we have defined several tables with all type of parameters to be monitored. For example, there are tables related to OS parameters or processes running on different machines. Finally, we have several tables with detailed information about all type of alarms which have been generated in our system.

Our first table concerns to all machines running on the testbed. An entry from this table has the following structure :

```
PuEntityEntry ::= SEQUENCE {
        puEntityIndex                   INTEGER,
        puEntityOS                      DisplayString,
        puEntityMachineType             MachineType,
        puEntityDHC                     DisplayString,
        puEntityStatus                  MachineStatus,
        puEntityScadaStatus             SCADAStatus,
        puEntityLastInitialisation      TimeStamp,
        puEntityIpAddress               IpAddress,
        puEntityAgentPort               INTEGER
        }
```

Besides, some information about operating system, we also contemplate attributes like machine type (Operation Server, Operation Client, SCADA Administrator, SCADA MMI, ...), the DHC the machine belongs to, status of this machine (managed or not managed, because some machines are temporarily separated from production systems to a development system so as to be checked when problems arised), SCADA status (running, standby, not running) and port where the management agent is listening to.

Next, we will explain more tables related to general parameters being managed. First of all we describe tables concerning to processes which must be running on the machine so as to ensure that applications are OK. There are two main tables. The former includes all general processes with high significance in our environment, indicating attributes like number of each process that have to be running or type of machine where this process runs. The latter table refers to specific processes running on each testbed machine. We identify attributes like number of current processes, number of PID variation of these processes and current alarm status.

```
PuGeneralProcessEntry ::= SEQUENCE {
        puGeneralProcessIndex           INTEGER,
        puGeneralProcessName            DisplayString,
        puGeneralProcessDescription     DisplayString,
        puGeneralProcessNumber          INTEGER,
        puGeneralProcessMachineType     MachineType
        }

PuSpecificProcessEntry ::= SEQUENCE {
        puSpecificProcessEntityIndex    INTEGER,
        puSpecificProcessProcessIndex   INTEGER,
        puSpecificProcessNumber         INTEGER,
        puSpecificProcessPIDVariation   INTEGER,
        puSpecificProcessAlarmStatus    AlarmStatus
        }
```

We manage PID variation of processes, because a few of them have their own vigilant, which reinitialises the process when it fails. If there is a severe problem, this process could be

continuously up and down. So when a process reinitialises a few times, the system rises an alarm to the AMS.

The next tables have information related to OS parameters. There is an important correlation between bad-functioned applications and these parameters. For example, though an application is running, its bad functioning can be detected due to the existence of several zombie processes. Also, when SCADA resumpts, the affected machine has a great CPU load. OS parameters the agents monitor are the following : number of free memory pages, page in, page out, CPU Load, number of zombie processes and so on.

```
PuGeneralParameterEntry ::= SEQUENCE {
        puGeneralParameterParamIndex        INTEGER,
        puGeneralParameterDescription       DisplayString,
        puGeneralParameterUpperBoundary     INTEGER,
        puGeneralParameterLowerBoundary     INTEGER
        }

PuSpecificParameterEntry ::= SEQUENCE {
        puSpecificParameterEntityIndex      INTEGER,
        puSpecificParameterParamIndex       INTEGER,
        puSpecificParameterValue            INTEGER,
        puSpecificParameterStatus           AlarmStatus
        }
```

The former table have general parameters to be taken into account. This table defines both upper and lower boundaries for alarm ranges. Specific parameters are defined in the latter table. For each specific parameter we store current value and alarm status.

