New Design Schemes for Lightweight TMN Mediation Device/Q3 Adapter

J.J. Lim,
Network Management S/W Lab.,
SAMSUNGELECTRONICS,
Bundang P.O.Box 32,463-050,Korea
jjlim@telecom.samsung.co.kr

K.W. Kim,
Switching S/W Lab.,
SAMSUNG ELECTRONICS,
Bundang P.O.Box 32,463-050,Korea
kkyewan@telecom.samsung.co.kr

Abstract

This paper describes new design schemes of telecommunication network management (TMN) agent for man machine language (MML)-managed network elements (NEs), especially for ATM networks. Most of currently developed ATM switches have simple MML interfaces for local operator crafts. This means that most operations, administration, maintenance, and provisioning (OAM&P) functions reside within the NE itself, and designing TMN agent as a proxy is considered to be most appropriate in that more intelligent event correlation and filtering can be done in NE itself. In TMN environment, proxy based management expects an agent to have a functionality of Q adaptation function (QAF), rather than network element function (NEF) itself. In contrast with NEF, QAF can be simplified by focusing on the message conversion, namely adaptation and mediation, between standard CMIP message and proprietary MML message or any kind of management message. Based on the observed fact above, this paper proposes several design schemes for Q3 agent and its companion gateway function, so that a Q3 agent, as a proxy, can rather be lightweight and mediation insensitive. The design schemes proposed in this paper are listed in the following: designing management protocol insensitive schema for NE-embedded database (DB), mapping of management information base (MIB)/management instances tree (MIT) accesses to DB queries and making direct use of DB queries, minimizing the retained information on MIT/MIB, and using an GDMO/ASN.1 based MML mediation capabilities with various message adaptation capabilities.

Keywords

MML-managed NE, NE-embedded database, mapping of MIB/MIT accesses to DB queries, GDMO/ASN.1 based MML mediation.

1. Introduction

In recent years, telecommunication networks have become more diverse and complicated. The telecommunication services also tend to be diverse as the capabilities of underlying telecommunication infrastructure evolve in terms of

software and hardware technologies. So efficient and effective network management becomes more significant to the orderly operation in large, multi domain networks. The telecommunication management network (TMN)[1] that was recommended by ITU-T provides the effective concepts to achieve interoperability and also provides efficiency to implement heterogeneous multi-domain management system.

TMN, based on the OSI system management concept, models the managed system using a set of managed objects (MOs). These MOs are classified into two classes: resource MO (RMO) and support MO (SMO). The former represents real resources in a managed system and the latter represents abstract functions conducted in a managed system. Those MOs constitute the MIB/MIT representing the information and operations for a managed NE. Currently many efforts have been made to define MIBs[2,3], by standardization organization such as ITU-T and ATM Forum, and its runtime aspects, namely management application framework (MAF), by many TMN toolkit vendors.

On the other hand, network management has a tendency to make heavy use of database. The database may be either normative relational DB or object-oriented DB, or any form of main memory resident DB. The MIB of a TMN agent is a typical example of main memory resident DB. Generally, several deployments of DBs bring off a DB consistency problem. This is true of a proxy based TMN agent. But typical implementations[4,5,6] of such a proxy agent tend to interpret MIB as real repository, in which case MIB has the same, but rather standardized management information as embedded DB of NE. Therefore it can be safely said that a proxy agent, or shortly Q3 adapter, has redundant information, thus having such a potential problem.

The intention of OSI manager-agent model still seems to be thought of having standardized management interface or message interface, although recent shift in software architecture of NE shows that features supported by built-in software are being handled and redefined through standardized MIBs, namely MIB supportable. This interpretation for proxy-based management enables to invent new design schemes for Q3 agent and its gateway function. With most OAM&P functions delegated to the real resource management functions in NE, an agent simply focuses on the adaptation and mediation of non-standardized messages to Q3 CMIP messages and the vice-versa. This interpretation is contrasted with the traditional approaches in that they treat MIBs as real repositories maintained and viewed at an agent itself, not as conceptual repositories translated and mediated by an agent.

