HEWLETT PACKARD

# Extending the Fusion Design Process for TMN Application Development

Paul Jeremaes
Intelligent Networked Computing Laboratory
HP Laboratories Bristol
HPL-95-50
May, 1995

object-oriented development, TMN, MIB, network management, information model

Fusion is an object-oriented development method that provides a framework of analysis and design models and processes to support the full software development lifecycle. It was designed as a generic object-oriented method in that it assumes a *green field* or *clean slate* development, working from a set of initial requirements through to implementation. In practice, text book developments are rare. There will typically be domain specific constraints placed on the development activity which influence the way the system is developed and in some cases will require modifications to be made to the development process.

In this paper we investigate extensions to the Fusion design process to support the development of TMN management applications. The Telecommunications Management Network (TMN) is an architecture for the management and control of telecommunication sever provider networks. A set of international recommendations and standards define the TMN architecture and the function of certain elements in a telecommunication network. In what follows, we show how aspects of this architecture affect the way in which management applications are developed and result in additional phases of design being required.

Internal Accession Date Only

# Extending the Fusion Design Process for TMN Application Development

Paul Jeremaes

Hewlett-Packard Research Laboratories
Filton Road
Bristol BS12 6QZ
U.K.

## *1. Introduction*

Fusion is an object-oriented development method that provides a framework of analysis and design models and processes to support the full software development lifecycle [1]. It was designed as a generic object-oriented method in that it assumes a *green field* or *clean slate* development, working from a set of initial requirements through to implementation.

In practice, text book developments are rare. There will typically be domain-specific constraints placed on the development activity. These may take the form of application design constraints, made explicit in the requirements, or perhaps aspects of the application's embedding infrastructure. No matter what form these constraints take, they will almost certainly influence the way the system is developed and in some cases will require modifications to be made to the development process.

In this paper we investigate extensions to the Fusion design process to support the development of TMN management applications. The Telecommunications Management Network (TMN) is an architecture for the management and control of telecommunication service provider networks [2]. A set of international recommendations and standards define the TMN architecture and the function of certain elements in a telecommunication network [3][4][5][6][7]. In what follows, we show how aspects of this architecture affect the way in which management applications are developed and result in additional phases of design being required.

## *2. TMN Application Development*

The main driving force behind the TMN standardization activity comes from the telecommunication service providers, who wish to ensure that management applications obtained from different vendors will interoperate. Therefore, the TMN standards define an architecture, functional components that populate that architecture, interfaces to those components, and guidelines for application developers.

The full details of the TMN architecture are beyond the scope of this paper. However, one aspect that is of central importance to the application developer is the Management Information Base (MIB). Conceptually, the MIB contains all the management information about network elements (e.g. transmission/signalling equipment), network configuration, customer services, etc. In practice, the MIB consists of a distributed, heterogeneous collection of data sources. Management applications are required to interact with the MIB to gain access to the structure and content of management information and thus provide the necessary management and control functions (figure 1).

The TMN MIB is defined in terms of *managed objects* and the network management standards that define TMN use many familiar object-oriented concepts to define these [8]. A managed object can represent anything deemed important to the management of the telecommunications network, e.g. multiplexers, switches, subnetworks, software, etc., and

provides a management view of the resource concerned [9]. It is defined in terms of the management operations that can be performed on it, the behaviour of those operations, the data attributes that are visible at the management interface and the notifications that it is allowed to make about events that occur. Note that one network element may be modeled by hundreds, possibly thousands, of managed objects.

