Experience of Modelling and Implementing a Quality of Service Management System

George Pavlou; University College London
Andreas Mann; IBM Heidelberg
Uffe Harksen; Copenhagen Telephone Company

Abstract

This paper presents the experience from modelling and implementing a Telecommunications Management Network (TMN) Quality of Service (QoS) management system, based closely on the principles of the evolving TMN standard [M.3010]. The latter is based the OSI Management model and suggests the use of its protocols. The management information models (Managed Objects - MOs) for each TMN layer, their relationships and the modelling decisions are presented, along with an implementation based strictly on real OSI management services and systems management functions. All management capabilities are expressed as managed object properties and all interactions are based on hierarchical manager-agent relationships of TMN Operations Systems (OSs). The full system development cycle, our experience from this modelling and implementation work and our conclusions are presented.

1. Introduction

In future Integrated Broadband Communication Networks (IBCNs) there will be complex provider and user schemes. Lower level service providers will sell their services to higher level service providers, the latter being their users. At the top of this layered service hierarchy, end users will be using underlying services.

The TMN prototype system developed models three "stratas" [ATMOS] of service providers:

a. a basic bandwidth provider, providing point-to-point links

b. an ATM connection provider, providing ATM connections

c. a service provider offering higher level services i.e. teleservices.

The teleservices span from telephony to file transfer, interactive data transfer, video broadcasting and conferencing etc. End users make use of these services which results in a layered service hierarchy consisting ultimately of four layers. The prototype management functions cover the top three layers (users, teleservices and ATM connections), basic bandwidth management being outside its scope. The service management functions in particular cover the teleservice and user layers and focus on Quality of Service (QoS) management.

The evolving TMN standards [M.3010] are based on the OSI Management model, enabling the management of resources (and thus services) through abstractions of them known as Managed Objects (MOs). A layered logical decomposition is suggested which recommends a similar structure for the associated management information models.

The service management part of the prototype has been developed to conform closely to the emerging TMN architecture and uses real OSI management protocols and systems management functions (event reporting, logging etc.) A hierarchical structure has been adopted for the service TMN management information model (MOs) and all the management capabilities regarding the associated resources (services, users etc.) are expressed as managed object properties. The service TMN Operations Systems (OSs) i.e. service management applications, are organised in a logical layered architecture and all the interactions between them are according to hierarchical manager-agent relationships.

In this paper the approach to the modelling, design and development of this TMN service management system is explained, highlighting models and sources of influence and explaining important design decisions and experiences gained through this process. Its realisation gives insight into the problems of structuring TMN applications and systems. It must be noted that in the absence of a real IBC environment, the network, services and users are simulated but the actual QoS management functions are based on real OSI management services.

The rest of the paper has the following structure: an overview of the system development cycle is given first, starting from the requirements and sources of influence until the actual prototype and other related outcome (experience, infrastructure etc.) Then the sources of input to the specification and design process are explained, comprising service and network management models and the particular prototype service management requirements. The result of this is the formal specification of the prototype service management information model complying to the TMN logical layered architecture, which is described next. Issues regarding the implementation architecture of the above model are then presented, including communication aspects and the structuring of management applications (OSs). Finally the experience from modelling, designing and implementing this system is presented along with our conclusions.

2. The System Development Cycle

Before examining in detail each of the various stages involved in modelling and implementing such a service management system, a first look at the full system development cycle will identify these and their relationships. Figure 1 gives an overview of the system development cycle.

The first source of input is the requirements specification. This is translated to the desired functionality of the service management applications, these being the handling of QoS complaints for service degradation and the generation of load predictions as explained later in detail. The application context is thus service management, focusing at Quality of Service as opposed for example to customer administration or other service management functional areas.

This brings up two important issues, *Management* and *Quality of Service*. Network Management and QoS models have being considered both by standardisation bodies and other RACE projects and together with the functional requirements they should be combined to provide the management solution, allowing for necessary deviations/extensions specific to the problem in hand.

Bearing in mind the conformance to the evolving TMN standards and the OSI management model, the target is to produce the prototype QoS service management information model in accordance to the TMN logical layered architecture. This will specify in effect all the capabilities related to the managed entities and will provide a framework for the development of management applications. The latter will use the information model (MOs) for their awareness and implementation functions according to the NETMAN ADI model [NETMAN], while the decision functions will be dictated by the system requirements (management policies).