Although this is the general case, interpretation in this way makes an agent heavy, especially for Q3 adapter. Based on the fact above, this paper proposes design schemes for a so-called lightweight agent targeted for proxy-based management over MML-managed NEs. The term "lightweight" means that designing and implementing MIB of an agent is done such that only containment and association relationships are represented. The accesses by managing system to the managed resources are through the adaptation and mediation of a proxy agent. For this, a proxy agent needs a direct access to embedded DB, rule database for

mediation, adaptive message interface to MML managed-NEs, and separation of agent's pure Q3 functionality from mediation function by defining a restricted Qx functionality around M interface.

Finding solutions for integrating TMN and legacy system may be worth challenging in that it has the essential points for accommodating the various management protocols or for interworking several managing systems. To overcome the fixed Q3 interface over multi-domain, multi-protocol environments, adopting more open technologies are prevailing on these network management trends. Besides the Internet SNMP, the World Wide Web (WWW)/Java [7,8] and common object request broker architecture [9,10] are good examples of current trends[11,12,13].

The expected result of this paper is to show the road to surviving integration methods to meet such emerging technological trends and needs. The organization of this paper is as following. Section 2 explains the proposed functional architecture and its philosophies. Section 3 goes into detail about each functional component and its implementation. Section 4 gives a concluding remark.

2. Architecture Descriptions

2.1. Functional Components and its Philosophies

Figure 1 shows the simplified architecture of an agent, which has interfaces of both Q3 and M interface [1]. Q3 interface is often described as a set of CMIP/S and FTAM, whereas M interface is where a gateway function resides for translation of standard information model into proprietary information model and the vice-versa. The underlying concepts of Figure 1 are listed in the following: extensibility and robustness, transparent distribution, and lightweight property.

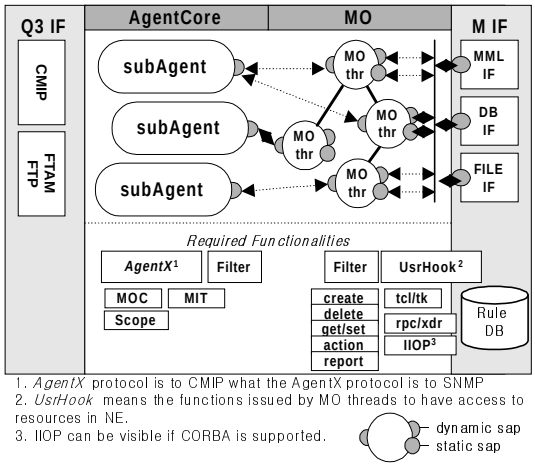

Figure 1: Simplified Views and its Functional Components for a Lightweight Agent.

- **Extensibility and robustness**: Extensibility means that when we design an agent, it is possible to keep an agent design simple and component-based on each device without taking the volume of managed information into much account. A sub agent takes charge of each managed resource independently of other sub agents. Thus robustness can be improved by making other sub agents running on separate process spaces.
- **Transparent distribution**: Partitioning the managed resources into a set of domains maintained by each sub agent counts, only if the partitioned domains are made available in a way that an managing system can access to and receive events from the managed resources transparently. As for managing system, it shall manage a logical view of MIB as a single unit without knowing the partitions and existences of multiple sub agents.
- **Lightweight property**: To lessen the burden on an agent and to rid an agent of DB inconsistency and update problems, interpreting an MIB as less as necessary to adapt and mediate messages matters. This can be achieved with well-defined DB schema in NE, mapping rules/APIs translating CMIP messages into DB queries. In this sense, an agent just depends on the embedded DB for modifying and retrieving the managed information. Any other real resource management is done by applications in NE, with their results being kept traced within the embedded DB.

