Lakshmi Raman
ADC Telecommunications

Chapter 1

Information Modeling and Its Role in Network Management

1.1. INTRODUCTION

Telecommunications, information, and entertainment network and service providers are experiencing new challenges today more than ever before due to advances in technology combined with growing demand from sophisticated users. In addition, the regulatory environment in many countries is undergoing rapid change. Competition for providing high-quality services is increasing. Competition introduces the need for exchanging information to support successful operation of network elements not only within a jurisdiction but also between jurisdictions (domestic as well as international). Customer network management is also requested by large business customers. Instead of considering network management as an afterthought, the trend is being reversed; product decisions are being made using network management features offered by the suppliers as a differentiator.

Today network management is provided using a variety of data communications protocols ranging from proprietary to de facto standards such as BX.25. Network management information, also referred to as operations messages, are specified in many cases using character strings with delimiters. This acceleration of communication protocols has resulted in islands being created with dependency on one or two suppliers in order to have interoperable interfaces between network elements responsible for providing the services and operations systems or supervisory systems that manage the network elements. This results in either delaying the introduction of new services or deploying them without adequate management in order to meet market demand.

These issues are not specific to network management applications. They have been encountered in building distributed applications in general. The fundamental requirement for successful communication is to have a common understanding of the information exchanged regardless of whether it is between two computer systems, between software processes, or between a software process and a database where the information is stored. This need was identified in database design in the early 1970s, and different techniques for information modeling exist in the literature. This chapter describes how information modeling principles have become an integral part of network management, a distributed application.

Two different approaches are available in the industry for developing information models for network management. One approach evolved from the need to manage simple data communications equipments such as bridges and routers, and the other stemmed from the need to manage more complex telecommunications equipment such as switching and transmission nodes. Even though the actual technique and details differ with the two approaches, the fundamental principle remains the same, namely, model the information across a communication interface so that both the sender and receiver of the information interpret it in the same way. Other information modeling efforts are in progress for development of portable software in distributed applications.

This chapter provides an introduction to information modeling, forming the foundation for an interoperable multisupplier network management solution. Even though several techniques are available for information modeling, the focus for this chapter is on using the object-oriented principles. These principles, which were initially found to provide a powerful foundation to develop reusable software, have been adopted to varying degrees in designing and building open distributed processing applications.

Sections 1.3 through 1.5 of this chapter discuss the need for information modeling to provide an interoperable management interface between the managed and managing systems. An introduction to different information modeling methods is provided in sections 1.6 through 1.8. Components of specific object-oriented information modeling principles used in support of telecommunications network management are discussed in sections 1.9 through 1.18. These components allow modeling complex telecommunications resources. The principles used for modeling data communications resources are simpler and are not discussed here in detail except to note some of the differences. Examples of information models available to support network management of telecommunications and data communications equipments are provided in sections 1.19 through 1.25. The goal of network management is to facilitate deployment of interoperable network equipments. Sections 1.26 through 1.29 discuss mechanisms available for suppliers to specify conformance of their products to requirements and for service providers to determine interoperability issues prior to testing and deployment. Advances in software development, specifically distributed processing, is influencing the future direction of network management specifications, specifically the Telecommunications Management Network (TMN). Since network management products are expected to embrace distributed processing concepts, probable future directions are discussed in sections 1.30 through 1.34. Conclusions are provided in section 1.35.

1.2. INFORMATION MODELING MADE EASY

Before discussing the various object-oriented concepts used to develop an information model, a simple approach to defining an object as part of an information model is presented in this subsection. An information model for an application may be composed of one or more objects to meet the different requirements. Because this chapter focuses on the role of information modeling in network management, examples are taken for that distributed application. The need for representing a resource in the information model must first be established. In other words, the object must meet some requirement for the application.

In developing the information model for network management applications, only the information that is relevant to management is modeled. For illustrative purposes, consider that it is a requirement to manage a hypothetical line card in a switch. The line card is a physical resource that exists to support call processing activities. In developing the information model for NM, only the information about the line card relevant to management is modeled.

Assume that the properties of the line card to be managed (either for the purpose of monitoring or control) are the following: the line card is produced by some supplier; it has a serial number, an equipment type specifying the type of line card; and a state indicating whether or not it is active; and, if the line card is active, then it is assigned to some telephone number. Information such as alarm status and equipment type will be required when the line card has a fault and needs to be replaced as part of fault management application. On the other hand, the telephone number property is required to provision a subscriber. The information model for representing these static properties may consist of the template illustrated in Table 1.1. The table identifies how these different properties are represented so that a common understanding exists between the managed switch where the line card is present and the managing system. The representation identifies the syntax of the information at communicating interface, and the properties describe the semantics of the managed information.

If the line card fails, it may emit an alarm message that will be sent to a Network Management System (NMS), and the NMS may request the following actions: activate/deactivate the card and replace the faulty card with another card of the

TABLE 1.1 TEMPLATE FOR A LINE CARD

Properties	Representation
supplier name	character string
serial number	character string
equipment type	character string
card active state	yes/no
alarm status	critical (0)/clear (1)
assigned phone number	numeric string
:	:

same type. In addition to the static properties included in the table, ability to emit an alarm notification will also be included as part of the information model for the line card. These properties of the line card can be applied to every line card, regardless of how it is physically implemented and in which switch it is present. The above template represents a class of objects called the line card.

Every actual line card in the switch is represented by one instantiation of the template described in Table 1.1. Table 1.2 illustrates such an instantiation.

The properties with specific values shown in Table 1.2 represent a line card regardless of the actual implementation of the card and the supplier of the card. The power of information modeling is to bring together in a representation characteristics of the interface an object supports, regardless of the actual implementation. A collection of instances belonging to different object classes form a repository. In the case of network management, this repository is referred to as the Management Information Base (MIB). The collection of objects that represent the entries in a directory is known as the Directory Information Base (DIB).

TABLE 1.2 A SPECIFIC LINE CARD OBJECT

supplier name	ADC
serial number	cu126781
equipment type	ISDN Channel Unit card
card active state	yes
alarm status	clear (1)
assigned phone number	946 2090
:	:

1.3. COMMUNICATING MANAGEMENT INFORMATION

Network elements are managed today using different methods varying from proprietary to open interfaces, as pointed out in the previous section. Because this chapter is concerned with operations information being transferred at the application level,[1] methods such as E-Telemetry are not discussed. Two methods are currently in vogue to manage the network elements that pertain to sending application-level information. The first method is a message-based paradigm, and the

[1] Application level is used in the context of the OSI Reference model. The information is exchanged between application processes in the OS (Operations System) and NE (Network Element).

second is an object-oriented paradigm. The object-based approach is used for managing both telecommunications and data communications resources. However, the actual details of what the object represents vary considerably in both cases largely because of the differences in the interface definition.

1.4. MESSAGE-BASED PARADIGM

Several systems are deployed today in the service provider's network that use the message-based approach. The information exchanged is specified in terms of messages. Languages used to specify the messages use a human friendly format. Usually, character strings are used to define the exchanged information. As an example, in North America, message sets are available as open generic specifications using a language called Transaction Language 1 (TL1) developed by Bellcore. The message sets are defined in Bellcore Generic Requirements for applications such as alarm and performance monitoring, testing, and provisioning. The messages are specified using either position-based values for management information separated by delimiters or a tag value scheme.

In either scheme the messages specify the type of operation or notification along with one or more entities being managed.[2] Taking TL1 as the language, we provide two examples of the messages between OSs and either NEs or Mediation Devices (MDs).[3]

```
^^^RDBKNJ35672^85-04-10^05:46:20
*C^101^REPT^ALM^T1
^^^''101:CR,T,SA,FEND,,261,259
(^ is used to indicate space)

ENT-LI:267943:143: :OE=003151026,MC=1P,CHT=FL,PIC=123
```

The first example specifies the content of an alarm report on the entity referenced as T1 101 reported by an NE to an OS. If an alarm is to be reported on a different entity such as a circuit pack, a different message will have to be specified. In other words, for the same type of notification or operations request,[4] because the managed entity is different, a new message will have to be created. When developing the software, except for the parsing routines, it may not be possible to reuse code designed to support one message with another. Powerful software engineering con-

[2] Bellcore requirements define messages for most of the applications except provisioning (also referred to as memory administration). In the latter case, a nonnormalized data model combined with verbs corresponding to database operations define the messages.

[3] Mediation device is sometimes referred to as supervisory system.

[4] The terms *notifications* and *operations* are used in order to conform to the terminology used in TMN standards. In TL1 these are usually called commands and autonomus notifications, respectively.

cepts such as abstraction and reuse are inherently not available with this paradigm. Specification of the complete message structure, however, makes it easy for a reader to understand the exact information exchanged without requiring knowledge in the intricate details of the protocol and for programmers to develop the software specification.

In the second example using the relational data model, an OS sends information on a subscriber to a switching system. The verbs used with the management information in the data model correspond to the basic database operations.[5] Defining the data model and using a defined set of verbs are appropriate for memory administration, given the large number of features available with the Integrated Services Digital Network (ISDN) and Advanced Intelligent Network (AIN). The database operations themselves are not different. The information entered in a switch varies depending on the services selected by the subscriber.

The message-based paradigm in general was developed to provide the same message definition for user-to-machine and machine-to-machine interfaces. The language chosen in many cases is derivative of the ITU Recommendation Z.300, known as Man Machine Language (MML). Another advantage of this approach is the ability to use a simple protocol analyzer on the communications link to verify the content of the exchanged information. Using a user friendly message specification viewed from the user perspective has disadvantages from the machine-to-machine point of view. The languages employed are constrained to use human readable character sets. This restricts the data types for message specifications. Other data types such as integer and boolean that are more applicable for machine operations are not used.

Even though systems using the message-based paradigm are deployed extensively, this approach lacks rigor in specification. As an example, in referencing an entity being subject to management, it is necessary to provide an unambiguous identification. In the TL1 approach, the specification assigns *n* number of characters to be used by the supplier. The format for the identification is influenced largely by the architecture of the supplier's product. Even across multiple network management applications, different formats have been recommended.

1.5. OBJECT-ORIENTED PARADIGM

In the telecommunications environment, the equivalent of the "messages" is specified using an object-oriented paradigm. Several information models using this paradigm have been developed as part of standards work on the Telecommunications Management Network (TMN). With differing details and complexity, object-based approaches are defined for managing Internet data communications network resources as well as for building software for distributed applications. The following subsections describe the approaches used in telecommunications and data communications network management.

[5] The database operations are referred to as CRUD to denote create, read, update, and delete.

1.5.1. System Management Architecture

The architecture for exchanging management information for interactive applications is shown in Figure 1.1.

In Figure 1.1, let us assume that an OS is managing a network element; this implies that the OS is the managing system and the NE is the managed system. In contrast to the message-based paradigm, the object-oriented approach specifies a set of remote operations.[6] These remote operations may be performed on different resources depending on the technology, services, and architecture being managed. The managing system issues operations requests on resources and receives notifications corresponding to the various events. In other words, taking the example used in TL1, the structure of the message for a notification is not redefined if the notification is an alarm from a termination point or a circuit pack. Similarly, the structure of the message to create a subscriber is not different from that of a log. Depending on the resource being managed, the properties to be sent with the create request will vary. Instead of defining new messages, the resources are modeled as managed objects with specific characteristics.

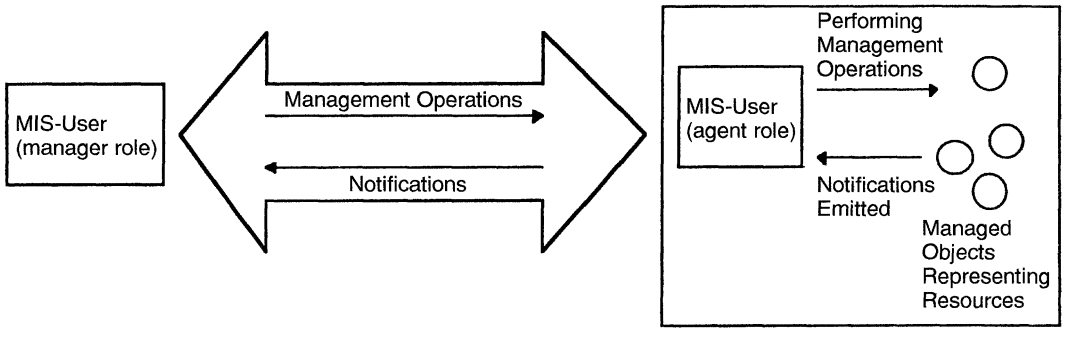

Figure 1.1 System Management Architecture.

1.5.2. Message Structure for Telecommunications NM

Common Management Information Protocol (CMIP), specified to provide the services defined in Common Management Information Service (CMIS),[7] is used for external communication. The various services offered by CMIS include the database

[6] The term *remote operations* is used here to distinguish it from commands where, for example, an OS requests the agent to perform a specific operation such as creation of a subscriber record. Both commands and notifications are to be considered as remote operations.

[7] Common Management Information Service Element (CMISE) is an OSI application service element consisting of a service definition known as CMIS and a protocol specification called CMIP.

operations to create, retrieve, delete, and modify data, along with resource-specific event reports and actions. Because the database operations are generic, depending on the resource, additional definitions (event and action types) will be required. A high-level structure for the "message" is shown in Figure 1.2.

Every message has a sequence number[8] and a value or code to identify the remote operation. A specific set of operation values are defined as part of CMIP, and these are reused across multiple management applications functions. For example, the same operation value and structure is used in the message to replace the value of window size of a protocol entity and the state of a cross connection. The resource and the property being modified vary with each message. The resources and their properties that can be managed are specified in terms of object-oriented information models. In other words, the message structure leads naturally to an object-oriented paradigm.

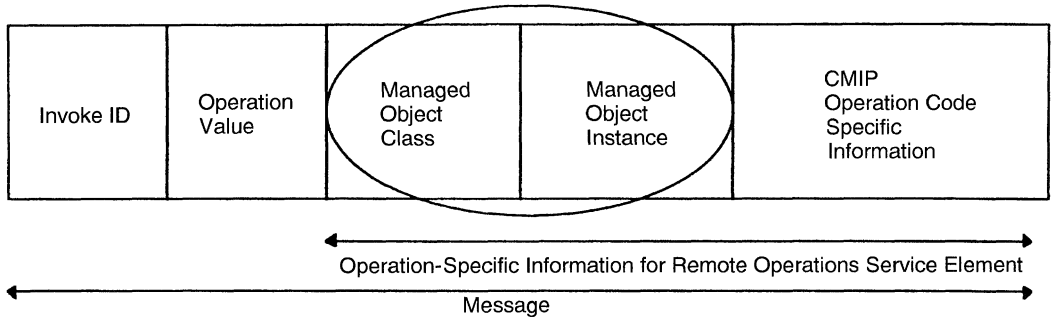

Figure 1.2 CMIP Protocol Data Unit (Message Using CMIP).

1.5.3. Message Structure for Data Communications NM

The system management architecture described in Figure 1.1 is suitable to describe the network management of data communications network, specifically the Internet. The concept of operations to be performed on objects, known as MIB variables, is the same, even though the types of operations are not identical. The ability to report notification is equivalent to trap messages indicating something has happened. Again the details of the types of notifications and the information exchanged with it are considerably different. The structure for the messages is shown in Figure 1.3.

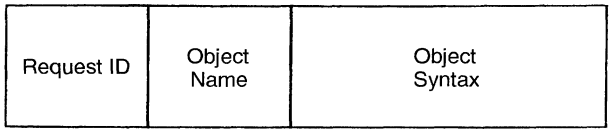

Figure 1.3 SNMP Protocol Data Unit (Message Using SNMP).

[8] The concept of sequence number to allow correlation between the request and response message is also present in the message paradigm such as TL1.

The figure is simplified in that only one reference to object name and syntax are shown. The repetitive nature of these two parameters is not illustrated in the figure. In addition, the type of operation is implicit by the tag of the message instead of an explicit operation value parameter shown in Figure 1.2.

Similar to CMIP, the number of operations is finite, and various requirements of network management are achieved by modeling the resources as objects. (The combination of object name and syntax is referred to as varbind.)

It should be noted that modeling application-specific information using an object-oriented approach does not impose requirements on software development. In other words, there is no requirement to implement using object-based languages. However, there are benefits to be gained in software development by object-based languages because the same principles are used.

Even though the concept of object-oriented modeling of information is discussed here in the context of message exchange, information modeling is not specific to external communication. While it provides a rigorous formalism to define the properties of managed resources for the purpose of network management communication, efforts are underway in ITU as well as other organizations such as the Object Management Group (OMG), to use an object-oriented approach for information modeling in general.

