INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

X.703

(10/97)

SERIES X: DATA NETWORKS AND OPEN SYSTEM COMMUNICATIONS

OSI management – Systems Management framework and architecture

---

Information technology – Open Distributed Management Architecture

ITU-T Recommendation X.703

(Previously CCITT Recommendation)

ITU-T X-SERIES RECOMMENDATIONS

DATA NETWORKS AND OPEN SYSTEM COMMUNICATIONS

PUBLIC DATA NETWORKS	
Services and facilities	X.1–X.19
Interfaces	X.20–X.49
Transmission, signalling and switching	X.50–X.89
Network aspects	X.90–X.149
Maintenance	X.150–X.179
Administrative arrangements	X.180–X.199
OPEN SYSTEM INTERCONNECTION	
Model and notation	X.200–X.209
Service definitions	X.210–X.219
Connection-mode protocol specifications	X.220–X.229
Connectionless-mode protocol specifications	X.230–X.239
PICS proformas	X.240–X.259
Protocol Identification	X.260–X.269
Security Protocols	X.270–X.279
Layer Managed Objects	X.280–X.289
Conformance testing	X.290–X.299
INTERWORKING BETWEEN NETWORKS	
General	X.300–X.349
Satellite data transmission systems	X.350–X.399
MESSAGE HANDLING SYSTEMS	X.400–X.499
DIRECTORY	X.500–X.599
OSI NETWORKING AND SYSTEM ASPECTS	
Networking	X.600–X.629
Efficiency	X.630–X.639
Quality of service	X.640–X.649
Naming, Addressing and Registration	X.650–X.679
Abstract Syntax Notation One (ASN.1)	X.680–X.699
OSI MANAGEMENT	
Systems Management framework and architecture	**X.700–X.709**
Management Communication Service and Protocol	X.710–X.719
Structure of Management Information	X.720–X.729
Management functions and ODMA functions	X.730–X.799
SECURITY	X.800–X.849
OSI APPLICATIONS	
Commitment, Concurrency and Recovery	X.850–X.859
Transaction processing	X.860–X.879
Remote operations	X.880–X.899
OPEN DISTRIBUTED PROCESSING	X.900–X.999

For further details, please refer to ITU-T List of Recommendations.

INTERNATIONAL STANDARD 13244

ITU-T RECOMMENDATION X.703

INFORMATION TECHNOLOGY – OPEN DISTRIBUTED MANAGEMENT ARCHITECTURE

Source

The ITU-T Recommendation X.703 was approved on the 24th of October 1997. The identical text is also published as ISO/IEC International Standard 13244.

FOREWORD

ITU (International Telecommunication Union) is the United Nations Specialized Agency in the field of telecommunications. The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of the ITU. The ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Conference (WTSC), which meets every four years, establishes the topics for study by the ITU-T Study Groups which, in their turn, produce Recommendations on these topics.

The approval of Recommendations by the Members of the ITU-T is covered by the procedure laid down in WTSC Resolution No. 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

INTELLECTUAL PROPERTY RIGHTS

The ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. The ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, the ITU had not received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementors are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database.

CONTENTS

INTERNATIONAL STANDARD

ITU-T RECOMMENDATION

INFORMATION TECHNOLOGY – OPEN DISTRIBUTED MANAGEMENT ARCHITECTURE

1 Scope

This Recommendation | International Standard describes the Open Distributed Management Architecture (ODMA). ODMA provides an architecture for the specification and development both of systems management as an open distributed application and of the management of open distributed applications. ODMA also provides the architectural framework for the development of the standards needed within the architecture. The management will be of a distributed nature; this implies:

- distribution of the managing activity;
- management of distributed applications; and
- management of resources that may be distributed.

ODMA is compliant with the ODP-RM, so that in a distributed environment OSI Systems Management can be used in combination with other techniques that are engineered and implemented according to ODP principles.

This Recommendation | International Standard is the base document of a (potential) range of standards and Recommendations to be developed within ODMA. Figure 1 provides an overview of the relationship between this Recommendation | International Standard and other standards.

Other standards that may be developed within ODMA are:

- ODMA supports: Based on the General Framework of ODMA, these standards give descriptions of specific systems support of ODMA. For example, the OSI Systems Management and CORBA support of ODMA have been identified.
- ODMA viewpoint notations: These component standards provide standardised notations for describing the ODP viewpoints for ODMA (see for example Annex D). These notations are described in separate documents for the ODMA viewpoint notations.
- ODMA functions: These component standards describe functions that are necessary for the construction of an Open Distributed Management System. Some example functions like the operation dispatcher function or the notification dispatcher function are outlined in this Recommendation | International Standard.
- ODMA inter-domain functions: These component standards describe the interworking between different paradigms providing support for ODMA, for example, between OSI Systems Management and CORBA.

As illustrated in Figure 1 this Recommendation | International Standard only elaborates a subset of supporting ODMA systems but allows for developments of other clauses. As a consequence, this Recommendation | International Standard consists of two sections:

1) General Framework

   This clause describes ODMA as a specific interpretation of the Reference Model of Open Distributed Processing for the purpose of management. It introduces general terms that are needed for open distributed management. It may also identify tools for the description of the open distributed management applications.

2) OSI management support for ODMA

   This clause describes the OSI management support for ODMA. It relates the current OSI systems management concepts to ODMA concepts. However, it extends the current systems management standards to support the distribution of the management activities and the distribution of resources to be managed. As this specific interpretation reflects the current OSI standards, limitations may be imposed. For instance, only a number of distribution transparencies may be supported by the (extended) OSI management mechanisms.

Table 1 illustrates which viewpoints are of relevance for which documents (indicated by plus sign). A plus sign indicates that a document describes the viewpoint.

Although the document is divided in clauses, the concepts of the ODMA are overarching and are meant to be used as a bridging architecture between the different paradigms supporting ODMA.

In Figure 1 the ‘standards and specifications based on ODMA’ represent all the specifications and standards that will be developed using the ODMA standards.

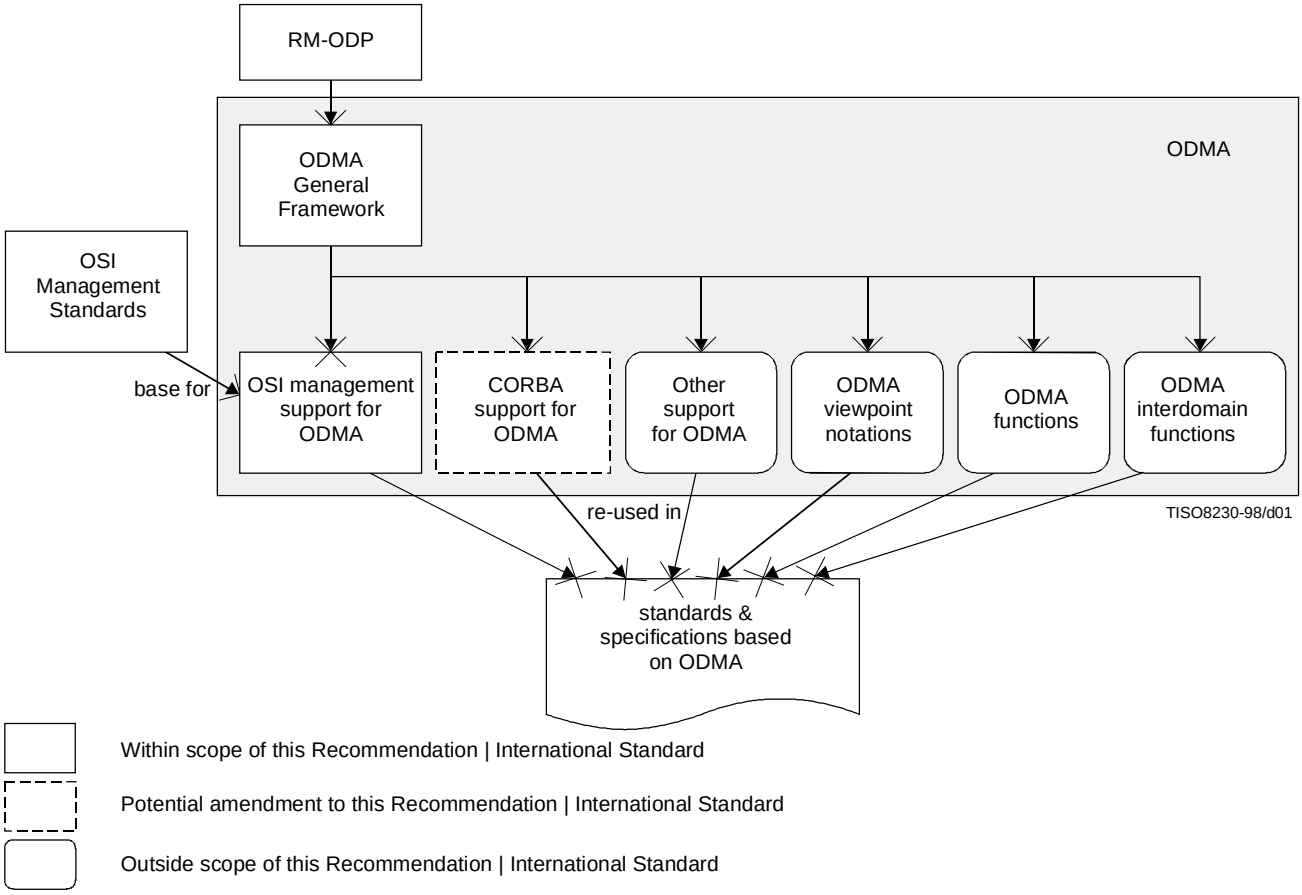


Figure 1 – ODMA document roadmap

Table 1 – Organisation of ODMA documents

	General Framework	OSI management support	CORBA support	...	ODMA functions
Enterprise	+				+
Information	+				+
Computational	+	+	+		+
Engineering	+	+	+		+
Technology					

2 References

The following Recommendations and International Standards contain provisions which, through reference in this text, constitute provisions of this Recommendation | International Standard. At the time of publication, the editions indicated were valid. All Recommendations and Standards are subject to revision, and parties to agreements based on this Recommendation | International Standard are encouraged to investigate the possibility of applying the most recent edition of the Recommendations and Standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards. The Telecommunication Standardisation Bureau of the ITU maintains a list of currently valid ITU-T Recommendations.

2.1 Identical Recommendations | International Standards

- ITU-T Recommendation X.500 (1993) | ISO/IEC 9594-1:1995, *Information technology – Open Systems Interconnection – The Directory: Overview of concepts, models and services*.
- ITU-T Recommendation X.701 (1997) | ISO/IEC 10040:1998, *Information technology – Open Systems Interconnection – Systems management overview*.
- ITU-T Recommendation X.702 (1995) | ISO/IEC 11587:1996, *Information technology – Open Systems Interconnection – Application context for systems management with transaction processing* .
- ITU-T Recommendation X.710 (1997) | ISO/IEC 9595:1998, *Information technology – Open Systems Interconnection – Common management information service*.
- CCITT Recommendation X.720 (1992) | ISO/IEC 10165-1:1993, *Information technology – Open Systems Interconnection – Structure of management information: Management information model* .
- CCITT Recommendation X.721 (1992) | ISO/IEC 10165-2:1992, *Information technology – Open Systems Interconnection – Structure of management information: Definition of management information.*
- 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.*
- ITU-T Recommendation X.725 (1995) | ISO/IEC 10165-7:1996, *Information technology – Open Systems Interconnection – Structure of management information: General relationship model* .
- CCITT Recommendation X.734 (1992) | ISO/IEC 10164-5:1993, *Information technology – Open Systems Interconnection – Systems Management: Event report management function* .
- CCITT Recommendation X.735 (1992) | ISO/IEC 10164-6:1993, *Information technology – Open Systems Interconnection – Systems Management: Log control function*.
- ITU-T Recommendation X.739 (1993) | ISO/IEC 10164-11:1994, *Information technology – Open Systems Interconnection – Systems Management: Metric objects and attributes*.
- ITU-T Recommendation X.749 (1997) | ISO/IEC 10164-19:1997, *Information technology – Open Systems Interconnection – Systems management domain and management policy management function.*
- ITU-T Recommendation X.750 (1996) | ISO/IEC 10164-16:1997, *Information technology – Open Systems Interconnection – Systems Management: Management knowledge management function* .
- ITU-T Recommendation X.901 (1997) | ISO/IEC 10746-1:1997 *Information technology – Open Distributed Processing – Reference model: Overview*.
- ITU-T Recommendation X.902 (1995) | ISO/IEC 10746-2:1996, *Information technology – Open Distributed Processing – Reference model: Foundations*.
- ITU-T Recommendation X.903 (1995) | ISO/IEC 10746-3:1996, *Information technology – Open distributed processing – Reference Model: Architecture.*
- ITU-T Recommendation X.920 (1997) | ISO/IEC 14750:1998, *Information technology – Open Distributed Processing – Interface Definition Language.*
- ITU-T Recommendation X.950 (1997) | ISO/IEC 13235-1:1997, *Information technology – Open distributed processing – Trading function: Specification.*

2.2 Additional references

- ITU-T Recommendation G.805 (1995), *Generic functional architecture of transport networks.*
- ITU-T Recommendation G.851.1 (1996), *Management of the transport network – Application of the RM-ODP framework.*
- ITU-T Recommendation G.852.1 (1996), *Management of the transport network – Enterprise viewpoint for simple subnetwork connection management.*
- ITU-T Recommendation G.853.2 (1996), *Subnetwork connection management information viewpoint.*
- ITU-T Recommendation M.3100 (1995), *Generic network information model.*
- ITU-T Recommendation Q.821 (1993), *Stage 2 and stage 3 description for the Q3 interface – Alarm surveillance.*

3. Definitions

Unless qualified by the word “managed”, the term “object” in this Recommendation | International Standard refers to an ODP object as defined in ITU-T Rec. X.901 | ISO/IEC 10746-1.

3.1 ODP-RM definitions

This Recommendation | International Standard makes use of the terms defined in ITU-T Rec. X.902 | ISO/IEC 10746-2 shown in the Table 2.

Table 2 – Terms taken from ITU-T Rec. X.902 | ISO/IEC 10746-2

abstraction;	distributed processing;	ODP system;
action;	distribution transparency;	open distributed processing;
activity;	entity;	permission;
architecture;	environment contract;	persistence;
behaviour;	failure;	policy;
binding;	fault;	portability;
class;	identifier;	postcondition;
client object;	information;	precondition;
communication;	instance;	prohibition;
compliance;	interaction;	quality of service;
composition;	interface;	role;
configuration;	interface signature;	server object;
conformance;	invariant;	state;
contract;	management information;	system;
creation;	name;	type;
data;	naming action;	viewpoint.
decomposition;	object;	
deletion;	obligation;	

This Recommendation | International Standard makes use of the terms defined in ITU-T Rec. X.903 | ISO/IEC 10746-3 shown in the Table 3.

Table 3 – Terms taken from ITU-T Rec. X.903 | ISO/IEC 10746-3

announcement;	dynamic schema;	parameter
basic engineering object;	engineering interface;	protocol object;
binder;	invariant schema;	reactivation;
channel;	node;	static schema;
cluster;	operation;	stub.
communication domain;	operation interface signature;	

3.2 OSI Management definitions

This Recommendation | International Standard makes use of the term defined in ITU-T Rec. X.950 | ISO/IEC 11235-1.

- trader.

This Recommendation | International Standard makes use of the terms defined in ITU-T Rec. X.701 | ISO/IEC 10040.

- agent;
- manager;
- managed object class;
- MIS-user.

This Recommendation | International Standard makes use of the term defined in ITU-T Rec. X.500 | ISO/IEC 9594-1.

- directory.

3.3 Additional definitions

3.3.1 computational management object: A specific name for ODP conformant computational objects that offer at least a managing or managed interface.

3.3.2 engineering management object: A specific name for ODP conformant basic engineering objects that offer at least a managing or managed interface.

3.3.3 linked reply operation: A sequence of operations between computational management objects in managing and managed role. The first operation is initiated by the object in the managing role. The subsequent operations are initiated by the objects in the managed role and convey replies to the managing object.

3.3.4 linked reply client interface: An operation interface of a computational management object that can emit multiple replies.

3.3.5 linked reply server interface: An operation interface of a computational management object that can accept multiple replies from multiple systems management operation messages.

3.3.6 management object: An object that can take either a managing or a managed role or both.

3.3.7 managed role: The behaviour of a computational management object with respect to its performing of systems management operations and its emission of systems management notifications in interactions with another computational management object.

3.3.8 managing role: The behaviour of a computational management object with respect to its handling of systems management notifications and its initiation of systems management operations in interactions with another computational management object.

3.3.9 notification: An interaction for which the contract between the invoking object (client) and the receiving object (server) is restricted to the ability of the server to receive the contents of information sent by the client.

3.3.10 notification client interface: An operation interface of a computational management object that can only emit systems management notification messages.

3.3.11 notification server interface: An operation interface of a computational management object that can only accept systems management notification messages

NOTE – Client and server are used in the ODP sense.

3.3.12 management-operation server interface: An operation interface of a computational management object that can only accept systems management operation messages.

3.3.13 management-operation client interface: An operation interface of a computational management object that can only emit systems management operation messages.

4 Abbreviations

For the purposes of this Recommendation | International Standard, the following abbreviations apply:

ACID	Atomic Consistent Isolated Durable
ACSE	Association Control Service Element
AE	Application Entity

API	Application Programming Interface
ASN.1	Abstract Syntax Notation One
CMIP	Common Management Information Protocol
CMIS	Common Management Information Service
CMISE	Common Management Information Service Entity
CORBA	Common Object Request Broker Architecture
GDMO	Guidelines for the Definition of Managed Objects
GRM	General Relationship Model
IDL	Interface Definition Language
lr	linked reply
lrc	linked reply client
lrs	linked reply server
MOC	Managed Object Class
moc	management-operation client
mos	management-operation server
nc	notification client
ns	notification server
ODMA	Open Distributed Management Architecture
ODP	Open Distributed Processing
ODP-RM	Reference Model for Open Distributed Processing
OSI-SM	OSI Systems Management
QoS	Quality of Service
RC	Relationship Class
RPC	Remote Procedure Call
SMA	Systems Management Architecture
SMASE	Systems Management Application Service Element
SNC	Subnetwork Connection
TP	Transaction Processing

5 Requirements

This clause describes a set of requirements that are to be fulfilled in order to allow open distributed management. ODMA shall support:

- management of resources including those resources that are required for management;
- modularity with the identification of the parts that may be distributed;
- delegation of responsibility from one manager to another manager for issuing management operations;
- co-ordination of distributed management activities;
- management of systems of any scale;
- modelling of open distributed management systems assuming distribution transparency;
- tools for selected distribution transparency;

- management specific notational techniques for object-oriented modelling;
- hand-over of management responsibility from one system to another;
- mechanisms to determine management responsibility with respect to components of a managed system;
- several forms of distribution transparencies are defined by ODP-RM. Not all of them are used for management applications or the underlying systems that have to provide all these transparencies;
- access transparency (e.g., CMIP or RPC based...) so as to allow several implementations of the same application specification parts of a given application to interwork if implemented in different ways (different APIs and different communication protocols);
- ensure interworking with pre-existing OSI-SM applications and systems and specify how concepts and notation of OSI-SM can be used to specify ODMA systems and applications;
- interworking of management and non-management applications (e.g. network management and intelligent network applications);
- portability of management applications;
- the use of existing management information models into ODMA with a minimal adaptation;
- guidelines on the development of new information models based on ODMA.