2.2. CMIP AgentX (Agent eXtensibility) Protocol

Defining functional components in operations system (OpS), configuring the functional components, and building OpS in a hierarchical way are well defined in ITU-T Rec. M.3010. Its underlying concept is logically layered architecture (LLA) and its real applications are described in other papers[14,15]. Less attention, however, are given to the framework of an agent than that of a manager. This is partly because managing system, together with its requirements for diverse user/application objectives, has been more complex, and partly because neither performance nor real-time property for an agent system has been considered to be critical.

However, our experiences during the development of TMN agent for ATM switch reveal that there is a necessity to divide MIB over a set of units on each switching subsystem. They also reveal that some management operations shall not block the processing of another independent management operations, or must not be influenced by the running states of other switching subsystems. The SNMP AgentX (Agent eXtensibility) protocol[16], as a proposed IETF standard, seems to be first publication handling this agent extensibility problem.

The proposed SNMP AgentX protocol allows multiple sub agents to make MIB information available in a way that is transparent to SNMP management applications. Although the AgentX protocol is only applied to SNMP, concept of sub agent can also be applied to CMIP (denoted by italic *AgentX* in Figure 1,*CMIP-AgentX*, or shortly *AgentX*). To keep the proposed transparent distribution

property, the *AgentX* protocol between the master agent and sub agents must be defined in a CMIP context. Moreover, each role to be conducted by the master agent and sub agents shall be discerned. The protocols, besides normal CMIP procedures, include a set of *AgentX* specific procedures: administration procedure for sub agents, dispatching procedure for CMIP messages.

The administration procedures of the *AgentX*, whose main features are naturally adopted from the SNMP AgentX, have the following sub procedures:

- **Session establishment/release procedures:** When a sub agent wishes to start/stop the communications with the master agent, this procedure is meaningful.
- **MIB registration/unregistration procedures:** When a sub agent is started, it contacts the master agent and registers/unregisters the various MIB domains for which it takes responsibility. A sub agent submits a descriptor for its MIB domain, and the master agent resolves any registration conflicts between sub agents. The registration description may take form of specifying which ObjectClass, under which ObjectInstance, what AVA (Attribute Value Assertion) conditions, and et.al.

The dispatching procedure is to forward the received CMIP message to a sub agent charge of handling it. Especially when an updating message such as M-SET is received, the master should apply its access control policy: best effort or atomic. Thus it has the following sub procedures:

- **Commit/Undo procedures:** Based on the received access control policy, the master agent must preserve its property, even when the designated MOs are accessed by different sub agents.
- **Prioritizing procedure:** For fault tolerant sub agent, it may be allowed to register multiple sub agents with the same MIB domains. A registered sub agent then may take precedence over the other sub agents with priority information. Thus received CMIP message is normally dispatched to and handled by a sub agent with highest priority.

2.3. Lightweight MO Definitions

As contrary to common implementations, each MO may be implemented with minimal set of in-memory attributes under the assumption that Q3 supportable embedded DB or well-defined mapping rules/APIs is possible. Realizing MOs with all the defined attributes in main memory or in persistent storage is natural. However, to update all the attributes values for a MO is time-consuming. For example, to set up an ATM-Forum M4[2] compliant point-to-point virtual circuit connection in ATM NE, it requires the traversals on several MOs like uni, intraNNI, vpTTPBirectional, vcCTPBidirecitional, atmFabric, and atmCrossconnection. Under light load on the system it can be negligible, but under heavy stress test it cannot be negligible. This is also true of millions of MO instances.

To lessen the load on an agent and to gain performance benefits, we choose to design MIBs with the minimal set of attributes for given MOs. The minimal set of

attribute represent only containment relationships, together with association relationships such as SBO (supportedByObjectList) and AOL (affectedObjectList). Thus MIB snapshot in any time may be like Figure 2.

