X/Open Guide

Systems Management: Reference Model

: Special Draft for XTP TWG, Sept 1995

X/Open Company Ltd.

X/Open Guide

Systems Management: Reference Model : Special Draft for XTP TWG, Sept 1995

ISBN: 1-85912-005-9
X/Open Document Number: G207

Published by X/Open Company Ltd., U.K.

Any comments relating to the material contained in this document may be submitted to X/Open at:

X/Open Company Limited
Apex Plaza
Forbury Road
Reading
Berkshire, RG1 1AX
United Kingdom

or by Electronic Mail to:

XoSpecs@xopen.org

Contents

List of Figures

Preface

X/Open

X/Open is an independent, worldwide, open systems organisation supported by most of the world's largest information systems suppliers, user organisations and software companies. Its mission is to bring to users greater value from computing, through the practical implementation of open systems.

X/Open's strategy for achieving this goal is to combine existing and emerging standards into a comprehensive, integrated, high-value and usable open system environment, called the Common Applications Environment (CAE). This environment covers the standards, above the hardware level, that are needed to support open systems. It provides for portability and interoperability of applications, and so protects investment in existing software while enabling additions and enhancements. It also allows users to move between systems with a minimum of retraining.

X/Open defines this CAE in a set of specifications which include an evolving portfolio of application programming interfaces (APIs) which significantly enhance portability of application programs at the source code level, along with definitions of and references to protocols and protocol profiles which significantly enhance the interoperability of applications and systems.

The X/Open CAE is implemented in real products and recognised by a distinctive trade mark — the X/Open brand — that is licensed by X/Open and may be used on products which have demonstrated their conformance.

X/Open Technical Publications

X/Open publishes a wide range of technical literature, the main part of which is focussed on specification development, but which also includes Guides, Snapshots, Technical Studies, Branding/Testing documents, industry surveys, and business titles.

There are two types of X/Open specification:

- *CAE Specifications*

  CAE (Common Applications Environment) specifications are the stable specifications that form the basis for X/Open-branded products. These specifications are intended to be used widely within the industry for product development and procurement purposes.

  Anyone developing products that implement an X/Open CAE specification can enjoy the benefits of a single, widely supported standard. In addition, they can demonstrate compliance with the majority of X/Open CAE specifications once these specifications are referenced in an X/Open component or profile definition and included in the X/Open branding programme.

  CAE specifications are published as soon as they are developed, not published to coincide with the launch of a particular X/Open brand. By making its specifications available in this way, X/Open makes it possible for conformant products to be developed as soon as is practicable, so enhancing the value of the X/Open brand as a procurement aid to users.

- *Preliminary Specifications*

  These specifications, which often address an emerging area of technology and consequently are not yet supported by multiple sources of stable conformant implementations, are released in a controlled manner for the purpose of validation through implementation of products. A Preliminary specification is not a draft specification. In fact, it is as stable as X/Open can make it, and on publication has gone through the same rigorous X/Open development and review procedures as a CAE specification.

  Preliminary specifications are analogous to the *trial-use* standards issued by formal standards organisations, and product development teams are encouraged to develop products on the basis of them. However, because of the nature of the technology that a Preliminary specification is addressing, it may be untried in multiple independent implementations, and may therefore change before being published as a CAE specification. There is always the intent to progress to a corresponding CAE specification, but the ability to do so depends on consensus among X/Open members. In all cases, any resulting CAE specification is made as upwards-compatible as possible. However, complete upwards-compatibility from the Preliminary to the CAE specification cannot be guaranteed.

In addition, X/Open publishes:

- *Guides*

  These provide information that X/Open believes is useful in the evaluation, procurement, development or management of open systems, particularly those that are X/Open-compliant. X/Open Guides are advisory, not normative, and should not be referenced for purposes of specifying or claiming X/Open conformance.

- *Technical Studies*

  X/Open Technical Studies present results of analyses performed by X/Open on subjects of interest in areas relevant to X/Open's Technical Programme. They are intended to communicate the findings to the outside world and, where appropriate, stimulate discussion and actions by other bodies and the industry in general.

- *Snapshots*

  These provide a mechanism for X/Open to disseminate information on its current direction and thinking, in advance of possible development of a Specification, Guide or Technical Study. The intention is to stimulate industry debate and prototyping, and solicit feedback. A Snapshot represents the interim results of an X/Open technical activity. Although at the time of its publication, there may be an intention to progress the activity towards publication of a Specification, Guide or Technical Study, X/Open is a consensus organisation, and makes no commitment regarding future development and further publication. Similarly, a Snapshot does not represent any commitment by X/Open members to develop any specific products.

Versions and Issues of Specifications

As with all *live* documents, CAE Specifications require revision, in this case as the subject technology develops and to align with emerging associated international standards. X/Open makes a distinction between revised specifications which are fully backward compatible and those which are not:

- a new *Version* indicates that this publication includes all the same (unchanged) definitive information from the previous publication of that title, but also includes extensions or additional information. As such, it *replaces* the previous publication.

- a new *Issue* does include changes to the definitive information contained in the previous publication of that title (and may also include extensions or additional information). As such, X/Open maintains *both* the previous and new issue as current publications.