1.6. FOUNDATIONS OF INFORMATION MODELING

As mentioned earlier, defining information models, also known as schema, is an activity that is not specific to network management. Data models using entity relationship concepts have been in use for a few decades. Entity relationship models define various entities and how they are related to each other. This section provides a brief introduction to two modeling approaches. Instead of discussing how to develop information models in general, emphasis is on how to define the schema within two network management contexts (datacom and telecom). Because the principles used in telecommunications information model are more complex, these are discussed below. Differences with the modeling principles used with data communications network management are noted.

A technical report published by ANSI T1 provides guidelines for specifying an information model. Several concepts that aid in developing an information model (specifically an object-oriented design) are introduced in the following sections. How these concepts are used in developing an information model depends on several criteria. Some questions to consider are:

- Is the model easy to understand?
- Is it implementable?
- Does it meet the requirements set forth at the start of the modeling process?
- Is it extendible and does it meet multiple supplier solutions to a technology?

The ultimate test is whether the model can be implemented efficiently to solve the specific application, for example, in the context of network management. It is possible to define a model that meets all requirements and is very elegant from the specification point of view. However, implementing this model may impose heavy processing and memory requirements. This will result in degrading the performance of the network being managed, thereby affecting the quality of services provided. Such a model will not be considered an acceptable specification. In developing information models for use with SNMP, simplicity was a main goal. *The Simple Book* by M. T. Rose discusses criteria for including an object definition in the schema. Even if the same criteria are not suitable in all applications, setting such guidelines can provide the balance between architectural purity and practical implementation.

In setting guidelines, one encounters requirements that may be complementary or conflicting. Understanding the tradeoffs and optimizing to meet a certain criterion will be necessary. In other words, it is fair to state that modeling is an art and there are no right or wrong answers.

1.7. E-R APPROACH

Early attempts at information modeling were done in database applications. As such, relationships between the components of the system were defined. Several books have been written on the topic of information modeling concepts (e.g., Matt Flavin, *Fundamental Concepts of Information Modeling*), and emphasis has been on identifying the relationships between the components of the data.

In the entity relationship approach, as the name suggests, the model defines various business entities and relationships between them. This approach to facilitate database design was first introduced by Chen (P. P. Chen, "The Entity Relationship Model—Toward a Unified View of Data," *ACM Transaction Database Systems*, Vol. 1, No. 1, 1976) and extended by Codd later (E. F. Codd, "Extending the Database Relationship Model to Capture More Meaning," *ACM Transactions on Database Systems*, Vol. 4, No. 4, 1979). Even though this approach has encountered criticism, the fundamental reason for its popularity is its simplicity. Identification of what constitutes an entity and what relationships should be modeled is somewhat subjective.

In addition to defining the entities and their relationship, these models are accompanied by diagrams describing the relationships between the entities along with the type of the relationship. Even though the object-oriented paradigm used in TMN information models has followed a different approach, standards have used an E-R like diagram to make the model human friendly. In other words, the notation used to define the models facilitates machine parsing rather than making it easy for a reader who is attempting to get an overview of the various objects and the relationships between them.

1.8. OBJECT-ORIENTED DESIGN

Unlike the E-R modeling approach, object-oriented design principles, simply stated, provide an abstraction that encapsulates data as an object. In terms of network management, the resources managed are specified as managed objects. In most cases, the resources exist to provide telecommunications service(s). The example of line card template in Table 1.2 is a managed object. Managed objects provide an abstraction for the properties of the resource that are manageable. In other words, details not relevant to management are not modeled.

The advantages of the object-oriented approach to model information are encapsulation, modularity, extensibility, and reuse. With the concept of encapsulation, the object is responsible for assuring integrity when it receives requests (messages) to perform operations. In other words, internal details of how the operation is performed by the resource is not visible at the boundary of the object. The abstraction of the resource defined as an object specifies the properties that reflect the result of executing the message.

The object-oriented approach naturally lends itself to modular specification. What aspects of management information should be considered as an object depends on the level of modularity desired. This can be illustrated by using the following examples. Let us consider the case where performance monitoring parameters are to be collected from a Synchronous Digital Hierarchy (SDH) line termination. An object may be defined to represent all the information, including the performance monitoring data pertinent to that resource. A more modular specification will separate the fundamental properties such as state from Performance Monitoring (PM) parameters. By including all the appropriate PM parameters within an object, the specification becomes modular, which automatically leads to flexible and extensible specification. New PM parameters may be introduced without affecting the definition of the termination point. Another example is provisioning a customer's subscription data for various services. Here again, modeling a subscriber with parameters corresponding to all the requested services will not provide a modular specification. Instead, modeling the parameters of a service as a separate object related to the subscriber results in a more modular specification. The flexibility is not just at the specification level, but it can also be reflected in software. However, flexibility is not without additional cost because the number of objects and relationships to be maintained increases. By appropriate design, modularity facilitates reuse both at the specification level and in software. Reusability at the specification level and in software is discussed in this and the following sections.

The paradigm used for defining SNMP information models does not use various object-oriented principles such as inheritance and encapsulation (identified above). The objects are simple atomic data elements. A complex structure such as a routing table is modeled as a table with various columns.

Information models developed by OMG/CORBA for the purpose of object-oriented software development use many of the principles mentioned above. Even though there are differences in the details of the modeling principles and information models, a common thread among these different applications is the need for an

information model. The models provide for interface definitions and facilitate exchanging information in an unambiguous manner.

Development of complex distributed systems necessitates system engineering requirements that include well-specified interfaces. The power of information modeling makes it a strong component of this upfront engineering and can potentially reduce development costs.

1.9. INFORMATION MODELING PRINCIPLES

Several variations of object-oriented design principles are being used for both information model specification as well as in software development. In this section, only the object-oriented design principles defined in ITU Recommendation X.720 | ISO/IEC 10165-1 are discussed. The term *Structure of Management Information* (SMI) is used to refer to the various concepts, as well as the representation techniques used to specify the information model. These principles form the basis for the information models that are available today as part of TMN standards. The subsections below describe various concepts that are used to design an information model. However, it is worth repeating that modeling is an art. As seen in the following discussion, there is more than one way to model the functions of a resource for management. Considerations such as optimizing data transfer for external communication, ease of retrieval and manipulation of data, granularity level for the information, and possible implications for implementations (memory size, processor power, speed of processing) will drive the decision between multiple choices.

1.10. MANAGED OBJECT CLASS DEFINITION

A managed object, as mentioned earlier, represents the management aspects of a resource. A resource may be physical (e.g., circuit pack) or logical (cross-connection map). Taking the example of the circuit pack, several managed objects[9] may be present in a network element to represent the various cards. A managed object class called circuitPack is used to define the properties that are common across different cards. Characteristics such as operational state identifying whether or not the circuit pack is working, type of alarm to inform users that the circuit pack has failed, and behavior stating that they are replaceable plug in units are applicable to all circuit packs regardless of the supplier and the actual function supported (power supply, line card or processor, etc.).

Generalizing the above example, we can state that a managed object class defines a schema or a template with properties shared by managed objects that are instances of this class. Figure 1.4 shows the difference between the managed object class definition and a managed object (instance) for circuit pack.

[9] The term *managed object* is sometimes referred to as a managed object instance.

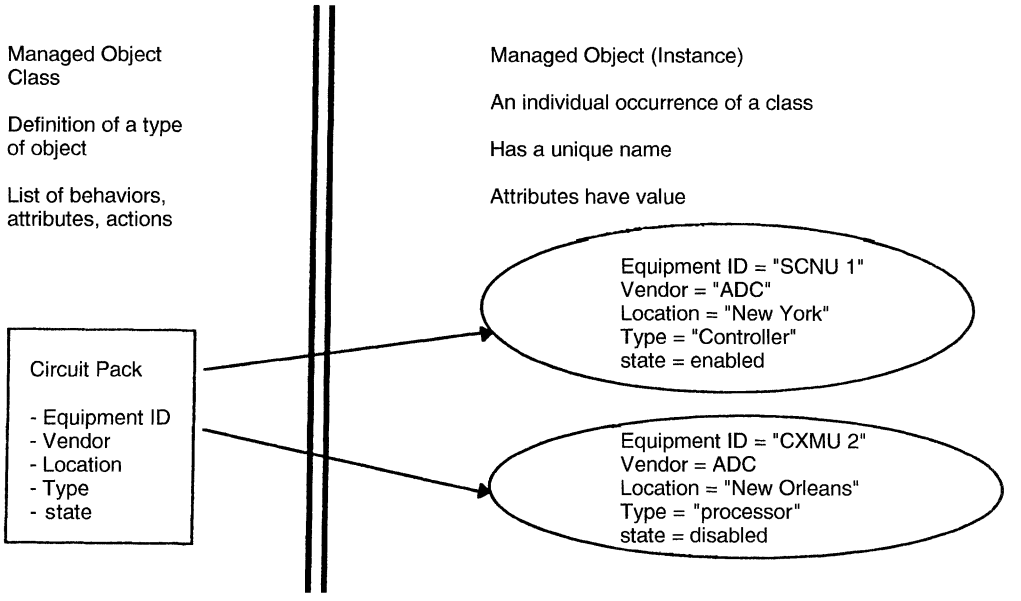

Figure 1.4 Example of Managed Object Class versus Instance.

A template for a circuit pack managed object class includes, in addition to the behavior, the following attributes: circuit pack ID to support unambiguous identification of a specific circuit pack, operational state to indicate whether an instance is working or has failed, and the type of circuit pack; and the following notifications: an alarm to inform the OS that the circuit pack has failed and a state change notification to indicate that the state of the circuit pack has changed to the value "disabled." Let us assume that an instance of a circuit pack is created to represent a power supply in a network element. The instance at any time contains values for the attributes and emits notifications whenever the events resulting in the notifications occur. In this example, all the properties defined in the template are expected to be present in an instance. Modeling of functions that are optional in a resource is discussed below.

1.10.1. Package Definition

The characteristics of a managed object class are defined in terms of behavior, attributes, notifications, and operations. In order to provide variations among instances, the concept of "package" has been introduced. A package is a collection of characteristics (behavior, attributes, operations, and notifications[10]) as shown in

[10] The semantics of attribute, operations, and notifications will be discussed later. For example, state representing whether the resource is active or standby may be modeled as an attribute, requesting a loop-back test may be modeled as an operation, and reporting a circuit pack failure, as a notification.

Figure 1.5. It is not required that every package include all the characteristics. However, if an instance includes a package, all the properties defined for that package must be present. In other words, a package is atomic, and further breakdown is not permitted.[11]

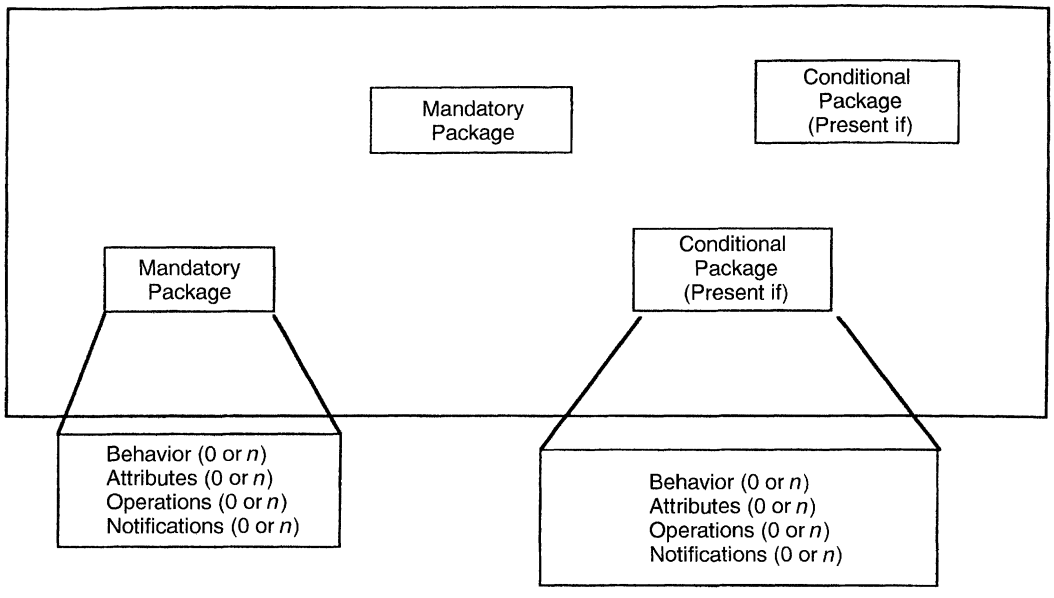

Figure. 1.5 Managed Object Class Definition.

Two categories of packages may be used in defining a managed object class. A *mandatory package*, as the name suggests, includes all properties that are present in every instance of that class. The second category is referred to as a *conditional package*. The properties defined for this category are included in a managed object during its creation if the condition for their presence evaluates to true. This concept of conditional packages has been used also to account for optionality where the condition is a user's option. As an example, consider an object class that represents an NE. Alias names may be assigned for the NE by the supplier to provide a human friendly name. The attribute for the alias name is modeled as part of a conditional package where the condition statement implies this is optional.

The concept of packages has been included in the information modeling principles in order to identify the collection of properties that may or may not be present. It should be noted that packages are defined only to aid in specification. A package does not exist outside of a managed object; properties included in a package are present only as part of a managed object. Properties belonging to packages can be

[11] In order to claim conformance to a definition, all properties of a package are required to be implemented. However, when phasing implementations, agreements may be made to include only a subset between suppliers of the managing and managed systems.

included in a managed object only at creation time. In other words, once a managed object is created, the properties of the object become part of the object and are available at the object boundary. They are not referenced or separated in terms of package boundaries. As a corollary, properties belonging to a package cannot be added after the creation of the managed object. If this is required, the original managed object should be deleted and a new one with the additional properties must be created. Similarly, during the lifetime of a managed object, existing properties cannot be deleted either individually or in terms of packages. As with addition, the deletion of properties also requires the deletion of the managed object followed by a creation without these properties.

A package is included only once within a managed object. In other words, multiple copies of a package cannot be present within a managed object. As an example, consider the conditional package mentioned earlier for alias name. Let us assume that it contains an attribute for a user friendly name. When the managed object class with this package is instantiated, the attribute is included only once. This is true even if there are two packages with the same attribute. Even though the specification gives the impression that there are two copies of the same attribute, when the object is instantiated, only one copy is present. This is consistent with the earlier discussion that knowledge of package boundaries is present only at specification time; the boundary is not maintained once the object is instantiated. When two packages have the same property such as an attribute, it is the responsibility of the managed object class designer to ensure that there are no contradictions.

The next subsections define the various components that can be included in a package definition.

1.10.2. Behavior

Behavior definitions are used to describe, for example, semantics and integrity constraints to be satisfied when the components of the package are included in a managed object. Even though specific behavior may be associated with each attribute or notification, when present at the package level, behavior definition is the glue to bring all the properties together. The circuit pack object class specified earlier, includes an attribute for the state and an alarm notification when there is a failure. The behavior describing that, when a critical failure occurs in the circuit pack, the value of the state attribute should be changed to the value "disabled" correlates the effect of an event to the modifications in the value of an attribute. Integrity constraints may be specified in terms of pre-, post-, and invariant conditions. Let us consider the case where the values of two attributes such as window size and packet size for a protocol entity are constrained by a discrete set of values for the ratio. The behavior definition is used to describe an invariant condition that the changes to the attribute value should not violate the integrity constraint on the ratio.

In addition to providing constraints among the properties defined within a package, behavior definitions may also describe how characteristics within a package may be influenced by the presence or absence of a conditional package. As an illustration, consider a managed object class definition that represents a log. Let

us assume that an attribute called availability status is defined as part of the mandatory package. This attribute in general may have several values to indicate multiple status information. The value "log full" is applicable for all instances of the log object. Suppose there is a conditional package that permits logging at scheduled intervals (during off hours 5 P.M. to 8 A.M. when no one is available to monitor alarm displays). Then behavior will be added to indicate that "scheduled off" is another valid value if the instance includes the conditional package for scheduling. As a result, when scheduling of the log is permitted, the availability status may assume one or both of the values "log full" and "scheduled off."

The examples provided here pertain to behavior corresponding to characteristics within a package or across packages within a managed object class definition. Another context in which behavior definitions are applicable is to describe the effects on a relationship among objects of the same class or different classes. For example, suppose an equipment holder managed object class represents a slot where a circuit pack can be plugged in. When the circuit pack is absent in the slot, the value of the attribute holder status will be empty. However, when a circuit pack is inserted, resulting in either creating a circuit pack managed object or updating the state of an existing managed object (for example, an existing circuit pack was pulled out and replaced), then the holder status must be updated to a value "occupied." Thus, behavior within a package definition is used to describe effects that are applicable to the managed object itself.

1.10.3. Attributes

A package includes zero or more attributes that reflect static characteristics of the object. In other words, properties described as attributes are distinguished from dynamic or active aspects such as notifications and ability to perform operations. An attribute is defined in terms of an identifier and has a value in a managed object.