Then in the implementation stage the information model together with the management policies should be translated to an implementation architecture. According to the Open Distributed Processing (ODP) viewpoints, the system model from the information viewpoint should be mapped onto an enginering model. Computational aspects should also be examined to provide the management policies and achieve the translation between adjacent layers of the information model. Finally, different technologies could be used to realise the system (technology viewpoint). The implemented system will be installed in the testbed and cooperate with the rest of the prototype which is system managing the ATM network. This cooperation assumes a common understanding of the management model, the communications infrastructure and the specific management scenarios.

The final outcome is not only the service management part of the prototype. Parts of the system that were considered generic and implemented that way may be reused in future applications in a similar context. As an example, the infrastructure for converting the management information model into a concrete MIB implementation and providing access to it i.e. the management platform, can be reused by any management applications following the same model. This is more facilitated if suitable technology, e.g. object oriented one, has been used for the implementation. More important, the experience of modelling the system in such a way may be used in future attempts to organise MIBs in a hierarchical fashion.

Figure 1. The System Development Cycle

3. Input to the Specification and Design Process

3.1 The Service Management Functions

The main objective of the complete prototype management system is to optimise the utilisation of the underlying ATM network, maintaining at the same time the quality of service to the levels negotiated with the service users. Service management can be viewed as a part of a control system, which should contain monitoring as well as active control functions. A complete service management system should contain applications to serve planning functions on all management levels. In line with the QOSMIC Timeline model [], these functions may be classified in three categories:

a. service level management, applications that support long term strategic planning of new services and phasing out of old services

b. user level management, applications that support the general management of service subscribers e.g. covering contract negotiation and user individual QoS monitoring and billing and

c. call level management, applications that monitor and verify the QoS of individual calls and provide active control functions.

The prototype focuses on the following types of service management applications:

a. monitoring and presenting service user behaviour over time and providing load predictions to the network TMN

b. monitoring and logging the provided QoS levels and instigate alarm generation in the case of degradation below certain thresholds and

c. monitoring lower strata QoS and complaining to lower strata service providers.

3.2 The Quality of Service Model

Before being able to measure the quality of service, a model is needed to define what quality of service is. For the purpose of this system, the following Network Quality of Service (NetQoS) parameters were considered as providing an indication of the "network" quality of service i.e. QoS at the level of the ATM connection provider:

N-CED:	Network Connection Establishment Delay
N-CSD:	Network Connection Signal Delay, the average end-to-end delay through the network
N-CSV:	Network Connection Signal Delay Variation, the maximum variation of the CSD
N-CDD:	Network Connection Disconnect Delay
N-CLR:	Network Cell Loss Rate, equivalent to the Bit Error Rate (BER)
N-CSQ:	Network Connection Signal Quality, a unitless number between zero and 1, where 1 denotes the perfect quality of service
N-CEP:	Network Connection Establishment Probability

For each teleservice, the equivalent Service Quality of Service (SrvQoS) parameters can be calculated based on the NetQoS ones provided by the ATM network. This requires the development of different calculation models for each teleservice, as the effect of e.g. cell losses will have different effect on a PCM-telephone and a Bulk File Transfer service.

As an example, in the PCM-telephone case the cell loss rate measured at the entry point into the ATM-network (N-CLR) will be mapped directly to an equivalent service one (S-CLR), indicating that there is no intermediate adaptation or compensation for cell losses. In the the Bulk File Transfer case, the S-CLR will always be zero, as there will be an intermediate cell loss detection and retransmission mechanism (e.g. the OSI FTAM), which in reality transforms cell losses to increased end-to-end delay (S-CSD) and reduces the effective mean bandwidth.

As a second example, a video conference is based on 6 or more ATM-network connections e.g. two for voice and 4 for the picture transfer. The calculation of the service connection establishment delay and probability (S-CED and S-CEP) is very complex and depends heavily on the connection establishment algorithm.