Corrigenda

Most X/Open publications deal with technology at the leading edge of open systems development. Feedback from implementation experience gained from using these publications occasionally uncovers errors or inconsistencies. Significant errors or recommended solutions to reported problems are communicated by means of Corrigenda.

The reader of this document is advised to check periodically if any Corrigenda apply to this publication. This may be done either by email to the X/Open info-server or by checking the Corrigenda list in the latest X/Open Publications Price List.

To request Corrigenda information by email, send a message to info-server@xopen.co.uk with the following in the Subject line:

```
request corrigenda; topic index
```

This will return the index of publications for which Corrigenda exist.

This Document

This document is a Guide (see above). It provides an architectural overview of the Systems Management Model, identifies the various components of the model, and describes the ways in which they interact.

The X/Open Systems Management Reference Model employs object-oriented specification techniques. Within the model different components perform various roles necessary to provide a fully functional, interoperable systems management architecture. X/Open has an agreement with the OMG to use OMG-compliant specifications for any work programmes where an object-oriented approach is taken. In addition, the OSI Network Management standards exist and have taken a somewhat different object-oriented approach. This document:

- presents a higher-level model which encompasses both models
- presents an approach for coexistence of both models
- outlines differences in terminology

This reference model does not go into detailed consideration of the various components (for instance it does not define which managed objects will exist) but addresses the general properties of components within the model, their means of interaction, and the properties of their interfaces.

Trade Marks

X/Open® is a registered trade mark, and the ‘‘X’’ device is a trade mark, of X/Open Company Limited.

Referenced Documents

The following documents are referenced in this guide:

CORBA 1.2
: X/Open CAE Specification, July 1994, The Common Object Request Broker: Architecture and Specification (ISBN: 1-85912-044-X, C432), in conjunction with the Object Management Group (OMG).

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

ISO/IEC 9595
: ISO/IEC 9595: 1991 Information Technology — Open Systems Interconnection — Common Management Information Service Definition.

CMIP
: ISO/IEC 9596-1: 1991, Information Technology — Open Systems Interconnection — Common Management Information Protocol, Part 1: Specification.

ISO/IEC 10040
: ISO/IEC 10040: 1992 Information Technology, Open Systems Interconnection — Systems Management Overview.

ISO/IEC 10164
: ISO/IEC 10164: 1992 Information Technology — Open Systems Interconnection — Systems Management (Parts 1 to 13 inclusive).

ISO/IEC 10165-1
: ISO/IEC 10165-1: 1992, Information Technology — Open Systems Interconnection — Structure of Management Information — Part 1: Management Information Model.

GDMO
: ISO/IEC 10165-4:1992, Information Technology — Open Systems Interconnection — Structure of Management Information — Part 4: Guidelines for the Definition of Managed Objects.

OMAG
: Object Management Group Architecture Guide, OMG, Issue 1.0, 1st November 1990.

OMGOM
: Object Management Group Object Model.

PS
: X/Open Snapshot, 1991 Systems Management: Problem Statement (XO/SNAP/91/010, S110).

RFC 1155
: RFC 1155, Structure of Management Information (SMI), M. Rose, & K. McCloghrie.

RFC 1157
: RFC 1157, Simple Network management Protocol (SNMP), J. Case, M. Fedor, M. Schoffstall, & J. Davin.

RFC 1442
: RFC 1442, Structure on Management Information for version 2 of the Simple Network

Management Protocol (SNMPv2), J. Case, K. McCloghrie, M. Rose, & S. Waldbusser.

RFC 1448
RFC 1448, Protocol Operations for version 2 of the Simple Network Management Protocol (SNMPv2), J. Case, K. McCloghrie, M. Rose, & S.Waldbusser.

XIMS
X/Open Snapshot, May 1992, Systems Management: Identification of Management Services (ISBN: 1-872630-30-8, S190).

XTP
X/Open Technical Programme, X/Open, 1993.

Chapter 1

Introduction

The X/Open Systems Management Problem Statement (see reference **PS**) describes several aspects of the problem, and also surveys some of the existing work in this area. This document, the X/Open Systems Management Reference Model, is intended to describe a framework for providing solutions for the problem.

The Reference Model describes its various components and how they interact. It does not give detailed descriptions of individual components, but addresses their general properties and their means of interaction. The model identifies, but does not define, the required management interfaces.

1.1 Background

As enterprises take increasing advantage of networking technology by interconnecting computing equipment supplied by a variety of vendors, they are finding it both difficult and costly to administer their collective systems.

Traditional systems management technology is neither open nor integrated. Management systems from different vendors do not interoperate and there is little or no integration in the management of different, but related, areas.

A network of heterogeneous systems has either to be managed as a series of sub-groups of systems, each from a single vendor, or end-users have to provide their own integration layer that implements common functionality across the complete set of machines.

Similarly, different aspects of the users' environment, for instance their interaction with the mail and printing systems, are managed using differing interfaces.

As a result, enterprises must employ significant numbers of skilled system administrators to manage the diverse features of these systems. When seen as increased cost of ownership, high system administration costs can act as a deterrent to continued investment in open systems.

1.2 Goals and Objectives

The goals of this Reference Model include the following:

- To identify the crucial aspects of the distributed systems management problem space, especially those that are unique to this topic.
- To establish common terminology.
- To establish a problem-oriented approach to the realisation of distributed systems management solutions.