The value of the attribute is determined by the syntax defined for external communication. The pair of items (attribute ID and attribute value) is sometimes referred to as Attribute Value Assertion, or AVA. That is, an attribute is asserted to possess a certain value. In addition to defining the syntax for external communication, attribute definitions may also include the types of checking appropriate for the values. For example, if the syntax of the attribute is defined to be an integer or real, then the value can be checked to be greater, equal, or less than a specific value. If the syntax is a complex structure consisting of multiple elements, then simple matching criteria will not be applicable. In some cases, special rules can be specified using behavior statements on how the values are to be checked.

Even though attributes are used to define properties such as state and circuit pack type, it is possible to indirectly invoke an operation by setting the value of an attribute to a specific value. A managed object class defined to represent processors may include an attribute called "restart" with syntax being Boolian. The behavior of this attribute will indicate that setting this value to true requires that the processor be restarted. That is, the indirect effect of changing the value of the attribute is to start a process.

Often, it is possible to specify the syntax, matching rules, and behavior specific to an attribute without taking into account the package (or indirectly the managed object class) where the attribute is included. The definition of the attribute by itself addresses to a large extent the syntax.[12] The semantics of what the attribute represents within the context of the package (actually, the managed object class representing the resource) is determined by the characteristics of the resource.

Attribute-oriented operations[13] are specified to reflect whether in the context of a specific managed object, the value of an attribute is read only versus modifications are permitted. It is possible that the same attribute may be allowed to be modified within one managed object class while this operation is not permitted in another object class. As an example, consider the counters for performance monitoring (PM) parameters. Let us assume that two classes of objects are defined to contain the PM parameters. One class is used to contain the values corresponding to the current collection interval, and another class contains values collected in the previous interval (historical information). Attributes for PM parameters, such as coding violation, will be defined to be readable as well as resettable to allow zeroing them at any time during the collection interval. However, when the same PM parameters become attributes of the class representing the historical data, then the only allowed operation is read. Any modification of historical information will be disallowed. It is the responsibility of the object to ensure that the integrity constraints defined using the pre-, post- and invariant conditions in behavior are not violated as a result of performing the requested modification.

Depending on the syntax, the value may be a single value (corresponding to, say, an integer syntax) or an unordered collection[14] of values. The latter is referred to as a "set valued" attribute. Additional operations such as adding and removing values to the set may be included with this type of attribute. The set of values is treated as a mathematical set. Adding an existing value, while not an error, will not include a second copy. Similarly, removing a nonexisting value is not considered an error.

Even though the value of an attribute is determined by its syntax, within the context of a managed object class, restrictions may be imposed on the specific values or range of values allowed for that resource. Two types of value restrictions may be specified. These are not mutually exclusive; either or both may be present in any managed object class definition. The set of permitted values is used to specify all the values that a resource may support. The set of required values specifies the minimum an instance of this class should support. For example, assume that an object class is defined to represent a modem with an attribute called "speed." Let us suppose that the syntax is real. A standard set of values such as 2.4, 4.8, 9.6, 19.2, 56, and 64 Kb are permitted for any instance of the modem class. Any other real value for the speed

[12] Behavior definitions may be used to specify semantics.

[13] Use of the term *attributed-oriented operations* should not be taken to mean that the operation requests are issued directly to the attributes. These are operations performed by the managed object affecting the values of attributes they contain.

[14] Multiple values that are to be considered in a specific order (for example, sequence of integers increasing in value) are treated as a single value.

will not be allowed. However, all instances may be required to support 9.6 Kb. This implies that in order to be conformant to this specification, all managed objects implemented to this definition must be capable of supporting 9.6 Kb. This example also points out that the set of required values must be either a subset of or the same set as the permitted values.

Another type of restriction that can be specified for the value of the attribute within the context of a managed object class is identifying a default value or an initial value. The difference between them is as follows: a default value implies that when the object is created and the value of this attribute is not provided, then the default value is used. This situation occurs irrespective of whether the attribute is part of a mandatory package or present in the conditional package that will be included (because the result of the condition evaluates to true). When an attribute has an initial value, this implies that during the creation of the managed object that attribute will not be permitted to have any other value. If a request to create supplies a value other than the initial value, the object will not be created. However, when a value is not supplied, the initial value is used and the behavior is similar to the default value.

In addition to providing a default value to use in the absence of one being specified in the request, default value can be used to reset the value at any time during the existence of the object (assuming no other integrity violations are encountered). Continuing the example of performance parameters, it is essential that counters representing the PM parameters are set to an initial value of zero before starting data collection. However, it is also true that the default should be zero so that, at any time, the OS may request a reset of the counters. The combination of both default and initial values in the specification will be required to meet the two requirements.

1.10.4. Attribute Groups

Attribute group is a handle to refer to a collection of attributes. It is a shorthand form to request a group of attributes. An attribute group has an identifier similar to an attribute. However, unlike an attribute, it has no value of its own. Restrictions as to what attributes may be included in the attribute group depend on the type of attribute group and are discussed below. The handle may be used with only two operations—read and set the value of the individual group elements to default values. In order to support the latter function of setting to default, the individual attributes included in the attribute group definition should have default values associated with them in the context of the specific managed object. With the read operation, instead of including the identifiers of all the attributes, referencing the identifier corresponding to the group will return the values of all the constituents. There is no limit on how many attribute groups may be included within a package.

Two categories of attribute groups may be defined: fixed and extensible. The fixed group, as the name implies, is a definition (the set of attributes that are included in the group) that cannot be modified later. If new attributes have to be added, then another attribute group must be defined. In order to include an attribute as a member of a fixed group, that attribute must be present in the same package as the attribute group.

Extensible group, on the other hand, refers to an attribute group where new attributes may be added after the original definition. These new attributes (because the original managed object class was extended to include additional properties) should either be present in the conditional package where the attribute group is included or be part of the mandatory package.

Let us now look at how this may be used in a model. Consider the case of performance parameters corresponding to DS1/E1 line included in a managed object class. All the parameters have a default value of zero. A fixed attribute group can be used to optimize data transferred in a request. Instead of listing all the PM parameters, a handle using the identifier of the attribute group can be provided to read all the parameters. Similarly, all the parameters can be set by using the set to default operation on the attribute group.

An example of use of extensible attribute group in the literature is for state attributes. Even though several state and status types are defined, not all are applicable in the context of a resource. Moreover, additional state/status types are expected to be defined that may be specific to a resource. In this case, a simple form of extensible attribute group is one with no attributes. Behavior is defined as follows: "components of the group are determined by the state and status attributes for the class containing this group." In other words, the elements in the group are dynamic. If this attribute group is included within a managed object class definition, then referencing this group for a read request will include, in the response, all state/status attributes present within the object class. If this group is included in the mandatory package, then the elements in the group may vary between different instances depending on whether state/status attributes are present in conditional packages that may or may not be included when the managed object is created.

1.10.5. Notifications

The previous subsections addressed the matter of how to model characteristics that are appropriate for database operations.[15] Let us now discuss characteristics that are modeled using the concept of notifications. These are triggered by either internal or external events.

Consider the case of internal events. Such events may occur for several reasons. Addressing the model of a circuit pack to represent a power unit, we see that failure of the unit or the board itself will result in generating an event of type equipment alarm. As a result of this alarm, the state attribute of the circuit pack will change to disabled. Suppose this power unit has a backup; then an automatic protection switch to this backup unit may occur. The status of the circuit pack managed object representing this backup unit will change from "standby" to "active." All three events, equipment alarm and two-state change notifications (state and status), are generated

[15] The phrase "static" has been sometimes used to describe attributes. An attribute, such as a PM counter, is dynamic in that the value is changed as a result of, let us say, detecting an errored second. Another way to think of attributes is information that is always present once included while creating the managed object.

as a result of internal events power unit failure and protection switch. The definition of the managed object class will include these notifications (as part of the mandatory or conditional package).

Examples of notifications arising as a result of external events are the following. Suppose two operations systems (OSs) are used in managing an NE. Assume that one of the systems is concerned with configuring the NE and the other with obtaining alarm information. The OS configuring the NE may send a request to create a circuit pack object when a new line card is plugged in a slot. When a circuit pack object is created in response to this request, an object creation notification will be generated by the newly created circuit pack to announce itself. This event may then be forwarded to the OS performing the alarm monitoring function so that it can understand alarms from this newly created object. Another example commonly used is the request to modify the value of one or more attributes. This is again an external event that gives rise to a notification referred to as attribute value change.

When the managed object class (through the package definition) includes a notification, behavior is used to explain the circumstances for generating the event. In addition to referencing the type of notification, it is also necessary to include the syntax for the information associated with the notification in order to communicate the event to an external system. For example, the parameters for an equipment alarm may include severity, probable cause, diagnostic information, and whether the resource has been backed up. Specific data types must be defined, similar to the syntax mentioned for attributes, in order to communicate details of the event to an external system. When specifying syntax of the notification, to promote reuse of the definition without restricting the ability to include additional information, it is recommended that extension capabilities be included in the syntax. The syntax of the extension itself will have to be determined once required extensions are identified.

In TMN modeling, it should be noted that occurrence of an event within a resource resulting in a notification must not be taken to imply that this is always communicated via the protocol to an external system. The next section defines a mechanism available in standards, and it is possible for the managed or managing system to configure the criteria under which a notification is communicated to a specific system.

1.10.6. Actions

This concept is used to model operations that the resource is capable of performing at the request of a managed system. These operations are distinguished from the attribute-oriented operations mentioned earlier. The latter refer to requests to read or modify attributes; even though the request references the object, the requested operation is on the attribute(s) only. Actions on the other contain requests to perform operation on the object as a whole. Associated with an action type is information required to perform that action successfully. The syntax of the information will have to be specified to communicate the request externally. One or more replies may be present for an action request. Syntax specification will be required to

explain how to interpret the response containing the result of performing the action request.

An action definition is usually accompanied by a description of the process for performing the operation. In discussing the various kinds of modifications on the attribute, it is noted that when attribute values are modified there may be indirect effects, as in the case of replacing the value of the restart attribute. That is, it should not be misconstrued that action is always needed when describing a process. Defining an action to describe a process does not provide any additional feature beyond what is available if modeled as a result of the modification of an attribute.

Examples of where action instead of set[16] is more appropriate are as follows: information issued in the request is not required to be an attribute of the object,[17] operation requires coordination of activities across multiple objects that may or may not be of the same class, information in the reply may not correspond to attribute values. The request to perform a test is modeled using an action. This is more appropriate because often initialization information sent in the request is required to begin the test. In order to perform a connectivity test on a termination point, the remote end will be provided. This is not required to be an attribute for the lifetime of the managed object but information required to start the test. Similarly, the result of the test "pass," "fail" is not appropriate to be modeled as an attribute. When using action to model the test request and response, it is also possible to send multiple responses indicating progression of the test. All these responses can be related to one request. X.722|ISO 10165-4 describes guidelines for using action versus set. As stated, they are only guidelines; it is still the judgment of a modeler(s) or a compromise between different members in a standards environment that influences the decision.

As with notifications, it may be appropriate to include extension capabilities with the syntax definition for the request and response to an action.

1.11. SYSTEMS MANAGEMENT OPERATIONS

The sections on attributes, attribute groups, and actions discussed two kinds of operations—attribute-oriented, where the database operations such as read and modify the values of an attribute are applicable, and object-oriented, where action requests are addressed to the managed object as a whole. These operations are sometimes referred to in the literature as SMI (Structure of Management

[16] Strong arguments have been advanced in some standards groups in the past on the use of set versus action. Some have argued that set allows generic software development versus action. On the other side, arguments have been made that with action you give the semantics of doing a process and this is more conrollable than if it was just a change in an attribute value. Both of these arguments can easily be refuted. When special behavior is associated with modifying an attribute such as checking integrity constraint, special software development will be required. Similarly, action does not give more control (giving only the appearance of control to a modeler) than a set with specific behavior.

[17] This information is not required to be retained for the lifetime of the managed object; it is required only to perform the action request successfully.

Information) operations. (Similarly, the notifications are called SMI notifications.) The operations and notifications specified as part of the information model define what is visible at the boundary of the managed object. In other words, the internal details about the resource are concealed. Mapping between these operations and notifications to a suitable communication protocol such as CMIP is required to transmit what is available at the managed object boundary to an external system.

Table 1.3 illustrates such a mapping when services and protocol of CMISE are used for external communication. The table is specified by mapping to the CMIS services; transforming from the service to the protocol transmitted across an external system/interface may be obtained from the CMIP standard.

TABLE 1.3 MAPPING BETWEEN SMI DEFINITIONS AND CMIS SERVICES

SMI Operation/Notification	CMIS Service
Get	M-GET
Replace	M-SET
Replace with Default	M-SET
Add Member	M-SET
Remove Member	M-SET
Create	M-CREATE
Delete	M-DELETE
Action	M-ACTION*
Notification	M-EVENT REPORT*

*CMIS Services where further syntax specification is required in the context of System Management Functions or resources.

Previous discussions have not addressed the question of how two basic operations prevalent in database—creation and deletion—are incorporated into an information model. The approach taken to include these operations in this paradigm is not in the definition of the managed object class itself; instead, rules are provided on how they are named or unambiguously identified outside of the class definition. This method provides flexibility, and allows creation and deletion at the instance level instead of requiring all instances of a class to follow the same rules. Naming rules, as discussed later, enables new schema to be included after the managed object class definition is completed. With this approach, some instances of the same class may be created by external request and others based on internal resource requirements.

Even though the information model is not used to capture the concept of synchronization, it is worth noting here in the context of operations. Using the management protocol defined for use in TMN, we can specify either best effort or atomic synchronization. The synchronization referred to here is not across multiple objects in different systems; the same managed system is responsible for the multiple objects. Because the requirement to support atomic synchronization is not part of the information model or schema, the support is left to local decision of the managed

system (in other words left to implementation). Note that in some cases behavior may be used to specify the need for using atomic synchronization.

1.12. MANAGEMENT INFORMATION BASE (MIB)

The distinction between managed object class and managed object (instance) was discussed earlier. The class definition is part of the information model or the schema. Managed objects that are instantiation of the classes in a system such as an NE form a repository referred to as the "Management Information Base." The actual database used to store the MIB may vary across different implementations and is not relevant for defining the schema. Figure 1.6 shows an example of the MIB[18] concept. Information in the MIB pertains to properties of the various managed resources that can be made visible across an external interface. When the MIB concept is realized in an implementation using various databases, additional information may be included to assist in the internal working of the managed system. These are not visible across the interface.

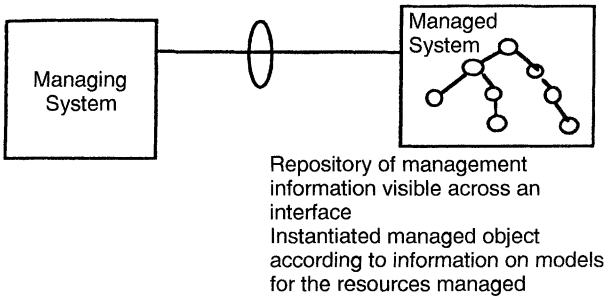

Figure. 1.6 Management Information Base (MIB) Concept.

The term *MIB* has been used in another context—the approach for managing the data communications equipment (bridges and routers) using the Internet management protocol, Simple Network Management Protocol (SNMP). Modeling principles defined in this approach do not make a distinction between a class representing a type of resource and a specific instance of that class similar to that made earlier in this section. The schema itself is referred to as MIB. Because of the much simpler principles used to define the schema and how the objects (data) are identified, this term is applicable when referencing the schema as well as the repository in an implementation. In other words, an Internet RFC containing the structure of management information is known, for example, as ATM MIB, SONET MIB.

[18] The reason why instances are depicted as being in a tree structure will be discussed later in this section. This is the result of how instances are to be named, which is defined as part of the information model.

1.13. EXTENDING MANAGED OBJECT CLASS DEFINITION

One advantage of using the object-oriented design, identified earlier, is reuse at the specification level as well as the implementation level, along with extendibility. When a managed object class is first defined, certain properties are included. Often, one encounters several reasons for extending the original specification to add any of the characteristics mentioned earlier. Extension may be required because new features are now included as a result of enhancements to a service, introduction of new technology, or a supplier interest in providing new features as a market differentiator for their product. The process of extending an existing definition with new properties is referred to as specialization.

Associated with the process of specialization is the concept of inheritance. When a new managed object class is defined by specializing an existing definition, it is said to inherit the properties of the original class. The specialized class is referred to as the subclass, and the original class is called the superclass. Inheritance is a powerful mechanism for building reusable specification. A new managed object class may be defined to inherit from multiple object classes. The restriction on the type of inheritance, regardless of single or multiple, is that it is a strict inheritance. In other words, the subclass includes by inheritance all the properties of the superclass(es) and adds new properties. The collection of managed object classes defined in this manner forms a tree, referred to as class hierarchy.