3.3 The Service Management Model

The prototype management model is based on the following models and principles:

- the Telecommunications Management Network standards [M.3010]
- the OSI Reference Model [OSI84] and in particular the Management Model [ISO91a][ISO91b], Service [ISO91c] and Protocol [ISO91d]
- the B-ISDN Reference Model [I.321]

- the ATMOSPHERIC Stratified Reference Model [ATMOS]
- the QOSMIC timeline model [QOSMIC]

The analysis of the input from these models showed that the design of a service management model for a layered services system requires a multidimensional model covering:

- the service layering, as proposed in [ATMOS] and [QOSMIC]
- the timeline concept as proposed in [QOSMIC]
- the protocol layering from [OSI84]
- the TMN layering as proposed in [M.3010]

A generic service management model for layered B-ISDN communication services complying to these dimensions was developed and is presented in [HARKS]. This model provides a generic way of relating MIBs of stratified management systems and the distribution of managed objects in the different management layers. The model comprises a generic dataflow model and a global conceptual information model. The dataflow model is based on the timeline one, which structures QoS management functions according to the following *timelines*:

- the Service Timeline, covering functions and information related to a service over its lifetime, scale being in the order of years
- the Customer-Provider Timeline, covering functions and information related to a customer-provider engagement over its lifetime, scale being in the order weeks or months
- the Call Timeline, covering functions and information related to the a single service call, time scale being in the order of seconds or minutes

4. The Service Management Information Model

The service management information model is the most important step towards the service management prototype as it specifies completely the management capabilities of the system. A managed object model has been developed for the service management information base, complying to the service and management models and principles described above. In particular, it is layered according to the TMN logical layered hierarchy, the managed objects of the higher layers providing increased abstraction of managed resources. This layering is *complete* as interactions between adjacent only layers are allowed.

At the time of the management information base specification for this prototype, there were no complete/detailed models for MIBs related to service management. The prototype MIB has been developed based partly on material from the following sources:

- the OSI generic management information model [ISO 10165-5]
- the TMN generic management information model [M.3010]
- the RACE Common Functional Specification (CFS) documents, mainly [CFSH550]
- the generic service MIB model described in [HARKS]
- the OSI Network Management Forum (NMF) object library [NMF90]

In the following two subsections we discuss the information model from two different points of view. Section 4.1 explains the classes of MOs arranged in the inheritance tree while section 4.2 discusses the instantiation of MOs, that is the containment hierarchies.

4.1 Inheritance Hierarchy

All MOs defined and implemented in the prototype are shown in Figure 2. This figure can be divided vertically and horizontally revealing different aspects of the information model.

Figure 2. Inheritance Hierarchy

Going through the inheritance hierarchy from left to right, three groups of MOs can be identified:

1. The three leftmost branches starting with actor, address and service are mainly derived from [CFSH550, NMF90]. MOs in this branch identify "resources" with respect to service management. It is obvious that some representation of service, service provider, user is needed. Some of them are further refined. As the main focus of the NEMESYS prototype lies on on-line performance management aspects as planning, accounting etc. are not reflected in the MOs properly. This is for effort reasons only, the general concept of these MOs has proven suitable.

2. The middle branch starting with serviceInfrastructure is mainly derived from [OSI 10165-5, M.3100], where network, node (to some extent similar to entity), serviceAccessPoint, protocolMachine and connection are defined. MO serviceInfrastructure is used to group all resources belonging to one service provider. It enables a service provider to distinguish resources required for different offered services. Last not least, when these MOs are instantiated a provider has to define carefully for what service this resource is actually needed. We found this MO helpful in structuring the layered service architecture.

3. The rightmost branch starting with profile contains all supporting MOs. Some of them as system, discriminator, log etc. are not shown. These support all kind of management applications and are defined commonly by OSI and CCITT [OSI 10165-2, CCITT X.721]. In this branch <figure 4.1> shows a specific NEMESYS MO called profile only. Profiles are needed to store various kinds of information, eg. how many network connections are needed for one service, what are the QoS requirements that are guaranteed to a user.

Going through the inheritance hierarchy from top to bottom, two different layers can be identified:

1. The upmost MOs of this figure are part of a somewhat "generic layer". MOs are independent of a service and are refined be for a bearer service offering pure network connections from A to B, say, be for a teleservice offering what NEMESYS calls a service association realising the transmission and computation facilities for a video conference, say. Most of the MOs in the generic layer as provider, user, service, network, node, connection, profile can be found in some standards or documents going to become standards.

2. MOs at the leaves of the inheritance tree are service specific. As mentioned above, a generic connection is refined to a networkConnection modelling the transmission facility a bearerservice provider offers to its users, that is to providers of teleservices. On the other hand, a generic connection is refined to a so called serviceAssociation modelling the facilities (comprising eg. transmission, adaptation, error correction) a teleservice provider offers to its users, the end user. Similar examples hold for the refinement of node, serviceAccessPoint and protocolMachine. For effort reasons only, we did not further refine eg. network, provider.