The Reference Model will describe the components and architecture necessary to build a comprehensive distributed systems management environment. It describes the environment in which distributed systems management can be performed without requiring that particular technologies be used.

Distributed systems management can be implemented using a variety of technologies. Any solution that does not implement all the concepts embodied in the Reference Model is probably deficient in some respect, and any technology that is not capable of implementing the concepts is probably unsuitable as an implementation base. The Reference Model identifies the mapping between the abstract concepts and some technologies that provide suitable implementation bases for the realisation of the Model.

It is a key objective of the X/Open Systems Management Programme to specify a systems management model that will simplify the whole area of system administration, allowing increasingly complex systems to be administered by lower numbers of less highly skilled personnel.

The model is intended to satisfy several high-level system requirements:

Portability
: The ability to make software on managed and managing systems portable in source code form between different vendors' systems by extending the X/Open Common Applications Environment (CAE).

Interoperability
: The ability of management systems, and components of such systems from different vendors, to interwork, thus allowing a network of heterogeneous systems to be managed as a single system.

Transparency
: The ability to administer Resources without the need to be explicitly aware of their location or details of their implementation.

Extensibility
: The ability to extend the scope and capabilities of the management system and to implement different management policies as required. This includes the ability to make use of new communications protocols.

Robustness
: The ability of the management system to provide integrity and the necessary levels of security and reliability.

The following requirements relate to the form of the interfaces that will be provided to access the management functionality:

Ease of Use
: The services and APIs should be simple to use, consistent with the complexity of the underlying functionality.

Consistency
: Wherever appropriate, stylistic inconsistency should be avoided in specification of interfaces.

The management model is intended to describe the vision of the X/Open Systems Management Working Group, to provide for the distributed management of distributed systems. It is the intent of the model to allow fully transparent management, with full interoperability, such that a network of heterogeneous, conforming systems can be managed from any system on the network.

1.3 Relationship to Implementation Technologies

The Reference Model is described in abstract terms, and is intended to be realisable in a variety of technologies. In the appendices, a mapping is provided from the abstract Reference Model to selected technologies. There is currently much industry development work in the area of distributed systems management, and the mapping provided reflects the technologies that are being used.

It is anticipated that the primary vehicle for implementation of the Reference Model will be the Object Management Group's Object Request Broker (ORB) technology. At the time of writing some major issues relating to the practical implementation of ORB based management systems are still to be resolved, particularly those relating to the interoperability of different ORB implementations.

Another significant implementation technology, particularly in the area of Network management, is that embodied by the ISO/CCITT and Internet management protocols, CMIP and SNMP. The X/Open Management Protocols API (XMP) provides a uniform access method to these technologies.

Appendix A and Appendix B describe how the above technologies may be mapped onto the Reference Model. However, this does not imply inherent portability or interoperability between these environments. Appendix C addresses some of the methods that are necessary when managing a hybrid environment, which includes OMG, OSI, and Internet components.

In addition to the above, which represent the anticipated future development of distributed systems management, the Reference Model can also be implemented using currently available technologies. These include those based on existing Remote Procedure Call (RPC) technologies, such as ONC NIS and DCE RPC.

1.4 Relationship to Legacy Systems

In order to fully meet the requirements to provide distributed systems management across the full range of IT systems that make up today's complex information management environments, it is also necessary to integrate the management of both open and legacy (proprietary) systems.

In this context, legacy systems are characterised by their use of management systems based on protocols and interfaces that are not conformant to open standards. The techniques that are described within the Reference Model in order to provide interoperability between the technologies that are expected to be used to implement the Reference Model may also be used to provide interoperability with legacy systems. These techniques allow legacy systems to be integrated into distributed, heterogeneous management systems, however, this integration is limited to interoperability between management systems, and will not provide for portability of management software between open and legacy systems.

1.5 Intended Audience

The Reference Model is intended for the following audience:

- Implementers of systems management frameworks
- Implementers of systems management applications
- IT strategists and decision makers
- Providers of standards and technology
- End-users responsible for the deployment of distributed systems management

Chapter 2

X/Open Systems Management Programme

The X/Open Systems Management Programme is defined in terms of a suite of documents that, taken together, will describe all the components needed to achieve the goals stated in Section 1.2 on page 2.

The first of these documents is the X/Open Systems Management Problem Statement (see reference **PS**). The Problem Statement, published in 1991 as a Snapshot, provides an overview of the problem, and also includes a review of activities current at that time.

The Reference Model builds on the Problem Statement, providing a framework in which the various components of the solution can be identified. The individual components will be defined in subsequent documents.

The current X/Open work program is developing a coherent family of documents that address the various components needed in order to provide an open, portable, interoperable management system. The documents can be divided into a number of groups according to their functionality. These groups are described below.

In the following sections, documents already completed are indicated by an asterisk (*). The descriptions given below are intended for overall guidance only. For more specific information regarding planned time-scales, refer to the Distributed Systems Management section of the X/Open Technical Programme (see reference **XTP**).

2.1 Strategic Documents

The first group of documents provides the framework and strategy that defines the overall approach, and consists of the following documents:

- Systems Management Problem Statement (*): Provides an overview of the problem space and a review of activities in the area of system management.
- Systems Management Reference Model (*): This document.
- Guide to Systems Management: It is envisaged that a tutorial-style document will ultimately be provided which will describe the use of the distributed systems management specifications developed under the systems management work program.