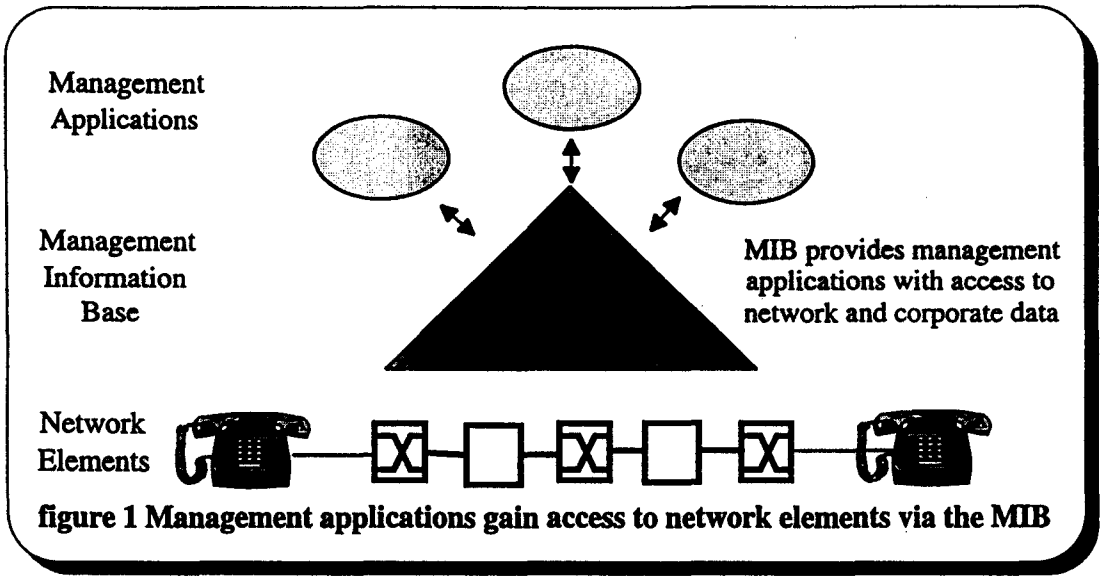


**figure 1 Management applications gain access to network elements via the MIB**

Managed objects in the MIB represent the network resources being managed and provide network information to a variety of management applications. Consequently, the developer of a management application has to produce a design with the contents of the MIB in mind. The design can not proceed based just on the operational and functional requirements of the particular application being developed. There is instead a constraint placed on the design activity by the information that is available from the managed objects in the MIB. This constraint can be looked upon as *bottom-up* design information (figure 2). A similar situation to this exists when a class library or application framework is to be used on a development project. The existing classes and structures in a framework, for example, have to be utilized in the application design, imposing bottom-up design constraints.

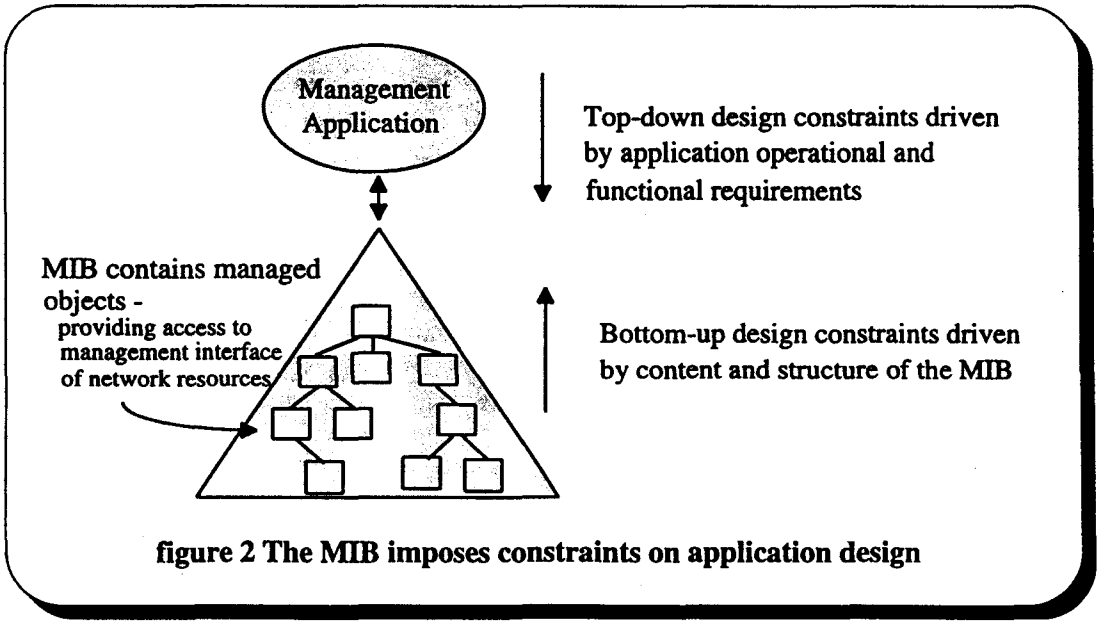


**figure 2 The MIB imposes constraints on application design**

The core Fusion method is essentially driven *top-down*, with the required *system operations* guiding the design process. The steps outlined in this paper show how the apparent conflict between top-down and bottom-up design constraints can be resolved when developing TMN applications requiring access to managed objects in the MIB. We start by considering the analysis and design models that are used in Fusion to establish the information model for an application. We then look to see how the information model has to be modified to take the MIB into account.