We found this concept of layering the inheritance tree into a generic and a service specific part very helpful. The separation of generic and specific parts in the inheritance hierarchy was recommended by other authors very recently [ENC paper - held at INDC’92 in

Helsinki <I will look for correct reference>]. Thinking of reusing the service management part of the prototype and its service information model in other projects we found no principle restrictions.

4.2 Containment Hierarchies

All classes of MOs instantiated in the prototype are shown in Figure 3. This shows the three levels of the service management system realised in the prototype and is a refinement of the Figure 4 in section 5.1. The functional decomposition regarding the management information trees is discussed.

Figure 3. Containment Hierarchies

The lowest level (SSA - Service Simulator Agent) implements a mediation function translating the service simulator language into a standard conformant one, eg. defining MOs, allowing CMIP communication. The communication between service simulator (NEMESYS's real resource) and SSA is private matter, communication between SSA and other managers is conformant to system management standards [OSI84, ISO91a, ISO91d].

The middle level (NELSM - Network Element Level Service Manager) manages the SSA and translates its information model into its own [MANN]. Among others, it implements a very important data reduction function, aggregating and compressing information over time. While SSA operates on real time data, NELSM aggregates information over some aggregation interval that may be changed by other management applications.

The upper level (SLSM - Service Level Service Manager) manages the NELSM and performs all kinds of administration work. Among others it aggregates and compresses information further. This condensed information may then be used (not implemented in the current prototype) by other management applications eg. to decide if a service should be stopped or the QoS values guaranteed to each user can be improved.

Again, Figure 3 can be divided vertically and horizontally revealing different aspects of the information model. Going through the figure from bottom to top three groups of MOs can be identified:

MOs instantiated at the SSA level represent mainly information out of the call timeline [QOSMIC]. This is especially true for serviceAssociation and networkConnection.

1. These MOs are created dynamically as often as some user requests a service. Thus, MOs at this level have a short lifetime compared to other ones as serviceAssociationAccessPoint.

2. These MOs are updated very often by the real resource; in NEMESYS the service simulator. Attributes of these MOs change values frequently and notifications may be emitted quite often.

3. The number of MOs to be handled at this level is huge and varies significantly over time.

1. MOs instantiated at the NELSM level represent mainly information out of the user timeline [QOSMIC]. A serviceAssociationAccessPoint exists as long as a user subscribes to a service. Together with the MO saProfile (instantiated in SSA) it reflects the contract between end user and teleservice provider and contains information eg. about the QoS the provider guarantees to deliver to this user.

2. MOs instantiated at the SLSM level represent mainly information out of the service timeline [QOSMIC]. MOs as service and provider exist as long as a service is offered. The users using this service may differ over time, but the service remains. Each service has default QoS values stored in the MO saapProfile. If in some contract more stringent QoS values are agreed

between user and provider, these values will be stored in MO saProfile as discussed above.

Going through the figure from left to right two different types of MOs can be distinguished:

1. MOs drawn with solid line represent resources controlled by the service manager, user, serviceAssociationAccessPoint, serviceAssociation being typical examples. For example a service management application can decide to base a service association on 4 instead of 3 network connections (say) in order to increase the efficiency of a high-speed FTAM service. Whenever a user asks for this service the corresponding saProtocolMachine has to get enough network connections from the network provider in order to set up the requested service association.
2. MOs drawn with dashed line represent resources provided by the underlying service provider (in NEMESYS the network provider). NetworkNode, networkConnection are typical examples. These resources are controlled by the network provider. The service manager needs a representation of them in order to measure delivered QoS and to express information to the network provider. For example, it makes no sense the service manager telling the network provider that 1000 (say) service associations of type A are expected to be set up during the next hour. Instead the network provider needs to know the estimated number of network connections for the next hour in order to provide enough transmission capacity in time. This is reasonable as the network provider does not know how many network connections make up one service association.

5. Implementation Aspects

5.1 The Overall Engineering Model

The management information model described should be translated to a set of hierarchically organised TMN operations systems. This organisation for the whole of the TMN prototype, including both service and basic bandwidth management functions are shown in Figure 4. The architectural model we described applies only to the service management functions, referred to for simplicity as "service manager". The basic bandwidth management functions are similarly referred to as "traffic manager". Last but not least, the IBCN network, the services and their users are simulated and referred to collectively as the "simulator".

Figure 4. The TMN Prototype Engineering Model

The network and services are found in the Network Element layer of the proposed TMN layered hierarchy, which comprises the managed elements. It should be noted that managed elements can be either physical or logical and services are logical managed elements manipulated by the service management functions. Services at that level exist physically at the customers premises and like other network elements (nodes etc.) they are outside the TMN. They should be managed though by functions in the Network Element Management layer above which is in the TMN.