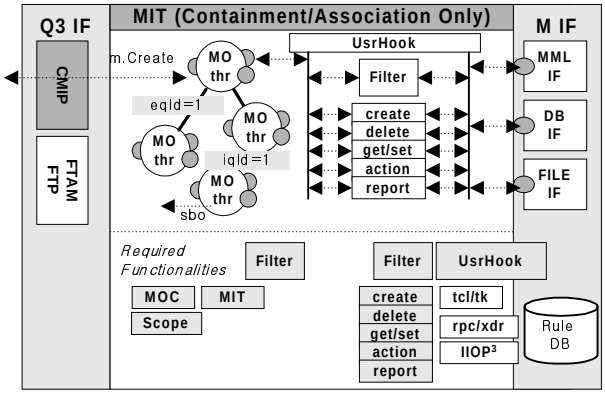

Figure 2: Lightweight MO Definitions with Containment Tree and Association Relationships.

To make the underlying assumptions realistic, it is required to implement the *AgentX* protocol. Furthermore, we should define mediation scenario for each received CMIP message with MO's behavior algorithm and defined user-hook functions. Fortunately, currently available TMN toolkits by many vendors meet this *AgentX*'s requirement. Thus there is no obstacle to do this. As for mediation scenario, it must describe the steps necessary to access the managed resources via defined M interface. The mediation may take approaches of either direct translation or abstract translation in [17], or both. As explained previously, however, all of these complexities depend on the Q3 supportabilities of both an embedded DB and NE's information model, and on the ease of getting services provided by applications in NE.

2.4. Mediation for M Interface

What is important to the mediation function is to avoid unnecessary coupling with a general Q3 functionality of agent side. To take examples from Figure 2, there are many implementations as to where the *UsrHook* functionality is placed: in either NE side or Q3 side, or as another entity. Therefore the more tightly coupled mediation is, the less modularity and reusability of an agent's Q3 functionality are achieved. The problems caused by a tightly coupled mediation may have lacks in accommodating the subtle changes of NE side software.

To avoid these problems we invent a loosely-coupled mediation architecture by defining the mediation-specific SMOs. All of mediation algorithms and differences between software versions are handled by them, whereas the existences of mediation-specific SMOs are hidden from a managing system. These mediation SMOs and CMIP operations for the *UsrHooks* constitute the proposed Qx functionality.

Another consideration for mediation is a binding scheme of scenario to the defined M interface. The scenario can be linked in either a dynamic binding or static binding. Dynamic binding means that we modify the scenario in run-time without affecting the running state of an agent, whereas static binding means that each scenario is determined in compile-time. Tcl/tk[18] script language is a good example of such dynamic binding, which can read each scenario from rule database, and its applications for management accesses are also described[19]. On the other hand, RPC/XDR[20] or even CORBA can be suitable for a static binding.

2.5. Adaptation for M Interface

Adaptation can be thought of translating messages from one form to another. This is dependent on the number of management protocols among them. Instead of defining management protocol dependent message structure, it would be nice to define a lot general message container. This property is more required when we design the NE interface. Therefore the desirable features of adaptation is analyzed on NE's part.

Stepped away from management protocols, we can simplify the relationships among the NE, GUI, and its underlying management protocols. Traditional MML-managed NEs can safely be said to have two invariant parts: GUI and NE, regardless of underlying management protocols such as CMIP, SNMP, and WWW/Java. This means that an operator accustomed to MML-managed NE expect to use a uniform interface, regardless of its management protocols. In this case, a management protocol is merely management middle-ware, whose main functions are just marshalling/un-marshalling messages, dispatching messages to appropriate object, not processing the messages.

Admitting of leaping forward in our assumptions, we design the proposed adaptation capabilities based on the following criteria. One is adapting simple ASCII strings for MMLs or SQL queries, another is adapting a meta data structure for MMLs. And the third is adapting the general byte streams, which aim to incorporate the concept of interface definition language (IDL). Given these capabilities, any kinds of interface messages are believed to be translated to the application's requirements accessing the NE.

3. Prototype Implementation

3.1. Overall Functional Configuration and Its Interaction

The implemented functional architecture for a lightweight agent in Figure 3 has typical functional components. There are general Q3 MO handling block and mediation blocks. And the functional architecture is component-based on its management functional area basis. This architecture is similar to that of our manager system[14,15], exactly a mirrored architecture. Each MO is conceptually grouped and constitutes the management functional areas such as configuration management, fault management. However, this grouping does not indicate any implementation addendum or restriction between MOs. Every MO has the same

interfaces to both *AgentCore* block and *UsrHooks* as another MO.