The most important table is the one concerning to alarm information. To develop this table, we have compiled information from X.733 [5] and X.736 [10] ITU-T Recommendations. In this table, information from SNMP traps and proprietary alarms are completed with additional value stored in DBs. There are two attributes which identify the alarm in the table, peErrorEntityIndex (concerning to the machine where the alarm arised) and peErrorIndex (an unique identifier for this alarm in the aforementioned machine). In addition to this, we have two attributes to identify neither the process nor the OS parameter affected by the fault.

```
PuErrorsEntry ::= SEQUENCE {
        puErrorsEntityIndex                     INTEGER,
        puErrorsIndex                           INTEGER,
        puErrorsProcess                         INTEGER,
        puErrorsParameter                       INTEGER,
        puErrorsEventType                       EventType,
        puErrorsEventTime                       TimeTicks,
        puErrorsProbableCause                   ProbableCause,
        puErrorsSpecificProblems                SpecificProblems,
        puErrorsPerceivedSecurity               PerceivedSecurity,
        puErrorsBackUpStatus                    BOOLEAN,
        puErrorsBackUpObject                    OBJECT IDENTIFIER,
        puErrorsTrendIndication                 TrendIndication,
        puErrorsActivatedThresHold              INTEGER,
        puErrorsThresHoldLevelValue             INTEGER,
        puErrorsThresHoldLevelHysteresis        INTEGER,
        puErrorsThresHoldMeasuredValue          INTEGER,
        puErrorsThresHoldReactivatingTimeStamp  TimeTicks,
        puErrorsNotificationIdentifier          DisplayString,
        puErrorsStateChangeDefinition           DisplayString,
        puErrorsAdditionalText                  DisplayString,
```

```
        puErrorsAdditionalInformation          DisplayString,
        puErrorsCurrentTime                    TimeTicks,
        puErrorsEventReply                     DisplayString,
        puErrorsError                          DisplayString
        }
```

From X.733 we got most of attributes of the table. We have completed several definitions like probable cause or event type with data from security alarm reporting function, X.736 [10]. Among the stored information, this table provide to us with data like:

- Event Type : Categorizes the alarm.
- Probable Cause : Defines further qualification as to the probable cause of the alarm.
- Specific Problems : Identifies further refinements to the Probable cause of the alarm.
- Perceived Severity : Provides an indication of how it is perceived that the capability of the managed object has been affected.
- Trend Indication : Specifies the current severity trend of the managed object.

We can also obtain information about correlated notifications and repair actions from additional tables as shown next. The former table is defined to be the set of all notifications to which one notification is considered to be correlated. The latter is used if the cause is known and the system being managed can suggest one or more solutions.

```
puErrorsNotificationsEntry ::= SEQUENCE {
        puErrorsNotificationsEntityIndex               INTEGER,
        puErrorsNotificationsAlarmIndex                INTEGER,
        puErrorsNotificationsIndex                     INTEGER,
        puErrorsNotificationsCorrelatedNotification    DisplayString
        }

puErrorsActionsEntry ::= SEQUENCE {
        puErrorsActionsEntityIndex                     INTEGER,
        puErrorsActionsAlarmIndex                      INTEGER,
        puErrorsActionsIndex                           INTEGER,
        puErrorsActionsProposedRepairAction            OBJECT IDENTIFIER
        }
```

With such a design, we are able to provide alarm reports with a standardized style, using a common set of notification types, with standarized parameters and parameter definitions, independent of particular managed objects.

6 Integration of AMS and SCADA-Operation Services

At this point we are going to explain our testbed and how it was integrated with several AMS. The testbed, shown in Figure 7, consisted of a reduced environment, formed by SCADA administrator, SCADA interface, Operation server and Operation client. The last one is acting as a gateway with the host. There is also a MMI in SCADA environment. All the applications are displayed onto the screen of this machine. In this testbed, the most significance elements of the real environment are represented.

Each one of these machines have an agent running. There are CMIP and SNMP agents which send CMIP events and SNMP traps respectively. All the alarms generated by these agents are

forwarded in X.733 format after their information contents have been completed and processed making use of the data maintained in the repository of the AMS. This conversion is made by the ICF (see section 2).