Access to the network elements in a real system will mostly occur through non fully compliant management protocol stacks because of the complexity and overheads involved in realising these by simple elements. Though some services may be able to support them, translation functions (mediation devices) will be needed for some others to provide an object-oriented view according to the OSI management model. In the case of this prototype, the unit implementing the simulated service functions had been developed before the rest of the service management system and used a simple message passing mechanism as its management hooks.

The Service Simulator Agent (SSA) is thus exactly a mediation function box between the latter and the rest of the service management system. It must be added that the SSA is partly within and partly outside the TMN.

In the Network Element Management layer, there is an operations system which translates the lower level information model to its own through aggregation and compression functions. It is also at that level that the complaints regarding poor quality of service are produced and sent to the bandwidth provider's management system (Link Banwidth Manager OS) to help optimise the usage of the network resources. This operation system is known as the Network Element Level Service Manager (NELSM).

Finally, at the Service Level of the TMN hierarchy there is another operations system which has its own information model as described in 4.2 and is responsible for taking a global view at the service network and producing load predictions which are used by another part of the bandwidth provider's management system (Virtual Path Bandwidth Manager OS) to optimise the allocation of bandwidth to virtual paths. This operation system is known as the Service Level Service Manager (SLSM).

It is mentioned that in the prototype there are no service management functions in the Network Level of the TMN hierarchy: this is a very subtle and debatable point as is the allocation of the management functionality to the layers suggested by the example TMN layered architecture []. This decision is in line with the particular example architecture but more important is the use of such a hierarchical layered decomposition rather than the debate of which label is put on each layer.

The overall engineering model for the service management part of the prototype thus consists of four different units, each implemented by a separate process in operating systems terms: Service Simulator, SSA, NELSM and SLSM.

5.2 The Communications Infrastructure

There are two aspects in the communications infrastructure used in the service management prototype: the communication within the TMN and the communication with the managed elements (services) outside the TMN.

The communication within the TMN is based on the use of the OSI management service and protocol CMIS/P realised over a full interoperable OSI stack. These are q3 reference points realised as Q3 interfaces over CMIS/P. There is always a hierarchical manager to agent relationship between the management units involved (OSs and mediation function blocks), with units higher in the hierarchy acting in a manager role with respect to the units of the level below which act as management agents in that instance of communication. This recursive relationship terminates at the lowest level where real resources (services in this case) are managed. Units at the same (peer) level may act in both roles with respect to each other: the basic bandwidth provider's management units to which the NELSM and SLSM send their information may be thought as such.

The communication between the mediation function block (SSA) and the managed services (Service Simulator) is realised in a proprietary fashion which is whatever the service boxes support as management hooks. In our case, this is a simple message passing facility requiring no presentation services and is implemented using the Internet Transmission Control Protocol (TCP). This is a qx reference point which is converted to a Qx interface though TCP.

The implementation of CMIS/P is realised by the OSIMIS management platform [] which in turn uses the ISODE OSI protocol stack []. The latter allows interoperable services over different transport-service stacks: TP0 over X.25, TP4 over CLNP or even TP0 over TCP/IP using the RFC 1006 method []. It is the latter we have mainly used in this experiment which mostly operates on a local area network environment.

The communication with the rest of the system (basic bandwidth management) does no use fully conformant Q3 interfaces as the rest of the system uses a proprietary management platform based on TCP. Were it using the same management platform, communication at the Network Element Management level (QoS complaints) would be naturally realised as CMIS event reports while communication at the Service Management level (load predictions) could be modelled either as event reports or as information stored in the service network managed object using aspects of the workload monitoring systems management function [], which any interesting party could periodically read.

5.3 The Service Management Applications

Using CMIS/P is only part of the problem of exploiting the powerful features of the OSI management model. The difficult part is how to convert the formal specification of management information into managed objects in software and also how to provide necessary OSI systems management functions such as event reporting, logging, workload monitoring, access control etc.

The solution to this comes from the genericity and object-orientation of the OSI management model: if the generic features of the latter are implemented carefully, they can then hide a large part of the complexity of using CMIS. The systems management functions can also be implemented in a generic way and they could be "hidden" from managed object implementors.