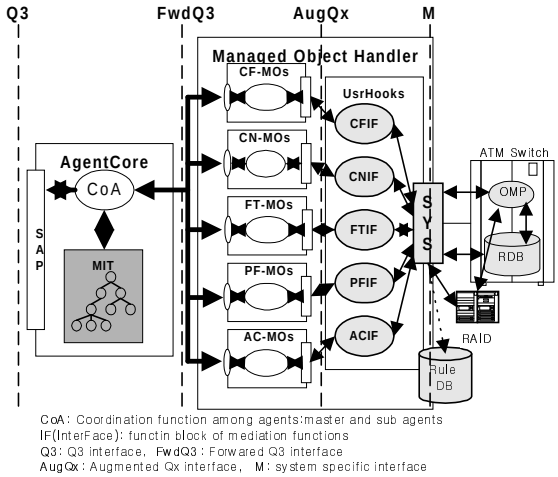

Figure 3: Overall Functional Configurations.

The *AgentCore* block takes responsibility for forwarding the received CMIP message to target MOs. This dispatching, as already mentioned in the previous section, is based on the containment and association relationships only. The *UsrHooks* block is an implementation of adaptation and mediation. Between them, MO threads implementations are placed. Each MO has the same control algorithm and interfaces with each other. Each CMIP message received by *AgentCore* is forwarded to the target MO threads. The MO threads extract the meaning of operation for the received message, and extract a list of {type,value} pair for managed attributes. Then they pass them as an augmented Qx M-ACTION request to its user-defined hook functions.

For autonomous messages from NE or response messages, the user-defined hook functions generate a list of Qx M-ACTION reply or M-EVENT-REPORT indication messages. And they pass them to a target MO thread. For both mediations to succeed, there must be well-defined database directing the mappings. The *Rule DB* provides the necessary information for both mediations. And the *UsrHooks* block communicates with the *OMP* by exchanging MMLs. It also accesses directly the embedded DB with SQL queries, or RAID with FILE interface. All of communications between the *UsrHooks* and the NE are provided by the adaptation services defined at the *SYS*.

3.2. Embedded DB Interface and Rule DB

To support CMIP attribute-oriented operation with a direct access to the embedded DB, we firstly implement the *DB provider* in both the *SYS* and NE. The NE side provider is called as a *DB Server* and the *SYS* side is called as a *DB Accessor*. In fact, the *DB Server* is just a SQL query processor with its DB engine. The *DB Accessor*, as a child process of the *SYS* block, has inter-process communication

facilities for other modules in the *SYS*. Furthermore, its GUI interface for an operator is also provided as shown in Figure 4. Using this facility an operator verify the results of queries issued by the *SYS*.

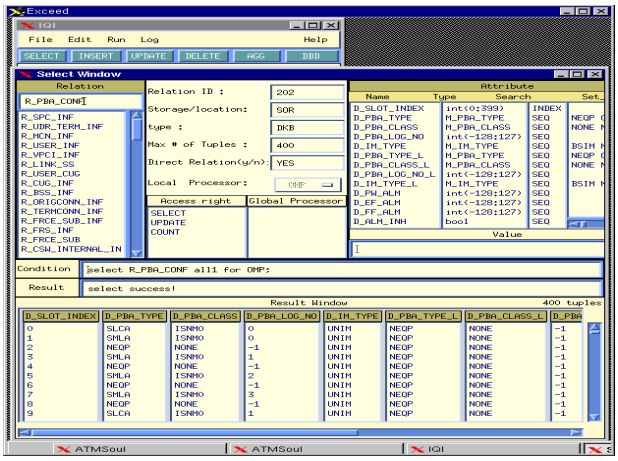

Figure 4: Screen Dump of a DB Accessor GUI.