### *3. Application Information Models*

A major output of the Fusion analysis and design activity is the application information model, designed to support the required system operations. This is realised in terms of object classes, data attributes of those classes and the object valued attributes, providing visibility links between classes.

Initial attention is given to the information model with the construction of the *object model* during analysis. This captures the static structure of information in the application and its environment. The *system object model* is a subset of this object model and provides the foundation for the information model of the application to be built (figure 3). The dynamic aspects of analysis are captured by the two components of the *interface model*, namely the *operation model* and the *lifecycle model* (not illustrated).

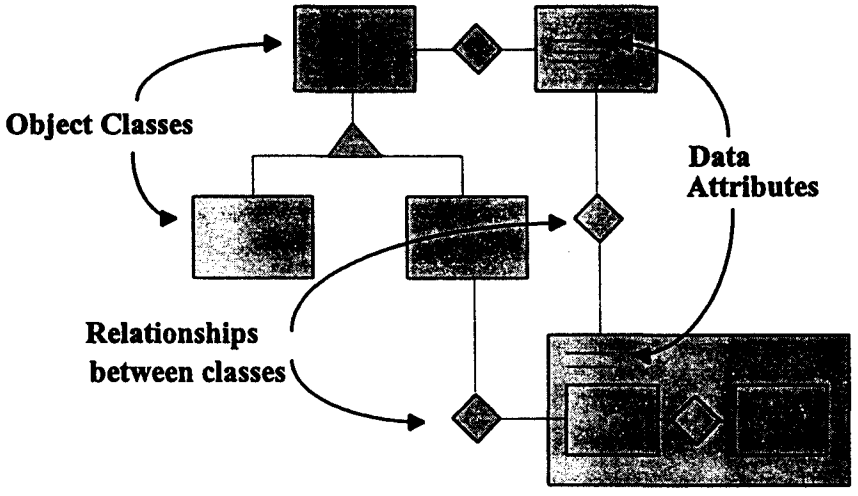


**figure 3 System object model captures static structure of information**

These analysis models establish a precise description of the requirements placed on the design activity. Although a skeleton of the application information model has been formed at this stage, it is not until the design really gets underway that the structure and content of information model starts to take shape.

The *object interaction graphs* have the greatest influence on this by identifying the need for specific object instances to send messages to other objects. It is this messaging behaviour that determines the visibility structures required by object classes, i.e. the object valued attributes which provide references to object instances to enable message passing (figure 4). These visibility structures *implement* the relationships, required by the design, that were initially identified on the system object model. There will also be data attributes introduced during the

development of the object interaction graphs to support the algorithms that implement the system operations. These data attributes, together with those identified on the system object model, also form part of the application information model.

For TMN application development it is the information model that has to be designed taking into account both top-down and bottom-up constraints. An information model has to be designed to adequately support the required functionality and also to derive much of its information from managed objects in the MIB. Cumbersome relationships may exist between an application and the MIB if the design of the information model is poor, leading to inefficient data access (causing unnecessary MIB *traffic*), redundancy and possible inconsistency of data.

To counter this potential danger, an appropriate stage in the design process has to be established where the top-down and bottom-up influences can be combined. In figure 5 the