Figure 1.7 is an example of how inheritance is used to define object classes. The example chosen is taken from the termination point model in Recommendation M.3100.

In Figure 1.7, termination point object class is specialized to form new object classes: trail termination point source and sink. The properties of termination point are applicable regardless of whether the trail termination is a source or sink for signal flow. By defining the generic object class called termination point, the specification is reused, and only additional characteristics for the trail termination source and sink such as downstream and upstream connectivity pointers are included. The specification is flexible and extendible because new subclasses for different technologies can be specified without affecting existing definitions.[19] The figure also includes an example of multiple inheritance when the termination point is bidirectional or when PM data are included for the termination point.

In order to utilize the inheritance concept, it is common practice to define object classes at higher levels of the hierarchy (in contrast to the leaf level) with generic properties. In the above example from TMN, generic source and sink terminations for the transmitted signal are defined, including properties such as states and the

[19] This can be also be a disadvantage if sufficient care is not taken because it may result in a proliferation of managed object classes. While it is not a serious concern from the specification point of view, having many flavors of classes with similar functionality makes it difficult for implementation. Unless an implementation can accommodate all the specializations, ability to interoperate will be reduced. This matter is discussed in a later section.

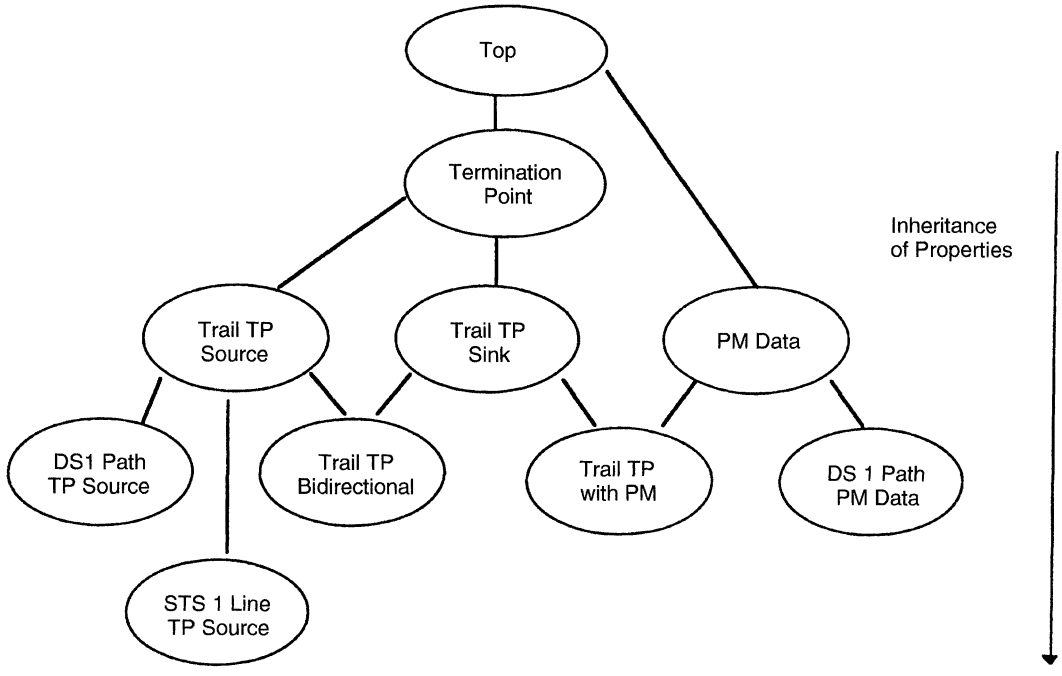

Figure. 1.7 Example of Single and Multiple Inheritance.

connectivity relationship. Using these generic definitions, we define specific subclasses for SDH and ATM. When generic managed object classes are defined, the requirement for strict inheritance should be carefully considered. Including properties that may not be applicable (sometimes it is difficult to know what types of specializations will be required at the time generic definitions are created) for all subclasses will lead to managed objects carrying extra baggage without serving a management purpose.

When defining new object classes using multiple inheritance, care must be taken to ensure that no inconsistency is introduced between characteristics inherited from multiple classes. The subclass definition should resolve these inconsistencies or contradictions. Furthermore, if an attribute, attribute group, notification, or action is included in multiple classes, only one occurrence will be in the managed object of the newly created subclass. However, values restrictions will be combined. As an example, if an attribute in one superclass has the permitted range of values 1..10 and another superclass 1 . . 15, the subclass defined by multiple inheritance from the two classes will permit the range 1 . . 15 for that attribute.

Figure 1.7 is drawn using "top" as the starting object class. All managed object classes defined for the purpose of management are subclasses of top at some level. The definition of top is contained in X.720 and X.721. The properties defined for top

are two mandatory attributes that identify the actual class of a managed object and an identifier referencing the naming rule used in instantiating the object. In addition, two conditional packages are included, with each one providing one attribute. One attribute is used to identify the registered[20] packages. The other attribute represents the ability to behave as different object classes.[21] Every managed object class defined as part of TMN includes these characteristics because of inheritance.

1.14. ALLOMORPHISM

Even though allomorphism[22] is not part of the principles used to define a schema, a chapter on information modeling using the principles applied in TMN will not be complete without a brief introduction of this concept. This concept facilitates software release independence to some extent.

Allomorphism enables an instance of a managed object class to be managed as an instance of another class. In order to support this capability, the two classes (the actual class of the managed object and the class used when managing it) must be compatible. Compatibility between two classes are governed by rules described in X.720. Examples of the rules are:

- For conditional packages, the result of evaluating the same condition must be the same regardless of the actual object class of the managed object.
- In order for the managed object to perform the same action included in compatible classes, regardless of its actual class, the mandatory parameters supplied with the request must be the same.

The concept of allomorphism is associated with a managed object. In other words, different implementations of the same class may not exhibit this property. In discussing the properties of top, it was mentioned that a conditional package is defined with an attribute that reflects this property.

[20] In order to understand the complete meaning of "registered," knowledge of the mechanism used to provide globally unique identification for the properties is required. For ease of understanding, assume that a package is given a globally unique number under certain conditions. For example, conditional packages are assigned a globally unique value. When a managed object is created, with conditional packages, the values of this attribute identify the packages included in the object.

[21] This is a simple way to describe the complex concept called allomorphism. Further explanation is provided later.

[22] In object-oriented programming, the term *polymorphism* is used to explain how the same function used with different resources will provide what is appropriate for that resource or different methods may be used for the same interface invocation. Even though there are some similarities at some level, allomorphism is different from polymorphism.

Let us now see how this concept facilitates release independence. Even though, in theory[23] a managed object may behave as classes that are not related by inheritance, let us take the case where a circuit pack object with some characteristics is defined in a standard. Suppliers, to meet capabilities unique to their product (not modeled in the standard object class definition), may define a subclass with these additions. Let us suppose that the circuit pack is managed by two systems. Also assume that the software corresponding to the extra capabilities was included in only one managing system when it was incorporated in the managed system. If the circuit pack managed object can exhibit allomorphism, then one managing system can manage using the new class with additional features. However, the other managing system can continue to manage as if the circuit pack were still the standard class. Different releases of the software to manage the same circuit pack can coexist without requiring flash updates in all the systems.

1.15. NAMING MANAGED OBJECTS

Characteristics of the managed resources are described in the schema or information model using the managed object class definition. Given a definition, instances are created to correspond to the resources being managed. When multiple instances of a class are created within the TMN, unambiguous identification of the managed object is required. The scope within which the managed object name is unambiguous may vary from global to relative to the managed system responsible for management of the resource.

In order to meet requirements for both global and local uniqueness, the containment relationship is used. Containment may or may not reflect physical containment. For example, in the case of a circuit pack, it is named relative to the slot in which it is placed. The name of the slot itself will be relative to the shelf and so on. However, a log record may be named relative to the log in which it is entered from a logical perspective, even though it is contained in a disk. Figure 1.8 illustrates naming using containment.

Even though names are pertinent only to instances and not classes, the structure for determining how an instance of a class will be named is specified in terms of classes. The specification of this structure is referred to as "name binding." Name binding expresses how an instance of a class, referred to as the "subordinate class," is named relative to another class, the superior class.[24]

[23] The reason for this statement is as follows. When referencing an object with a class other than the actual class, the agent implementation must first recognize the name unambiguously. As we will see later, depending on the naming rule, instances of different classes not related by inheritance often have different naming sequence. In order to successfully implement the behavior rules for allomorphism, the practical approach is when the classes are related by inheritance.

[24] When learning the information modeling concepts discussed here, often confusion arises between the inheritance tree and the naming tree (resulting from the containment relation and name binding rules). Inheritance and naming trees are distinct. Classes defined in name binding as superior and subordinate should not be mixed with superclass/subclass defined as part of the inheritance hierarchy.

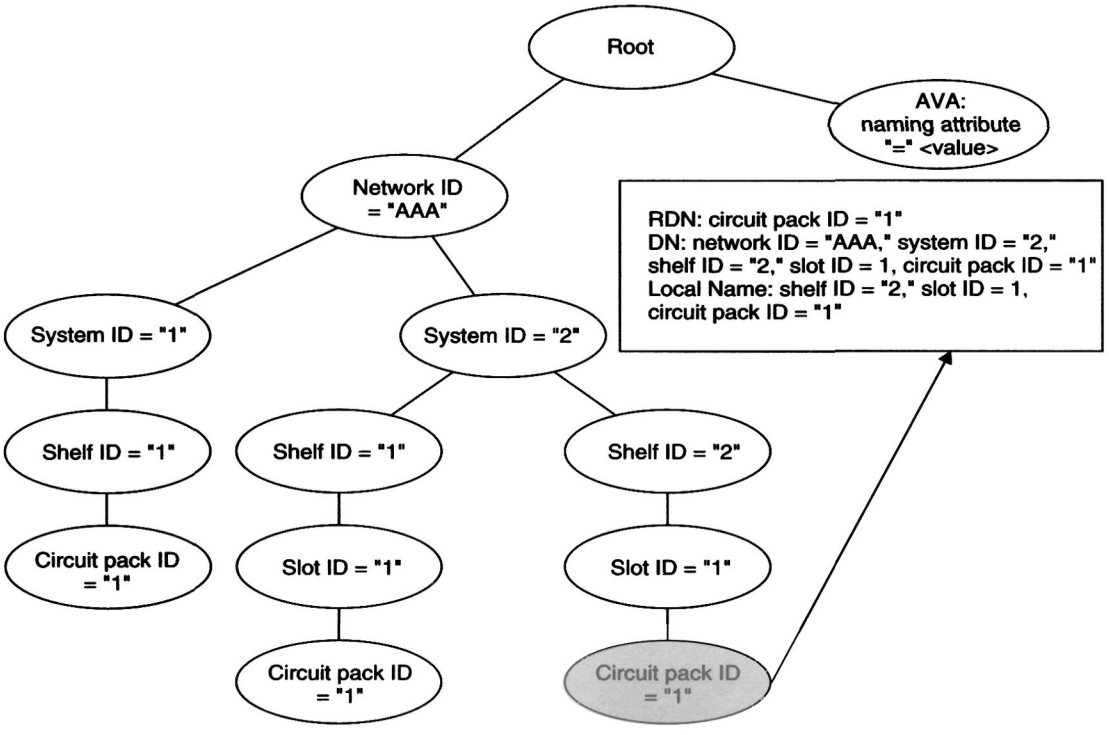

Figure. 1.8 Example Naming Tree.

In addition to identifying the superior and subordinate object classes, the name binding specification also includes the attribute used to name an instance of that class. The value of this naming attribute must be unique across multiple instances of the same class contained in an instance of the superior class. In the figure, slot ID is a naming attribute. A second slot contained in the same shelf (Shelf ID = 2) should have a different value for the slot ID in order to achieve unambiguous identification.

The example in Figure 1.8 also illustrates how to construct local and global names. The local name is relative to the managed element object that represents the NE. Once the managing system establishes communication with the NE, using local names automatically implies that the object is contained relative to that NE. Global name, on the other hand, provides global uniqueness. This is appropriate in cases where a network level management system needs to understand references to objects that may be in different NEs. The global name, however, is long. Another point to recognize here is that the global names are constructed by containing the tree of managed objects relative to objects available from The Directory.[25]

[25] The Directory referenced here is the standard X.500 series. The object classes such as organization are defined in X.521.

1.16. MODELING RELATIONSHIPS

The concepts described so far have addressed the properties of the objects. Relationships between objects are expressed in one of three ways: (1) an attribute in one managed object pointing to another managed object, (2) the containment relationship used for naming an instance, and (3) a managed object class that represents the properties of the relationship between two objects. Examples of the three approaches are: equipment that has a pointer to the objects that are affected if there is a failure in that equipment, naming a circuit pack relative to the equipment holder (slot) where it is inserted, and cross connection representing the properties between a from and to end points within a network element that are connected together. Even though relationships could be modeled with the concepts mentioned above, such an approach does not adequately address the specification of a relationship.

Some of the deficiencies are concepts such as role, cardinality, and integrity constraints associated with the relationship itself. These are not explicitly expressed except via behavior statements. To solve this concern, a separate standard was developed (X.725|ISO/IEC 10165-7) to consider relationship modeling known as Generic Relationship Model (GRM). However, the result of such a model should still be expressible across the interface in terms of the managed object concepts mentioned earlier.

In GRM, the approach taken was to consider relationship as an abstraction without being concerned with the actual realization. That is, relationship should be defined regardless of the representation as an attribute, naming relation, or a managed object. The relationship is defined in terms of roles, behavior, relationship management operations, and notifications. The roles themselves are defined in terms of cardinality, various managed object classes participating in the relationship, and how they come into or are removed from the relationship. The managed relationship is defined in terms of relationship classes. Relationships are realized using relationship mapping of the various components of the relationship class in terms of managed object classes, relationship objects, attributes, and operations.

Notational support for defining relationship classes and relationship mapping are provided.[26] In support of reuse in specification, a managed object class called generic relationship object has been defined. This is specialized from "top" mentioned in the previous section. Additional attributes included are: relationship name, relationship class, and relationship mapping.

In TMN, the work in progress for network level models is planning to use this technique. Relationships span managed objects in multiple systems at the network level and above; it is therefore natural to look at this approach as an appropriate technique to use. Without the introduction of GRM, it is difficult to represent information such as role cardinality, permitted operations on the relationship, and how a resource may enter or depart from a relationship.

[26] Existing TMN models and ISO standard models do not include the concepts identified by the general relationship model. The first example of its use is in the Domain and Policy management function.

1.17. REPRESENTING INFORMATION MODELS

Using the principles for information modeling defined in previous subsections, we can identify components of the managed resources. However, without a standard representation technique, different designers may choose to specify models in a variety of ways ranging from using a formal description language, pseudocode, or text. This obviously leads to difficulties in interpretation and implementation of the models. As part of the standards, a semiformal[27] technique is defined in ITU Recommendation X.722|ISO/IEC 10165-4. This technique is called Guidelines for Definition of Managed Objects (GDMO). In addition to representing the semantics of the information model using this technique, the syntax of information such as the value of an attribute, parameters sent with a notification for external communication are represented using Abstract Syntax Notation one (ASN.1),[28] defined in the ITU Recommendations X.680 series.[29]

GDMO provides templates to be followed when specifying the various components of an information model: managed object class, package, attribute, attribute group, notification, action, name binding, and parameter.[30] The template consists of key words in describing much of the semantics and hence facilitates parsing by machine. The aspect that is not machine parsable is the behavior description. The behavior is written in text form with the natural consequence of being prone to ambiguity and misinterpretation. Formal Description Techniques such as "Z," "object Z," "SDL," and "LOTOS" may be used instead of text. However, there is no one recommended language to use; some of these languages have been used in the industry for automatic code generation. Contributions were presented in ITU SG 15 working on management of transmission systems using "Z." The main concern with using these languages is the additional knowledge (some of the languages are quite complex) the modelers require to ensure that the description is correct. Support for this approach is therefore not unanimous.

[27] The term *semiformal* is used to indicate that not all parts of the specification technique are machine parsable.

[28] ASN.1 was developed to provide an abstract representation of application-specific information exchanged between two systems. Different encoding rules are applied to the specification using ASN.1 for generating the actual octets transmitted across an interface.

[29] Reference is provided to the revised version of the standard commonly known as X.208. To a large extent X.208 and X.680 (not the others in the series) are identical. X.680, in addition to extensions, incorporates corrections based on implementation experience.

[30] The parameter template is used to specify details when an extension field is included in the syntax. To amplify, the concept of providing extension capability was mentioned in the earlier discussion on notifications, actions, and compatibility. In the syntax of a notification, for example, it is common practice to include a field that provides a hole that may be filled in later if the notification is reused in another managed object class. The parameter template is used to specify the syntax to be used for the hole.