Management applications based on ODMA must be capable to adapt to changes in their environment. These environment changes include but are not limited to:

- internal administrative organisations: different administrative organisation within enterprises have different static configurations of management functionality. Reconfiguration of management functionality may result from administrative decisions. ODMA should allow permanent management application specifications to be produced irrespective of the variety of possible management organisations;
- quality of service of management applications: time constraints, reliability, availability and others;
- growth of management network size: ODMA should permit the evolution from centralised management to distributed management as small networks grow beyond a size that can be effectively managed by a centralised management system;
- evolution of management services: ODMA must support the evolution of existing management services into distributed management services. The specification of new services must be possible without the need to refer to the location of existing services;
- technological change: the specification of a management system must persist over changes in implementation technology.

6 General framework

The Reference Model for Open Distributed Processing is a joint ISO/ITU Standard which provides a framework for the specification of large scale, heterogeneous distributed systems. It defines an architecture with a set of five viewpoints concentrating on different parts of the distribution problem and a set of functions and transparency mechanisms which support distribution. The resulting framework is being populated by more detailed standards dealing with specific aspects of the construction and operation of distributed systems. ODMA provides such a specialised architecture reference model for the distributed management of distributed resources, systems and applications. The following description of ODMA focuses on the specific features or requirements of management which are not already reflected in the ODP-RM. When unqualified, the term object refers to the abstraction of ODMA objects with respect to the ODP viewpoint under consideration in the current clause. Unless otherwise indicated, the concepts presented in this clause are as specified in ODP-RM.

6.1 Foundations

The general framework of ODMA is based on the following foundations:

- computational management object;
- engineering management object;
- managed role;

- managing role;
- management-operation server interface;
- management-operation client interface;
- notification client interface;
- notification server interface.

6.2 Architecture

In the following, each viewpoint within ODMA is described using those concepts of the ODP architecture and ODP foundations, which are necessary for management purpose. Additional ODMA concepts are described in the respective viewpoint, which are not defined in ODP but needed for management purpose and defined in 6.1, Foundations.

6.2.1 Enterprise viewpoint

The enterprise viewpoint is a view on the system and its environment that focuses on the purpose, scope and policies of the system.

The enterprise viewpoint description within ODMA is not different from similar descriptions in other applications of ODP-RM. However, the roles of special interest to ODMA are various cases of managing role and managed role, and that there is possibly a more frequent requirement for real-time data acquisition.

An enterprise specification should define contracts between objects in relation to the managing and managed roles.

An object performing in the managing role may require one or more objects performing the managed role to execute some management activity subject to a given contract.

ODMA does currently not prescribe the use of any particular notational technique for specifying the enterprise viewpoint (i.e. enterprise viewpoint notation). However, the description must unambiguously identify (i.e. name) the various constituent parts of the enterprise viewpoint description, for example, as illustrated in Annex F. The constituents are:

- Contract;
- Enterprise role;
- Community;
- Policy;
- Action;
- Activity.

6.2.2 Information viewpoint

The information viewpoint is a view on the system and its environment that focuses on the meaning of the information manipulated by and stored within the system. A detailed description can be found in the ODP-RM documents (Parts 1 and 3).

The information viewpoint description within ODMA is not different from similar descriptions in other applications of ODP-RM. Except that there is possibly a more frequent requirement that the information involved must correspond to actual values concerned with the equipment represented by the information objects.

The information specification has to ensure that the interpretation of the information handled by the objects in the system is consistent independently of the way the information processing functions themselves are distributed (defined in the computational viewpoint). This requires a specification of the invariant, static and dynamic schema.

The information objects together with their relationships are specified by a static schema. Assertions expressed specify the initial state of each object at a certain point in time. The relationship between the information objects should reflect the invariant schema expressing the invariants.

The dynamic schema is used to express how the information changes over time. It is used to specify the valid changes of states of the information objects. An information viewpoint specification, compliant with ODP-RM Part 3, may include definitions of dynamic schema, by specifying the valid state transitions of one or more information objects. In contrast, operations on interfaces that can trigger the state transitions are specified in the computational viewpoints.

An example specification of the static schema using OMT's[1)] (Object Modelling Technique, Rumbaugh) basic and enhanced object model is shown in Annex C.

6.2.3 Computational viewpoint

The computational viewpoint is a view on the system and its environment that enables distribution through functional decomposition of the system into objects which interact at interfaces. A detailed description can be found in the ODP-RM documents (Parts 1 and 3).

6.2.3.1 Computational management object template specification

A computational management object template specification comprises a set of computational interfaces which the object can instantiate, a behaviour specification and an environment contract specification.

Computational interface: A computational interface is characterised by a signature, a behaviour and an environment contract. An operation interface signature defines the set of operations supported at the interface, whether the operations are related to systems management operations or notifications, and the interface role (client or server).

Behaviour specification: The behaviour specification of an object is defined as the sequencing constraints, timing constraints, and concurrency constraints applicable to the object. It defines the overall behaviour of the object which might constrain the behaviour as specified for each interface that is supported by the object.

Environment contract: The environment contract specification of the object template applies to the object as a whole including the interfaces it supports. Examples of items specified in the object environment contract could be:

- that the object can only be located in a certain domain (security constraint, location constraint);
- that the object has a specified maximum probability of failure (reliability constraint).

This implies that for each computational object (including binding objects) a template should be specified that contains the elements described above.

6.2.3.2 Computational management interfaces

There are three kinds of computational management interfaces:

- management-operation;
- notification; and
- linked replies.

NOTE – Whenever the term 'operation' appears without qualifier it is used as defined in ODP-RM.

A computational management interface is an operation interface which can have one of the following roles:

- managing client role: which invokes operations on management-operation client interfaces;
- managed server role: which receives operations from management-operation server interfaces;
- managed client role: which invokes notifications on notification client interface;
- managing server role: which receives notifications from notification server interfaces;
- managed client role: which invokes operations on linked reply client interfaces;
- managing server role: which receives operations from linked reply server interfaces.

For management purposes, operations (e.g. get, replace, action operations as defined in GDMO) can be either announcements or interrogations. Operations are emitted by a management-operation client interface and received by a management-operation server interface. Notifications are emitted by a notification client interface and received by a notification server interface.

---

1) The Object Modelling Technique (OMT) is a method developed by J. Rumbaugh.

In the computational viewpoint, the interface is specified by an ODP operation interface signature, composed of:

- an indication of the management interface role;
- notifications or operations signatures (name of the notification or operation invocation, names, number and types of parameters, action templates for possible terminations).

NOTE – Within this Recommendation | International Standard, parameter is used as in ODP and is not to be confused with the GDMO term.

Examples of notational templates that can be used to express management interfaces are presented in Annex E.

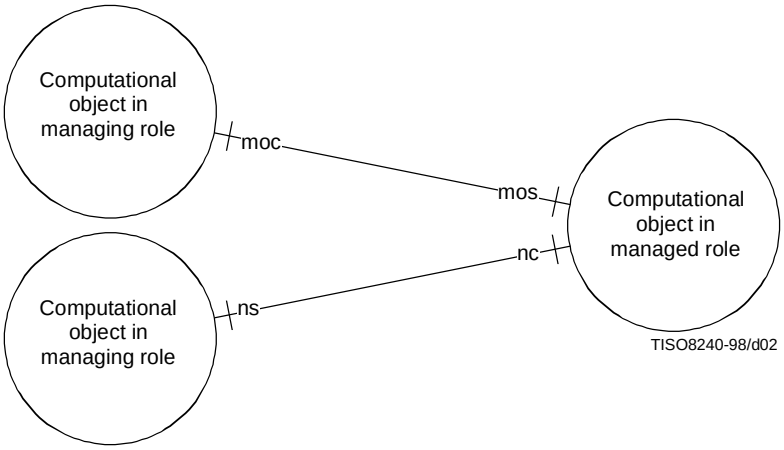
A computational management object can have multiple managed and multiple managing interfaces. For the OSI management support this enables the use of existing managed object class descriptions for describing managed interfaces for different purposes, e.g. one for configuration management (according to ITU-T Rec. M.3100) and one for fault management (according to ITU-T Rec. Q.821). Also having the possibility to define multiple management interfaces can lead to simpler relationship descriptions between the various interfaces of a management object. The managed system modeller therefore has the alternative of modelling management information for different purposes either by supplying multiple management interfaces to a management object or by providing different management objects for different purposes.

Table 4 describes the types of computational interfaces associated with roles within ODMA.

Table 4 – Type of computational interface associated with role

Interface role	Type of ODMA computational operation interface
managing client role	management-operation client (moc) interface
managing server role	notification server (ns) interface
managed client role	notification client (nc) interface
managed server role	management-operation server (mos) interface

Figure 2 depicts the types of interfaces together with the roles of the management objects.



NOTE – Due to the nature of notification operations, they may be invoked by a management object upon a notification distribution object, which can forward the contents to multiple destinations.

Figure 2 – Example of the relationship between roles and types for operations and notifications

Various computational interface definition notations can be used, some optimised for expressing interfaces to be carried by a particular engineering protocol objects. There may be benefit in a neutral computational interface definition notation, which refers to information objects and behaviour specified in the information viewpoint. Such neutral notations can be mapped via specification translations to the various interface definition notations.

From a management perspective there is a need to give identifiers (i.e. unambiguous names) to interfaces of objects in the managed role. Naming of the interfaces to objects in the managing role may be necessary when there is a need to bind to it. For example, naming of the interface of an object in the manager role is needed, if a notification dispatcher may push events to that object.

6.2.3.2.1 Linked replies operation

This subclause describes an example for the capability to support linked reply operations. Linked replies are used to return data as soon as they become available. Associated with the link reply operation is a final termination reply to indicate the operation has completed.

NOTE – The linked reply operation is a special operation of OSI management and not reflected in ODP-RM.

To provide this capability ODMA has to introduce two new interfaces. Table 5 describes the linked reply interfaces associated with roles within ODMA.

Table 5 – Type of computational interface associated with role

Interface role	Type of ODMA computational operation interface
managed client role	linked reply client (lrc) interface
managing server role	linked reply server (lrs) interface

Using the ODP-RM concepts these linked reply operations may be modelled as operations, which have to be linked. The linked reply (lr) interface is an interface in which all the interactions are operations and are related to the operation client interface.

The lr interface in the managing server role is related to the operation interface in the managing client role that initiates the action. Both belong to the same computational object in the managing role. The operation interface in the managed server role is related to the lr interface in the managed client role that responds to the action and belongs to the same computational object in the managed role.

The linked replies are specified in the operation invocation signature with an identifier for multiple operations.

NOTE – The operation interface may support an operation to stop the emission of replies.

For each operation in the lr interface signature an additional parameter is necessary to indicate the link to the requested operation.

Figure 3 shows how to deal with linked replies using the ODP concepts.

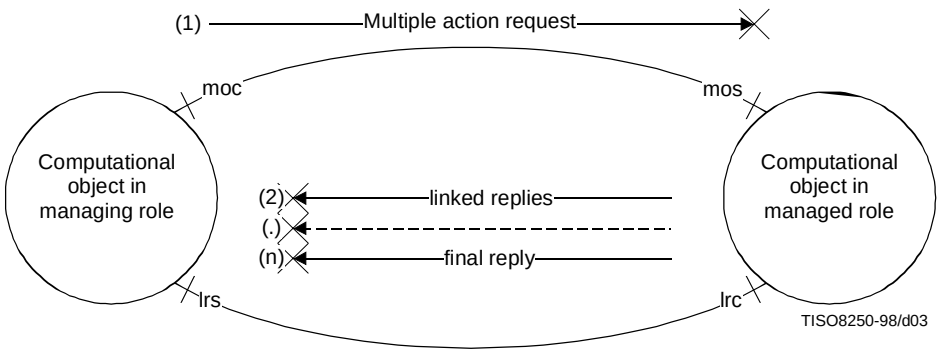
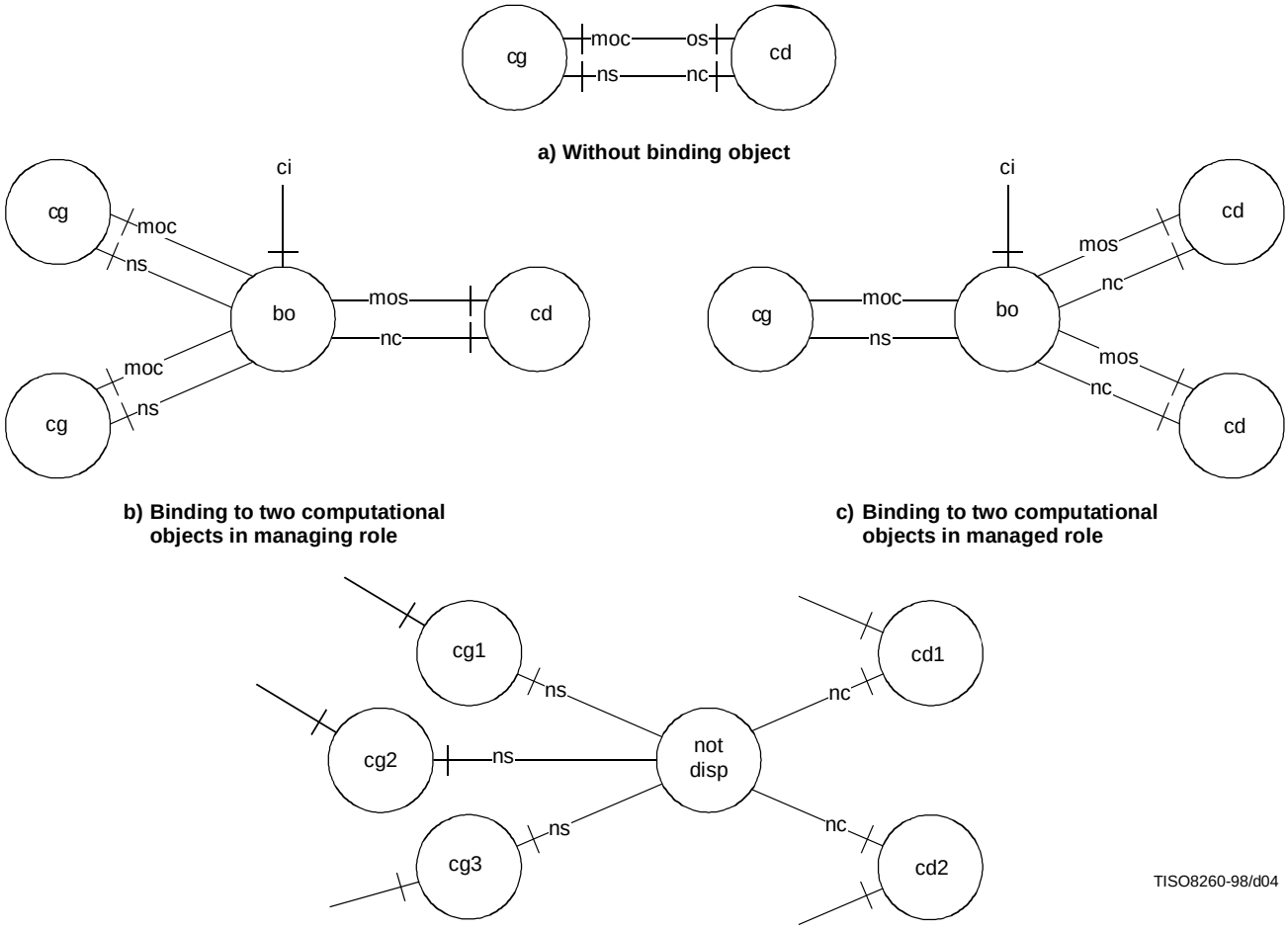


Figure 3 – Example of multiple linked replies from a single object

6.2.3.3 Computational bindings

Two objects can interact through a connection established between a managing interface and a managed interface. In the simple case the interfaces of objects in the managing and managed role are bound without binding objects [see Figure 4 a)]. However, in a number of cases one might want to introduce a binding object to control the binding between the objects. For example, when the communication between computational management interfaces is not simply point-to-point, a binding object is required to control the binding between the computational management objects. Examples for bindings with a binding object are given in Annex B, where the binding of interfaces is supported by ODMA functions for operation dispatching and notification dispatching.



mos	management-operation server
moc	management-operation client
ns	notification server
nc	notification client
ci	control interface
bo	binding object
cg	comp. object. in managing role
cd	comp. object. in managed role

Figure 4 – Computational management objects with and without binding objects

Examples of such cases are:

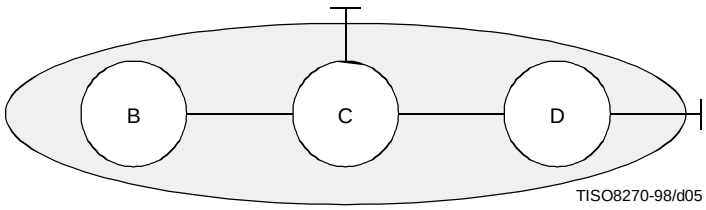
- to manage the access of one or more management objects in the managing role to a single management object in the managed role [Figure 4 b)];

- to manage the access of one or more management objects in the managed role to a single management object in the managing role [Figure 4 c)].

The notification dispatcher in Figure 4 d) is an example of a specific binding object. This binding object can manage the access of one or more management objects in the managing role to one or more management objects in the managed role.

6.2.3.4 Composition

The ODP composition concept is used to group ODMA computational management objects. A composite object exhibits multiple management interfaces. Interfaces between management objects within the composite object are not visible to the outside. This mechanism is illustrated in Figure 5. Here three management objects are grouped into a composite computational object. Management interfaces cannot be combined or decomposed in the computational viewpoint. Composition of objects with multiple interfaces can lead to an object with multiple interfaces, with the exception of the interfaces that exist between the objects in the composition (see Figure 5).



NOTE – A composite object and its components do not coexist in the same computational specification. Instead they belong to two different computational specifications at different levels of abstraction.

Figure 5 – Example of a computational object composition

6.2.4 Engineering viewpoint

The engineering viewpoint is a view on the system and its environment that focuses on the mechanisms and functions required to support distributed interactions between objects in the system. Furthermore, the functionality of objects supporting distribution transparencies is identified. A detailed description can be found in the ODP-RM documents (Parts 1 and 3). For the purpose of ODMA, this generic model has to be refined.

As an example of how to use the engineering viewpoint, Figure 6 presents a possible specification of the management bindings previously introduced by Figure 4. This example focuses only on the channels description (it does not consider the capsules and clusters configuration) and is independent of any specific implementation solution. Clause 7, describing the OSI-SM support of ODMA, shows how OSI-SM can be used to specify such management bindings.