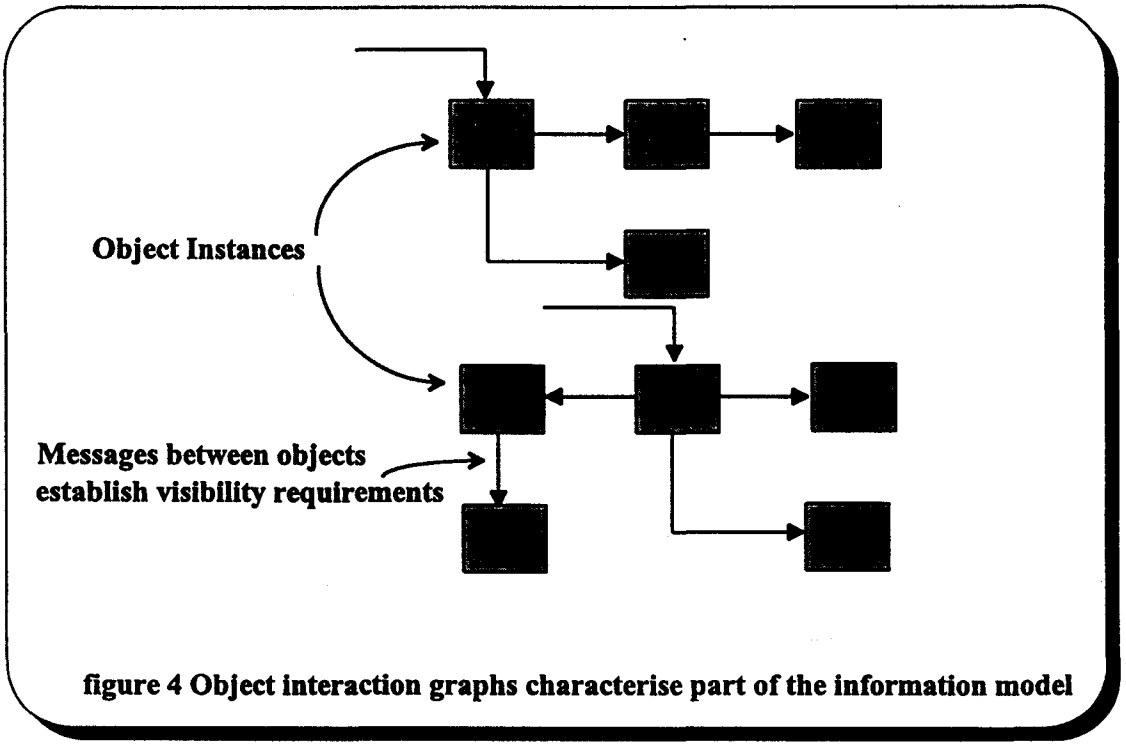


**figure 4 Object interaction graphs characterise part of the information model**

application information model is represented by the system object model and object interaction graphs. These two models show graphically the possible *points of contact* that may be required between the application information model and the MIB, namely:

- data attributes - data values obtained from managed object attributes.
- object valued attributes - derived from relationships between managed objects in the MIB.
- object behavior - application functionality implemented by managed objects in the MIB.

The figure shows the MIB represented by a containment tree of managed object instances. Note that in the TMN standards the containment relationship is of primary importance in structuring the MIB. This is used as a naming relationship to provide unique names for all managed object instances. Internally there may also be other relationships represented between managed objects in terms of attribute values that refer to the object identities of other managed objects (i.e. object valued attributes).