Because this chapter is concerned with information modeling more than with how to represent them, further details are deferred. The reader is directed to the standard or books available in the literature.

1.18. DIFFERENCES IN INFORMATION MODELING PRINCIPLES

Section 3 pointed out that, even though information modeling is used in many applications, the details differ. Some differences are summarized in this subsection.

Information models developed by TMN groups and OMG use object-oriented design concepts such as inheritance and encapsulation. The structure of management defined for data communications network management specifies objects that do not possess these properties. The distinction between an object class and an instance of the class is not present. As a result, multiple inheritance or strict inheritance is not applied when developing the information models. Each object type definition in SNMP can be considered to be roughly the equivalent of attribute definition. A collection of information that needs to be grouped together (similar to an object class) is defined using tables. Differences also exist with regard to how an object is named. Because of lack of distinction between class and instance, the name for the type is also the same for an instance except with table representation. In the latter case, an additional index is used to reference a specific row of the table. Even though ASN.1 syntax is used to represent the information exchanged across an interface, properties of the object (data) are specified using a different notation (ASN.1 macro) instead of GDMO. Some major differences are indicated in Table 1.4.

Information models from OMG define interface types similar to object class. Multiple inheritance (with some differences) is used to form an interface hierarchy/graph. While the types of interfaces of the objects in TMN and SNMP are for communications, OMG definitions support programmatic interfaces to facilitate application portability.

TABLE 1.4 SUMMARY OF DIFFERENCES BETWEEN INFORMATION MODELING PRINCIPLES

TMN Models	Internet Management Models
Object classes are collections of properties associated with a resource and are reusable.	Object types are atomic data or tables and are not reusable.
Object classes may be specialized using multiple and strict inheritance.	The concept of inheritance is not used.
Object classes may contain optional attributes and coexist with mandatory attributes.	All variables (object types) within an object group (such as a table) are mandatory.
Containment relation is used for naming objects and results in globally unique names.	The concept of containment does not exist, and the name is unique only within a single system.
No restrictions on the ASN.1 types are used for specifying the syntax of the exchanged information.	Only simple ASN.1 constructs and restricted basic types are permitted for defining the syntax.

1.19. EXAMPLES OF INFORMATION MODELS FOR TMN

Having discussed the principles to be used in developing an information model, let us now take a look at some examples of information available from standards for TMN. Before discussing these examples, an overview of the different modeling efforts for TMN is presented.[31]

1.20. TMN MODELING EFFORTS

ITU[32] SG 7 has developed information models to meet the requirements of systems management functions in general. These functions are generic in the sense that they are applicable to management of both components used for data communications as well as telecommunications. Regardless of the technology used to provide a service, resources emit alarms to indicate fault or failure. The generic function "alarm reporting" specifies five types of alarms (equipment, communication, environmental, processing, and quality of service) and information associated with alarms (probable cause, severity, diagnostic information, specific problems, etc.). This definition is used to support the alarm reporting function in the TMN function set "alarm surveillance." The above case is generic from the perspective that these alarms may be associated with resources supporting various technology. Another function modeled is "Event Report Control." In the discussion on notification, it was noted that a resource (managed object) emitting a notification does not imply it will be communicated to an external system. Criteria may be associated with what events should be sent to a particular system. The model to define and apply the criteria is generic regardless of the notification type, TMN function, or resource. This model is discussed in more detail later in this chapter. These generic functions promote one form of reuse. The software for this model, once developed, can be reused for all the TMN functions associated with different resources. For the TMN X[33] interface, the model to support trouble administration function (X.790) was defined incorporating the information models developed in ANSI T1 and NM Forum Omni*Point1* specification.

ITU SG 4 has developed a recommendation on generic network element information model (Rec: M.3100-GNIM). The model at the present time supports superclasses that are technology independent and suitable for both switching[34] and

[31] The standards covered here are those from ITU. In North America, standards for X-interface and specialization of ITU Recommendations to meet specific North American needs such as performance monitoring of DS1, DS3 (line and path), and SONET terminations have been developed. Similarly, ETSI has developed specialization for European requirements. Other organizations where information models for use in TMN are defined include ATM Forum, NM Forum, and Bellcore.

[32] Previously known as CCITT.

[33] TMN Standard Rec. M.3010 discusses various interfaces for exchanging management information. Q3 is the interface between a Network Element and OS or between a mediation device and an OS. X interface is between OSs in different administrations.

[34] Support for switching network elements is minimal. The standard was developed using the principles outlined for generic (G.805) and SDH-specific (G.803) transmission architecture.

transmission network elements. Work is in progress to address other levels of abstraction such as network and service. Examples of the managed object classes defined are: managed element (NE), different classes to support point to point, and point to multipoint cross connections, and termination points.

ITU SG 15 has developed information models to support management of SDH and ATM network elements. The SDH models documented in the G.774 series define subclasses specialized from Recommendations M.3100 and Q.822 (Performance Monitoring). A model is also provided at a generic level for protection switching. The model for ATM has been completed recently and is in the final ITU approval process. Part of SG 15 work (network level modeling) has been moved to SG 4 since November 1996.

ITU SG 11 has developed information models to support both specific functions independent of the resource (technology independent) and specific technologies. A service provisioning model provides a framework for administering subscriber information in an ISDN switch. The standard defines superclasses and explains how to model bearer services, supplementary services, optional user facilities for the packet mode, directory number, access port profile (service characteristic associated with an access port), and resource aspects. However, to provision specific services, the framework must be extended.

Models containing the framework for TMN function sets, alarm surveillance, performance monitoring, and traffic management have been completed, using the generic model for performance monitoring and ITU SG 15 developed SDH-specific subclasses. Work is in progress to support the V5 interface (configuration, fault, and performance monitoring functions) and usage metering (call detail record) function. Management models are close to completion for network elements in the SS7 network. Starting in November 1996, TMN interface work in SG 11 has been moved to SG 4.

Three examples are included in this section: (1) a model for event report control function, (2) a cross-connection model, and (3) the framework for performance monitoring. Instead of including GDMO definitions, the models are discussed in terms of how various modeling concepts described earlier are used. Referenced standards should be used if the reader is interested in the formal definitions.

1.21. EVENT REPORT MANAGEMENT

A model for event report management should meet the following requirements:

- Identify either a single or a group (for multi-casting) of destinations where the event[35] should be forwarded via a communication interface.
- Specify criteria such as forward the notification if it is of type communication alarm and the severity is critical or major.

[35] The phrases "event" and "notification" are used synonymously. In the standard, notification is used to describe what a managed object emits, and event report is used when the notification is communicated to an external system.

- Schedule the times to forward the notifications.
- Control the initiation, suspension, resumption, and termination of the event report activity.
- Identify the backup destination if communication with the primary destination is not available.
- Configure whether the event report should be communicated requesting confirmation from the managing system.

Except for requirements related to destination and the need for confirming the event report, other requirements are applicable to any activity, including reporting events. In order to provide for reuse, a superclass called discriminator was defined with the properties shown in Figure 1.9.

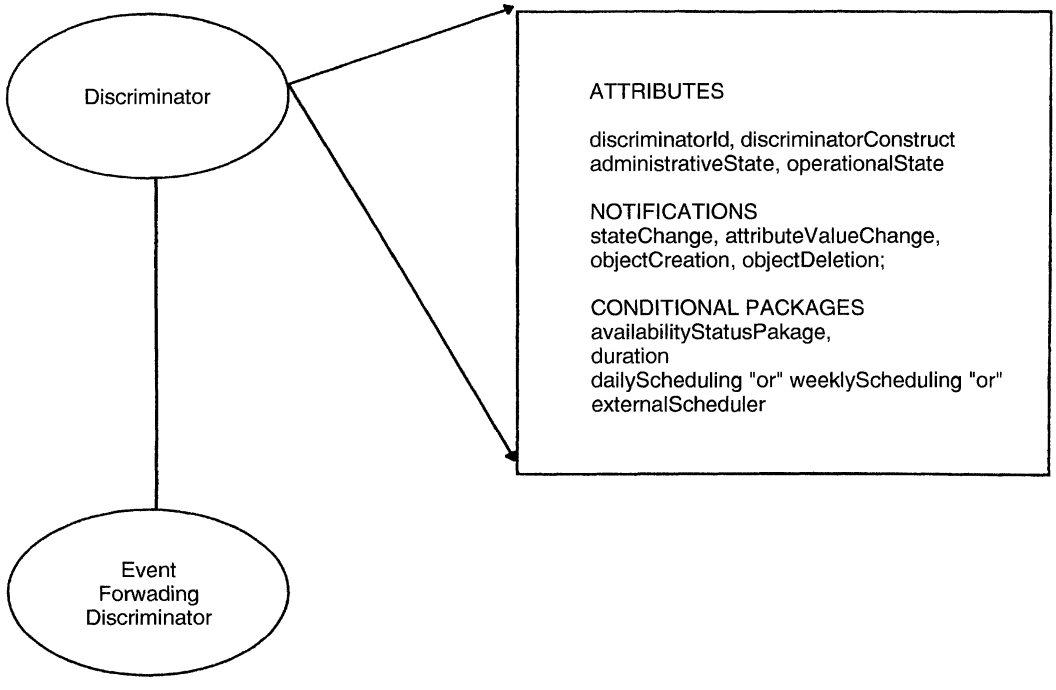

Figure. 1.9 Class Hierarchy for Event Report Control Function.

The inheritance used for discriminator is from "top." In addition to the new attributes defined here, four attributes specified earlier for "top" are also present. The semantics of the components defined for the discriminator managed object class are as follows:

discriminatorID: This naming attribute is used to identify uniquely an instance of either the discriminator class or its subclasses relative to the containing object.[36] The value is settable at creation time.

discriminatorConstruct: This attribute specifies the criteria to perform an activity. It is an expression defined using logical operators. In order to perform the activity, the criteria must evaluate to true. An example of defining a criteria is "perform this activity if (attribute $x > 5$ and attribute y is not present or attribute y has an initial string equal to "sri") is true." The logical expression may be modified by the managing system.

administrativeState: By setting this state attribute to locked or unlocked, the service offered by the discriminator is suspended or resumed. (Initiation is equivalent of resumption if at creation time the value is unlocked.) The value is settable by an external system

operationalState:[37] This attribute reflects the operability of the managed object. The values are enabled or disabled. The values are not controllable by an external system. The value of the state is changed internally based on whether or not the service offered by the discriminator is available.

stateChange, attributeValueChange, objectCreation, and objectDeletion notifications: These correspond to events that are generated as a result of changes in the values of the attributes, changes in states and status, and creation and deletion of the discriminator, respectively. Note that even though states are also attributes, separate notifications are defined to distinguish from changes to any attribute in general. State, status attributes generally applicable across multiple objects, and state change notification are defined in ITU Recommendation X.731, X.721|ISO/IEC 10164-2, 10165-2.

Conditional packages for scheduling: Availability status, included in a package, is used to indicate whether the discriminator activity is scheduled on or off. Different types of scheduling are specified: the duration is intended to provide the start time (which may be the time when the object is created) when the services of the discriminator are made available and an end time (includes continual availability as long as the object is in existence). Within the duration, one can schedule on a daily basis, for example, availability only between 8–5 every day or for each day of the week, or on a per week basis (for Mon–Fri 8–5 and not available on Saturday and Sunday). Instead, an external scheduler such as the ones defined in ISO/IEC 10164-15 |X.746 may also be used. The external scheduler has the advantage that the same scheduler may trigger activities in multiple objects (same or different classes), whereas the daily and weekly scheduling packages affect only the object containing them.

[36] TMN specifications define a managed object class called managed element to represent a network element. This is used as the superior object for discriminator and its subclasses.

[37] For an object like a discriminator, the semantics of operational state may be difficult to comprehend. This is an object that represents the logic used to program this function. On the other hand, it is more natural to understand that the operational state of a circuit pack will change to disabled when the card fails.

The event forwarding discriminator (EFD) is specialized from discriminator, with the special behavior that the activity performed relates to forwarding events to one or more destination. The EFD object class, in addition to this enhanced behavior, adds the mandatory attribute destination. Destination is defined to be either one application entity (an application in a specific system) or a group of entities. The latter case supports the broadcasting environment.

The following two conditional packages are included:

- backUpDestination: This package includes two attributes—a list of destinations to be used in the given order if communication to the system identified by the destination attribute fails,[38] and the attribute active destination to specify the currently active one based on what has been selected from the backup list. Because the active destination is set as a result of selecting one of the items in the backup list, only a read operation is permitted.
- mode: This package, if present, facilitates a managing system to configure the method of receiving event reports. The protocol CMIP specifies two methods of issuing event reports. In the confirmed mode, the manager sends a response acknowledging receipt of the event report. By setting the value of mode attribute to confirmed or otherwise, the managing system can define how the event reports should be issued. If this is not present, then the decision is local to the managed system.

The model for event report management forwards all events to all EFDs.[39] Depending on the result of evaluating the criteria, an event may or may not be forwarded to a specific destination. The standard defines the concept of a preprocessing function. The syntax for a notification may include information that is to be determined by a process outside of the managed object itself. For example, if an alarm is issued with a minor severity and later with a major severity, indication that the two alarms are correlated is done by the preprocessing function prior to including it in the event report. The standard also accounts for an EFD that is not manageable, in other words, the logic for determining which events should be forwarded is internal to the managed system and is not configurable by the managing system.

[38] New work is in progress for an enhanced event report control function. This introduces a new object class disseminator in which the notifications that could not be issued because of communication failure are queued and disseminated later when the links are set up again.

[39] Even though, conceptually, all notifications are seen by all EFDs, in practice this is not required to be implemented in this manner. Depending on the number of EFDs and the criteria to be checked, this may not be optimum for software development. In some cases, specific name bindings are provided to restrict the above behavior so that program logic can be simplified.

1.22. CROSS-CONNECTION MODEL

The model to support cross-connection assignments was developed as part of the Generic Network Information model in ITU Recommendation M.3100. The requirements to be met include:

- Assign a particular channel or time slot for specific use (designation as active Embedded Operations Channel to support access arrangements between a switch and an access node).
- Consider cross connecting a group of terminations with a certain bandwidth used to carry services with another group of terminations with the same bandwidth.
- Provide one or more point-to-point or point-to-multipoint cross connection with a single request.
- Cross connect a termination point selected from an available pool to another termination (either a specific one or selected from a pool).
- Create pre-provisioned cross connection in a state where traffic will not flow until later (by turning on with a management request).
- Create the cross connection as uni- or bidirectional.
- Trace connectivity within the network element when flexible cross connections are present.

In addition to the above generic requirements, within North America when a cross connection is part of a sensitive circuit (line to the President), it is identified as a "red-lined cross connection". The ability to create these specialized cross connections should be supported in some administrations. Details on how to use the generic cross-connection model and examples of its application in SDH (using termination points defined in G.774) are described in an annex to M.3100 (1995).

The object class hierarchy is shown in Figure 1.10. Managed object class fabric is responsible for receiving requests to create and delete cross connections. In addition, it supports actions to create and modify the pool of available terminations and group of terminations for concatenated payloads. The original definition of fabric developed in 1992 was further specialized in the revised Rec. M.3100 to support switching one of the end points of the cross connections to a different end point (termination).

In addition to the naming and state attributes of fabric, the fabric model defines the following actions:

- Add termination point(s) to a group termination point (used for concatenated pay loads).
- Remove termination point(s) from a group termination point.
- Add termination point(s) to a pool.
- Remove termination point(s) from a pool.
- Connect to create one or more cross connections.
- Disconnect to delete cross connection(s).

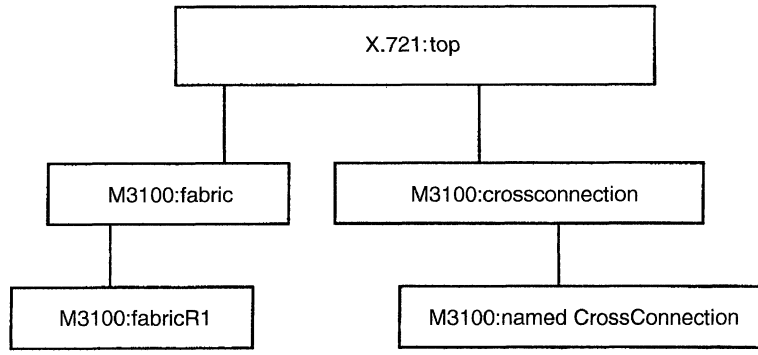

Figure. 1.10 Cross-Connection Model Class Hierarchy.

Cross-connection object definition defines state attributes, from and to termination points of the cross connection, and directionality to indicate uni- or bidirectional flow and signal type. The definition of multipoint cross-connections specifies only the from termination. The "to terminations" are determined from the cross-connection objects contained in the multipoint cross-connection object.