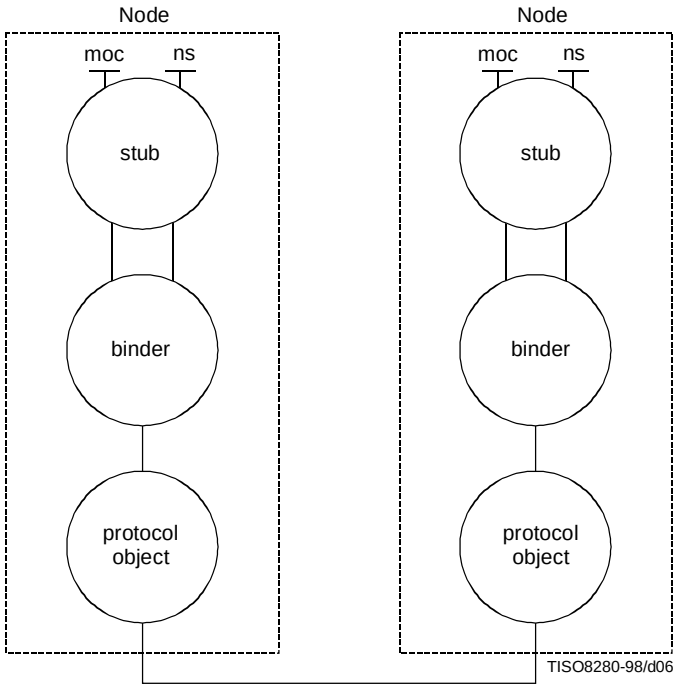


Figure 6 – Example of an engineering model for the bindings presented by Figure 4 a)

Figure 7 presents an example of a more complex system possibly using different technologies. The shaded graphical elements represent a stack consisting of a stub, binder and protocol object as illustrated in more detail in Figure 6.

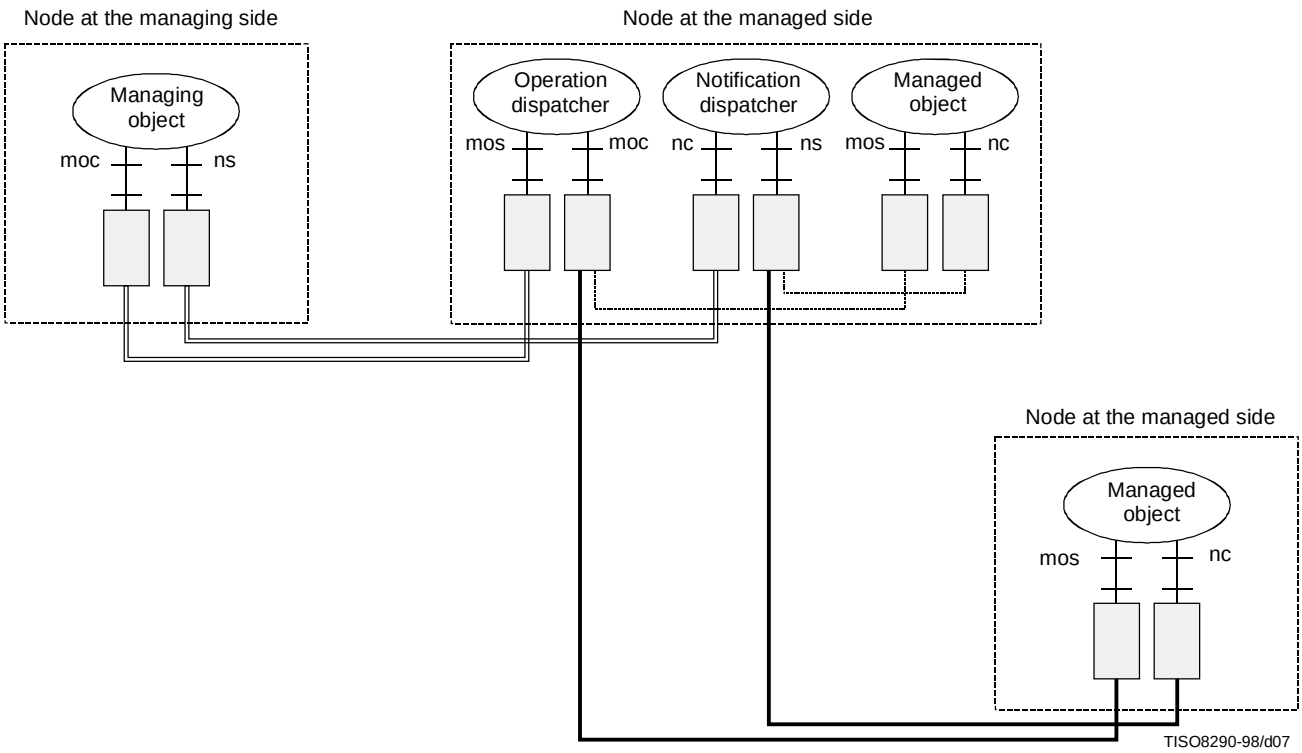


Figure 7 – More complex example of an engineering model for multiple bindings

6.2.4.1 Support of distribution transparencies

The following distribution transparencies may have to be supported but are not required in all cases. However, location and access transparency have to be supported in all cases.

6.2.4.1.1 Location transparency

Location transparency masks the location in space of interfaces: location transparency for interfaces requires that interface identifiers do not reveal information about interface location. This also enables managed objects to be location transparent.

In some cases location awareness (e.g. a particular presentation address for managed object access) may be needed. This requires that information about this location has to be available to other objects.

6.2.4.1.2 Relocation transparency

Relocation transparency masks relocation of an interface from other interfaces bound to it. It applies to objects in clusters and is achieved by co-operation between cluster managers and relocaters.

6.2.4.1.3 Migration transparency

Migration transparency masks from an object the ability of the migration function to change the location of that object. Migration of objects, for example, is used in:

- mobile networks to optimise access speed; and
- dynamic load balancing.

The underlying architecture is responsible for migration resolution.

6.2.4.1.4 Access transparency

Access transparency masks from an object the differences in data representation and invocation mechanisms. This enables integration of heterogeneous environments and the use of different technologies. In the ODP-RM, access transparency is provided by stubs.

This implies that ODMA can provide mechanisms for interworking between OSI management and other paradigms like IDL, SQL, etc. However, full access transparency will not be possible. Instead, an identification has to be made of a small set of access transparencies that can currently be achieved, e.g. GDMO-IDL interworking.

6.2.4.1.5 Failure transparency

Failure transparency masks, from an object, the provisioning of fault tolerance for that object.

6.2.4.1.6 Persistence transparency

Persistence transparency masks from an object the use of the deactivation and reactivation function to vary the processing, storage and communications resources provided to an object.

6.2.4.1.7 Replication transparency

Replication transparency masks the use of a group of behaviourally compatible objects to support an interface.

6.2.4.1.8 Transaction transparency

Transaction transparency masks co-ordination of activities among a configuration of objects to achieve data consistency.

6.2.5 Technology viewpoint

The technology viewpoint expresses how the specifications for an ODP system are implemented. It deals with hardware, software, installation, etc. The technology viewpoint is also the viewpoint concerned with conformance verification of implemented systems against standard specifications.

The technological choices of hardware, software, etc., are outside the scope of the ODMA.

The ODMA provides guidelines for conformance statements for the specification of standards within ODMA and specification of systems according to ODMA.

NOTE – Compliance to ODMA is for further study.

6.3 Reuse of ODMA specifications

The ODP viewpoints, as will be applied to describing specifications and standards, are related to a specific problem space. These problem spaces may overlap, as is shown in Figure 8. This may lead to a situation where objects in the computational viewpoint of a problem space have been defined before as part of another (overlapping) problem space. As Figure 8 shows, there may be several information viewpoints (e.g. several Standards | Recommendations | Specifications that conform to ODMA), each related to a specific problem space.

The figure shows two problem spaces as might be described by two ODMA specifications. Suppose specification 1 was written before the development of specification 2. In that case, the part of the computational viewpoint 2 that overlaps with computational viewpoint 1 has already been defined in specification 1. Specification 2 may therefore refer to specification 1 and does not have to repeat. Indirectly, also a part of information viewpoint 1 is reused.

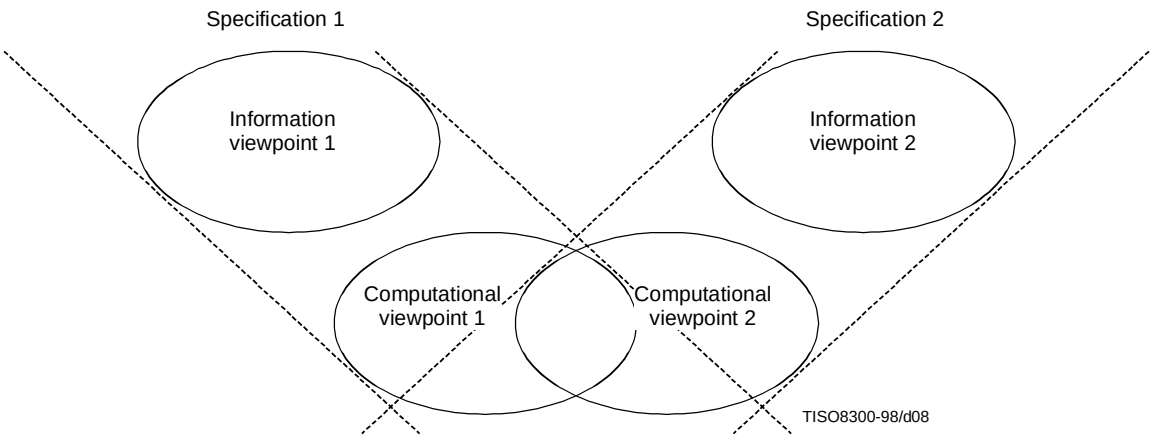


Figure 8 – Overlapping problem spaces

In each viewpoint, parts of other viewpoints defined in other specifications may be reused. In addition, a viewpoint may also refer to other viewpoints in other documents, e.g. a computational viewpoint in specification 1 that refers to information viewpoint in specification 2.

7 OSI management support for ODMA

This clause describes how the current available OSI Systems Management concepts can be used to support ODMA, i.e. the distribution of management applications and the distribution of resources to be managed.

The OSI Management support for ODMA is supplementary to OSI Systems Management Architecture (SMA). It describes how SMA can be reused in a distributed environment. Using the ODMA framework, it is shown how aspects of OSI Systems Management can be embedded. In the limiting case of interaction between a single system carrying out management activity and another single system where the resources being managed are located, OSI management as defined in ITU-T Rec. X.701 | ISO/IEC 10040 may apply. When applications or resources to be managed or managing activity is distributed, one should use the ODMA architecture.

The enterprise and information viewpoint descriptions for the OSI management support of ODMA are the same as the corresponding enterprise and information viewpoints in clause 6, General Framework.

7.1 Computational viewpoint

This subclause shows how OSI management concepts can support the computational viewpoint described in clause 6.

The explicit identification of the managing role and managed role in this computational viewpoint are extensions to the current OSI Systems Management Architecture (SMA). Where SMA defines a manager and an agent as being specific roles of an MIS-user, ODMA defines a managing role and a managed role as being specific roles of a computational management object. An object can have multiple interfaces of which some are (OSI) management interfaces.

MIS-users are still a valid concept within ODMA; however, these are engineering concepts and will not be visible at the computational viewpoint. For example, the agent (i.e. the MIS-user in the managed role) will not be visible in the computational viewpoint in any form.

Another important basic element of the SMA is a managed object. A managed object is visible at the computational viewpoint. Within ODMA a managed object describes a management-operation server interface and a notification client interface (or only one of them). The interpretation is still that the managed object is the manager's view on the resource. However, the representation of the managed object is a managed interface to an object. The managed interface is described using the GDMO template.

The management-operation server interface and the notification client interface of one managed object belong to the same unit of distribution.

A computational management object may also possess a managing interface. The current SMA does not provide tools to describe the managing interface. However, since the managing interface will be a mirror of the managed interface, additional tools are not necessary. GRM can be used to describe relationships between interfaces of a server and an associated client of a computational management object. An example can be found in Annex D.

Figure 9 shows an example of four computational management objects participating in a management system. The Circuit object is currently involved in three relationships. In its relationship with the object in the managing role the Circuit object takes on the managed role. In its relationship with the two Leg objects the circuit object has the managing role. The computational behaviour of the Circuit management object will determine how operations received on its managed interface will affect the operations sent out on its managing interfaces.

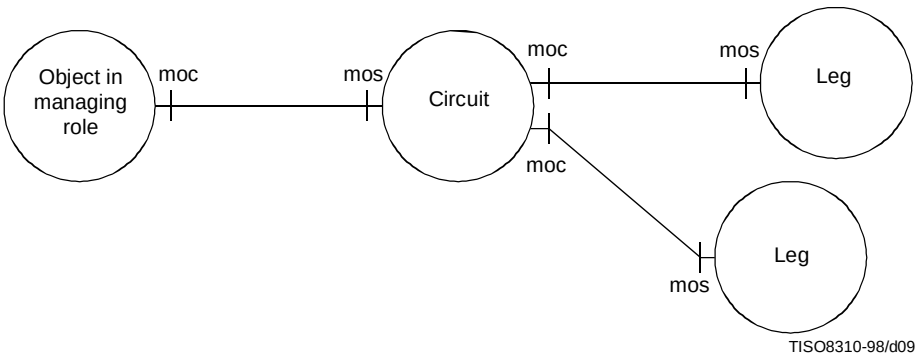


Figure 9 – Example of a computational viewpoint for ODMA

Here, GRM Relationship Class (RC) definitions can be used to describe the interfaces and behaviour of the computational management objects Circuit and Leg. The interface of a computational management object corresponds to a role of the relationship class. So, the Circuit relationship class has two roles, e.g. Circuit Operation Server and Legs Operation Client. Furthermore, the Leg managed relationship class has one role, e.g. Leg Operation Server. The behaviour of the relationship class describes the interactions between the interfaces (roles) of the computational management object (relationship class). This is shown in Figure 10.

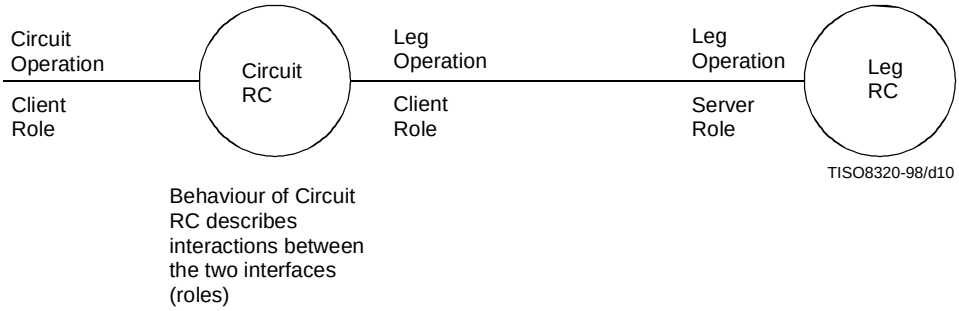


Figure 10 – Relationship Class description of computational management objects

Thus the interfaces of the computational management objects correspond to GRM roles. As roles need to be defined in terms of compatible managed object classes, the interfaces of the computational management object need to be described in terms of managed object classes. So a Managed Object Class (MOC) definition describes the characteristics of a management-operation server interface and a notification client interface. In addition, the same managed object class can also be used to describe a management-operation client interface and a notification server interface on an instance of another object class. This is a method in which managed object classes are used to describe the managing side of an interface by mirroring the management-operation server interface and notification client interface. The last principle is essential for the method of using GRM in combination with GDMO.

Still, the behaviour of the managed object class will only describe the behaviour at the managed side. So, suppose that we have a managed object class LegInterface that is used to characterise both the Leg Operation Server and Leg Operation Client interface. Then the behaviour of LegInterface will only apply for the Leg Operation Server role. The associated behaviour for the Leg Operation Client role will be described in the behaviour of the Circuit relationship class. This is shown in Figure 11.

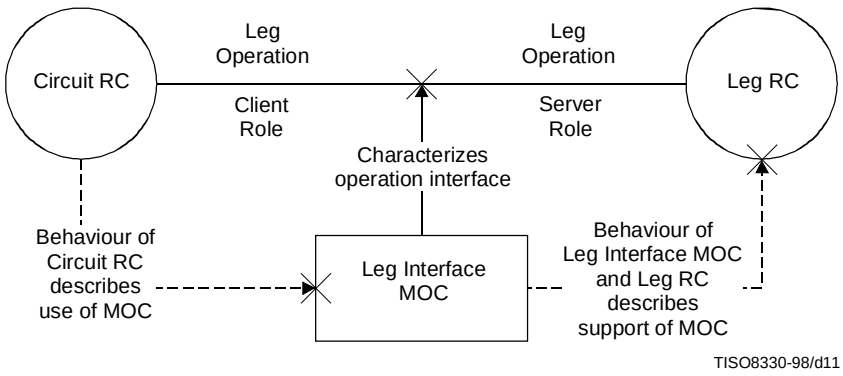


Figure 11 – Managed Object Class characterisation of operation interface

Note that the behaviour of the Leg Interface MOC, as well as the behaviour of the Leg RC can describe the performing of get operations on the attributes of the Leg Interface managed object. However, the behaviour of the Circuit RC can only specify the invocation of a get operation on an attribute. The reason is that from the viewpoint of the Circuit computational management object, the operation is actually performed at the remote side of the interface to the Leg computational management object.

The use of GDMO alone does not make a complete specification of the management computational management interfaces. Since there is no single (standardised) CMIS operation interface signature, several computational designs of operation interfaces are possible.

There are two ways of handling this problem: either extend GDMO with CMIS parameters (e.g. for scoping and filtering) or complete the operation interface signature of a computational management interface, based on a GDMO specification, with (parts of) a standardised CMIS operation signature.

NOTE 1 – Both options are for further study. A solution would be especially useful for the interworking with existing OSI management systems.

Thus GDMO, GRM and CMISE together can be used to constitute a complete computational design. Such a computational design would be optimised for working within a CMIP based communication domain.

NOTE 2 – For the purpose of developing standardised computational models based on CMISE, it is sufficient to use GDMO and GRM for specifying computational interfaces. This does not imply the use of CMIP as an engineering viewpoint protocol.

NOTE 3 – The GDMO, GRM and CMISE specification may be derived from a more abstract computational notation as mentioned in 6.2.3.

7.1.1 Linked replies

In OSI Management linked replies use CMIP.

7.1.2 Levels of computational abstraction

There are different types of operation interfaces possible depending on the underlying mechanisms. Besides the interface signature related to the problem domain (e.g. a managed object class for OSI management), also CMIS-parameters may be provided. Examples of these parameters are:

- scoping and filtering parameters;
- event forwarding parameters.