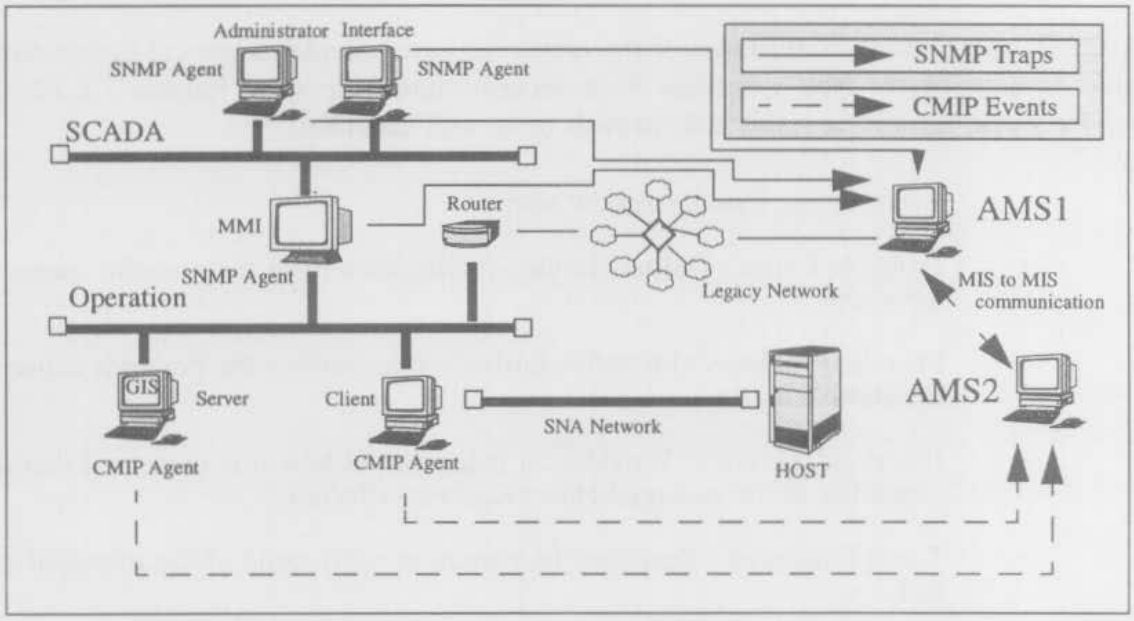

Figure 7 *Testbed for integrating AMS with SCADA-Operation Services*

There are two MIS, one of them (AMS1) with a RDBMS, storing information about generated alarms. The two MIS exchange information about received alarms. The first one, AMS1, stores information related to alarms arisen in SCADA environment. Meanwhile, AMS2 stores information related to alarms arisen in Operation environment. The actual configuration makes one MIS to be in charge of SCADA and Operation alarms. The reason for this choice has been the correlation between the alarms in both environments within this project. Currently we have only taken into account correlation between OS parameters and SCADA or Operation status. An alarm redirector module has been installed in another management platform that provides metamanagement facilities. When the secondary platform receives an event, it is configured to re-send the received event to the primary platform.

When the prototype is installed in a real environment, there will be three AMS. These AMSs will manage two, two and three DHC respectively, taken into account the significance and size of the DHC.

AMS in receives all SNMP traps from SCADA environment, which includes the MMI machine. On the other side, AMS receives all CMIP events from Operation. We have configured AMS 1 to redirect received traps to AMS with the trap redirector module.

In this integration, AMS supports alarms from SCADA machines which have been isolated from production environment and incorporated to a development environment. This type of machines usually has SCADA applications in inoperative state. That's the reason why these alarms are disabled. Also alarms from SCADA gateways in SCADA environments where there are UCSs active are disabled.

So as to complete our testbed, we have designed a graphical sender of SNMP traps which allows us to test our management system in extreme situations, with AMS supporting lots of SNMP traps proceeding from SCADA environments.

6.1 Example of SCADA alarm

Now we will see an example of how alarm information is completed with aditional data mainteined by the AMS. The AMS has stored relatively static information like the backup status

of the managed objects and probable causes related with specific problems.
Let's assume that we have a DHC called "CMD-Castilla". This DHC has an SCADA MMI, and its related entry at the puEntityTable has the following data :

puEntityIndex	190
puEntityOS	"Sun Solaris 2.5"
puEntityMachineType	operationServer
puEntityDHC	"CMD-Castilla"
puEntityStatus	MANAGED
puEntityScadaStatus	NC
puEntityLastInitialisation	4326149
puEntityIpAddress	"193.144.50.11"
puEntityAgentPort	2010

The operation server is continuously checking the state of the SCADA application. When the test fails, the SNMP agent located at the server raises an alarm to the AMS. The only information the server receives is the machine that has generated the notification (in case of Operation machines, it will be a notification, see Figure 7),

puErrorsEntityIndex	190
puErrorsIndex	1902
puErrorsEventTime	4782934
puErrorsSpecificProblems	enterprisesSpecific