The *SYS* side provider makes it possible to access directly the relations and their tuples of the managed resources. These relations and tuples for a given attribute of each MO are also defined in the *Rule DB*. In reality, the cores of *Rule DB* are just the specification of lists of {*mo, attribute – relation, tuple*} relationships. The permitted access operations for current implementations are select, update, and count. The select operation gets the tuples matching given SQL queries for a designated relation. The update operation set the vales of tuples matching given SQL queries. And the count operation just counts the number of tuples matching given SQL queries.

3.3. Mediation for M Interface

To provide the proposed lightly coupled mediation, we define the simple Qx interface. The Qx interface between the Q3 agent and its companion gateway has the only form of restricted CMIP messages: M-ACTION and M-EVENT-REPORT. All of the received Q3 CMIP messages are converted into Qx M-ACTION only. The behavior implementation of M-ACTION is sequencing the steps for mediation and adaptation. Based on the defined mapping rules in *Rule DB*, their action information and notification types are determined. In this way we can eliminate the dependency of Q3 agent to the NE as minimum.

Although both behaviors of M-ACTION and triggering condition for a specific NE may be different from another, The Q3 agent will see the same information under the same GDMO/ASN.1 specifications. Because any agent using the same GDMO/ASN.1 specifications shows the same information retained in GDMO/ASN.1 specifications only. All the differences are mediated at the

UsrHook function using Qx functionality. Figure 5 shows the implemented mediation configurations.

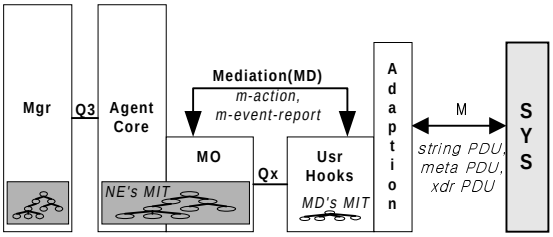

Figure 5: Loosely-coupled MD Configuration.

3.4. Adaptation for M Interface

Legacy operations systems has a set of system specific MMLs to control and monitor the NE. The MML is eligible for system diagnostics, but in case of further processing such a non-human processor it has deficiency in decoding the MML messages. This is the case for interworking the other managing systems using the different management protocols for further processing.

To overcome this problem, three types of protocol data unit (PDU) for generalized inter-process communications are defined: *string* PDU, *meta* PDU, and *xdr* PDU. The *string* PDU is just prettily printed char streams of ASCII strings. The *meta* PDU has a generalized *meta* data structure to convey a sequence of type-value pairs with format effectors for any MML message. And the *xdr* PDU incorporates the concept of IDL which is currently most promising technology for the exchange of information. Current limitations allow only *xdr* PDU to contain any kind of interface messages. And other two types, *string* PDU and *meta* PDU, are only allowed to convey the MML interface messages and DB SQL queries.

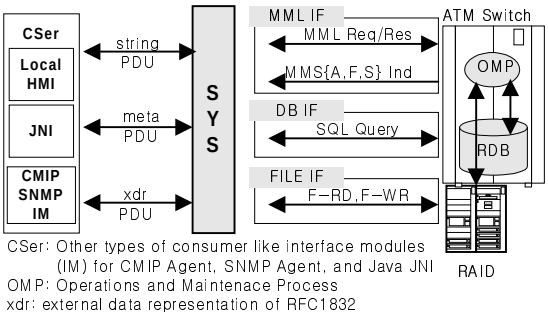

Figure 6: Adaptation Capabilities.

Given these adaptation approaches and capabilities, we are able to freely attach other types of message adapters into MML-managed NEs. This adapter contains the Java native interface (JNI) interface modules for CMIP, SNMP, and needless to say for legacy HMI console. This indicates the ease of integration with other

management protocols and means that a generalized NE interface can be possible. Figure 6 shows such examples of deploying local operator craft and any other consumer modules with the proposed adaptive interface.