2.2 Managed Object Definitions

The second group is concerned with the definition of Managed Objects. It consists of the following:

- Managed Object Guide (XMOG) (*): Provides a guide to the principles, techniques and processes used in defining Managed Objects.
- Guide to Translating GDMO to XOM (*): Provides a set of rules for translating object definitions based on the ISO Guidelines for the Definition of Managed Objects into the XOM form required by XMP.
- Guide to Translating GDMO to IDL: This document will provide guidance for translating Managed Object definitions defined using ISO GDMO into IDL definitions required for use with OMG CORBA-based technology.
- DMI Contents Package(*): This document provides the necessary definitions of management support objects, based on the OSI Definition of Mangement Information, for use with the XMP API.
- Managed Object Definitions: There will be series of documents that provide the object definitions corresponding to the Resources that need to be managed. In most cases this will be achieved by reference to existing definitions.

2.3 Management API Specifications

The third group will address APIs to Management Services, including communications services:

- Identification of Management Services (XIMS) (*): Identifies the various services required for implementation of distributed management systems.
- Management Protocols API (XMP) (*): Defines programming interfaces to management communications services provided by CMIP and SNMP.
- Common Object Request Broker Architecture (CORBA) (*): This document, developed by the Object Management Group and published as an X/Open specification, is not formally a part of the systems management program. However, it forms an integral part of the systems management APIs and is a key technology in the realisation of many distributed systems management systems.
- Management Services APIs: Interfaces will be defined to provide access to the underlying Management Services provided by the framework.

2.4 Interworking Specifications

The fourth group addresses interoperability issues, namely the protocol profiles necessary to achieve interworking between different implementations of conformant systems.

- Management Protocol Profiles (XMPP) (*): Describes the protocol options to be used in an X/Open System Management implementation. This document makes reference to the International Standardised Profiles (ISPs) relevant to CMIP, and the appropriate RFCs for SNMP.
- ISO/CCITT and Internet Management: Coexistence and Interworking Strategy (*): This document addresses the issues arising from the need to accommodate both OSI and Internet management protocols within a common environment.

2.5 Management Application Specifications

The fifth group of specifications will address specific functional areas corresponding to real end-user requirements, and will provide the definitions necessary to provide portability and interoperability in the development of solutions to those requirements. Functional areas expected to be covered include the following:

- Networked Backup and Restore: This specification will provide the necessary object definitions and interfaces to provide networked backup and restore capability. It will provide a framework for extending backup and restore beyond file systems, allowing other sub-systems to be backed up, and promoting a *plug'n'play* approach to the provision of the various components of a backup and restore system.
- Performance Management: Performance management specifications will provide the low-level data gathering functionality that is needed to allow portable high-level interpretation tools to be developed.
- Accounting Management: As with performance management, it is anticipated that work in this area will be concerned with the low-level data gathering functionality, thus providing access to the information needed by Resource accounting packages for billing and other purposes.
- Software Management: It is is anticipated that work in this area will be based on the work currently being undertaken within the POSIX 1003.7 System Administration working group.
- Printer Management: It is is anticipated that work in this area will be based on the work currently being undertaken within the POSIX 1003.7 System Administration working group.

Chapter 3

Overview

3.1 Foundations

At a fundamental level, XSM recognises that there are Administrators[1], performing ‘‘Management Tasks’’, who exercise some form of control and/or wish to be kept informed about the operation of one or more ‘‘Resources’’ (see Figure 3-1 below). In order to achieve these objectives the XSM Support Environment will need to provide communication between the Management Task and the Resource such that information about the Resource can flow to the Management Task and the Management Task can influence the Resource. In addition to a communications service, the environment will need to provide other services that provide access to other management functionality

The Reference Model does not specify what Management Tasks or Resources exist, and it does not make any restrictions on how Management Tasks and Resources interact, although it does permit a many-to-many interaction. The purpose of XSM is to provide an environment that will permit Management Tasks to be constructed in a way that encourages portability from one conformant system to another and interoperability between heterogeneous systems.

Figure 3-1 Fundamental Components of Systems Management

1. In this document, the term Administrator refers to a person. The term Manager (and its derivatives, such as Management Task) refers to a component of the Reference Model.

XSM is based on the exploitation of several concepts, all of which are required to enable Systems Management to take place. These concepts are:

Management Tasks
: Management Tasks represent the management activities performed by administrators. As such, they are abstract entities which make use of the underlying functionality of the management system in order to achieve the desired action.

Management Tasks do not appear explicitly within the Reference Model. However, they are the essential functionality that the Model exists to support. The Model provides the framework necessary to support the various components that need to exist in order to implement the operations that are represented by Management Tasks.

An example of a Management Task is adding a user, which involves the creation and manipulation of several different Resources within the system.

The definition of Management Tasks is outside the scope of the Reference Model, which is primarily concerned with the underlying functionality required to implement a rich set of Management Tasks.

XSM Support Environment
: The XSM Support Environment consists of the capabilities and interfaces that are necessary in order to support the other components of the Reference Model. These capabilities are provided in the form of General and Management Services.

General Services are the normal range of services available to all applications. They include services such as file and process manipulation, device input/output services, and basic security mechanisms.