By the *specific-trap* field attached to the *trap* PDU, we can distinguish the kind of alarm it arised. In that way, the AMS can query its database and fill the rest of empty fields in the alarm MIB. When SCADA applications are not detected, this could be symptom of several faults. The most frequent one is a fault in the own SCADA application. Also the windows system crashes from time to time, but when this fact occurs, the machine blocks, so no trap would be provided by the SNMP agent. When SCADA applications crash at the MMI, this could be cause the momentary stop of SCADA applications at the rest of SCADA machines in the same DHC. This issue helps us to detect a first-level alarm correlation.
The information that will be filled by the AMS through the database will be the following :

puErrorsEventType	processingErrorAlarm
puErrorsProbableCause	softwareError
puErrorsSpecificProblems	enterprisesSpecific
puErrorsPerceivedSecurity	CRITICAL
puErrorsBackUpStatus	FALSE
puErrorsTrendIndication	MORESEVERE
puErrorsAdditionalText	"Crash of SCADA system"

The MIBs relating to correlated notifications and repair actions will be filled in with several data too. In the correlated notifications MIB will be included all the SCADA alarms raised at the same time by machines joining the same DHC. In the proposed repair actions MIB, the first entry will suggest the following processes : restart of SCADA applications, restart of windows system (which also includes restart of SCADA applications) and warning to the system operators.

7 Conclussions and future work

SNMP products have a great market share. Despite this, CMIP technology is increasing its significance in the world of management. Both protocols coexist, and its integration is a vital point in the current developments. This situation gets more complex because of the existence of proprietary protocols in great legacy networks like power utilities.

Our development proposes an integration of all these management protocols through a system which allows the enrichment of the information by accesing to repositories of data, with techniques of DBMS. With this scheme, we can manage events, traps and proprietary alarms through a single management system. The system is flexible enough to allow distributed management with filtering and redirection functions of all these multiprotocol alarms.
Applying this schema to alarm management in problematic environments like SCADA, which have a great amount of faults, we have obtained a single way to handle all alarms and distribute its treatment among several management systems, increasing the performance to take recovery actions.
Our system can be integrated in TMN management, allowing to include SNMP in current telecommunications management platforms which have neglected this protocol a lot.
New areas of research have been defined to improve the AMS. Nowadays, AMS only supports one-level alarm correlation. Besides, all the achieved management is developed around network-element level. Taking ideas and concepts from TMN standards, we are trying to design a more object-oriented model, focussing our efforts in CMIP management, and extending the management to network level studies.
Another research field is the distribution of functions of the AMS among several modules, avoiding problems inherent to metamanagement, like the filtering of alarms at the manager. We are working on intelligent agents which perform filtering and scoping, decreasing the tasks of the manager and improving its performance.

8 References

[1] ITU-T M.3010, *Principles for a Telecommunications Management Network*, 1992.

[2] ITU-T X.710. ISO/IEC 9595, *Information Technology - Open Systems Interconnection - Common Management Information Service - Part 1: Specification, (CMIS)*, 1990.

[3] ITU-T X.711. ISO/IEC 9596-1, *Information Technology - Open Systems Interconnection - Common Management Information Protocol - Part 1: Specification, (CMIP)*, 1990.

[4] ITU-T X.721. ISO/IEC 10165-2, *Information Technology - Open Systems Interconnection - Structure of Management Information - Part 1: Management Information Model*, 1991.

[5] ITU-T X.733. ISO/IEC 10164-4, *Information Technology - Open Systems Interconnection - Systems Management - Part 4: Alarm Reporting Function*, 1992.

[6] ITU-T X.734. ISO/IEC 10164-5, *Information Technology - Open Systems Interconnection - Systems Management - Part 5: Event Reporting Management Function*, 1992.

[7] Jakobson G. and Weissman M.D., *Alarm Correlation*, IEEE Network, pp. 52-59, November 1993.

[8] RFC 1157, *Case, J et. al., A Simple Network Management Protocol*, 1990

[9] Muñoz, F., Carneiro, V.M, *Integrating SNMP and CMIP Alarm Processing in a TMN Management Environment, IFIP97, Italy, 1997.*

[10] ITU-T X.736. ISO/IEC 10164-7, *Information Technology - Open Systems Interconnection - Systems Management; Security Alarm Reporting Function, 1992.*