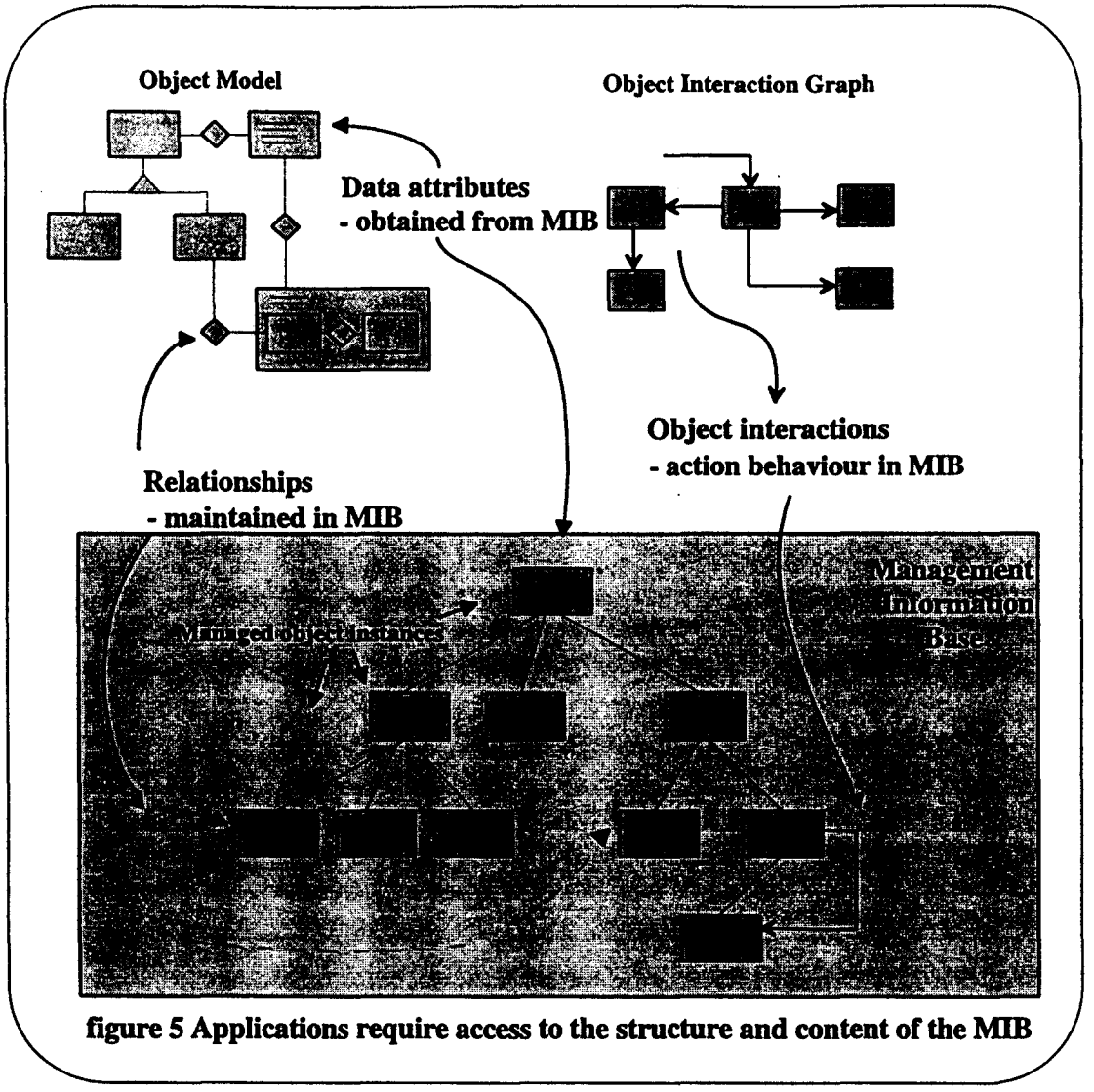


**figure 5 Applications require access to the structure and content of the MIB**

### *4. Gaining access to the MIB*

It has already been pointed out that the top-down/bottom-up design issue for TMN applications is similar to the design of an application using a pre-defined class library or application framework. Here there are pre-defined (managed) object classes, instances of which have to be used in the design to meet the requirements of the application. For the developer of TMN applications the MIB is a design constraint which has to be factored into the design process.

However, the major difference for the TMN application developer is that the MIB is typically regarded as an *active* information base. Many applications have access to it and these may change the structure and content of data in the MIB. The network elements are also continuously notifying the managed objects in the MIB of changes in status, network events, error conditions, etc.

There are two main consequences of these MIB characteristics. First, the application developer has to be able to *explore* the MIB to determine what managed objects exist.

Dependant upon the application development environment, appropriate tools are required to discover the data available in the MIB. For example, a browser would be used to explore the structure of the MIB and to obtain details of the data held by managed object instances.

Second, a different computational model exists between the management application and the MIB from that which exists inside the management application itself. Within the application we assume the computational model of, say, C++ or Smalltalk, i.e. message passing between object instances is synchronous and the invocation of an object method corresponds to a procedure or function call in more traditional programming languages such as C or Pascal.

The TMN standards define protocols and management information services to enable applications to access and modify the MIB data. The computational model employed is asynchronous. This means that for any managed object in the MIB that the management application needs to access, the application has to provide an agent or proxy to manage interactions with that managed object. We refer to these as *proxy objects* in the remainder of this paper. In the Fusion models this is similar to the introduction of monitor or terminal objects to handle asynchronous interactions with agents external to the system interface [1].

Much of the additional design activity that has to be introduced because of the influence of the MIB is concerned with the introduction of proxy objects into the application information model and to extending the design models to interact with these objects.

### *5. Extending the Fusion Design Process*

As stated in section 3, an appropriate stage in the design process has to be found where the requirements of the application and the constraints of the MIB can be resolved. We believe that this should take place early on in the design process when the object interaction graphs are being developed. Experience has shown that if the design is allowed to develop too far it becomes difficult to re-engineer the application information model and the cost of repair is greater.

The following process steps give a flavour of the additional design activity required to introduce the influence of the MIB.