Management Services are the common management functionality available to all management applications. They are typically built upon common underlying system services and management specific services. Examples of Management Services include security services, consistency services, collection services, and event services.

Resources
: Resources are the entities with a system or network of systems that require management. Resources can include physical entities (such as printers or routers) as well as logical entities (such as users or groups). Not all Resources require management, and such Resources are beyond the scope of XSM.

These concepts will be realised as concrete implementations using the specifications of XSM as defined in the suite of XSM documents.

3.2 Resource Interactions

The basis of XSM is the manageability of system Resources. The Resources that are to be managed represent the capability of systems to perform functions which, ultimately, are of benefit to users. Some Resources represent capabilities that are of direct benefit to users and, hence, are directly consumed by users. Such Resources present users with a *functional interface* and participate in *functional interactions*. These represent the normal (non-management) activity of the Resource, the reason for which the Resource exists. This is shown in Figure 3-2.

Other Resources exist that aid in the management or consumption of Resources on the system or in the network. Such Resources may present little or no *functional interfaces*, and only take part in *management interactions*.

For a Resource to be managed it must provide a *management interface* and take part in *management interactions* such that it becomes a Managed Resource. In Figure 3-2, therefore, the Resource is acting as a Managed Resource and has a dual nature, one representing its functional interactions and one representing its management interactions.

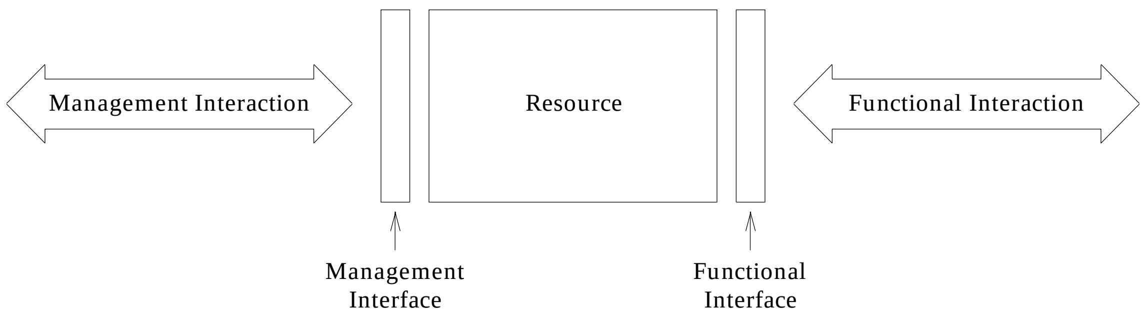

Figure 3-2 Resource Interactions

XSM does not specify what form functional interfaces may take, nor which standards (if any) they will adhere to. There is no implication that methods used to define the management interface should be used to define the functional interface.

The functional and management interfaces are quite separate, the interactions taking place independently of each other. However, the results of one type of interaction may affect the way in which the Resource performs the other interaction. Therefore, a management interaction may affect the functional operation of the Resource and thereby its functional interactions, and *vice versa*. For example, some activity at the functional interface may cause an alarm notification to be sent to a Manager using the management interface, or an operation at the management interface may change the configuration of the Resource causing it to respond differently at the functional interface. The way in which this relationship between management interface and functional interface operates is outside the scope of XSM, being a function of the Resource itself and the way in which its management interface and its interpretation of the management interactions are defined.

Chapter 4

Realising the Model

The abstract concepts described in Chapter 3, are further elaborated upon in this chapter. At this point the precise software components that are required to realise the abstract concepts are described.

The X/Open Systems Management Programme makes use of object-oriented techniques to describe the encapsulation of Resources and the interactions between managed and managing entities.

The X/Open Systems Management Reference Model uses object-oriented techniques in the specification of systems management. These techniques are derived from those used in the OSI Management Model, as well as the Object Management Group Common Object Request Broker Architecture. The use of such techniques for specification does not require an object-oriented implementation, although in many cases there would be benefits in adopting such an implementation.

The benefits of adopting an object-oriented approach are described in Appendix D.

The Reference Model consists of 3 basic components:

Managers	which implement Management Tasks and other composite management functions.
Managed Objects	which encapsulate Resources.
Services	which provide the XSM Support Environment.

For management of Resources to take place it is necessary that management interactions are possible. If that were all that was provided, every Manager would need to know specific details about each Resource and understand all aspects of its operation in terms that were unique to the Resource. So that management of heterogeneous systems is possible, it is necessary that the management view of a Resource achieves isolation from the implementation of that Resource and reflects the need for management of the Resource. The management view should be expressed in terms that enable Managers to perceive Resources as being the same from a management perspective, even when their implementation and functional interface are quite different. This view of the Resource is in terms of Managed Objects, the definition of which encapsulates the management characteristics of the Resource and isolates those characteristics from the implementation of the Resource. It is also necessary that different types of Resources, as well as different implementations of the same Resource, are expressed in the same form.

Figure 4-1 represents the relationship between a Manager and a Resource expressed in terms of Managed Objects that represent the Resource as one or more instances of these objects. The definition of these Managed Objects provides the common semantic knowledge required by both the Manager and the Managed Object implementation which allows the management of the underlying Resource to be performed. Other objects may be defined related to the Resource, indeed the functional interface may be expressed in terms of objects, but this is outside the scope of XSM.