These parameters can be made visible in the computational viewpoint if:

- there is a need to support these parameters in a generic way. Figure 12 for example, has a specific object, NW, that distributes the *set time* operation to multiple destinations, managed objects E1, E2 and E3. Computational visibility of scoping and filtering is not required to make this happen;
- the system can provide the required mechanisms to support these parameters e.g. support generic scoping and filtering. Figure 13 shows scoping and filtering exposed computationally to support generic browsing.

A computational design to satisfy a specific enterprise concern of setting network time is shown in the top of Figure 12. In this computational design the managing object on the left of Figure 12 also provides a managed interface to other managing objects (not shown). Furthermore, scoping and filtering parameters in this design do not have to be exposed in the computational viewpoint, because the objects (E1, E2, E3) are hidden by the object (NW).

The computational viewpoint diagram at the top of Figure 12 shows Network Object, NW, distributing the set time operation to multiple objects, E1, E2 and E3

As such, the NW (network) object is an abstracted notion from the ‘scoping and filtering parameters’. Note that scoping and filtering are two different functions modelled using two engineering objects: a scope (for managing object) and a filter (for managed object).

NOTE – Composition can be used to combine the filtering and managed objects.

A computational design to satisfy another enterprise concern for a generic browser capability, is shown in Figure 13. In this case, the scoping and filtering parameters are exposed computationally, and each object in Figure 13 is a candidate for distribution.

The computational viewpoint diagram at the top of Figure 13 shows a generic browser object making use of scoping and filtering objects.

Because there is no standardised CMIS-API, multiple signatures for scoping and filtering may be defined.

The computational design shown in Figure 12 is an example of what one ODMA component standard solving one enterprise concern, might specify, while that shown in Figure 13 is an example of what another ODMA component standard, solving yet another enterprise concern, might specify. Each computational design expresses different candidates for distribution, thus cannot be expected to inter-operate.

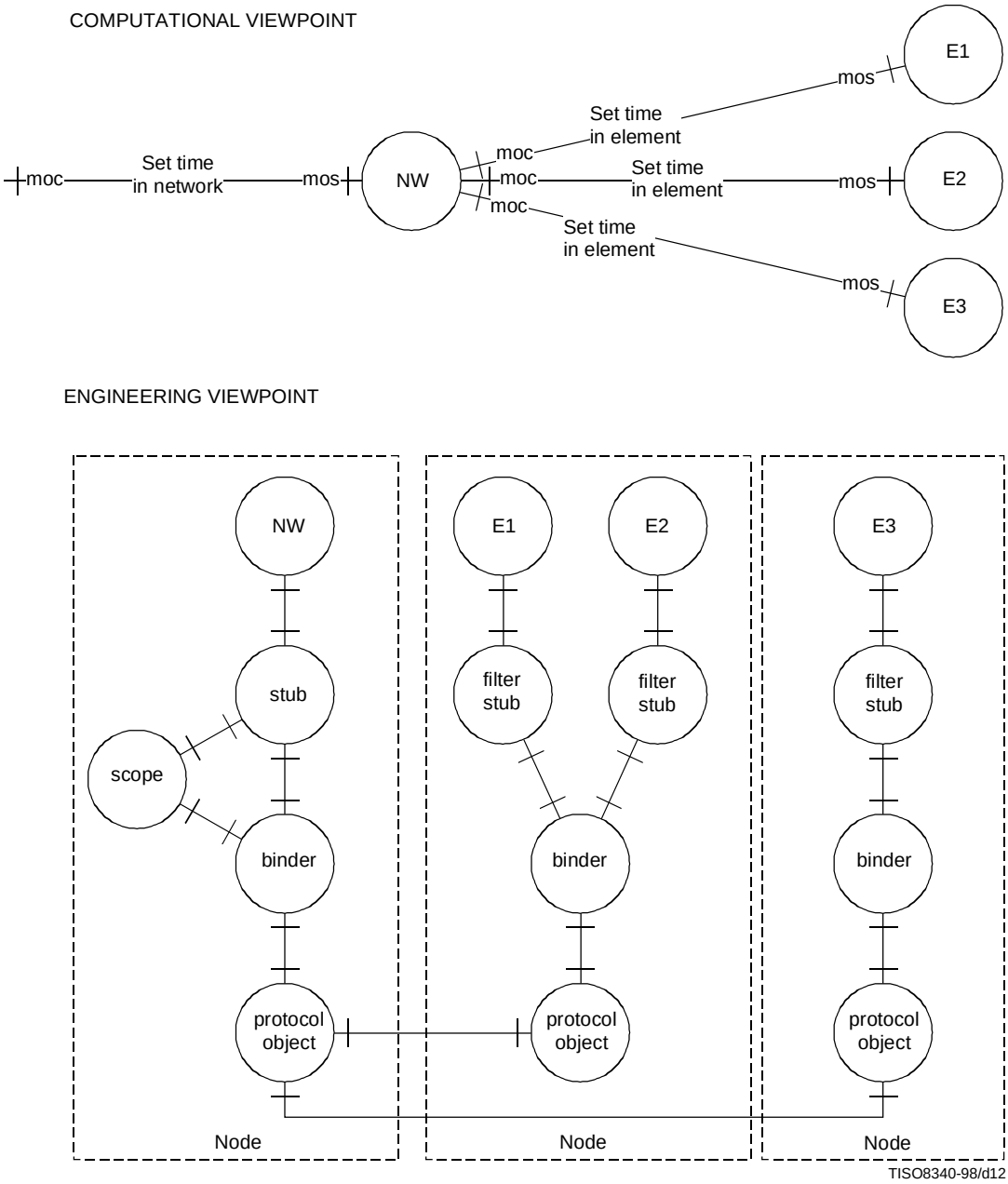


Figure 12 – Scoping and filtering as an engineering optimisation (not exposed computationally)

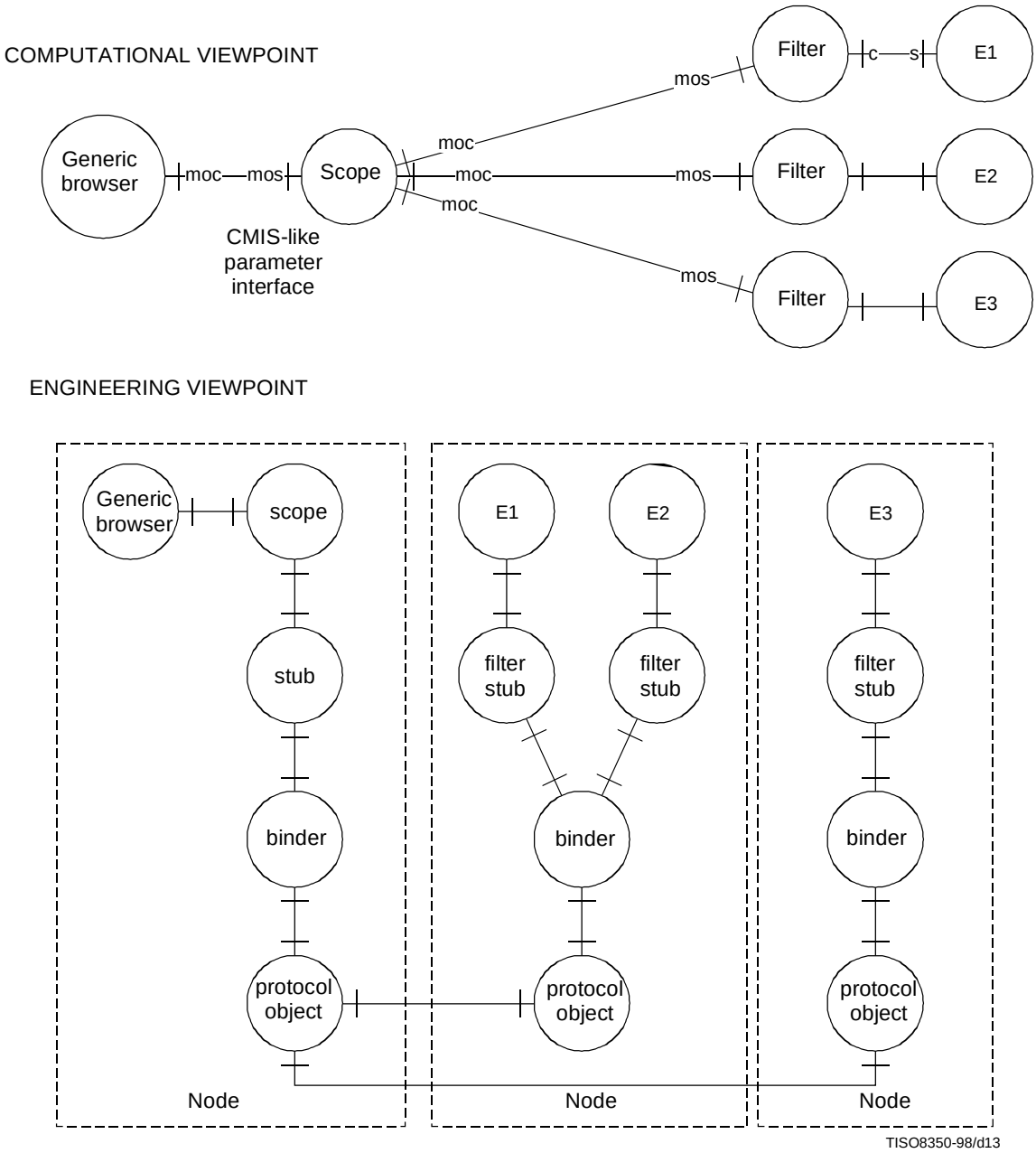


Figure 13 – Scoping and filtering exposed computationally

Besides scoping and filtering, other CMIS-parameters may be exposed to the computational viewpoint, like for instance event forwarding parameters. Suppose we have a computational management object that can emit notifications by one of its (clients) interfaces. In this case an Event Forwarding Discriminator may be used to describe a managed interface to this computational management object to control the flow of event reports, as shown in Figure 14:

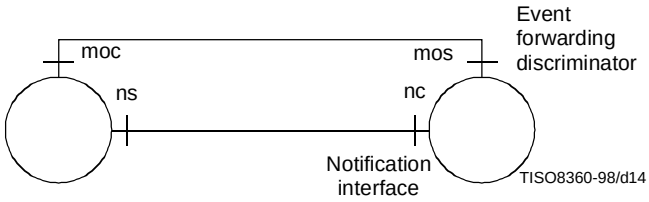


Figure 14 – Event forwarding exposed computationally

These kind of figures make assumptions on underlying implementations. In this ODMA Recommendation | International Standard, we only use the simple signature as described by a managed object class. In addition, ODMA makes the same abstraction at the managing side. We abstract from the detailed computational management object design related to several supporting mechanisms by means of composition.

Additional standards will have to be developed within the scope of ODMA to standardise interface signatures. For example, for the event forwarding function signatures can be defined for scoping, for filtering etc.

7.2 Engineering viewpoint

This subclause discusses how the various forms of distribution transparency can be supported using OSI management mechanisms.

It will first be shown how the current Systems Management Architecture provides for engineering concerns. Figure 15 gives an example of how a MIS-user in both the manager and agent role can be engineered. In this way, the two interfaces of the engineering object at the agent side corresponds to a managed object class. In Figure 15 it is shown that both client and server management interfaces are bound into a single management association to the protocol object.

The part of OSI Systems Management needing consideration with respect to Open Distributed Management is the role of the MIS-user. The agent is always identified in an open system. It is identified by an AE-title. In Figure 15 it is illustrated that both interfaces are bound into a single management association. For Open Distributed Management the agent has to be split up into (manageable) parts. There is no single agent object in the case of Open Distributed Management. The stub, binder and protocol objects can be considered as parts of the agent functionality as can be seen in Figure 15.

Engineering objects may need to be defined for various reasons, e.g.:

- controlling management session (initiation and termination);
- creation and deletion of managed objects;
- handling of operations requests including access control, synchronisation, scoping and filtering;
- co-ordinating amongst managed objects;
- notification dissemination.

For OSI Systems Management, the engineering viewpoint describes the functionality needed to provide communication between objects in the managing role and in the managed role using CMISE in conjunction with possibly other protocols such as Transaction Processing. It also describes the application entities that exist within the distributed management system.

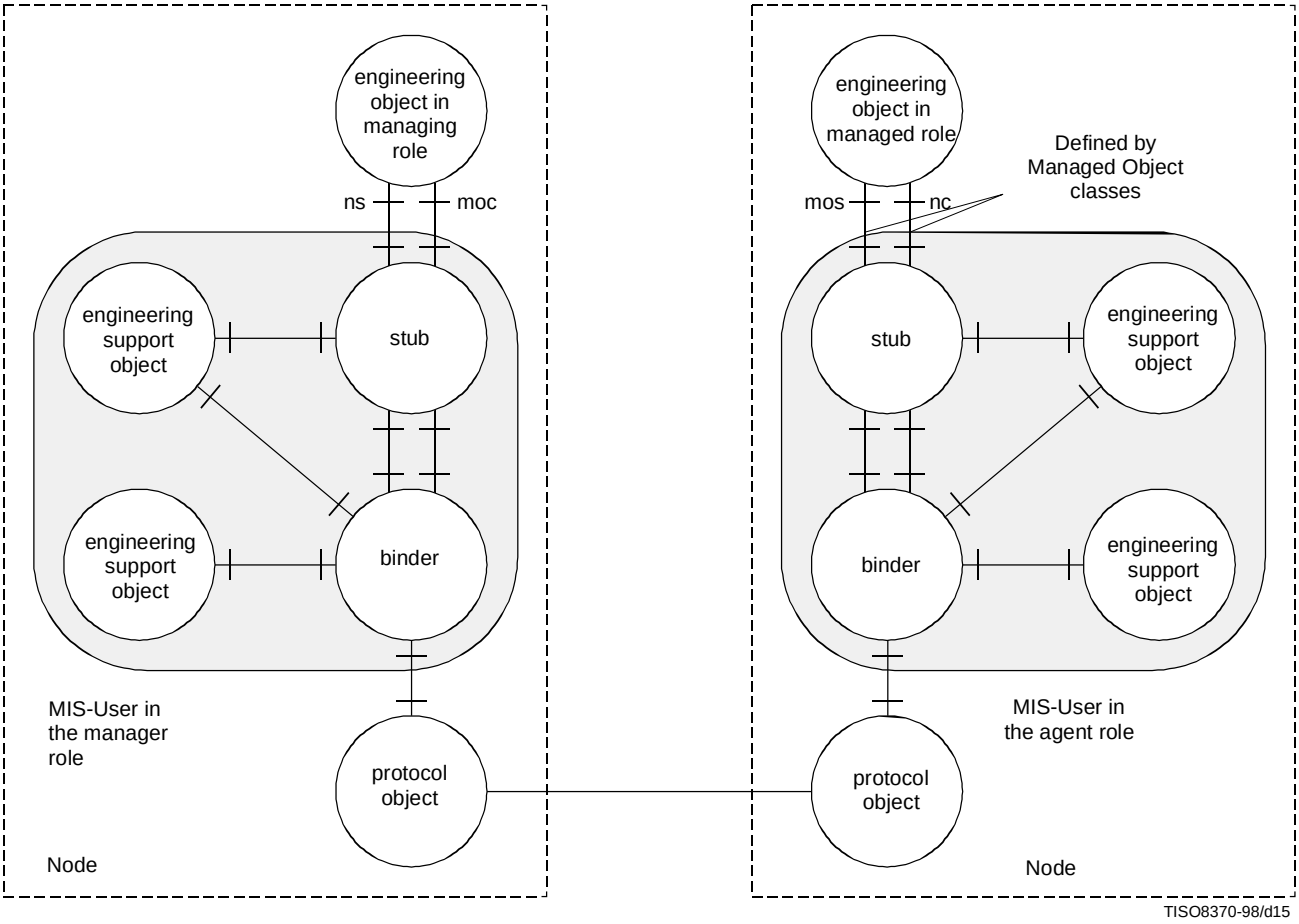


Figure 15 – OSI Systems Management (1991) model for the engineering viewpoint

The protocol object represents a protocol stack, e.g. a 7-layer OSI protocol stack that provides an application layer service.

Binders address many of the problems associated with distribution. They are responsible for maintaining the end-to-end integrity of the channel and deal with object failures. Consequently, binders have to handle configuration and communication changes. A binder has to establish the binding when the channel is created and has to keep track of endpoints if objects move, are replaced or fail. The binder object is involved in the process of object relocation and in the provision of many of the distribution transparencies described previously.

A managed object, which has both a client notification interface and a server management-operation interface, must, in some cases, be able to have all of its interactions mapped onto a single CMIP association. Thus a binder must have the capability to map multiple engineering interfaces onto a single Protocol Object association.

The stub object provides adaptation functions to support interaction between interfaces in different nodes. This adaptation may mean that an operation is translated to a coding that is understandable for the engineering objects (i.e. marshalling/unmarshalling).

The following example of composed engineering objects shown in Figure 16 is introduced for use in other figures of this subclause. The engineering management object is a shorthand for the basic engineering object and the stub and can be a possible way of engineering in a case where access transparency is not required in the engineering representation of a computational management object. The management association object combines the functionality of the protocol and binder object into a single engineering object that corresponds to an existing management association. This can be a possible way of engineering the case where the use of multiple protocols is not required in the engineering representation of a computational management object. Finally, the engineering support objects represent additional systems management functions like, for instance, the notification server.

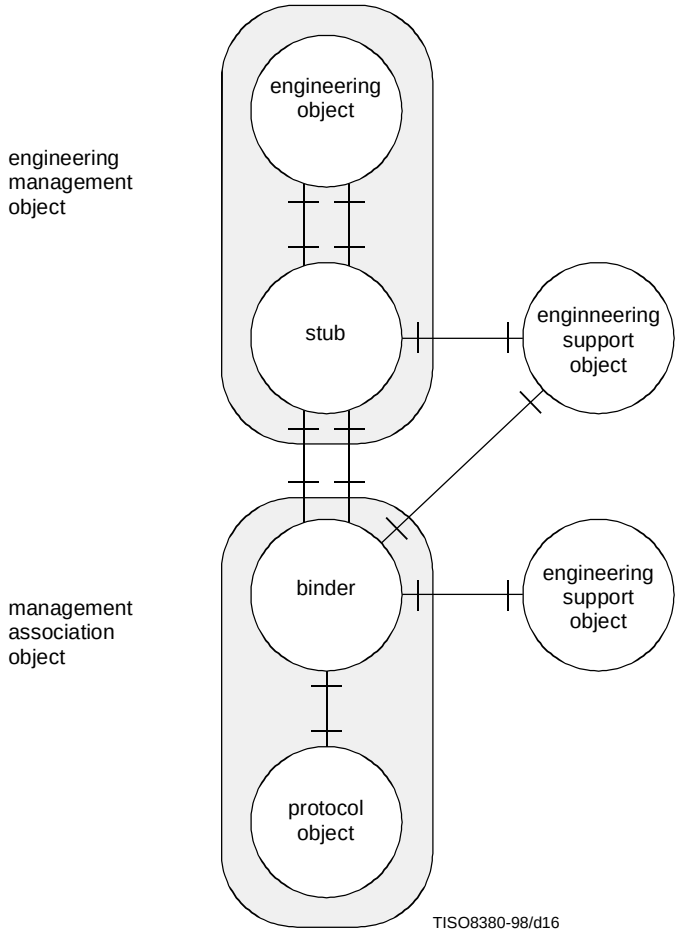


Figure 16 – Example of ODMA basic engineering objects

The services offered by the management association objects (by its interface) correspond to the CMIS primitives. The same holds for engineering support objects: they use CMIS primitives from the management association object and they provide (a subset of) CMIS primitives to other engineering objects.