The cross connections created are named as shown in Figure 1.11 relative to the fabric that was requested to create the cross connection. In the case of point to multipoint, the multipoint cross-connection object is contained in the fabric and, as stated above, contains the various cross-connections to the multiple to terminations. Figure 1.11 also shows that when a group termination point (GTP) or TP pool (pool of terminations) is created, it is contained relative to the fabric.

One of the requirements is to include enough information to facilitate tracing connectivity. This is seen in Figures 1.11 and 1.12. The cross-connection objects

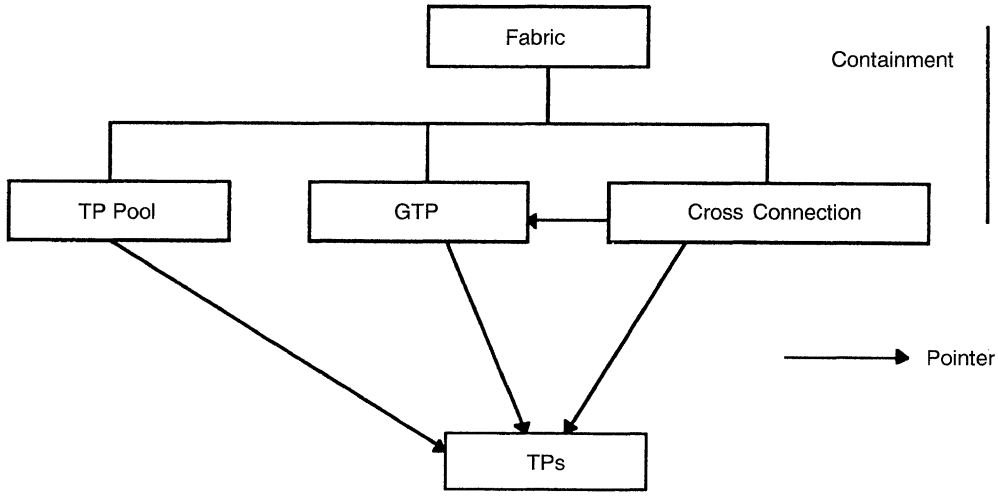

Figure. 1.11 Relationships in Cross-Connection Model. (User-friendly names, instead of M3100 names, are used.)

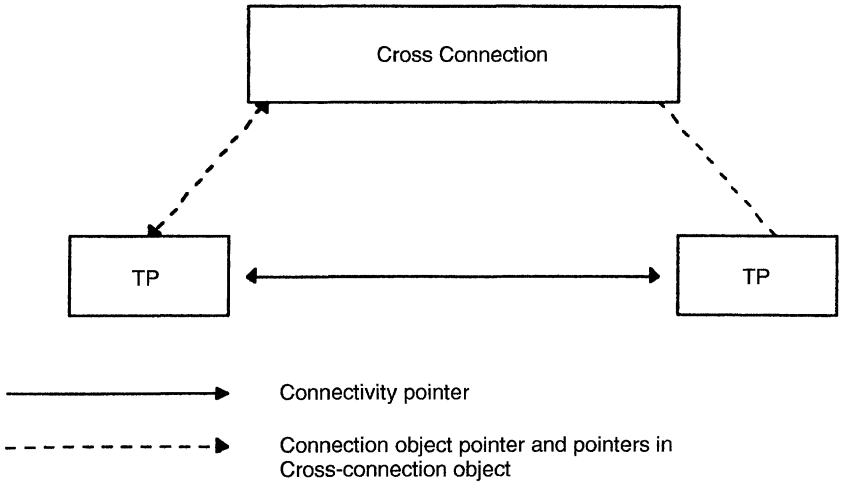

Figure. 1.12 Connectivity Relationships Between Termination Points.

(simple point to point case) have pointers used to identify the connectivity between the terminations via the cross-connection object. The termination point model also supports the backward relation by pointing to the cross-connection object. When there is no flexible cross connection, the two termination points forming the connection will point to each other.

Figure 1.13 illustrates the use of cross connection in an integrated digital loop carrier system. This figure is taken from the model defined in Bellcore GR 836 (defines termination points shown in the figure subclassed from ITU Rec. M.3100) and GR 2833 (to support fiber in the loop, IDLC architectures). The cross connection shown is unidirectional and connects a ds0 within a ds1 to a ds0 (contained within the analog line termination) on the customer side. The use of pointers is quite excessive; however, the complete connectivity information can be traced.

1.23. PERFORMANCE MONITORING FRAMEWORK

The framework for performance monitoring model was developed by ITU SG 11 and is documented in Recommendation Q.822. This is a framework because the managed object classes defined in Q.822 for collecting performance monitoring information are not implementable without additional technology-specific information. In addition to the requirements resulting from the definition of TMN performance monitoring functions (scheduling data collection, reporting on threshold crossing, etc.), examples of the criteria considered in developing the model include:

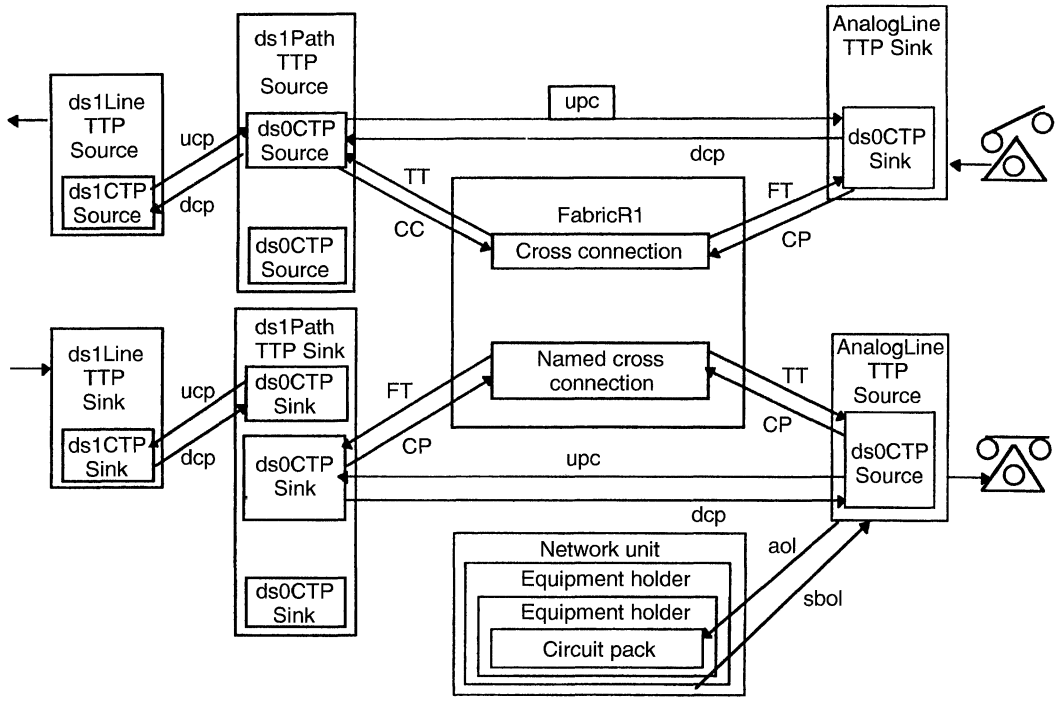

Figure. 1.13 Application of Cross-Connection Model in IDLC System.

- Flexibility to add new performance parameters in two cases: (a) new capability available in the resource, and (b) add or remove dynamically parameters for collecting data for a specific interval.
- Addition or removal of the collected PM parameter values without affecting the existence of the managed object representing the monitored resource.
- Ability to retrieve history information pertaining to individual parameter values.
- Storing object threshold values of the PM parameters applicable to multiple managed resources.

Figure 1.14 shows the managed object class hierarchy for the PM framework. This framework has also been used in the traffic management model. The figure includes both the framework and an example of the specialization required for specific technologies. The scanner object is defined as part of the Metric Objects and attributes (used in generating statistical information), with the behavior to scan and report the scanned data. In addition, scanner includes attributes such as granularity period, which determines how often the parameters are scanned, and attributes for scheduling the scanner activity. The subclass current data include the mandatory

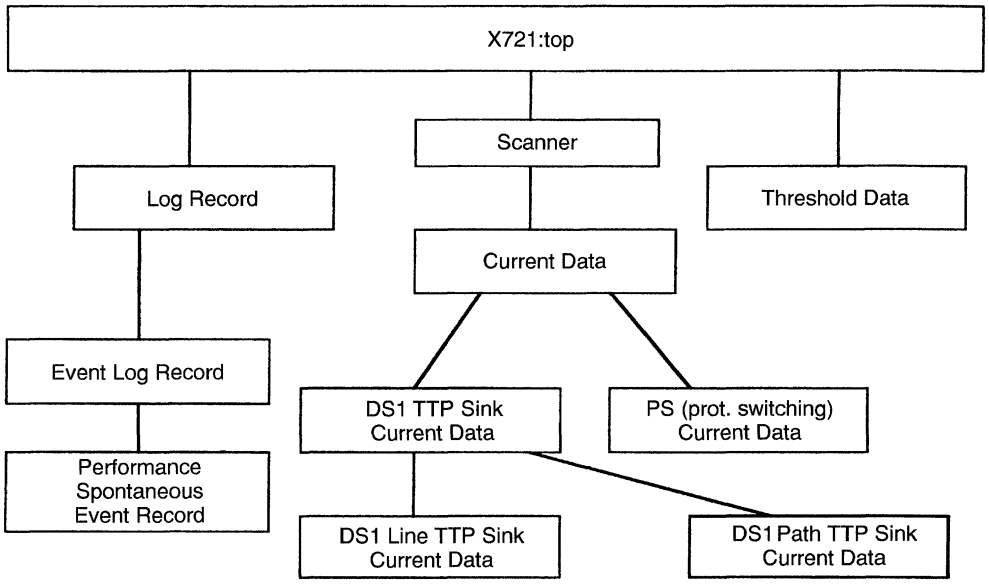

Figure. 1.14 Performance Monitoring Class Hierarchy.

attributes to represent the elapsed time during the collection period and a flag to indicate the validity of the collected data.

Several conditional packages are included. To support the requirement for adding or removing new parameters, the measurement list attribute in the conditional package is used. Because this is a list of attributes, even after the object is created, additional parameters that require collection for special-purpose analysis can be included. Another package of interest contains the attribute that specifies how long the collected information must be retained. In performance monitoring, if the resource is working properly, many of the performance parameters will have a value of zero. Instead of recording several intervals with zero values, using the zero suppression package, it is possible to suppress them, thus avoiding records with just zero values. Two conditional packages are defined to contain two notifications. One notification defines the structure for reporting PM data collected at the end of each interval. The second notification is a quality of service alarm corresponding to threshold crossing alerts.

The instantiable managed object classes are defined by subclassing current data. The subclass includes attributes corresponding to the performance parameters appropriate to that technology. In Figure 1.14, the DS1TTP sink current data is the superclass that contains performance parameters common to both path and line.[40] Further subclasses add parameters that are specific only to line or path.

[40] Recent work in ANSI T1 for SONET and PDH has shown that splitting of line and path current data objects in terms of two classes each (for near and far end performance monitoring parameters) is required.

Several subclasses for collecting PM parameters on SDH-specific terminations are defined in G.774.03.

Based on the attribute in the history retention package (defined in the superclass current data), the collected data will be retained in an instance of the appropriate subclass of history data. The threshold data-managed object class is used to include threshold values for the PM parameters. The collected values are compared against the value in the appropriate threshold data object (if present) to detect a threshold-crossing condition.

The notifications may be logged, and two specific log record object classes are defined corresponding to the two notifications.

The flexibility requirement is met by using the containment relation between the current data (actually instantiated subclasses) and the monitored object. For example, a ds1line current data object will be contained in a ds1line object. By using containment, PM parameters are separated from the observed object itself. This implies that new current data objects (PM parameter collections) can be created and deleted without affecting the observed object. In addition, creating subclasses of an already defined technology specific current data makes it modular without having to redefine the observed object. (This would have been required if the PM parameters were defined to be part of the managed object class representing the monitored resource.) However, the number of managed objects is increased to support this flexibility.

1.24. INFORMATION MODELS IN STANDARDS

As mentioned earlier, several information models have been developed or are in progress to support management of telecommunications network elements. A list of models available in ITU and ISO Recommendations are provided in Table 1.5. The table provides the various ITU Recommendation, status, and a brief description of the areas addressed. For work in progress, the status, may have changed by the time of publication of this book. Readers are encouraged to check information available in the public domain such as the World Wide Web.

TABLE 1.5 LIST OF INFORMATION MODELS IN STANDARDS

Document Number (ITU/ISO/IEC)	Title	Status	Description
X.721/10165-2	Structure of Information—Part 2: Definition of Management Information	IS	Definitions to support OSI Systems Management function such as event report control, log control. In addition, generic object classes top, system are included.

TABLE 1.5 *Continued*

Document Number (ITU/ISO/IEC)	Title	Status	Description
X.734/10164-5	Systems Management—Part 5: Event Report Management Function	IS	Model in text for event report control function.
X.735/10164-6	Systems Management—Part 6: Log Control Function	IS	Model (in text) for log control function.
X.746/10164-15	Systems Management—Part 15: Scheduling Function	IS	Definitions of different types of schedulers to allow periodic, daily, and monthly scheduling of an activity.
X.730/10164-1	Systems Management—Part 1: Object Management Function	IS	Model (in text) for managing creation, deletion, changing values of attributes along with generic notifications applicable to any managed resource.
X.731/10164-2	Systems Management—Part 2: State Management Function	IS	Textual description of state model, attributes and notification applicable in general to several managed resources.
X.732/10164-3	Systems Management—Part 3: Attributes for Representing Relationships	IS	Textual description of relationship attributes and notification applicable in general to several managed objects.
X.750/10164-16	Systems Management—Part 16: Management Knowledge Management Function	DIS	Model and definitions to discover the schema implemented in the managed system.
X.751/10164-17	Systems Management—Part 17: Change Over Function	IS	Model and definitions to support the changeover between the active/standby or backup/backed-up relation between managed objects (resources).
X.744/10164-18	Systems Management—Part 18: Software Management Function	IS	Model and definitions to support software activation, deactivation, and interactive aspects of software download.
X.749/10164-19.2	Systems Management—Part 19: Management Domains and Management Policy Function	DIS	Model and definitions to support identifying domains and policies to be applied for management.

TABLE 1.5 *Continued*

Document Number (ITU/ISO/IEC)	Title	Status	Description
X.743/10164-20	Systems Management— Part 20: Time Management Function	DIS	Model and definitions for managing time of day synchronization and accuracy.
X.733/10164-4	Systems Management— Part 4: Alarm Reporting Function	IS	Model in text of the five types of alarms and the information associated with them. Applicable to several managed resources.
X.745/10164-12	Systems Management— Part 12: Test Management Function	IS	A general framework and generic definitions for testing.
X.737/10164-14.2	Systems Management— Part 14: Confidence and Diagnostic Test Categories	IS	Definitions of specific test categories using the framework mentioned in the previous row.
X.739/10164-11	Systems Management— Part 11: Metric Objects and Attributes	IS	Model and definitions of various metering monitors to sample an attribute value over time and calculate statistics such as mean, variance, and percentile.
X.738/10164-13	Systems Management— Part 13: Summarization Function	IS	Model and definitions to scan attribute values from several objects for specific time periods and provide one packaged report.
X.742/10164-10	Systems Management— Part 10: Usage Metering Function	IS	Model and definitions for a framework to collect usage measurements from resources and report according to triggers.
X.736/10164-7	Systems Management— Part 7: Security Alarm Reporting Function	IS	Model (in text) of the different types of security alarms and the associated parameters. Applicable to several managed resources.
X.737/10164-8	Systems Management— Part 8: Security Audit Trail Function	IS	Model and definitions of the information logged to facilitate auditing the security violations.

TABLE 1.5 *Continued*

Document Number (ITU/ISO/IEC)	Title	Status	Description
X.741/10164-9	Systems Management—Part 9: Objects and Attributes for Access Control	IS	Model and definitions to manage the security information used for controlling access to managed resources.
M.3100	Generic Network Information Model	Approved Rec.	Generic Network Information Model to support transmission and switching network elements (concentration is on transmission NEs).
G.774	Synchronous Digital Hierarchy (SDH) Management Information Model	Approved Rec.	SDH specific network element model based on M.3100 and G.803, transmission architecture.
G.774.01	Synchronous Digital Hierarchy (SDH) Management Information Model for Performance Monitoring	Approved. Rec.	SDH specific definitions based on Q.822 to support performance monitoring of SDH NE.
G.774.03	Synchronous Digital Hierarchy (SDH) Management Information Model for MS Protection Switching	Approved Rec.	Generic and SDH specific information model to support different types of protection switching arrangements.
G.774.04	Synchronous Digital Hierarchy (SDH) Management Information Model for Connection Protection	Approved Rec.	
G.774.02	Synchronous Digital Hierarchy (SDH) Management Information Model for Configuration of Payload Structure	Approved Rec.	Information model for the payload configuration management of SDH networks. The functions addressed are used to configure various SDH adaptation functions.
Q.751.1-MTP	Network Element Manager Information Model for Message Transfer Part (MTP)	Approved Rec.	