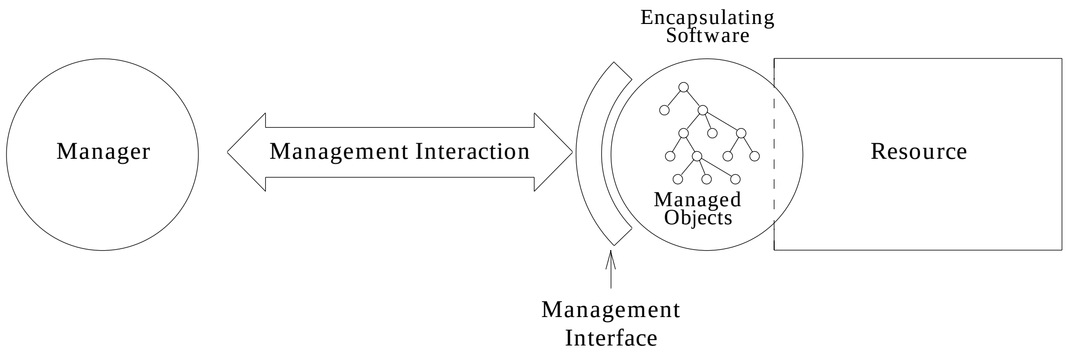

Figure 4-1 Managed Objects

4.1 Managers

A Manager is the initiator of a management interaction. It is a software component that requests some operation to be performed by a managed Resource.

Within an OSI environment, this functionality would be described as the performance of a Manager Role. In general distributed computing terminology, this functionality would be described as a client, making requests of a server.

Managers may provide a user interface which thus form the means of invoking Management Tasks. Managers may invoke other Managers which provide common management functions.

Managers may be invoked by an Administrator, either directly by means of a command line interface (CLI) or a graphical user interface (GUI), or indirectly by a scheduler, programmed to invoke a particular task at a specific time, or upon detection of a specific condition. Managers may also be invoked by other Managers which need to make use of some composite function that this Manager provides.

A Manager invoked by another Manager appears to be the same as a Managed Object. It will take part in a management interaction with the requesting Manager and will perform the requested function. It may in turn initiate further management interactions with other Managers and with other Managed Objects.

This behaviour describes an important aspect of the Reference Model, the concept of ‘‘cascading’’. Although management is often simply described in terms of the interaction between Managers and managed Resources, in reality, there are often multiple layers of management interaction between the original initiator of the management request and the ultimate target Resource(s) that are affected by it. Cascading may be performed for various reasons, such as delegation, policy implementation, or ease of provision of composite management functions. The cascading of Managers illustrates the transient nature of management roles, with a given software entity acting sometimes as a Manager, sometimes as a Managed Object.

4.2 Managed Objects

In order to transform a Resource into a Managed Object it is necessary to *encapsulate* it with software that provides the necessary management interface. This encapsulation may be extremely simple, or it may involve significant complexity. The simple case is that of a Resource which has implemented the necessary management interface directly. No additional encapsulation is necessary in this case.

Within an OSI environment, this functionality would normally be described as an Agent, and would perform an Agent Role in a management interaction. In general distributed computing terms, such a software entity is acting as a server, responding to requests originated by a client.

Any type of Resource may be encapsulated with a management interface. Indeed, some Managed Objects may not correspond to any real Resource within the system, but rather to an abstract element of functionality that is relevant to the management of some other Resource. In this way, Managed Objects can be defined which represent some aggregation of disparate real Resources that neeed to be managed as a coherent whole. This is explored further in the examples in Chapter 5.

The interface between the encapsulating software and the Resource that it is managing may conform to standards, or they may be entirely specific to a particular implementation of a Resource. One of the major purposes of the encapsulating software is to provide a standard management interface to diverse implementations of common Resources. If different implementations provide a standard interface for use by the encapsulating software, then it is possible to envisage the development of a portable implementation of the encapsulating software.

As has been described in previous sections, a Managed Object may represent a ‘‘real’’ Resource, (e.g. a file system), a ‘‘logical’’ Resource, (e.g. a user), or a unit of management functionality, providing the capability of cascading Managers.

4.3 Services

Services exist to provide the common facilities that must be provided by the XSM Support Environment in order to support distributed systems applications. Services can be divided into 3 major classes:

- General Services,
- Management Services, and
- Application Services.

This classification derives from the relationship of a specific service to the specific problem space being addressed.

General services are characterised as being of use to a wide range of different problem areas.

Management Services are common facilities which have been specialised for XSM distributed management. Areas of specialisation might include: policies for more centralised control of security, policies for configuring and distributing applications, and the ability to control the location of objects.

Application Services are services that are specific to some particular functional area within the overall management problem space. While these services are not of general use to a wide range of management applications, they provide common services to implementations addressing that particular area. An example might be a catalogue service provided for the use of multiple backup and restore applications.

A fuller discussion of management and general services is given in the X/Open Systems Management: Identification of Management Services Snapshot (see reference **XIMS**). This section summarises some of the services that are needed by distributed management systems.