1. Initial idea exists of the structure and behaviour required in the application information model (analysis and early object interaction graphs).
2. A data value or behavior is identified that depends, either directly or indirectly, on data/behavior in the MIB.
3. Locate the data/behavior of interest in the MIB and obtain details of the managed object(s) involved.
4. Modify the design of the OIGs and other models as appropriate to accommodate proxy objects providing access to managed objects in the MIB.

This design activity is iterative in nature and the developer may have to design new object classes based on the results of the MIB exploration. There are a number of possible outcomes which include:

- The application information model is modified to accommodate new object classes that (naturally) map onto managed object classes in the MIB. For example, finer granularity object modeling may be required to accommodate the detailed containment relationships in the MIB. This in turn will require rework on the operation model and object interaction graphs.
- The MIB is modified to include new abstractions that contain data more suitable for the management application being developed. Although this is not an option always open

to the application developer, the possibility of re-engineering the MIB should not be overlooked. The interested reader should refer to [2] to gain further insight into MIB modeling techniques.

- New abstractions are included in the application information model that mediate between the application and the MIB. For example, a required class in the application may have to be formed as a subclass of a more abstract class, instances of which correspond to managed objects represented in the MIB. Conversely, the application may only be required to provide a restricted view of a managed object in the MIB, or perhaps provide a view of data derived from a number of managed objects.

There are clearly other stages during the Fusion analysis and design activity where the influence of the MIB could be taken into account. For example, this could happen early on in analysis when the domain object model is being developed. Restrictions could be made to only use object classes in this modeling activity that corresponded to managed object classes in the MIB. However, if the constraints imposed by the MIB are allowed to influence the application analysis too strongly, this can lead to a poor object-oriented design. For example, the dominance of the containment relationship in the structure of the MIB can smother more appropriate relationships required by the application. At the other extreme, if the impact of the MIB is not considered until late in the design activity, this can lead to unacceptable run-time inefficiencies.

### *6. Conclusions*

The guidance provided by Fusion establishes an essentially top-down approach to object-oriented analysis and design. The models produced, together with the supporting development processes, are driven by the operational and functional requirements of the application being developed. The method provides a systematic approach, moving from an initial set of requirements through to implementation. In practice, many deviations and iterations are required to investigate alternative design choices and to factor other constraints into the design process. This paper has addressed the issue of domain specific constraints being placed on the design of the application information model.

The particular domain of TMN application development introduces the MIB as a design constraint. This contains a specialised object-oriented structure, providing the management interface to the network elements and other sources of network data. The goal of the application developer is to design an information model that supports the required management functionality but at the same time minimizes any disparity between the data abstractions represented in the MIB and those required by the application.

Our experience to date has been that if attention is given to the structure and content of the MIB early on in the design process then appropriate modifications can be made to the application information model to satisfy both the top-down and bottom-up constraints. However, care must be taken not to allow the structure of the MIB to dominate the design of the application information model and, as a consequence, introduce inappropriate object classes and object visibility structures.

The Fusion object interaction graphs prove to be the most effective models for considering the two design influences. Their initial design, based on the operation model from the analysis phase, provides a suitable foundation for exploring the MIB to discover a source for the required data. Proxy objects are introduced into the object interaction graphs to manage

the asynchronous messaging between the application and the MIB. Additional object abstractions are introduced into the application information model and/or into the MIB to mediate between the two potentially different information structures. Further research in this area aims to develop mechanisms to partially automate the development of information model viewpoints of projected data from the MIB.

*References*


[1] Coleman, D., Arnold, P., Bodoff, S., Dollin, C., Gilchrist, H., Hayes, F., Jeremaes, P., *Object-Oriented Development: The Fusion Method*, 1994, Prentice Hall.

[2] Bapat, S., *Object-Oriented Networks: Models for Architecture, Operations, and Management*, 1994, Prentice Hall.

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

[4] *TMN Interface Specification Methodology*, ITU-T Rec. M.3020, 1992.

[5] *Generic Network Information Model*, ITU-T Rec. M.3100, 1992.

[6] *TMN Management Services: Overview*, ITU-T Rec. M.3200, 1992.

[7] *TMN Management Functions*, ITU-T Rec. M.3400, 1992.

[8] *Management Information Services - Structure of Management Information - part 4: Guidelines for the Definition of Managed Objects*, ITU-T Rec. X.721.

[9] *OSI Management: Technical Guide*, NCC Blackwell, ISBN 1-85554-187-4, 1992.