The following subclauses describe in more detail how management functions and distribution transparencies can be supported by specific engineering support objects.

7.2.1 Creation and deletion of managed objects

For creation of a single manarged object, the following steps can be taken (see also Figure 16):

1) A managing object obtains information (e.g., AE-title, presentation address) on the desired interface (managed object) possibly by requesting this information from the Directory.

2) The trader provides the managing object with the requested information (interface reference).

   NOTE – Trader is defined in ODP-RM.

3) The managing object issues a creation request to the appropriate AE.

4) Node management creates the interface (managed object).

The Directory can reside in another node. In that case, a separate communication channel is needed.

ODP node management is responsible for interface creation and deletion. It exists in every node. See ITU-T Rec. X.903 | ISO/IEC 10746-3, (Subclause 12.1) for details.

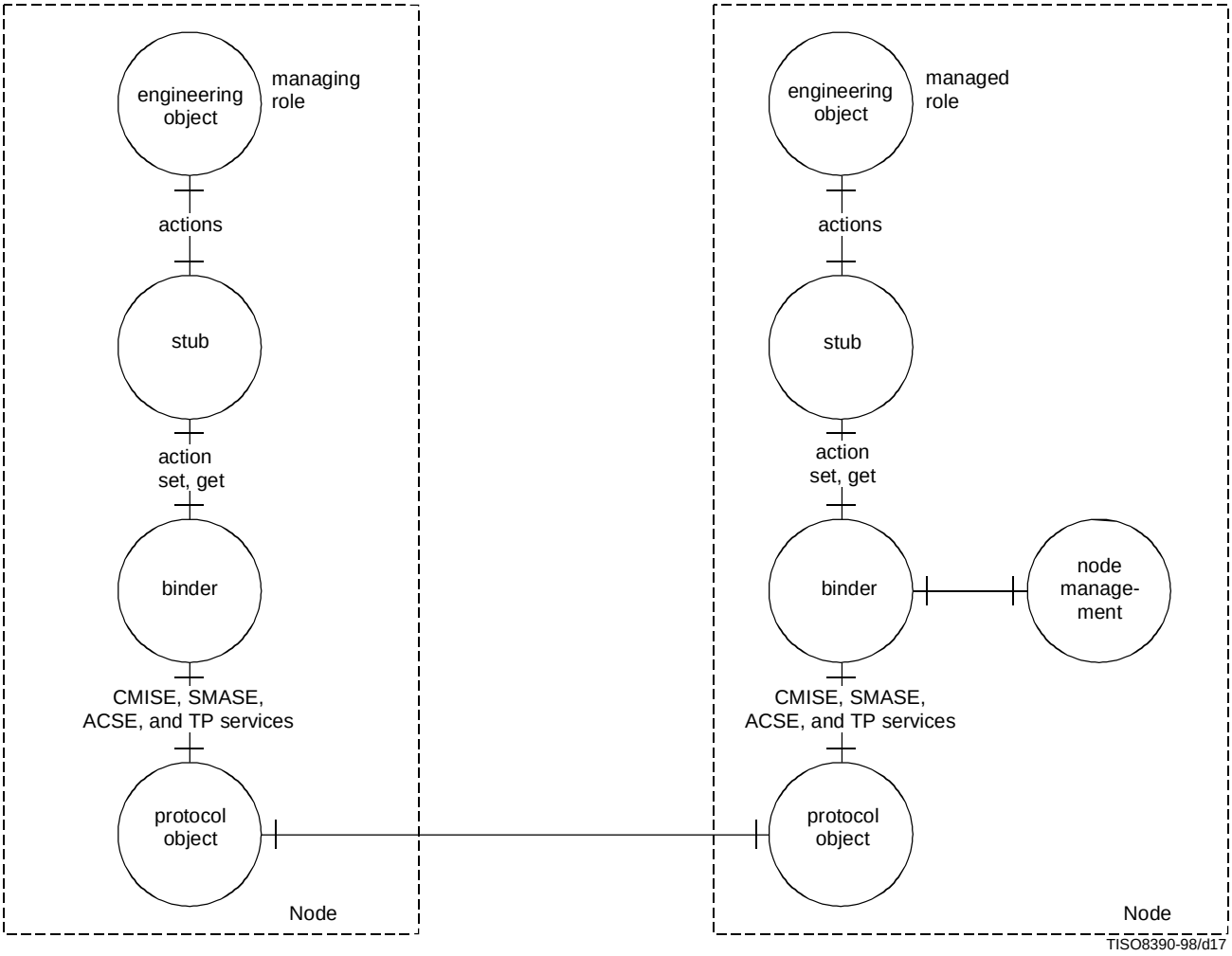


Figure 17 – An example of an engineering viewpoint to support creation, deletion

7.2.2 Handling of operations requests

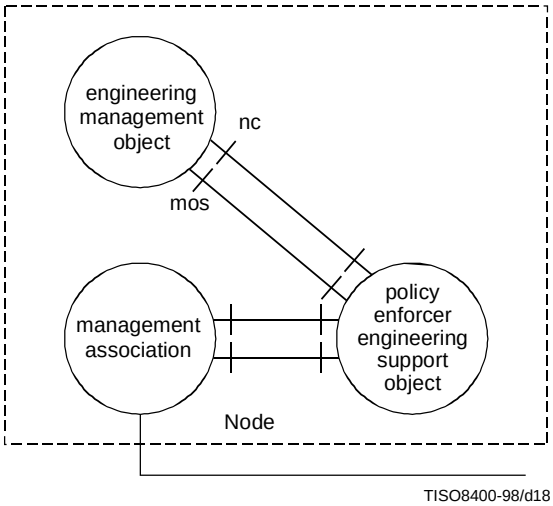
A managed object offers operations on attributes (GET, REPLACE), actions and notifications. Part of the agent functionality is the translation of CMIS primitives into the local representation of these operations/notifications on managed objects, for instance, mapping of M-GET on a GET and mapping of a notification on an M-EVENT-REPORT. This translation should typically be done within the engineering viewpoint.

The objects in the managing and managed roles communicate by means of the protocols object. Their interactions are not directly with the protocol object, but are adapted by the stub object by marshalling/unmarshalling of operations. The stub object can translate operation invocations from their local representation into CMIS primitives.

The managing object invokes computational management-operations. These invocations are translated by the stub from their local representation into CMIS primitives, and, at the other end of the channel the CMIS primitives are translated into local representations of operations. For instance, the object in the managing role invokes a GET operation that is translated from its local representation into a CMIS M-GET by its stub. The stub at the other end translates the M-GET into a GET which results in a GET at the object in the managed role.

7.2.3 Handling of policy enforcing

Policy enforcing can be solved by a Policy Enforcer engineering support object on the managed side, intercepting all management-operation and notification interactions between all policed engineering management object and the environment, as illustrated in Figure 18.



NOTE – Access control is a special case of policy enforcing.

Figure 18 – Policy enforcing engineering support object at the managed side

7.2.4 Handling of scoping across multiple systems

This is a possible solution for handling scoping across multiple systems.

OSI Managed Objects are named according to a naming schema using name binding templates as specified in GDMO. In case of ODMA, a global naming tree will be the most practical solution for managed objects. In that tree, the selection of objects using scoping needs to be applicable. Scoping identifies the objects to which filtering is applied, e.g. by identifying a root of a subtree. Filtering is used to select a subset of the objects, identified by scoping.

In combination with the Directory, ODMA can support scoping. In this case, the Directory should store information on portions of the management information tree that are stable enough for storage in the Directory. The Management Knowledge Management Function provides repertoire directory objects to support this storage. The repository needs to be able to do the scoping and to respond with the list of all selected managed objects. The list of (references to) managed objects is sent to the Operation Dispatcher engineering support object, see Figure 19. The Operation Dispatcher engineering support object represents functionality to distribute the operation (invokes and replies) to all applicable AE-titles and managed objects behind that AE-title.

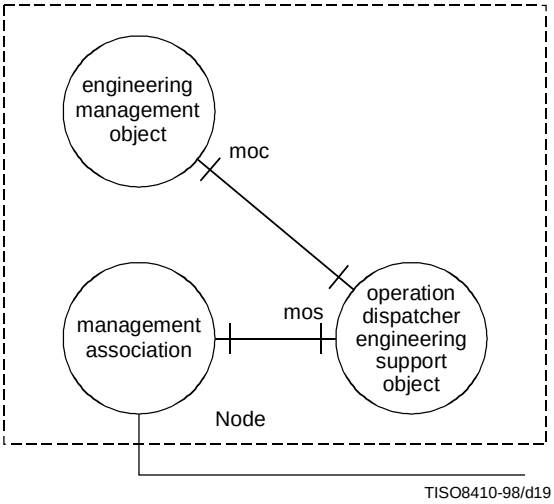


Figure 19 – 'Operation Dispatcher' engineering support object at the managing side

The naming schema is composed of relations between management object classes. These relations expressed by the naming templates are relevant to the information viewpoint.

NOTE 1 – As a repository for management knowledge the Directory could be used. For instance an object that makes visible the management knowledge by means of its interfaces (managed objects).

NOTE 2 – The Operation Dispatcher support object is a potential ODMA function. An example of such an object is described in Annex B.

7.2.5 Handling of filtering

NOTE – Filtering is for further study.

Filtering will have to be supported at the managed interface. An example can be found in Figure 12.

7.2.6 Handling of notification dissemination

Currently, two approaches are identified for notification dissemination:

- as a separate service (ODP event notification function) through which objects can subscribe to either send notifications to (in the resource role) or receive event reports from (in the managing role); or
- as part of an engineering object related to a computational management object, i.e. the event forwarding function is part of the computational object.

In the first case, there is one service that has a management interface with managed objects with the functionality of the event forwarding discriminator.

Such a service is provided by a notification dispatcher support object. This object is used to forward and filter events generated by objects that have been registered within the notification dispatcher support object. An object that generates events is called an emitter and an object that receives event notification is called a recipient. Note that notifications from the emitter should be considered as operations on the notification dispatcher support object. When an emitter is created, it can inform the notification dispatcher support object that it wishes to emit events of a certain type. The notification dispatcher support object will then provide an interface that is capable of receiving such events.

When a recipient wishes to be notified about certain events, it can instruct the notification dispatcher support object to forward these events. This allows many recipients to receive an event from one emitter. The notification dispatcher support object can also be instructed to perform filter functions, including security checks for authorised recipients, on the events it receives to determine which events are forwarded.

An engineering view of the notification dispatcher support object in relation to objects in the managed role is presented in Figure 20.

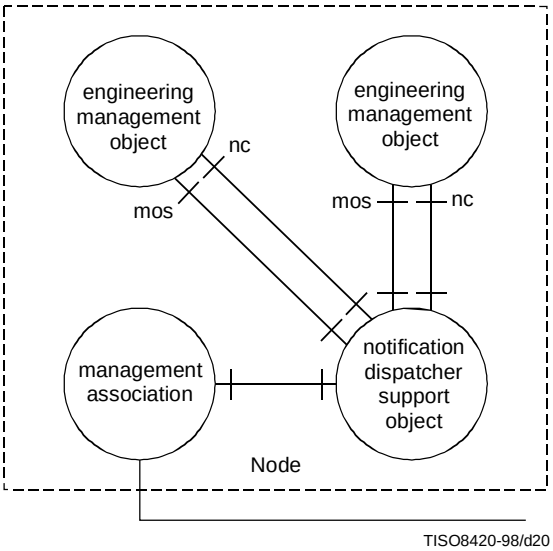


Figure 20 – Engineering view of the notification dispatcher support object at the managed side

The objects in the managed role may issue notifications that are sent to the notification dispatcher support object. Via its management association, the notification dispatcher support object will forward the notifications to those sites where objects in the managing role are located that are subscribed to these notifications. Therefore, the notification dispatcher support object has an administration to which management association objects it has to send which notifications. For more details on notification dispatcher support object in relation to the managing role, see 7.2.7.

NOTE – In different (transport) protocol environments, this functionality may not be needed.

Interfaces to the notification dispatcher support object are described by the OSI Event Report Management Function (see ITU-T Rec. X.734 | ISO/IEC 10164-5).

In the second case, each object will have its own event forwarding discriminator in the form of a managed object. Note that a hybrid approach is also possible.

7.2.7 Managing role

OSI management does not make many assumptions about the characteristics of the manager. So, in the context of Open Distributed Management, it is easy to consider the manager also as an object. An object in the managing role may delegate responsibility to another object in the managing role for issuing management-operations and/or handling received notifications.

The object in the managing role is identified by a local identifier within the scope of its AE-title. These local identifiers are administrated by the dispatcher and are used for addressing notifications and management-operation replies to the right managing interface. The engineering view is presented in Figure 21.

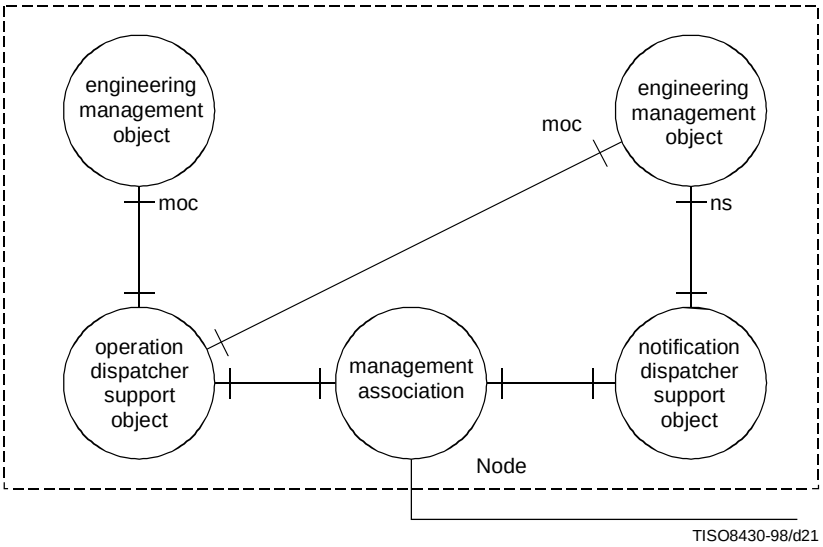


Figure 21 – Engineering view of the dispastcher at the managing side

The Operation Dispatcher support object administrates the operations that are issued from one of the management-operation client interfaces within its scope. The Management Association object is addressed by the usual address information (AE-title, presentation address).When a reply to a management-operation is received, the dispatcher takes care that the reply is sent to the right management-operation client interface. For identifying the management-operation client interface, the Operation Dispatcher may use a local identifier.

A Notification Dispatcher is used to send notifications to the notification server interfaces according to subscription. For this purpose also the local identifiers of the notification server interfaces are used. The notification dispatcher support object administrates which notification server interfaces are subscribed to which notifications.

7.2.8 Support for location transparency

In OSI management location transparency can be achieved by global naming, that is, each managed object must have a global unique name independent of systems that provide access to those objects (i.e. agents).

In order to realise location transparency, the following two requirements need to be fulfilled:

- In the case that a manager knows the name of a managed object, it must be able to retrieve the addresses of the Application Entities that make that managed object visible. This requirement can be fulfilled by the Management Knowledge Management Function. The Directory can be the central point for storage of information that relates managed objects to their addresses; and

- A managed object shall have a global distinguished name. The principle of naming managed objects within the context of one management system only at which it is visible is disadvantageous for location transparency. For managed objects that can be flexibly located on different managed systems, a location independent global distinguished name shall be chosen.

7.2.9 Support for transaction transparency

Transaction transparency in OSI management can be achieved using the systems Management Application Context with Transaction Processing (ITU-T Rec. X.702 | ISO/IEC 11587).

NOTE 1 – This can be used for maintaining relationships between objects across multiple systems. ODMA functions for relationship management are for further study.

NOTE 2 – Not all OSI management implementation support transaction processing.

One object in the managing role may select multiple managed interfaces to an object in the managed role. The object in the managing role may issue an operation on these managed interfaces of an object in the managed role that needs to be synchronised according to ACID properties.

The synchronisation can be ensured by group invocations. In this way multiple interfaces can be accessed atomically by a single manager.

7.2.10 Support for persistence transparency

Although OSI Management is primarily concerned with interfaces, persistence transparency is needed and supported by most existing managed object implementations.

7.2.11 Support for replication transparency

With this transparency, managed objects can be visible at one or more agents. Although OSI Management is primarily concerned with interfaces, replication transparency is needed in some cases. ODMA functions to support this feature are for further study.

Annex A

OSI Management corresponding terms

(This annex does not form an integral part of this Recommendation | International Standard)

The following table uses ODMA terminology to explain existing concepts from OSI Management.

OSI systems management term	Corresponding ODMA term
Managed object	The combination of an operation server and notification client interface to a computational management object
Managed object class	Description of management-operation server and notification client interface, including the signature and the behaviour
Agent	A collection of engineering support objects which supports communication with objects in the managed role
Manager	A collection of engineering support objects which supports communication with objects in the managing role
Managed system	A node containing engineering objects in the managed role
Managing system	A node containing engineering objects in the managing role
Notification	An interaction for which the contract between the invoking object (client) and the receiving object (server) is restricted to the ability of the server to receive the contents of information sent by the client
Systems management operation	An operation performed by an object in the managing role on objects in the managed role
Attribute	Part of the signature of a management-operation interface (a shorthand notation for all access operations on attribute values)
Systems management function	One or more engineering support objects and a set of computational management interface types, which satisfy a set of logically related user requirements.
CMIP protocol entity	Part of the engineering protocol object
Naming tree	A tree of identifiers of operation interfaces, of use for the purpose of scoping
Scoping	Scoping is a function of an engineering object that enables a single invocation on the scope object to identify a set of management-operation server interfaces to determine whether the invocation will be propagated.
Filtering	Filtering is a function of an engineering object which applies a test to a management-operation server interface to verify if an operation invocation has to be performed on this interface.
NOTE 1 – In the right hand column the term ‘operation’ is used in the ODP sense and should not be confused with the notion of systems management operation. NOTE 2 – When the term object is not qualified it applies to both computational and engineering objects and is used in the ODP sense.	

Annex B

ODMA functions

(This annex does not form an integral part of this Recommendation | International Standard)

These functions are specific functions to support the distribution of management activities or the distribution of resources to be managed. Specific functions such as the operation dispatching function or the notification dispatching function provide binding between multiple computational management interfaces. Its semantics can be further refined in computational management objects performing generic management functions. For operation dispatching and notification dispatching, the following descriptions give one way of designing such functions. These are abstract and should be refined in particular paradigms (e.g. this has been done for the operation and notification dispatcher in clause 7). Furthermore, only information and computational viewpoints descriptions are provided.