This is exactly what is provided by the Generic Managed System (GMS) of the OSIMIS platform which provides an object-oriented Application Program Interface (in C++) for implementing managed objects with minimal complexity. It allows implementors to concentrate in the intelligence built into the managed objects and the communication with the associated resource rather than the mechanics of CMIS access and the complexity of the systems management functions. It should be added that complex but powerful aspects of CMIS such as scoping and filtering are completely hidden. Support is also provided for periodical polling functions and communication to the associated real resources though arbitrary communications mechanisms. This is what we have used to convert the information models for each layer into operations systems with agent capabilities (SSA, NELSM, SLSM).

An additional problem though comes from using CMIS in a manager role to access subordinate management information models e.g. SLSM to NELSM etc. Though this is not as complex as the agent part, a special object-oriented API was developed which made transparent the use of remote management information bases. The is known as the Shadow MIB (SMIB) method and allowed implementors to concentrate on the management policies rather than the mechanics of CMIS access. This infrastructure is now an integral part of the OSIMIS platform.

The object-oriented implementation architecture of the generic operations system is shown in Figure 5. Each such system is implemented as one process in operating system terms and handles an MIB instance (a management information tree). It offers an agent interface to peer and higher levels and acts as a manager itself with respect to peer or lower information models which its uses to implement its management policies. It may also translate lower information models into its own providing higher levels of abstraction.

Figure 5. The Generic OS Implementation Architecture

6. Experience Gained

(Is is really feasible to organise a TMN by modelling all management capabilities as managed object properties? In addition, is it possible to layer the information models hierarchically without allowing for violation of the layering principle? What are the pros and cons in organising a system like this).

(How easy it is to convert the specification of such a system into a concrete implementation? Which parts can be identified as generic in order to provide a reusable management platform?)

(Finally, which type of implementation technology we have found helpful in meeting our target.)

7. Conclusions

(Our conclusions.)

REFERENCES

[ATMOS]	Tuan Haduong et al., *Stratified Reference Model, An Open Architecture Approach for B-ISDN.* XIII International Switching Symposium Stockholm 1990 (From RACE Project 1014 ATMOSPHERIC)
[M.3010]	CCITT Draft Rec. M.3010, *Principles for a Telecommunication Management Network*, Rev. Melbourne March 1991
[NETMAN]	NETMAN Del. 6, *Draft Telecommunications Management Specifications*, Doc.id: 24/BCM/RD2/DS/A/006/b2, September 1990
[OSI84]	ISO/IS 7498, *Information Technology - Open Systems Interconnection: Basic Reference Model*, 1984
[ISO91a]	ISO/IS 10040, *Information Technology - Open Systems Interconnection - Systems Management Overview*, August 1991
[ISO91b]	ISO/IS 10165-4, *Information Technology - Structure of Management Information - Part 1: Management Information Model*, August 1991
[ISO91c]	ISO/IS 9595, *Information Technology - Open Systems Interconnection - Common Management Information Service Definition, Version 2*, July 1991
[ISO91d]	ISO/IS 9595, *Information Technology - Open Systems Interconnection - Common Management Information Protocol Specification, Version 2,* July 1991
[ISO91e]	ISO/DIS 10164*Information technology - Open Systems Interconnection Various parts (mainly 5 and 6 - Event Report and Log Control Functions)*, 1991
[ISO91f]	ISO/IS 10165-4, *Information Technology - Structure of Management Information - Part 4: Guidelines for the Definition of Managed Objects,* August 1991
[I.321]	CCITT Draft Rec. I.321, *B-ISDN Protocol Reference Model and its application*, Rev. June 1990

[QOSMIC] QOSMIC Deliverable D1.4B, *QOS Verification Methodologies*

[HARKS] U. Harksen, Copenhagen Telephone Company, *Service Modelling in NEMESYS*

[CFSH550] CFS H550, *Telecommunications Management Objects RACE Concertation Technical Workshop STG 4.3*, R1024 NETMAN, sec. 6, 18 June 1991

[NMF90] *Forum Library of Managed Object Classes, Name Bindings and Attributes*, OSI/Network Management forum 006, Issue 1.1 June 1990

[MANN] A. Mann; IBM Germany, G. Pavlou; University of College London, *Quality of Service Management in IBC: an OSI Management Based Prototype*, Proceedings of Fifth RACE TMN Conference

[KNIGHT] G.Knight, G.Pavlou, S.Walton, *Experience of Implementing OSI Management Facilities*, Proceedings of the IFIP Second International Symposium on Integrated Network Management, Washington, April 1991