TABLE 1.5 *Continued*

Document Number (ITU/ISO/IEC)	Title	Status	Description
Q.751.2-SCCP	Network Element Manager Information Model for the Signaling Connection Control Part (SCCP)	In Res. 1 Procedure	
Q.821	Stage 2 and Stage 3 Description for the Q3 Interface—Alarm Surveillance	Approved Rec.	Generic objects to support alarm surveillance function set. Uses X.733 alarm reporting definitions.
Q.822	Stage 1, Stage 2, and Stage 3 Description for the Q3 Interface—Performance Management	Approved Rec.	Framework for collecting and reporting performance management data—applicable to both performance monitoring and traffic management.
Q.823	Functional Specification for Traffic Management	Submitted for final approval	Model to surveil, audit, traffic data from circuit switches and SS7 network elements and for different types of controls.
Q.824.0	Stage 2 and Stage 3 Description for the Q3 Interface—Customer Administration—Common Information	Approved Rec.	Framework model to support provisioning analog and ISDN services. Expected to form the basis for other technologies.
Q.824.1	Stage 2 and Stage 3 Description for the Q3 Interface—Customer Administration—Integrated Services Digital Network (ISDN)—Basic and Primary Rate Access	Approved Rec.	Specialized from the general framework above, model for service aspects as well as resource aspects to provision both basic rate and primary rate interfaces.
Q.824.2	Stage 2 and Stage 3 Description for the Q3 Interface—Customer Administration—Integrated Services Digital Network (ISDN)—Supplementary Services	Approved Rec.	Framework for developing supplementary services. Examples of how to use the framework to support specific services included.
Q.824.3	Stage 2 and Stage 3 Description for the Q3 Interface—Customer Administration—Integrated Services Digital Network (ISDN)—ISDN Optional User Facilities.	Approved Rec.	Model to support packet mode bearer service.

TABLE 1.5 *Continued*

Document Number (ITU/ISO/IEC)	Title	Status	Description
Q.824.4	Stage 2 and Stage 3 Description for the Q3 Interface—Customer Administration—Integrated Services Digital Network (ISDN)—ISDN Tele Services	Approved Rec.	Model to support teleservices in ITU Recs.
Q.942	Stage 2 and Stage 3 Description for the Q3 Interface—Customer Administration—Integrated Services Digital Network (ISDN)—Service Profile Verification and Service Profile Management	In progress	Model based on the generic framework of Q.824.0 and service descriptions for the switch to CPE interface.

IS—International Standard Status
DIS—Draft International Standard Status

1.25. EXAMPLE INFORMATION MODELS FOR DATA COMMUNICATIONS

The information models developed by Internet groups for managing data communications resources fall into two primary categories:

- **Request for Comments (RFC) models:** These are Internet "standards" that have been formally approved by the Internet Advisory Board (IAB). They are developed under the Internet Engineering Task Force (IETF) by less formal working groups than the national (ANSI) or international (ISO and ITU) standards groups, and consist primarily of individual technical contributors rather than national body representatives.
- **Enterprise-specific models:** These are developed by individual contributors and, therefore, are not formally approved. They may be made available to the Internet community as public domain or may remain proprietary.

The initial set of objects to manage TCP/IP networks were developed in MIB-I [RFC 1155]. Since then, the use of MIB-I has been deprecated with the introduction of MIB-II [RFC 1213]. MIB-II is an updated version of MIB-I with the introduction

of several new object groups, modification of some variables, and deprecation of one object group for address translation. MIB-II object group definitions include system, interfaces, Internet Protocol (IP), transmission, transmission control protocol (TCP), and SNMP. Enterprise-specific information can be added to MIB-II by adding new variables. Other MIBs available in the public domain include DS1 [RFC 1406], DS3 [RFC 1407], SMDS Interface Protocol [RFC 1304], and others. Internet RFCs are also available for customer network management of SONET and ATM.

The system group is a collection of object types, each being an atomic data. These are distinguished from the example below where a group of properties are collected together in a table. Information modeled as part of the system group are: system description, system object identifier, system up time, system contact, system name, system location, and system service. The access definition indicates whether a specific information may be only monitored or modified. System up time, for example, cannot be modified, whereas changes can be made by management exchange to system contact. These access properties are similar to those discussed for TMN models in section 5, even though there are fewer allowed attribute operations than with the TMN paradigm.

For the IP group, object types include tables to store IP addresses, IP routing tables, and IP address translation tables. An IP routing table is composed of Route entries. Each entry has the following columns: IP Route Address, IP Route Index, IP Route Metrics 1 to 4, IP Route Next Hop, IP Route Type, IP Route Protocol, IP Route Age, and IP Route Mask. The information model for this group further defines representation of these entries. Management of this information is obtained by reading or writing values for a row of this table. Each column is defined as an object type, and a table is composed of these object types. A table, similar to a managed object class in TMN, includes a set of properties; however, it cannot be specialized to add another column. Referencing a row is using an index relative to the table reference.

All the object types within a MIB are mandatory for an implementation. The conditional packages that allowed for optionality in TMN information models are not available. Having no optionality, as can be seen from the next section, results in simpler and interoperable interfaces.

1.26. CONFORMANCE AND INTEROPERABILITY

The major objective of the information models for network management is to promote interoperability in a multisupplier communications network and thus enable efficient management of that network. The previous sections discussed how information models to enable unambiguous interpretation of the management information exchanged between the managing and managed system are defined. From the examples presented, it quickly becomes obvious that reasons such as compromises between members developing the standard, differences in how various administrations set up their network and offer services, and flexibility of the model introduce

options in the model. For example, conditional packages with the present condition "if an instance supports it" is nothing more than leaving it an option for the equipment provider to implement the feature. Regardless of whether the communication protocol or management information exchange (application level) is considered, options translate to potential issues with interoperability. This has been demonstrated during the early implementations of X.25 where different options were chosen by different implementors.

It is important to understand the differences between conformance and interoperability. ISO has developed a framework and methodology for conformance testing as part of the ISO 9646 series. An implementor can claim conformance by identifying the options chosen and any restrictions to value ranges, syntax, and so on. These statements from the implementors are tested by organizations such as the Corporation for Open Systems (COS) with a well-defined set of test suites and certify whether the product passes the conformance requirements and check for validity of the claims in the statements from the supplier. Two suppliers may have products that are certified. However, except for the features mandated by the protocol or information model, variations may be found among the various suppliers on the actual options selected.

Let us illustrate the difference between the two concepts using the event forwarding discriminator discussed in section 5. Let us assume that the managed system is implemented without the mode package that allows the manager to configure the EFD regarding receiving notifications as confirmed or nonconfirmed. If an OS sends a request to create an EFD, including the mode attribute, it will fail because the managed system did not implement it. Even though the managed system can respond with the error of unsupported attribute and the managing system can resend the create without the mode attribute, this exchange implies that there is an interoperability issue. Another example is restrictions on the value ranges. Supporting different value ranges by the two communicating partners also leads to interoperability issues.

1.27. CONFORMANCE STATEMENTS

The protocol requirements for the TMN interfaces are specified by selecting options available in the international standards. As part of the international standards specifying the protocols, Protocol Implementation Conformance Statements (PICS) are defined. The conformance statements reflect in a tabular form for every protocol data unit (PDU) which parameters are to be supported and which are optional. In addition, a comment or description column is used to provide any value restrictions and other additional information. These statements, defined as part of the protocol standards, are referred to as static conformance. In other words, the implementor claims by filling the support column what parameters have been implemented relative to the PICS requirements. Implementing a parameter does not imply that it will be sent or received in every exchange of the message.

This is referred to as dynamic conformance, which will be discussed in the section on Profiles.

In addition to the framework and testing methodology, a notation known as Tree and Tabular Combined Notation (TTCN) has been developed to assist in the specification of abstract test suites.

Similar to PICS for protocol, conformance statements are also provided for the information models. The framework for writing conformance statements to the various components of an information model (managed object class, attribute, notification, action, name binding, parameter) is defined in ITU Rec. X.724 |ISO/IEC 10165-6, "Requirements and Guidelines for Implementation Conformance Statement Proforma Associated with OSI Management." The various statements (Management Conformance Summary—MCS, Managed Object Conformance Statement—MOCS, Management Information Definition Statement—MIDS, Managed Relationship Conformance Statement—MRCS) may be used by the implementors to indicate to their customer what is available in their product. The set of conformance statements defined by the above standard addresses conformance from the perspective of the managed system. An amendment is close to final approval for stating conformance from the managing system view.

These statements are specified in a tabular form with the goal of making them machine readable. It is therefore possible to automate the comparison of the conformance statements from the suppliers of the managing and managed system and to determine potential problems with interoperability.

1.28. PROFILES AND INTEROPERABILITY

To facilitate interoperability between suppliers of OSI Standards, the concept of International Standardized Profiles was introduced by the OSI implementors' workshops in North America, Europe, and Asia. ISO developed technical report for the framework and taxonomy of International Standardized Profiles (ISP) in TR 10000-1.

The major goal of the ISPs is to increase the probability of interworking between products from different suppliers. In specifying the profiles, consideration should be given not only to the static aspect of conformance mentioned earlier but also to the dynamic aspects related to the communications exchange. As an example, a parameter within a Protocol Data Unit (PDU) may be defined to be optional. If the static conformance specifies that the parameter is mandatory, this implies that the product must implement that parameter. However, if the protocol defines this parameter to be optionally present in a PDU, then the dynamic conformance will be made optional. In addition, differences may exist relative to sending versus receiving the PDU. A parameter defined as optional may not be present in every exchange of that PDU. However, the receiver must be capable of receiving it if it is sent in the PDU.

1.28.1. Network Management Profiles

The concept of an A-profile has been introduced in the industry by standards and implementation groups. This refers to the requirements on the protocols for the OSI layers 5 through 7. For network management, the requirements for session, presentation, and ACSE are documented in ISP 11183 Part 1. Two profiles called AOM11 (ISP 11183 Parts 1 and 3) and AOM12 (ISP 11183 Parts 1 and 2), are developed for CMISE to support network management. These profiles developed by the implementors' workshops in the United States, Europe and Asia are approved international standards.

It was pointed out earlier that the protocol requirements for the Q3 and X interface are being revised. Requirements in the revised recommendations are specified using the network management profiles.

In addition to the protocol profiles, profiles for each system management function (e.g., log control, event report management) are standardized in ISO 12059 parts. Using these profiles as building blocks, we can define a set of profiles in ISP 12060 parts. For example, ISP 12060-2 includes both alarm reporting and state management capabilities.

The above-mentioned profiles are generic functions that are commonly defined between ISO and ITU and used in TMN. Similar profiles do not exist for TMN applications. Efforts are underway in various organizations to begin work on TMN profiles. The delay is because the conformance statements are not available for all the models in TMN. At the time of writing this chapter, conformance statements exist only for the Generic Network Information Model (Rec. .3100) in ITU Rec. M.3101.

1.28.2. Information Model Ensembles

The concept of ensembles was first introduced in the NM Forum to specify the collection of components of the information models required to meet a specific function. The standards for Q3 interface include management information that is suitable to different types of NEs (e.g., ADM, DCX) and different functions (e.g., cross connection). Ensembles define the collection of objects that are necessary to support different network element functions. (Note that the collection should be specified in terms of logical functions performed by the NEs to allow variations in vendor products that combine different functionalities in a product.) Recently, the concept of solution sets has been introduced by the NM Forum. The solution sets may be generic, such as alarm monitoring, or specific, such as LAN alarm interface. The solution sets provide a high-level description of the problem to be solved, along with references to appropriate documents from regional/international standards and NM forum for the details. All these efforts are aimed at facilitating service providers to request interoperable products from various suppliers without requiring detailed knowledge of how the management information is modeled or the functions are supported by each standard.

1.29. CONSIDERATIONS FOR INTEROPERABLE TMN INTERFACES

The following can be considered a checklist or steps (not necessarily in sequence) required for implementing an interoperable CMISE based interface.

Agreement on the minimum functionality to be deployed in the network to support operations for a specific domain such as alarm surveillance and performance monitoring.

Agreements on the protocol features in CMISE as well as in the lower layers required in order to achieve the above functionality.

Subdivision of the minimum functionality required to support operations in terms of atomic units.

Agreement on the application context to be used.

Subset of the schema required when managing a specific type of NE supporting a specific technology.

Determination of the administration-specific requirements for the selected schema.

Selection of the structure rule for naming instances of the selected object classes.

Security requirements may also have to be addressed in interfaces such as customer network management and between different administrations.

1.30. FUTURE DIRECTIONS

The information models developed in support of TMN addresses the interface between two systems. This emphasis is appropriate for the initial goal of improving interoperability when the network includes multisupplier products. In addition to improving interoperability using standard interfaces, recent advances in software engineering on distributed processing are starting to influence future directions. Some technical reasons[41] for promoting the move toward the new concepts are discussed below. Before introducing these new directions, let us first look at some of the concerns expressed by implementors in building TMN applications based on current specifications.

An important time-to-value consideration in building and deploying products adhering to TMN standards is the availability of the infrastructure components required to easily build TMN applications. The standards do not address implemen-

[41] *Soap box*: As will be seen later, new notational techniques are being introduced in some cases, in addition to describing requirements and information models using new terminology from distributed processing. Based on the work in open distributed management architecture, the need for these complex notations (even though the final product is expected to be GDMO-based information models) is very questionable.

tation aspects such as interobject interactions, moving management information across multiple nodes, replication of data for redundancy, and portability of applications using programming interfaces and recovery scenarios. The existing TMN standards considered these issues as being specifically outside the scope of the work. Until recently sufficient tools have not been available in the marketplace to aid the implementors in developing TMN applications without becoming versatile with many new concepts (e.g., different protocols, syntax transformation, object-oriented concepts, relation between managed objects and resources).

To assist in the rapid development of network management products using CMISE and associated information models, specifications providing bindings to programming languages C and C++ have been (are being) developed by the Open Software Foundation and X/Open committees. Platform products are now available (though limited) with infrastructure components such as

- Application programming interfaces
- Processes for database manipulation
- Generation of transfer syntax
- Generation of appropriate communication protocol data units

These components follow the specifications from X/Open, thus facilitating the portability of applications developed using these components.

1.31. DISTRIBUTED PROCESSING AND TMN

While the development of the tools aid in implementations, it should be recognized that the current specifications are based on the fact that the managed resources and management activity are within a single system. The advent of distributed processing concepts such as client/server has produced several benefits. Some of the advantages are load balancing, failure resilience, increased reliability by redundancy, and increased performance because of concurrent executions. Building network management as a distributed application allows distribution of managed resources across multiple network nodes and makes use of the aforementioned benefits. However, this is not a panacea. As always, there is a price to pay to reap these benefits. Examples of the challenges to be resolved are:

- Connection of disparate systems
- Management of partial failure or differences in availability schedules
- Access and location transparency for the information
- Federation of administrative and technology domains
- Security concerns
- Need for clock synchronization

In addition to the above-mentioned general issues for any distributed processing application, additional problems to be solved specific to TMN include:

- Determination of the agent responsible for specific resources
- Global naming of managed objects to handle location transparency
- Maintaining integrity constraints between objects distributed in different systems
- Correlation of distributed management activities
- Migration of existing information models into a distributed environment with minimal adaptation

A first step toward solving some of the issues for network management is the new work on Open Distributed Management Architecture (ODMA). Before describing ODMA, let us look at some of the concepts developed for open distributed processing (ODP).

1.32. OPEN DISTRIBUTED PROCESSING

The open distributed processing standards (X.901-903|ISO/IEC 10746-1 to 4) are being progressed jointly between ISO and ITU. The ODP reference model includes concepts, modeling approach, and levels of abstractions required in building a distributed system. In addition, an object-oriented framework[42] has been adopted for modeling some of the specifications. The five levels of abstractions discussed are:

Enterprise viewpoint: requirements that address the business goals, policies, and environment for the system independent of how the system is distributed.

Information viewpoint: semantic information that should be stored and processed in the system along with the information flow between the source and sink for the information.

Computation viewpoint: further details on how the information can be decomposed into objects that interact via interfaces. The components are defined to meet the requirements for implementing a distributed system.

Engineering viewpoint: realization of the computational model in a specific environment, for example, using specific protocol mechanisms. Furthermore, mechanisms to support various transparencies associated with distribution and infrastructure components are provided.

[42] The principles described earlier to define a management information model, though object-oriented, are not exactly the same in ODP.