NOTE – These are outlines of examples of ODMA functions. They may not correspond with actual ODMA functions.

B.1 Operation dispatching function

The operation dispatching function is used to control the binding between management-operation server interfaces and a management-operation client interface. It facilitates dynamic changes to the group of management-operation server interfaces.

B.1.1 Information viewpoint

The following information objects need to be described:

a) Operation Request

An Operation Request is composed of an Operation Name, a set of parameters and a Requested Destination Interface Set.

b) Operation Name

A name of the operation which is dispatched.

c) Requested Destination Interface Set

A Requested Destination Interface Set defines what destinations may receive the operation.

d) Generalised Scope

A Generalised Scope defines the list of Managed Resource References. It can be changed dynamically.

e) Maximum Destination Interface Set

A Maximum Destination Interface Set is a subset of the Generalised Scope. It can be changed dynamically.

f) Managed Resource Reference

A Managed Resource Reference is a potential target of the dispatching function. It is associated with a named set of Operation Names.

g) Actual Destination Interface Set

An Actual Destination Interface Set is the result of the Maximum Destination Interface Set intersected by the Requested Destination Interface Set. It is associated with each Operation Request.

Static and dynamic schemas can be further defined for specific paradigms (e.g. OSI-SM).

B.1.2 Computational viewpoint

In the computational viewpoint, the ODMA Operation Dispatching function is embodied by a computational binding object called an Operation Dispatcher. This object maintains a list of the management-operation server interfaces to which it is bound.

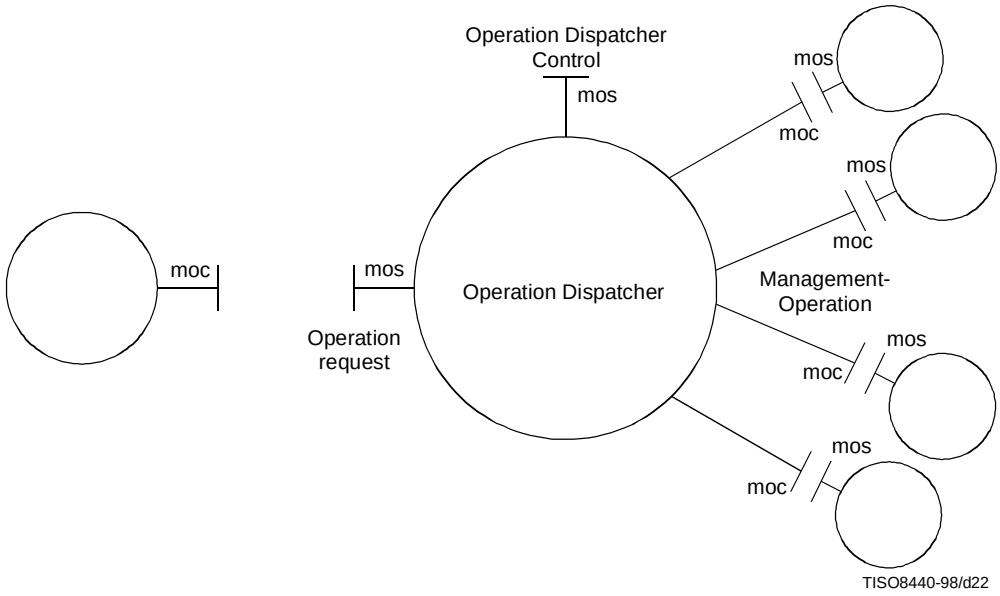


Figure B.1 – Operation Dispatching in the computational viewpoint

B.1.2.1 Computational interfaces

The Operation Dispatcher computational object should support the following interface types as server:

- Operation Request Interface;
- Operation Dispatcher Control Interface;

The Operation Dispatcher computational object should also support, for each object bound to it as client, the following interface type:

- Management-operation Interface.

The Operation Request Interface enables client ODMA objects to “push” operations towards a set of other ODMA objects. This interface contains one operation called Operation Request. This operation has at least three parameters: the first two respectively include the name of the Operation and the Information to be sent with the operation. The third one is optional and may comprise a specific list of destinations for the Operation.

The Operation Dispatcher Control interface enables other ODMA objects to dynamically manipulate the set of bound Management-operation Interfaces which may receive the dispatchable Operations. This interface should also facilitate the selection criteria of the Operation names. Such an interface contains at least two operations. The first one, Modify Destination Set, enables an ODMA object to modify the Destination List with one parameter representing the set of Management-operation Interfaces either explicitly with a list or explicitly through a selector. The second one, called Modify Permitted Operation Information, is used to modify the selection criterion of the Operations; the new criterion is passed as parameter.

B.1.2.1.1 Behaviour specification

The information contained in the request of an ODMA object via the Operation Request Interface must have been delivered to the destination set of bound Management-operation Interfaces that satisfy the criteria of the Operation Dispatcher related to the Operation name provided by the ODMA object which requested the dispatching.

The Management-operation contains the information of the Operation Request.

B.1.2.1.2 Environment contract

The Operation Dispatcher is intended to offer concurrent access.

B.2 Notification dispatching function

The notification dispatching function is used to control the binding between notification server interfaces and a notification client interface. It facilitates dynamic changes to the group of notification server interfaces.

NOTE – This is a simplified example which does not have the capability to determine destinations based on values of the parameters of the notification. For simplicity, it determines destinations based on the notification name.

B.2.1 Information viewpoint

The following information objects need to be described:

a) Notification Request

A Notification Request is composed of a Notification Name and a list of parameters.

b) Notification Name

A name of the notification which is dispatched.

c) Generalised Destination

A Generalised Destination defines the set of notification server interface references. It can be changed dynamically.

d) Maximum Destination Interface Set

A Maximum Destination Interface Set is a subset of the Generalised Destination. It can be changed dynamically. Each member of the subset is associated with a set of selected Notification Names.

e) Notification Server Interface Reference

A Notification Server Interface Reference is a potential target of the dispatching function. It is associated with one Notification Name.

f) Actual Destination Interface Set

An Actual Destination Interface Set is associated with each Notification Request. It is the result of the Maximum Destination Interfaces Set filtered by the Notification Names.

Static and dynamic schemas can be further defined for specific paradigms (e.g. OSI-SM).

B.2.2 Computational viewpoint

In the computational viewpoint, the ODMA Notification Dispatching function is embodied by a computational binding object called a Notification Dispatcher. This object maintains a list of the notification server interfaces to which it is bound.

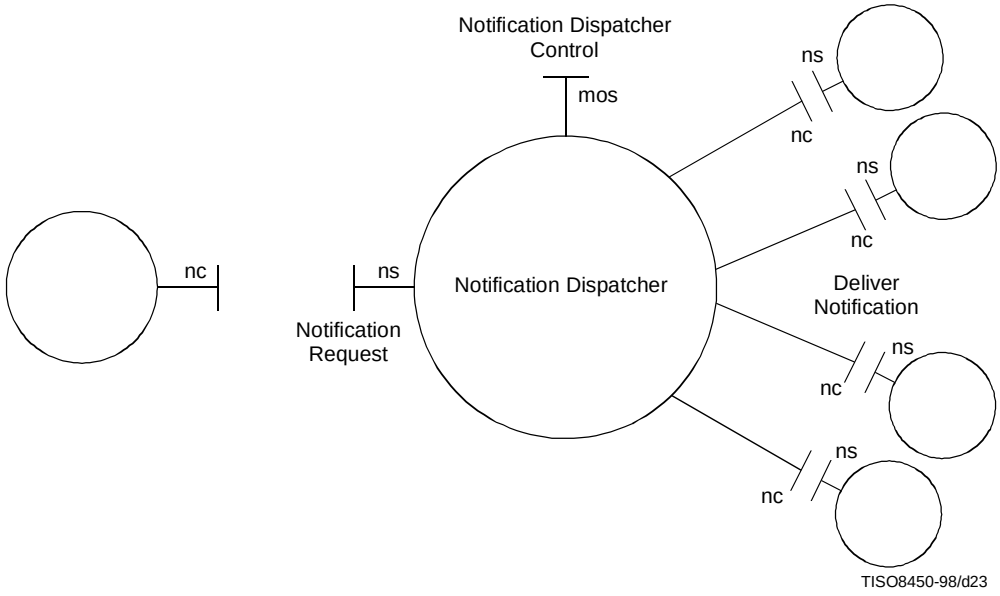


Figure B.2 – Notification Dispatching in the computational viewpoint

B.2.2.1 Computational interfaces

The Notification Dispatcher computational object should support the following interface types as a server:

- Notification Dispatcher Control Interface;
- Notification Request Interface.

The Notification Dispatcher computational object should also support the following interface type as a client:

- Deliver Notification Interface.

NOTE – All ODMA objects that are potential receivers of this notification must support the Deliver Notification Interface as server.

The Deliver Notification Interface (comprising a single operation Deliver Notification) is a notification client interface that dispatches notifications. Each instance of this interface is bound to an ODMA object supporting this interface as server.

The Notification Request Interface enables client ODMA objects to “push” notifications towards a set of other ODMA objects. This interface contains one operation called Notification Request. This operation has at least three parameters: the first two respectively include the name of the notification and the information to be sent with the notification. The third one is optional and may comprise a specific list of destinations for the notification.

The Notification Dispatcher Control interface enables other ODMA objects to dynamically manipulate the set of bound Deliver Notification Interfaces which should receive the dispatchable notifications. This interface should also facilitate the selection criteria of the notification names. Such an interface contains at least two operations. The first one, Modify Destination Set, enables an ODMA object to modify the Destination List with one parameter representing the set of Deliver Notification Interfaces either explicitly with a list or implicitly through a selector. The second one, called Modify Permitted Notification Information is used to modify the selection criterion of the notifications; the new criterion is passed as parameter.

B.2.2.2 Behaviour specification

The information contained in the request of an ODMA object via the Notification Request Interface must have been delivered to the destination set of bound Deliver Notification Interfaces that satisfy the criteria of the Notification Dispatcher related to the notification name provided by the ODMA object which requested the dispatching.

The Deliver Notification operation contains the information of the notification Request.

B.2.2.3 Environment contract

The Notification Dispatcher is intended to offer concurrent access. As a result, the Notification Dispatcher must offer atomic operations for its server interfaces.

B.3 Policy enforcing function

The policy enforcing function is a function which is used to ensure that management policies are enforced, and that attempted management policy violations are reported. Management policies are described in ITU-T Rec. X.701 | ISO/IEC 10040 and ITU-T Rec. X.749 | ISO/IEC 10164-19.

B.3.1 Information viewpoint

The information objects which needs to be specified are:

a) Policy Specifications

A Policy Specification defines the policy applied.

b) Violation Reports

A Violation Report is issued when a policy violation is attempted.

In addition, the operations and notifications enforced by the policy have to be specified.

B.3.2 Computational viewpoint

Two possible computational designs are illustrated:

- one (Figure B.3) where interactions between objects in the managing role and objects in the managed role are intercepted by a single Policy Enforcer computational binding object;
- another (Figure B.4) where each object in the managed role is bound via separate Policy Enforcer computational binding objects.

NOTE – Another possible solution is an entity that exists independent of the managed object to which the policy is being applied, and which inspects all interactions with all managed objects that are subject to the policy, and in turn raises appropriate reactions in case of policy violation.

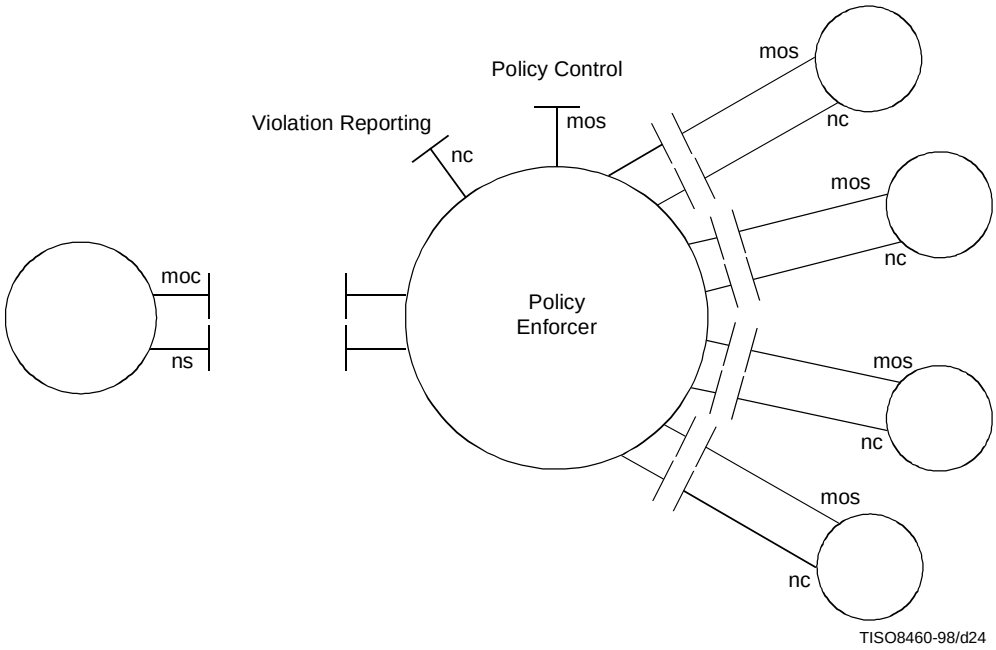


Figure B.3 – Computational design of Policy Enforcing with a single policy enforcer object

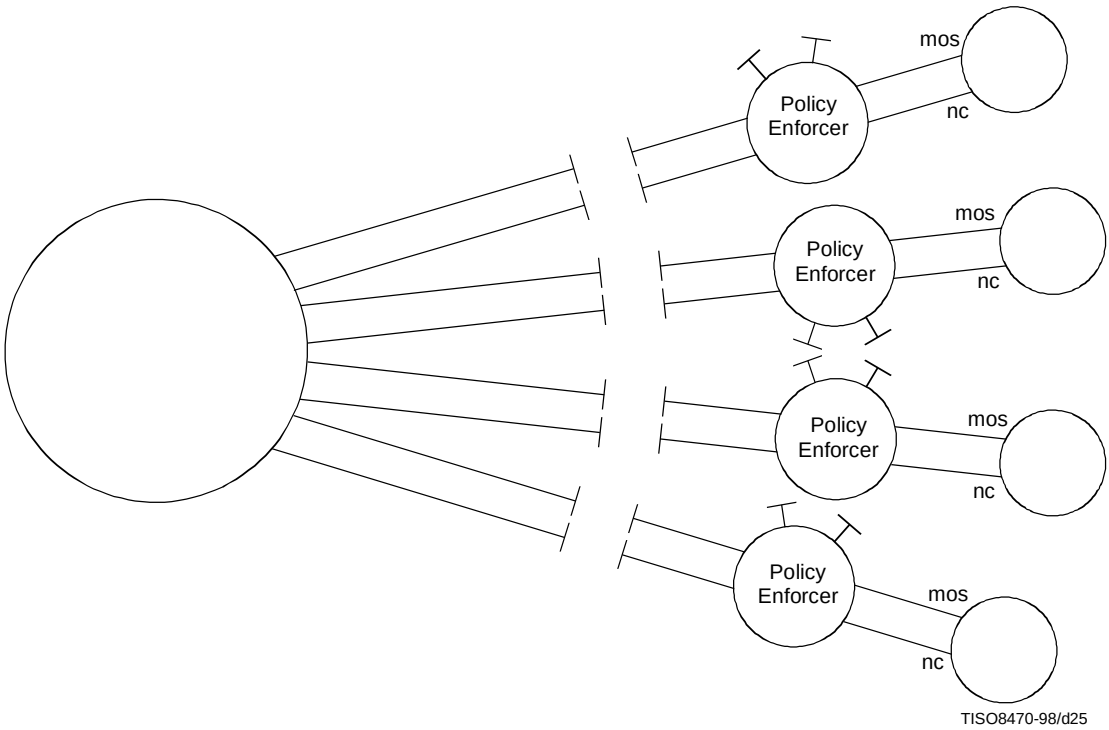


Figure B.4 – Computational design of Policy Enforcing with multiple Policy Enforcer objects

B.3.2.1 Computational interface

Apart from the obvious binding interfaces, the Policy Enforcer object should support the following special interfaces:

- the policy control interface is used to control the applied policies;
- violation reporting (notification client interfaces) is to report policy violations.

The difference between the two computational management object designs is expressed in terms of two different behaviour specifications and environment contracts.

B.3.2.2 Behaviour specification

- Solution 1 (Figure B.3):

  The Policy Enforcer is used to intercept interactions between objects in the managing role and in the managed role by a single computational binding object. Then the binding object must enable the setting of the policy and ensure the notification of the attempted policy violations. The Policy Enforcer must be able to intercept management interactions with all objects subject to the policy.

- Solution 2 (Figure B.4):

  The Policy Enforcer is used to intercept interactions between objects in the managing role and in the managed role by a set of computational binding object, where each Policy Enforcer is bound to a single object in the managed role and a single object in the managing role.

B.3.2.3 Environment contract

- Solution 1 (Figure B.3):

  There is a need to secure the access to the Policy Enforcer.

- Solution 2 (Figure B.4):

  Means must exist to ensure that policies are set consistently in all Policy Enforcer objects involved in enforcing one single policy. There is a need to secure the access to the policy enforcer.

Annex C

Example of specifying OSI management using ODP-RM

(This annex does not form an integral part of this Recommendation | International Standard)

This annex presents an example of the use of the ODP viewpoints in order to describe an Open Distributed Management application. In this example, the Log Control systems management function is described (see CCITT Rec.X.735 | ISO/IEC 10164-6), Log control functions.

NOTE – The method shown in this annex is not the only method for use within ODMA.

C.1 Enterprise viewpoint

The enterprise viewpoint text is a copy of the text in CCITT Rec. X.735 | ISO/IEC 10164-6, Log control functions.

For the purpose of many management functions it is necessary to be able to preserve information about events that may have occurred or operations that may have been performed by or on various objects. In a real open system various resources may be allocated to store such information. In OSI management these resources are modelled by logs and log records contained in the logs.

The management needs for the type of information that is to be logged may change from time to time. Furthermore, when such information is retrieved from a log the manager must be able to determine whether any records were lost or whether the characteristics of the records stored in the log were modified at any time.

The above needs give rise to the following requirements to be satisfied:

a) the definition of a flexible log control service which will allow selection of records that are to be logged by a management system in a particular log;

b) the ability for an external system to modify the criteria used in logging records;

c) the ability for an external system to determine whether the logging characteristics were modified or whether log records have been lost;

d) specification of a mechanism to control the time during which logging occurs, for example, by suspending and resuming logging;

e) the ability for an external system to retrieve and delete log records;

f) the ability for an external system to create and delete logs.

C.2 Information viewpoint

Information is needed on:

– the logged information (log and log records);

– the managed objects that model the source of the logged events.