4. Conclusion

This paper proposes new design schemes for a lightweight TMN Q3 agent and its loosely-coupled gateway function. The term "lightweight" means that designing and implementing MIB of an agent is done such that only containment and association relationships are represented. The accesses by managing system to the managed resources are through the adaptation and mediation of a gateway. For this, there are several inventories including a direct access method to the NE-embedded DB, a rule database for mediation, an adaptive message interface to MML managed-NEs. In an effort to make these schemes successful, required capabilities are proposed and implemented partially.

One of current issues in telecommunication network is to integrate legacy systems into TMN. The other is to allow for more open interfaces such as WWW/Java, CORBA, SNMP, instead of strict adhering to CMIP based TMN. To meet the current trends, the proposed schemes and techniques are meaningful and effective for employing the integrated network management systems. Finally, remaining works, mentioned but not covered in current implementation, are the definition of protocol-insensitive DB schema from the standardized MIBs definitions. Applying of agent extensibility protocols and fully implementing the lightweight MIB/MIT are still in progress. The continuing researches on the not resolved design schemes, on the problems of implementing the proposed schemes, and on the performance comparison with the traditional agent system will be conducted in near future.

References

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

[2] The ATM Forum af-nm-0027.000, "CMIP Specification for the M4 Interfaces", Sep. 1995.

[3] Adrian Manley, Clare Thomas, "Evolution of TMN Network Object Models for Broadband Management", IEEE Comm. Mag., pp 60 – 65, October 1997.

[4] M. Feridun, et.al, "Implementing OSI Agent/Managers for TMN", IEEE Comm. Mag. Sep. 1996, pp 62~67.

[5] Roch H. Glitho, et. al, "Approaches for Introducing TMN in Legacy Networks: A Critical Look", IEEE Comm. Mag. Sep. 1996, pp 55 ~ 60.

[6] Tomoaki Shimzu, et.al, "Implementing and Deploying MIB in ATM Transport Network Operations System", E-mail:shimizu@ntttsd.ntt.jp

[7] Java Soft, "Java Management API", Specification 1.0, Sep, 1996,

[8] Free Range Media, Inc., "Industry Leader Propose Web-Based Enterprise Management Standard Efforts", July 17,1996,

[9] Douglas C. Shmidt, "An Overview of the Common Object Request Broker

Architecture (CORBA)",
[10] Object Management Group, "CORBA-based Telecommunication Network Management System", OMG Whitepaper Draft2, January 1996.
[11] Qinzheng Kong and Graham Chen, "Integrating CORBA and TMN Environment", CiTR Technical Journal, an later version of IEEE/IFIP Network Operations and Management Symposium 1996.
[12] Subrata Mazumdar, "Inter-Domain Management between CORBA and SNMP: WEB-based Management – CORBA/SNMP Gateway Approach", DSOM '96, L'Aquila, Italy, October 28-30, 1996.
[13] Subrata Mazumdar, "Inter-Domain Management: CORBA, OSI, SNMP", IM '97, San Diego, May 12, 1997,
[14] S.H. Lee, J.J. Lim, W.S. Kim, "Layered Architecture of TMN Management Application Framework for ATM Switch in NEML and Sub-NML Layer", ICT '97, Melbourne, Australia
[15] S.H. Lee, W.S. Kim, J.J. Lim, "A Proposal on Design Scheme of TMN NEML Management Application Framework for ATM Switching Systems", ICC 97, Montreal, Canada
[16] RFC 2257, "AgentX Protocol Version 1.", M. Daniele, B. Wijnen, D. Francisco. January 1998.
[17] Eckhart Koerner, "Design of a proxy for managing CMIP agents via SNMP", Computer Communications, pp 349-360, 1997
[18] Ousterhout, "The Tcl Language and the Tk Toolkit, Reading, MA: Addison-Wesley, 1994.
[19] George Pavlou, et.al, "A CMIS-Capable Scirpting Language and Associated Lightweight Protocol for TMN Application", IEEE Comm. Mag., Sep. 1996, pp82~87.
[20] IETF RFC 1832, "XDR: External Data Representation Standard", August 1995.