4.3.1 Communications Service

XSM specifies a means by which management interactions can take place, namely the Communications Service. This service provides a defined interface which is specified as part of XSM. The Communications Service provides access to communications mechanisms. The Communications Service provides:

- confirmed and non-confirmed services, so that a Manager can optionally receive notice that a request has been accepted by a Managed Object;
- encoding of object requests in a form understood by XSM-conforming end-systems, as defined by vendor agreed profiles (for example, the CMIS Package in XMP);
- provision of security by the authentication of both the originating Manager and destination Managed Object in any communications;
- standardised ways of describing operations on all Managed Objects and receiving notifications from them;
- selective location transparency, so that the Manager does not need to be aware of the location of the Managed Object. The Manager is not precluded from determining an object's location.

The Communications Service may use a local transport mechanism for communications within the same system.

4.3.2 Persistent Storage Service

XSM defines an interface to a Data Store which can be used to store information about Managed Objects. This interface reflects the object structure and naming of these objects. The Data Store itself may be implemented as a conceptual repository, thus supporting implementations based upon different vendor database systems.

4.3.3 Security Service

XSM requires a security model which addresses several components of secure access to Managed Objects including:

- Authentication schemes which support bilateral authentication of Manager and Managed Object during object request execution, and which support principal identities for all types of Managers and Managed Objects;
- Authorisation schemes which support access control to the Managed Objects, including management service objects, and managed Resource objects;
- Establishing valid process identities for Managed Object implementations so that authorisation schemes provided by object implementation underlying subsystems (such as file systems and data Managers) operate correctly;
- Delegation schemes which support valid access checking as high-level object requests are broken into a series of lower-level object requests, some of which may be to remote systems. Delegation schemes solve the cascading authentication problem in distributed object systems;
- Trust schemes which support the above delegation schemes.

These components of security are used to construct application specific security policies for the management applications, as well as site-specific security policies defined by administrators.

4.3.4 Consistency Services

XSM will provide services to ensure the consistency of the Managed Object state in the following cases:

- Multiple simultaneous access to a Managed Object by several Managers. As there are several methods for achieving this form of consistency, not all possible methods will be covered. These consistency mechanisms can be built upon mechanisms such as object locking.
- Operations on multiple objects by a Manager acting as a single, consistent atomic operation. Typically, a Management Task will require that requests are made to more than one Managed Object, some of which may be located on remote systems. To ensure consistency among Managed Objects, it is necessary to support mechanisms which enable all related operations to complete successfully, or none at all. Various techniques may be used to address this problem, including transaction processing techniques and retrying failed operations. The solution adopted is both an implementation and a policy issue, and is currently beyond the scope of XSM. It may be possible to use transaction processing services, when such services become available.

Two other cases are not considered applicable to the XSM consistency services. The first is guaranteeing consistency of the state within a single Managed Object, both in lieu of an object request failure and modification of the managed Resource outside the XSM reference model implementations. It is left to the Managed Object implementation to address these consistency issues. The second case involves managed Resources interacting in ways that are not reflected in the Managed Object definitions. This should be considered a deficiency in the Managed Object definition.

4.3.5 Collection Services

XSM provides services to enable the establishment of relationships among Managed Objects, thus supporting the ability for management applications to operate on a set of Managed Objects (such capability is important to implement scalable management applications and tasks, as well as providing a mechanism for organising Managed Objects for searching, filtering, and browsing). Some relationships of interest are containment, connectivity, and activity.

4.3.6 Selection Services

Administrators require a mechanism to determine the set of Managed Objects that are to be acted upon by a Management Task. The techniques for scoping, filtering and finding provide mechanisms for supporting such services. Scoping and filtering rules are expressed as one or more conditions based on the relationships between Managed Objects and/or the attributes of Managed Objects.

4.3.7 Event Services

Many Managed Objects have the ability to generate asynchronous event notifications associated with the encapsulated managed Resource. XSM addresses services which support the generation, registration, filtering, and forwarding of such event notifications to management applications and other management objects. The services support the ability for a management application or task to explicitly specify which event notifications should be forwarded to it. Such specification can be based upon one or more conditions relating to the values of the parameters in the notification.

4.3.8 Naming Service

The Naming Service provides a common interface to underlying naming servers (such as X.500, DCE CDS, and ONC/NIS+). This service encapsulates a federated naming model, providing operations which are naming syntax independent.

4.4 Basic Reference Model

The overall structure of the Reference Model is shown in Figure 4-2.