Figure C.1 specifies some of the information viewpoint for log control:

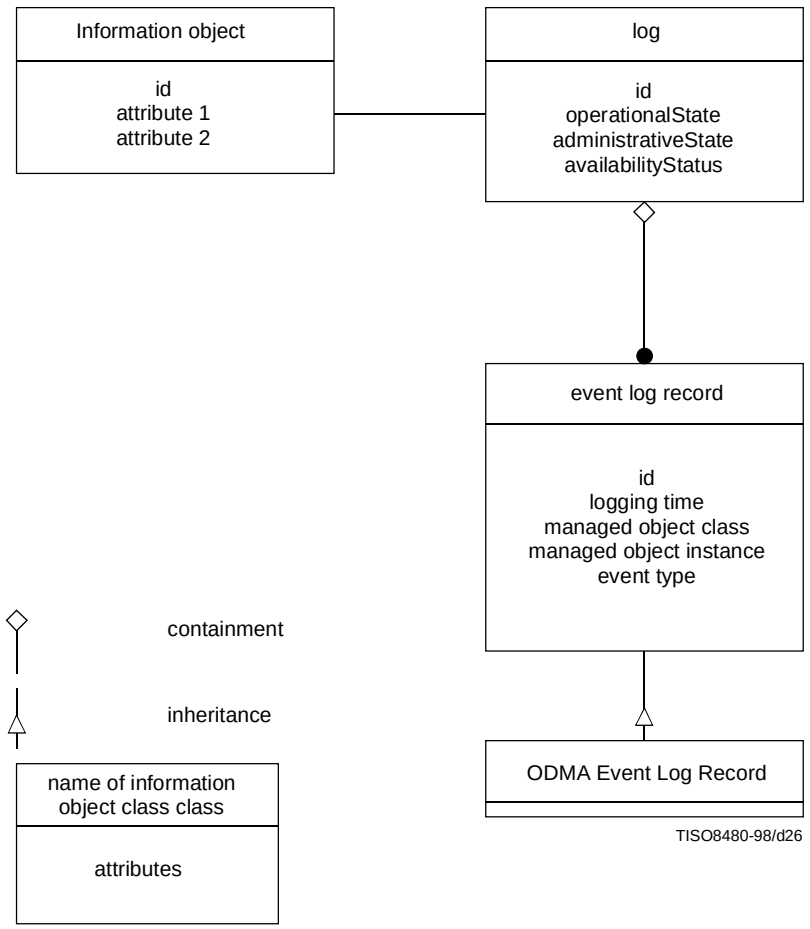


Figure C.1 – Example of information view of log control using the OMT notation[2)]

The specification by formal notations is also part of the information viewpoint, but is not elaborated in this example.

C.3 Computational viewpoint

In the computational viewpoint, the operation interfaces between computational management objects are visible. For the log control example, the following computational management objects are identified: log, log record, object that issues the notification and two manager objects.

The available GDMO specifications (without taking into account the conditional packages) are related to the interfaces that are identified in the computational viewpoint.

The following box lists the original GDMO template for the log Managed Object Class, as taken from CCITT Rec. X.721:

```
log      MANAGED OBJECT CLASS
DERIVED FROM      top;
CHARACTERIZED BY
--  see CCITT Rec. X.735 | ISO/IEC 10164-6 for the description of this managed object class.
logPackage     PACKAGE
        BEHAVIOUR
        logBehaviour  BEHAVIOUR
        DEFINED AS "This managed object is used to store incoming event reports and local system notifications. Additional
details are defined in CCITT Rec. X. 735 | ISO/IEC 10164-6. ";;
        ATTRIBUTES
        logId   GET,
```

2) The Object Modelling Technique (OMT) is a method developed by J. Rumbaugh.

```
        discriminatorConstruct        GET-REPLACE ,
        administrativeState    GET-REPLACE,
        operationalState       GET,
        availabilityStatus PERMITTED VALUES Attribute-ASN1Module.LogAvailability
                        REQUIRED VALUES Attribute-ASN1Module.UnscheduledLogAvailability GET,
        logFullAction  GET-REPLACE;
        NOTIFICATIONS
        objectCreation,
        objectDeletion,
        attributeValueChange,
        stateChange,
        processingErrorAlarm;;;
REGISTERED AS     {smi2MObjectClass 6};
```

In Figure C.2, manager object 1 receives a notification over notification server interface ns1. In addition, a log record object is created that can be read by manager object 1 over management-operation client interface oc1.

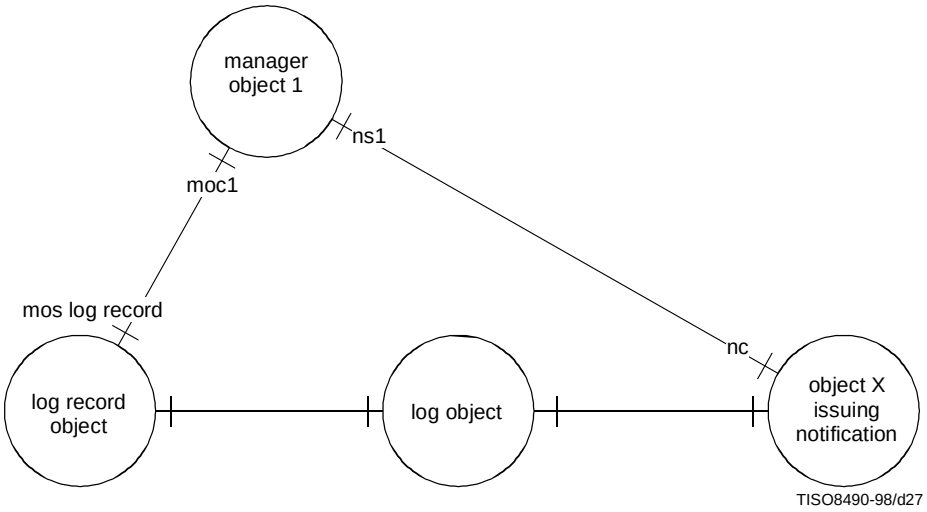


Figure C.2 – Example computational view on log control (normal operation)

In Figure C.3, another manager object (2) can read the log object through interface oc2 and the log record object through interface oc3.

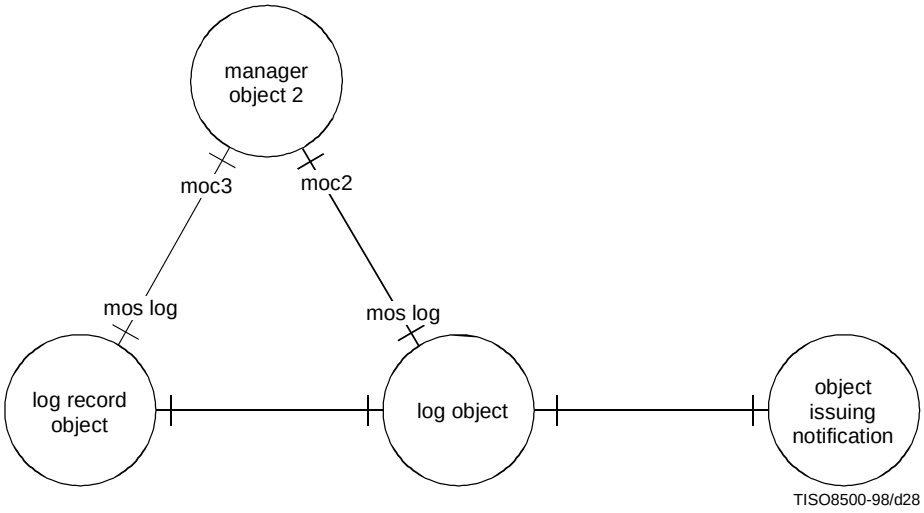


Figure C.3 – Example computational view on log control (log management)

C.4 Engineering viewpoint

For OSI Systems Management, the engineering viewpoint describes the functionality needed to provide communication between objects in the managing role and in the resource role using CMISE and, for instance, Transaction Processing.

For the log control example, the engineering viewpoint is presented in the Figure C.4:

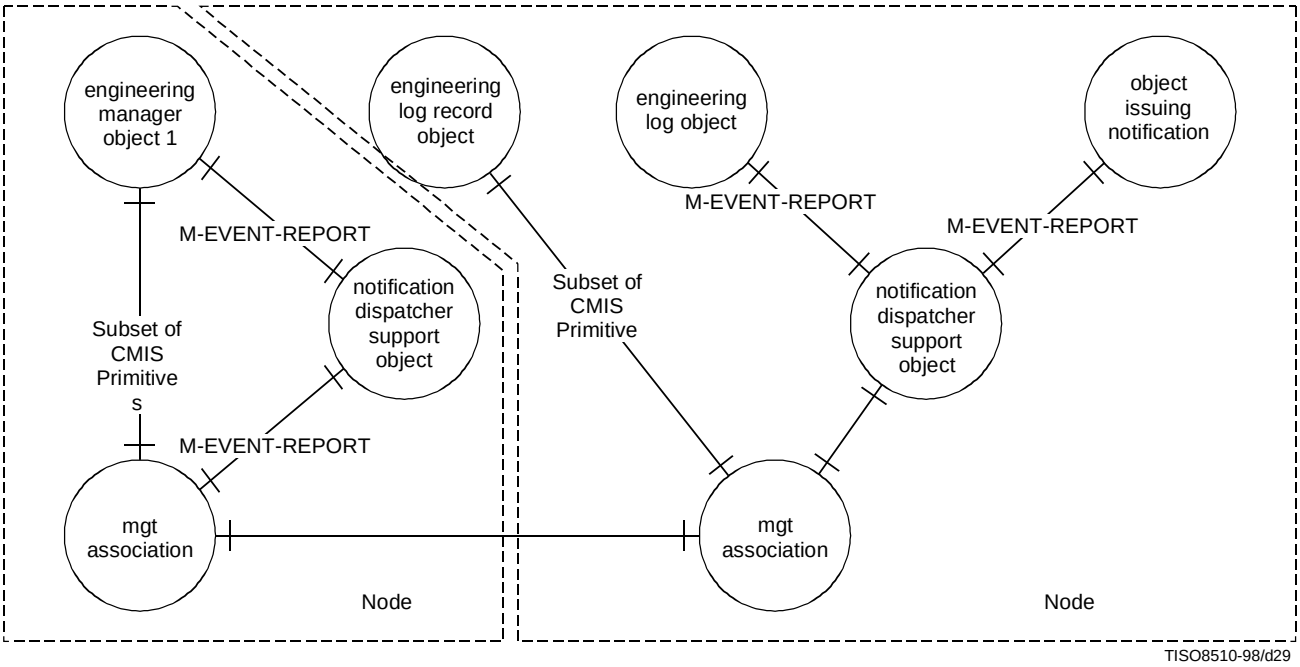
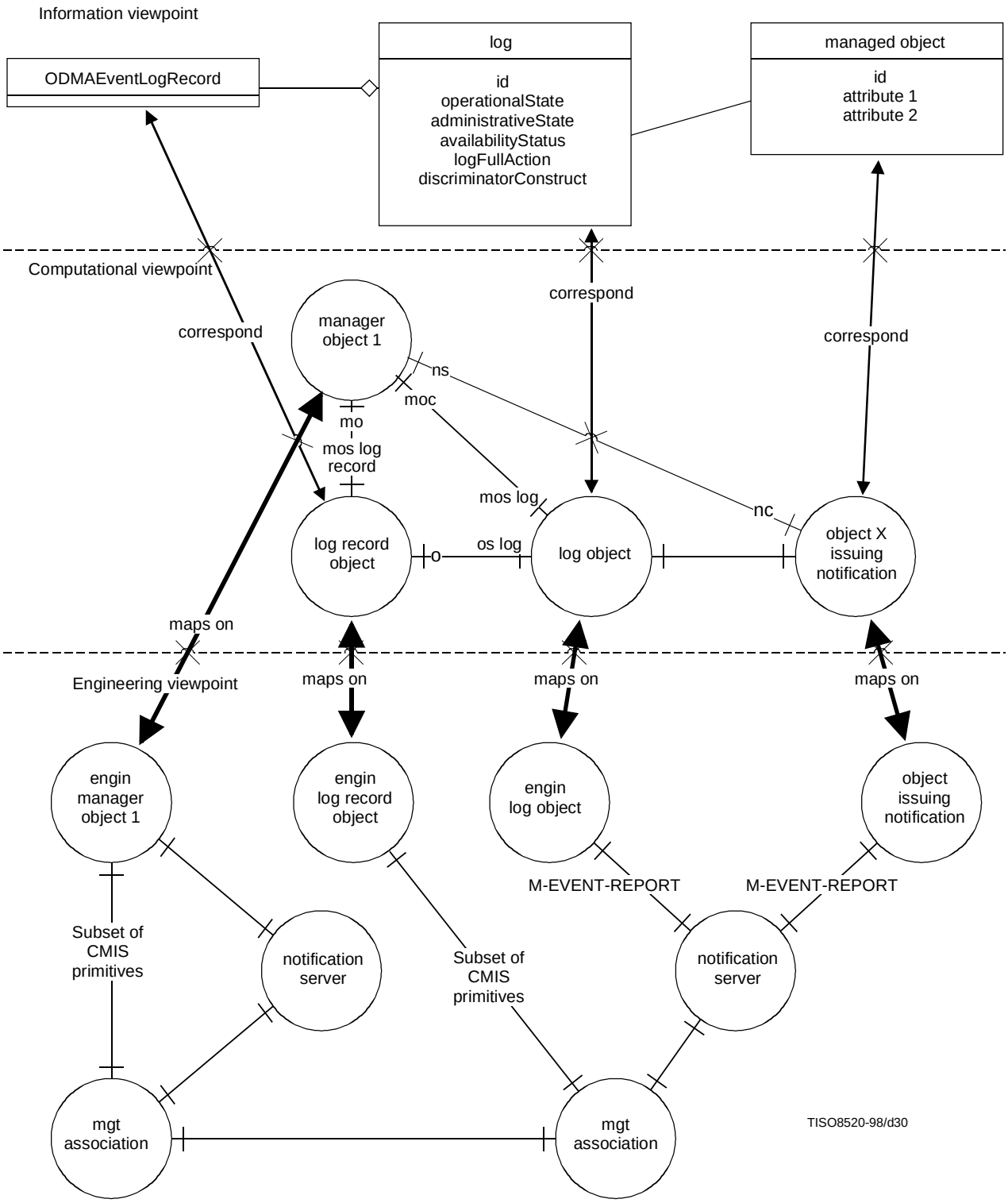


Figure C.4 – Example of engineering view on log management

For this particular example, the log and object issuing events exist within the same communication node. The location of the notification server is not made explicit in this figure, it may exist in the same node, but also in a different one. In the latter case, it needs its own communication channels.

C.5 Relationships between viewpoints



NOTE – In general it is not necessary to have a one-to-one mapping between objects in the various viewpoints.

Figure C.5 – Relationship between viewpoints

Annex D

Monitor metric example

(This annex does not form an integral part of this Recommendation | International Standard)

The purpose of this example is to illustrate how a ‘management-operation client’ interface can be exposed, and in fact also fully specified, by the use of GDMO and GRM.

NOTE 1 – The method shown in this annex is not the only method for use within ODMA.

NOTE 2 – A ‘notification server’ interface can be treated similarly, but this is not covered by the present example.

The example is based on the rather simple distribution structure of the monitorMetric managed object as defined in ITU-T Rec. X.739 | ISO/IEC 10164-11. Formally the GRM notation (ITU-T Rec. X.725 | ISO/IEC 10165-7) is used only to specify relationships between the ‘management-operation server’ of one computational management object and the ‘management-operation server’ interface of another computational management object (see Figure D.1).

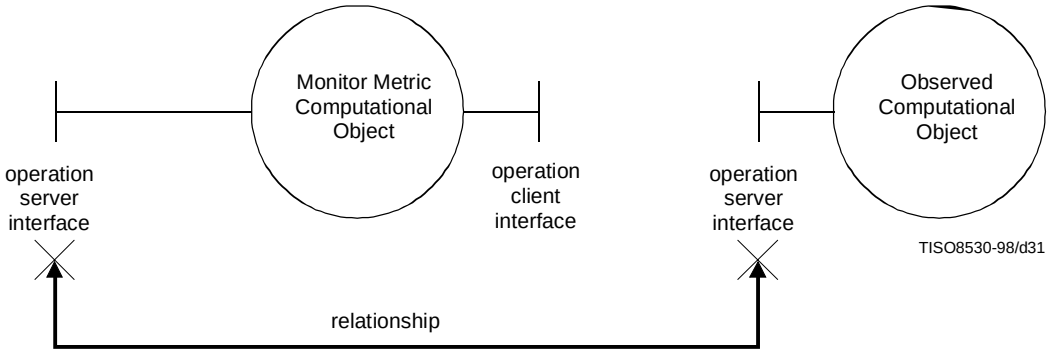


Figure D.1 – Relationship specification

However, it is relatively straightforward to expose ‘management-operation client’ interface by claiming the ‘management-operation client’ interface to the mirrored image of the ‘management-operation server’ interface as defined using GDMO, and subject to the constraints imposed by the specified relationship (see Figure D.2).

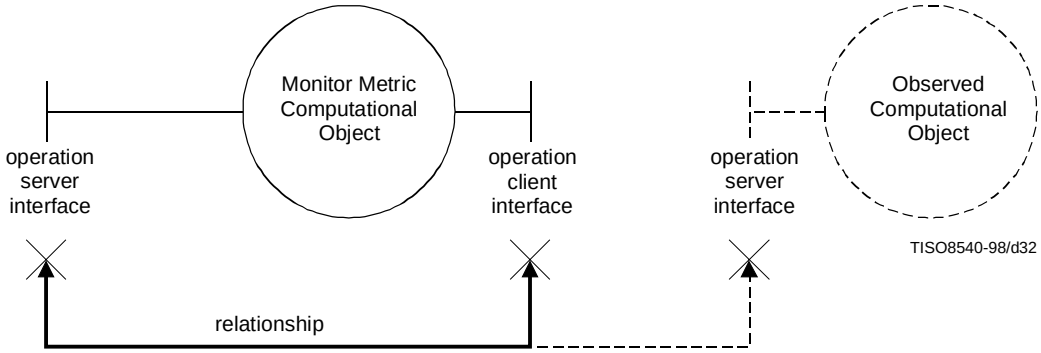


Figure D.2 – Managing interface specification

The observed object is solely for the purpose of identifying the management-operation client interface of the monitor metric object.