Technology viewpoint: implementation details of the components required for building distributed systems. This viewpoint is considered to be outside the scope of standardization.

Various distribution transparencies are discussed within the ODP standards. Not all transparencies will be applicable for all applications. Examples of transparencies discussed are:

- Access transparency to mask the variations in data representation and invocation mechanism (for example, regardless of the protocol used to invoke an operation).
- Failure transparency to hide the failure of the object from itself (provides for building resilient systems).
- Location transparency to shield from an object the exact locations of the objects with which it interacts.
- Persistence transparency to mask the activation and deactivation of an object so that it appears to be always present for interaction.
- Transaction transparency to hide the coordination (scheduling, monitoring, and recovery functions) of activities across multiple objects to achieve data consistency.

In addition to the prescriptive and descriptive models, infrastructure components to support some of the above-mentioned transparencies are in progress. These components allow users to obtain information about the available services and access them. One such component defined is the "trading" function. This allows servers to advertise the services offered and clients to discover them, thus decoupling the clients and servers. Other components include the distributed object manager to bind and initiate invocations of services provided by server objects, binder, access function, and type manager.

1.33. OPEN DISTRIBUTED MANAGEMENT ARCHITECTURE

ITU-T SG 7 (now in SG 4) and ISO have started new work (ODMA) that expands the OSI management architecture to allow the distribution of resources being managed. The work applies the ODP concepts mentioned above to management. Efforts are also beginning in TMN to consider extensions of the architecture to allow distributed management.

The requirements considered include: delegation of management activities from one manager to another and the resulting need for coordination of distributed management activities, support for distribution transparency, transparency to different communication protocols, portability of management applications, and guidelines for migration of existing models in a distributed environment.

ODMA is described using the five viewpoints of ODP, with the understanding that the extended architecture will have to accommodate systems based on the traditional peer-to-peer approach defined in X.701|ISO/IEC 10040. A mapping between the terminology used in ODP and OSI Systems management is included in ODMA. For example, "request for an operation or notification" is equivalent to the notion of "invocation." An example is also provided as to how to specify existing OSI management standards in the ODP format. Enterprise viewpoint is used to describe the requirements for a function. ODMA uses Rumbaugh's technique (*The Object Modeling Technique* by J. Rumbaugh) to describe the information without the object interface details. The computational viewpoint is provided by the GDMO and GRM definitions. The interface signatures may be specified using the Interface Definition Language (IDL) from Object Management Group (OMG) or CMIS services. The engineering viewpoint is specified in terms of the communication protocols, specifically using the features available in CMIP.

ODP as well as ODMA standards define concepts such as the viewpoints and apply them to management as a distributed application. However, no notational techniques (except in text) for defining these viewpoints are specified in these standards. ITU-T SG 15 has applied these viewpoints concepts for developing a network-level model for transmission. To define the computational and information viewpoints, a GDMO like notation was introduced.[43]

In order to support distribution in management, ODMA has introduced generic functions (using the viewpoints) and the following computational objects: operations dispatcher, notifications dispatcher, and policy enforcer. In describing the object interactions, ODMA introduces manager role object, which was not present in the existing systems management architecture.

1.34. COMMON OBJECT REQUEST BROKER ARCHITECTURE (CORBA)

The above two sections addressed the concepts introduced for developing distributed application specification in general and management as a specific case. These architectures do not address the technology viewpoint that pertains to

[43] *Soap box*: Some have claimed the reason for introducing these notations is to develop a protocol-independent information model. However, it is clear that the schema has been developed with two or three protocols in mind because of some of the restrictions placed on the syntax for communications exchange. While a more rigorous formalism for defining behavior within a schema is very useful, claims that the new notation results in a protocol-independent model have not been proven. A structured text at a higher level than syntax details is possibly a better approach for a protocol-independent information model. Introducing new notations may be a nice academic exercise but may not be palatable to implementors. New compilers will have to be developed to ensure that the syntax is correct and will further delay implementations.

implementation details. One important goal of CORBA is to build portable implementations.

Addressing the implementation issues is an architecture developed by Object Management Group, known as CORBA. Details are available in a series of specifications from X/Open. This architecture provides a flexible approach to integrate a variety of object system implementations, regardless of whether or not these objects are supporting management application. The architecture is built on the software engineering concepts of client and server similar to that described in ODP. The Object Request Broker (ORB) shields from the client the details of the object implementation, the programming language used, and its location. The client may use specific stubs or the interface independent method to invoke operations on an object. The core of the Object Request Broker locates the specific implementation and transfers parameters to a skeleton. An object adapter may be required for the object implementation to request services such as security, and mapping the object reference to implementation. Once the requested operation is completed, the result is returned to the client. The object interfaces within the CORBA architecture defined by OMG uses Interface Definition Language (IDL). As part of CORBA, Application Programming Interfaces are defined to ease the development of distributed applications.

In this chapter, we have either discussed in detail or referenced three different object modeling technologies: models used with CMIP, those with SNMP, and CORBA. These were developed to meet different goals. In order to provide an environment where the strengths of these approaches are combined, a Joint Inter-Domain Management group (JIDM) was formed by X/Open and Network Management Forum (NMF). As part of this effort to provide an interoperable environment, translations of concepts and notations between the three methods are provided. Because of the differing power of the three techniques, reconciling the models for differences is also included. The aim here is to provide much of the mapping that can be automated in a gateway so that interoperability between the varied implementations can be achieved.

It is expected that some of these concepts may be applied when TMN architecture is enhanced in the next study period. Arguments for introducing some of the complexity relative to the benefits offered continue in the standards group; however, the jury is still out on what will succeed in the marketplace. Combining powerful techniques is a goal, and several of the concepts are still maturing. Time will tell the winner.

1.35. SUMMARY

This chapter focuses on information modeling, specifically that used in network management. Several network management systems that exist in the service providers network use a message-based paradigm. The concept of modeling management information, thus providing a rigorous formalism, is an essential part of two-network management methods in the industry. This chapter discusses the various

object-oriented information modeling concepts such as inheritance, relationships, and how to model various characteristics of the resource being managed. These concepts were described using examples drawn from existing TMN and Internet management. A glimpse of other modeling efforts and the influence of distributed processing from the latest advances in software development are provided to suggest future directions to the reader.

References

[1] CCITT Recommendation X.720 (1992) | ISO/IEC 10165-1 (1992). Information Technology—Open Systems Interconnection—Structure of Management Information: Management Information Model.

[2] CCITT Recommendation X.721 (1992) | ISO/IEC 10165-2 (1992). Information Technology—Open Systems Interconnection—Structure of Management Information: Generic Management Information.

[3] CCITT Recommendation X.722 (1992) | ISO/IEC 10165-4 (1992). Information Technology—Open Systems Interconnection—Structure of Management Information: Guidelines for the Definition of Managed Objects.

[4] ISO/IEC 18824 (1990). Information Technology—Open Systems Interconnection—Specification of Abstract Notation One (ASN.1) | CCITT Recommendation X.208. (1988). Specification of Abstract Syntax Notation One. Geneva.

[5] CCITT Recommendation G.773 (1990). Protocol Suites for Q-Interfaces for Management of Transmission Systems. Geneva.

[6] CCITT Recommendation G.774. SDH Management Information Model for the Network Element View.

[7] CCITT Recommendation M.3010. Principles for a Telecommunications Management Network (TMN).

[8] CCITT Recommendation M.3020. TMN Interface Specification Methodology.

[9] CCITT Recommendation M.3100. Generic Network Information Model.

[10] CCITT Recommendation M.3180. Catalogue of TMN Managed Objects.

[11] CCITT Recommendation M.3200. TMN Management Services: Overview.

[12] CCITT Recommendation M.3300. F-Interface Management Capabilities.

[13] CCITT Recommendation M.3400. TMN Management Functions.

[14] CCITT Recommendation Q.811. Lower Layer Protocol Profiles for the Q3 Interface.

[15] CCITT Recommendation Q.812. Upper Layer Protocol Profiles for the Q3 Interface.

[16] CCITT Recommendation Q.821. Stage 2 and Stage 3 Description for the Q3 Interface.

[17] CCITT Recommendation X.701 | ISO/IEC 10040: 1992. Information Technology—Open Systems Interconnection—Systems Management Overview.

[18] CCITT Recommendation X.710. (1991). Common Management Information Service Definition for CCITT Applications.

[19] CCITT Recommendation X.711 | ISO/IEC 9596-1: 1991 (E). Information Technology—Open Systems Interconnection—Common Management Information Protocol Specification—Part 1: Specification, Edition 2.

[20] CCITT Recommendation X.712 | ISO/IEC 9596-2: 1992 (E). Information Technology—Open Systems Interconnection—Common Management Information Protocol—Part 2: Protocol Implementation Conformance Statement (PICs) Proforma.

[21] CCITT Recommendation X.723 | ISO/IEC 10165-6: 1992. Information Technology—Open Systems Interconnections—Structure of Management Information—Part 6: Requirements and Guidelines for Implementation Conformance Statement Proformas Associated with Management Information.

[22] CCITT Recommendation X.724 | ISO/IEC 10165-5: Information Technology—Open Systems Interconnections—Structure of Management Information—Part 5: Generic Managed Information, ISO/IEC JTC1/SC21 N6572, February 20, 1992.

[23] CCITT Recommendation X.730 | ISO/IEC 10164-1: Information Technology—Open Systems Interconnections —Systems Management—Part 1: Object Management Function, ISO/IEC JTC1/SC21 N6355, October 15, 1991.

[24] CCITT Recommendation X.731 | ISO/IEC 10164-2: Information Technology—Open Systems Interconnections—Systems Management—Part 2: State Management Function, ISO/IEC JTC1/SC21 N6356, October 15, 1991.

[25] CCITT Recommendation X.732 | ISO/IEC 10164-3: Information Technology—Open Systems Interconnections—Systems Management—Part 3: Attributes for Representing Relationships, ISO/IEC JTC1/SC21 N6357, October 15, 1991.

[26] CCITT Recommendation X.733 (1992) | ISO/IEC 10164-4: 1992. Information Technology—Open Systems Interconnection—Systems Management: Alarm Reporting Function.

[27] CCITT Recommendation X.734 | ISO/IEC 10164-5: 1992. Information Technology—Open Systems Interconnection—Systems Management: Event Report Management Function.

[28] CCITT Recommendation X.735 | ISO/IEC 10164-6: 1992. Information Technology—Open Systems Interconnection—Systems Management: Log Control Function.

[29] CCITT Recommendation X.736 | ISO/IEC 10164-7: Information Technology—Open Systems Interconnection—Systems Management—Part 7: Security Alarm Reporting Function, ISO/IEC JTC1/SC21 N6367, October 15, 1991.

[30] CCITT Recommendation X.740 | ISO/IEC 10164-8: Information Technology—Open Systems Interconnection—Systems Management—Part 8: Security Audit Trail Function ISO/IEC JTC1/SC21 N7039, June 2, 1992.

[31] CCITT Recommendation X.737 | ISO/IEC 10164-14: Information Technology—Open Systems Interconnection—Systems Management—Part 14: Confidence and Diagnostic Test Categories, 1995.

[32] CCITT Recommendation X.738 | ISO/IEC 10164-13: Information Technology—Open Systems Interconnection—Systems Management—Part 13: Summarization Function, 1994.

[33] CCITT Recommendation X.739 | ISO/IEC 10164-11: Information Technology—Open Systems Interconnection—Systems Management—Part 11: Workload Monitoring Function, 1993.

[34] CCITT Recommendation X.741 | ISO/IEC 10164-9: Information Technology—Open Systems Interconnection—Systems Management—Part 9: Objects and Attributes for Access Control, 1995.

[35] CCITT Recommendation X.742 | ISO/IEC 10164-10: Information Technology—Open Systems Interconnection—Systems Management—Part 10: Accounting Meter Function, 1994.

[36] CCITT Recommendation X.745 | ISO/IEC 10164-12: Information Technology—Open Systems Interconnection—Systems Management—Part 12: Test Management Function, 1993.

[37] CCITT Recommendation X.746 | ISO/IEC 10164-15: Information Technology—Open Systems Interconnection—Systems Management—Part 15: Scheduling Function, 1994.

[38] ISO/IEC 7498-4: 1989. Information Processing Systems—Open Systems Interconnection—Basic Reference Model—Part 4: Management Framework.

[39] ISO/IEC ISP 11183-1: Information Technology—International Standardized Profiles—OSI Management—Management Communications Protocols—Part 1: Specification of ACSE, Presentation and Session Protocols for the use by ROSE and CMISE, May 1992.

[40] ISO/IEC ISP 1183-2, Information Technology—International Standardized Profiles—ISO Management—Management Communications Protocols—Part 2: AOM12—Enhanced Management Communications, June 1992.

[41] ISO/IEC ISP 1183-3, Information Technology—International Standardized Profiles—OSI Management—Management Communications Protocols—Part 3: AOM11—Basic Management Communications, May 1992.

[42] ISO/IEC ISP 12059-0, Information Technology—International Standardized Profiles—OSI Management—Common Information for Management Functions—Part 0: Common definitions for management function profiles, 1994.

[43] ISO/IEC ISP 12059-1, Information Technology—International Standardized Profiles—OSI Management—Common Information for Management Functions—Part 1: Object Management, 1994.

[44] ISO/IEC ISP 12059-2, Information Technology—International Standardized Profiles—OSI Management—Common Information for Management Functions—Part 2: State Management, 1994.

[45] ISO/IEC, Information Technology—International Standardized Profiles—OSI Management—Common Information for Management Functions—Part 3: Attributes for Representing Relationships, 1994.

[46] ISO/IEC ISP 12059-4, Information Technology—International Standardized Profiles—OSI Management—Common Information for Management Functions—Part 4: Alarm Reporting, 1994.
[47] ISO/IEC ISP 12059-5, Information Technology—International Standardized Profiles—OSI Management—Common Information for Management Functions—Part 5: Event Report Management, 1994.
[48] ISO/IEC ISP 12059-6, Information Technology—International Standardized Profiles—OSI Management—Common Information for Management Functions—Part 6: Log Control, 1994.
[49] ISO/IEC 12060-1, Information Technology—International Standardized Profiles AOM2n OSI Management—Management Functions—Part 1: AOM211—General Management Capabilities, 1994.
[50] ISO/IEC 12060-2, Information Technology—International Standardized Profiles AOM2n OSI Management—Management Functions—Part 2: AOM212—Alarm Reporting and State Management Capabilities, 1994.
[51] ISO/IEC 12060-3, Information Technology—International Standardized Profiles AOM2n OSI Management—Management Functions—Part 3: AOM213—Alarm Reporting Capabilities, 1994.
[52] ISO/IEC 12060-4, Information Technology—International Standardized Profiles AOM2n OSI Management—Management Functions—Part 4: AOM221—General Event Report Management, 1994.
[53] ISO/IEC 12060-5, Information Technology—International Standardized Profiles AOM2n OSI Management—Management Functions—Part 5: AOM231—General Log Control, 1994.
[54] ISO/IEC TR 10000-1, Information Technology—Framework and Taxonomy of International Standardized Profiles—Part 1: Framework.
[55] Sloman, M. (ed), *Network and Distributed Systems Management*, Addison-Wesley 1994.
[56] Shaler S., and Mellor, S., *Object Lifecycles—Modeling the World in States*, Prentice-Hall, 1992.
[57] Shaler, S., and Mellor, S., *Object-Oriented Systems Analysis—Modeling the World in Data*, Prentice-Hall, 1988.
[58] Flavin, M., *Fundamental Concepts of Information Modeling*, Prentice-Hall, 1981.
[59] Stallings, W., *Networking Standards—A Guide to OSI, ISDN, LAN, and MAN Standards*, Addison-Wesley, 1993.
[60] Draft ITU-T Recommendation X.703|ISO/IEC DIS 13244, Information Technology, Open Distributed Processing Management Architecture (ODMA), 1997.
[61] ISO/IEC 10746-1|ITU Rec. X.901 Information Technology—Open Distributed Processing—Reference Model—Part 1: Overview and Guide to Use, 1995.
[62] ISO/IEC 10746-2|ITU Rec. X.902 Information Technology—Open Distributed Processing—Reference Model: Foundations, 1995.
[63] ISO/IEC 10746-3|ITU Rec. X.902 Information Technology—Open Distributed Processing—Reference Model: Architecture, 1995.

[64] ISO/IEC 10746-4|ITU Rec. X.902 Information Technology—Open Distributed Processing—Reference Model: Architectural Semantics, 1995.
[65] Draft ISO/IEC 13235|ITU Rec. X.9tr Information Technology—ODP Trading Function.
[66] Common Object Request Broker: Architecture and Specification, OMG and X/Open Revision 1.2, 1993.
[67] X/Open Preliminary Specification: Part 2 Object Model Comparison, April 1995.