D.1 Definitions for Metric Objects

The following definitions are made:

- meanMonitorMetric relationship class representing the meanMonitorMetric computational object, the computational interfaces of this object are interpreted as specific role described by a managed object class;
- meanMonitorControl managed object class that describes the characteristics of managed object instances that can be used to control the meanMonitorMetric;
- scanObservedObjectValue managed object class that describes the characteristics of managed object instances that can be observed by the meanMonitorMetric;
- qualityOfServiceAlarm managed object class that describes the characteristics of managed object instances that can be used to emit notifications of the meanMonitorMetric

Figure D.3 gives a graphical representation:

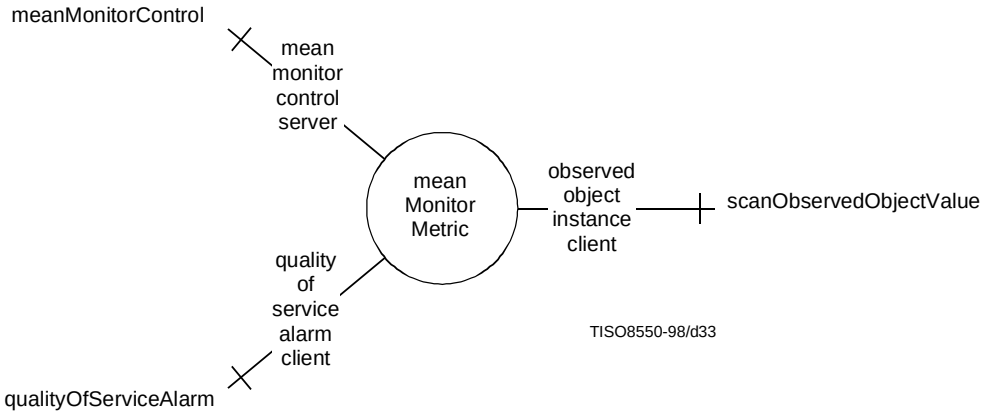


Figure D.3 – Mean monitor metric computational object

D.2 Relationship Class definition

meanMonitorMetric RELATIONSHIP CLASS

BEHAVIOUR meanMonitorMetricBhv BEHAVIOUR DEFINED AS

> **"The meanMonitorMetric computational object invokes a get operation on the observed object instance in the observedObjectInstanceClient role to obtain the current observed attribute value. if the scan operation has not resulted in the return of the observed object value before the start of the next granularity period, then that scan is invalidated, and is not usable for the metric algorithm update process.";;**

ROLE meanMonitorControlServer COMPATIBLE WITH meanMonitorControl;

ROLE observedObjectInstanceClient COMPATIBLE WITH scanObservedObjectValue;

ROLE qualityOfServiceAlarmClient COMPATIBLE WITH qualityOfServiceAlarm;

REGISTERED AS { };

D.3 Managed Object Class definitions

meanMonitorControl MANAGED OBJECT CLASS
DERIVED FROM "CCITT Rec. X.721 | ISO/IEC 10165-2":top;
CHARACTERIZED BY meanMonitorControlPkg PACKAGE
BEHAVIOUR meanMonitorControlBhv BEHAVIOUR DEFINED AS
"the control operations must affect the parameterised behaviour of the metric algorithm for scans originating after the attributes are set.";;
ATTRIBUTES
observedObjectInstance GET-REPLACE,
observedAttributeId GET-REPLACE,
granularityPeriod GET-REPLACE,
movingTimePeriod GET-REPLACE,
derivedGauge GET-REPLACE
notificationTriggerThreshold GET-REPLACE,
re-armThreshold GET-REPLACE,
operationalState GET-REPLACE;;
REGISTERED AS { };

scanObservedObjectValue MANAGED OBJECT CLASS
DERIVED FROM "CCITT Rec. X.721 | ISO/IEC 10165-2":top;
CHARACTERIZED BY scanObservedObjectValuePkg PACKAGE
BEHAVIOUR scanObservedObjectValueBhv BEHAVIOUR DEFINED AS
"Compatible classes of this managed object class shall have at least one attributethat has an integer or real value syntax.";;
REGISTERED AS { };

NOTE – We have to use the compatibility mechanism because a metric object works on arbitrary managed object classes.

qualityOfServiceAlarm MANAGED OBJECT CLASS
DERIVED FROM "CCITT Rec. X.721 | ISO/IEC 10165-2":top;
CHARACTERIZED BY qualityOfServiceAlarmPkg PACKAGE
BEHAVIOUR qualityOfServiceAlarmBhv BEHAVIOUR DEFINED AS
"Compatible classes of this managed object class will need the qualityOfServiceAlarmNotification in case there is a positive crossing event for the notification trigger.";;
NOTIFICATION
"ISO/IEC 10164-4":qualityOfServiceAlarmNotification;;;
REGISTERED AS { };

D.4 Example for Computational Metric Objects

Here we give an example of how the Metric Object may be used. The following is defined:

- observedObjectExample computational object represented as a relationship class;
- metricObjectExample relationship mapping;
- fullMeanMonitorControl managed object class;
- observedObjectInterfaceExample managed object class.

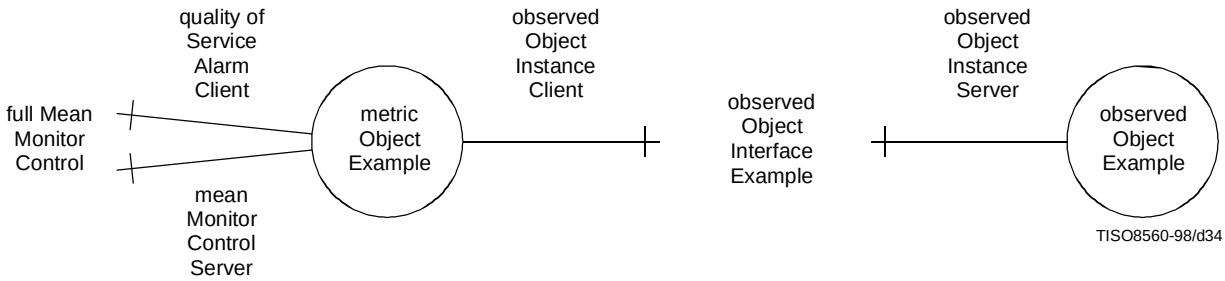


Figure D.4 – Example for Computational Metric Object

D.5 Relationship Class example

```
observedObjectExample RELATIONSHIP CLASS
BEHAVIOUR observedObjectExampleBhv;
ROLE observedObjectInstanceServer COMPATIBLE WITH observedObjectInterfaceExample ;
REGISTERED AS { };
```

D.6 Relationship Mapping example

```
metricObjectExample RELATIONSHIP MAPPING
RELATIONSHIP CLASS meanMonitorMetric;
RELATIONSHIP OBJECT fullMeanMonitorControl;
ROLE meanMonitorControlServer RELATED CLASSES fullMeanMonitorControl;
ROLE observedObjectInstanceClient RELATED CLASSES observedObjectInterfaceExample
        REPRESENTED BY RELATIONSHIP-OBJECT-USING-POINTER observedObjectInstance;
ROLE qualityOfServiceAlarmClient RELATED CLASSES fullMeanMonitorControl;
REGISTERED AS { } ;
```

D.7 Managed Object Class examples

```
fullMeanMonitorControl MANAGED OBJECT CLASS
DERIVED FROM meanMonitorControl, qualityOfServiceAlarm;
REGISTERED AS { };

observedObjectInterfaceExample MANAGED OBJECT CLASS
DERIVED FROM "CCITT Rec. X.721 | ISO/IEC 10165-2":top;
CHARACTERIZED BY observedObjectInterfaceExamplePkg PACKAGE
BEHAVIOUR observedObjectInterfaceExampleBhv;
ATTRIBUTES
    observedValue     GET-REPLACE;;
REGISTERED AS { };
```

Annex E

Examples of computational templates

(This annex does not form an integral part of this Recommendation | International Standard)

This annex provides examples of different types of notation that can be used as computational templates.

NOTE – The method shown in this annex is not the only method for use within ODMA.

E.1 ITU-T Rec. G.851.1 computational template

In the description of an operation in the computational viewpoint, the schemes in the information viewpoint are referenced to define the pre-conditions that the system[3)] must verify so that this operation can be performed, and the post-conditions that the system must verify after this operation. The GRM relationship class specifies a set of systems by providing:

- objects involved in all the instances of the systems class (through the use of the <role> clause);
- possible constraints on their states (through the use of <invariant> clause);
- relationships involved in the systems class (through the use of <invariant> clause);
- possible constraints on combined states described in relationships (through the use of <invariant> clause).

The operation parameter will select which system instance will be addressed by the operation in the pre-condition and which system instance will be provided as the result of the operation in the post condition. Therefore the behaviour has to include:

- a pre-condition clause referencing a system class of the information viewpoint specification;
- a post-condition clause referencing a system class of the information viewpoint specification;
- a parameter matching rule clause to give information about this selection.

E.2 Examples of use of the computational template

This example shows a computational interface and one of its operations which provides the transition from ssccNotConnected to ssccConnected in the simple subnetwork connection community (i.e. sets up a subnetwork connection between a given a-end point and a given z-end point).

Label References

The following information static schema, and ASN.1 productions are referenced in this annex:

Fully Qualified Label Reference	Local Reference Used
<"Rec. G.853.2", STATIC_SCHEMA:ssccNotConnected >	<ssccNotConnected>
<"Rec. G.853.2", STATIC_SCHEMA:ssccConnected >	<ssccConnected>

Fully Qualified ASN.1 Production Reference	Local Reference Used
"M.3100:1995 : ASN1DefinedTypesModule"::Directionality	Directionality
"M.3100:1995 : ASN1DefinedTypesModule"::UserLabel	UserLabel

3) The term ‘system’ refers to an information system.

simple SNC performer interface

The simple SNC performer interface is required to satisfy the enterprise requirements stated in

<"Rec. G.852.1", COMMUNITY:sscc, ACTION:sccc1 > ,

<"Rec. G.852.1", COMMUNITY:sscc, ACTION:sccc2 > .

```
COMPUTATIONAL_INTERFACE simpleSncPerformerIfce {
                    OPERATIONS          <setupSubnetworkConnection>;
                                        <releaseSubnetworkConnection>;
                    BEHAVIOUR "Any Computational Object Supporting this interface as server must meet requirements
stated in BEHAVIOUR clauses of the OPERATION templates for each of the operations in this interface.";
}
```

setupSubnetworkConnection operation

```
OPERATION           setupSubnetworkConnection {
                    INPUT_PARAMETERS
                            subnetwork : SubnetworkId ::= REF(snQueryIfce);
                            snpa : SnTPId ::= REF(snTPQueryIfce);
                            snpz : SnTPId ::= REF(snTPQueryIfce);
                            dir : Directionality;
                            suppliedUserLabel : UserLabel ;
                                    -- zero length string implies none supplied
                            serviceCharacteristics: CharacteristicsId ::= REF(serviceCharacteristicsQueryIfce) ;
                                    -- reference can be used to determine any QoS or routing characteristics) ;

                    OUTPUT_PARAMETERS
                            newSNC : SNCId ::= REF(sncQueryIfce) ;
                            agreedUserLabel : UserLabel ;

                    RAISED_EXCEPTIONS
                            incorrectSubnetworkTerminationPoints : SnTPId;
                            subnetworkTerminationPointDisabled : SnTPId;
                            subnetworkDisabled : NULL;
                            subnetworkTerminationPointConnected : SnTPId;
                            operationFails : NULL;
                            wrongDirectionality : Directionality;
                            userLabelInUse : UserLabel;

                    BEHAVIOUR
                            PARAMETER_MATCHING
                            subnetwork: < ssccNotConnected , ROLE:involvedSubnetwork > AND
                                        < ssccConnected , ROLE:involvedSubnetwork > ;
                            snpa : < ssccNotConnected , ROLE:potentialAEnd > AND
                                    < scmConnected , ROLE:connectedAEnd > ;
                            snpz : < ssccNotConnected , ROLE: potentialZEnd > AND
                                    < ssccConnected , ROLE:connectedZEnd > ;
                    dir : < ssccConnected, ROLE: involvedSubnetwork , ATTRIBUTE: directionality > ;
                    newSNC : <ssccConnected, ROLE: involvedSubnetwork> ;
                    suppliedUserLabel : <ssccConnected, ROLE:involvedSubnetwork, ATTRIBUTE: userLabel >
                                        OR <> ; -- The user does not have to supply a user label value
                    agreedUserLabel : <ssccConnected, ROLE:involvedSubnetwork, ATTRIBUTE:
                    userLabel > ;
                    serviceCharacteristics : < ssccConnected , ROLE:involvedServiceCharacteristics > ;

                    PRE_CONDITIONS < ssccNotConnected> ;
-- The scmNotConnected schema defines a schema type with two non-connected
-- networkTP information objects subtypes candidates to the point to point connection
-- management service.
                    POST_CONDITIONS < ssccConnected> ;
                -- The scmConnected schema defines the schema type of two connected networkTPinformation
                -- objects candidates to the point to point connection management service.

                    EXCEPTIONS
```

```
IF PRE_CONDITION <inv_1> NOT_VERIFIED RAISE_EXCEPTION
        incorrectSubnetworkTerminationPoints ;
IF PRE_CONDITION <inv_2> NOT_VERIFIED RAISE_EXCEPTION
        subnetworkTerminationPointConnected ;
IF PRE_CONDITION <inv_3> NOT_VERIFIED RAISE_EXCEPTION
        subnetworkTerminationPointConnected ;
IF POST_CONDITION <inv_1> NOT_VERIFIED RAISE_EXCEPTION
        operationFails ;
IF POST_CONDITION <inv_2> NOT_VERIFIED RAISE_EXCEPTION
        operationFails ;
IF POST_CONDITION <inv_3> NOT_VERIFIED RAISE_EXCEPTION
        operationFails ;
IF POST_CONDITION <inv_4> NOT_VERIFIED RAISE_EXCEPTION
        userLabelInUse ;

INFORMAL
        "This operation sets up a subnetwork connection between a given a end sntp and a given z end
        sntp." ;
}
```

Annex F

Example of Enterprise Community Specification

(This annex does not form an integral part of this Recommendation | International Standard)

NOTE – The method shown in this annex is not the only method for use within ODMA.

F.1 ODP Enterprise viewpoint concepts

This annex repeats ODP-RM definitions in order to improve the understanding of the templates in F.2.

Contract

Contract: An agreement governing part of the collective behaviour of a set of objects. A contract places obligations on the objects involved. The specification of a contract may include:

- a specification of the different roles that objects involved in the contract may assume, and the interfaces associated with the roles;
- quality of service attributes;
- indications of duration or periods of validity;
- indications of behaviour which invalidates the contract; and
- life and safety conditions.

Enterprise role

Subsets corresponding to specific functionalities can be extracted from the behaviour of an object. Such subsets are called enterprise roles. When an object is viewed in terms of a enterprise role, only a named subset of its actions is of interest, and other actions are abstracted away – possibly to other enterprise roles. Each object may have several enterprise roles at a given time depending upon its interactions, and may take on different sets of enterprise roles at different times.

The enterprise roles in community are there because they provide a particular function with respect to the objective of the community.

The enterprise roles of special interest to ODMA are various cases of managing role and managed role.

Community

The requirements are captured by first identifying the communities in the ODP system where a community is a group of roles that have come together for some objective. The objective of the community needs to be expressly articulated. Typically, the objective of the community is the provision of a particular service.

Community: A composition of objects formed to meet an objective. The objective is expressed as a contract which specifies how the objective can be met.

The community is defined by its purpose (i.e. the common objective of the roles involved in the community), the definition of each role implied in the community and the policy applicable to the entire community.

Policy

The community policy is specified as a set of permissions, obligations and prohibitions applicable for either the client or the provider with regard to the community.

- Policy: A set of rules related to a particular purpose. A rule can be expressed as permissions, obligations or prohibitions;
- Permission: A prescription that a particular behaviour is allowed to occur;
- Obligation: A prescription that a particular behaviour is required; and
- Prohibition: A prescription that a particular behaviour must not occur.

Action

Action: Something which happens.

Each action is defined by the action name and the specification of the action policy.

The action policy is specified as a set of permissions, obligations and prohibitions applicable for roles with regard to the community.

Activity

ODP-RM gives the following definition of an activity:

Activity: A single-headed acyclic graph of actions, where occurrences of each action in the graph is made possible by the occurrence of all immediately preceding actions.

F.2 Example Enterprise Community specification

COMMUNITY Simple Sub-Network Connection Management

Community purpose:

"The objective of the community is to manage point-to-point subnetwork connections between a specified set of endpoints, previously populated on the boundary of a subnetwork.

This is accomplished by providing 2 generic actions namely to

- set up; and
- release

the subnetwork connection in question."

Community roles

caller "This role reflects the client of the Simple Subnetwork Connection Management service."

provider "This role reflects the server of the Simple Subnetwork Connection Management service."

port "This role reflects the two G.805 ports involved in the Simple Subnetwork Connection Management community."

sn "This role reflects the G.805 Sub-network for which the Simple Subnetwork Connection Management service is defined."

snc "This role reflects the G.805 Subnetwork Connection involved in the Simple Subnetwork Connection Management Community."

Community Policies

None defined.

Community Action descriptions

Setup Point-to-Point SNC

"This action sets up a point-to-point subnetwork connection between two ports on the same subnetwork".

Action Policies

OBLIGATION1

"The caller shall identify two ports which must be part of the subnetwork domain".

OBLIGATION2

"If the service request is uni-directional, one source and one sink port must be identified by the caller."

OBLIGATION3

"In case of service rejection, the provider shall let the caller know which policy has been violated."

OBLIGATION4

"In case of service establishment, the provider shall provide the caller with the following connection information:

- a unique Subnetwork Connection identity;
- whether it is uni- or bi-directional;
- the ports which make up the endpoints of the Subnetwork Connection."

PERMISSION1

"The caller may specify whether it is a Uni-directional or Bi-directional Subnetwork Connection"

PERMISSION2

"The caller may specify a route restriction based upon Link ID or Subnetwork(Matrix) ID."

PERMISSION3

"The caller may specify an availability grade of service."

PERMISSION4

"The caller may specify a Quality of Service(QOS) identifier."

PERMISSION5

"The caller may specify a bandwidth."

PROHIBITION1

"The provider cannot satisfy the request if one of the ports are in use."

Modify Point-to-Point SNC

"This action is used to modify a simple subnetwork connection."

Action Policies

OBLIGATION1

"The connection must be part of the subnetwork domain."

OBLIGATION2

"The caller must identify the connection as part of the request."

PERMISSION5

"The caller may specify the new bandwidth requested."

Release Connection

"This action is used to release a simple subnetwork connection."

Action Policies

OBLIGATION1

"The connection must be part of the subnetwork domain."

OBLIGATION2

"The caller must identify the connection as part of the request."

OBLIGATION3

"The provider will make all resources available after the connection is cleared."

OBLIGATION4

"The provider shall provide a reason if the action is rejected."

OBLIGATION5

"The provider shall provide the unique connection identity of the cleared connection."

ITU-T RECOMMENDATIONS SERIES

Series A	Organization of the work of the ITU-T
Series B	Means of expression: definitions, symbols, classification
Series C	General telecommunication statistics
Series D	General tariff principles
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	TMN and network maintenance: international transmission systems, telephone circuits, telegraphy, facsimile and leased circuits
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	**Data networks and open system communications**
Series Y	Global information infrastructure
Series Z	Programming languages