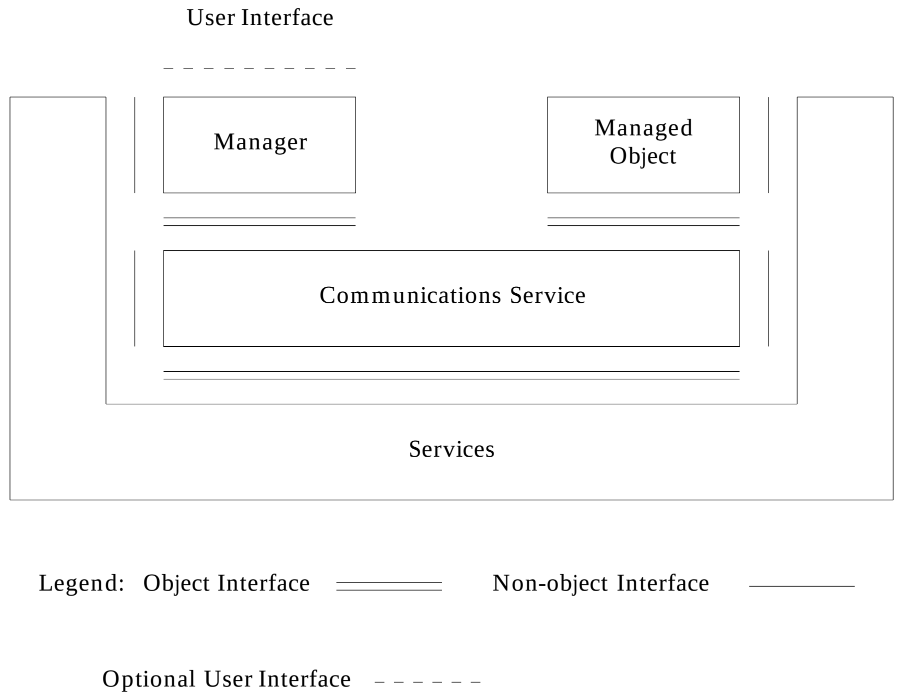

Figure 4-2 Basic Reference Model

The Reference Model illustrates the relationship between the fundamental components.

Within this model, the Communication Service has been specifically singled out for special treatment as it forms the core function of conveying management requests and their responses between Managers and Managed Objects. It may also provide access to Management Services, especially those that are defined using object-oriented techniques.

It is important to note that there is no direct access between the Manager and the Managed Object, except via the Communications Service. The Communication Service is defined as providing all the functionality necessary to provide transparent communications between Managers and Managed Objects. In the case where the Manager and the Managed Object are on the same system, then the Communication Service may make use of efficient local transport mechanisms, such as IPC.

Two forms of interface are present in the model. The primary interface to the Communication Service is presented as an object-oriented interface in which requests and responses are expressed in object terms. Non-object interfaces are also shown as it is expected that many services will be expressed in terms of traditional, functional interfaces. This is especially true of pre-existing general services which are not specific to management.

Chapter 5

Examples

This chapter contains examples of applying the Reference Model to real-world scenarios.

5.1 Backup and Restore

An administrator typically has a requirement to back up the system — which usually means the ability to save sufficient information about the current state of the system to enable all or selected parts of that state to be restored at some later time. It goes without saying that this should be achievable with the minimum of effort!

Most often in current practice, the backup is of files or file systems. However, the concept of backing up an application; e.g., a database, is also quite common. The notion of backing up a user is more abstract but is one that is not completely unknown in current practice.

Let us look at the major components that we might expect to find in a typical backup system. The most obvious component is some form of user interface which an administrator can use to initiate the backup or restoration processes. This interface would allow the user to select the data to be backed up or restored. Such a selection might be on the basis of date or name, for example. In order for a backup process to be useful there must be some component of the system which provides the repository for the backed-up data. Frequently this will be some for of magnetic tape or perhaps optical disk. Of course, there would be no point in having a backup system at all without the fundamental component of some form of data store to be backed up! Although at first sight it may appear that this is all that is required in a backup system, anyone who has had experience with managing the backup of large amounts of data will recognise the need for one other component: an inventory. Keeping track of which data is backed up to where is a significant administrative burden, so this task needs to be substantially automated in any reasonably featured backup system.

So, there are four basic components of our backup system:

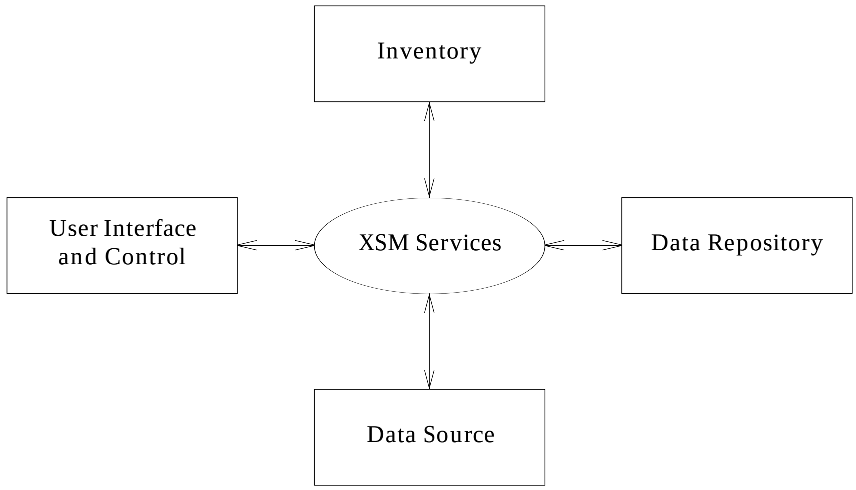

Figure 5-1 Possible Components of a Backup and Restore Service

Of these components, the user interface is the place from where control originates. It is the backup application as far as the administrator sees it. The inventory is a necessary component for performing the overall backup task and it will obviously be used in the task of restoring data. It would also be useful for providing information for browsing. Thus while an administrator would perform separate backup and restore operations, perhaps using very different applications, the services of the inventory would be used by both. If we consider the data source, it would seem to be beneficial if all sources of data — files, databases, users or whatever — could be manipulated in similar ways by backup applications. It is probably meaningful to say ‘‘back up all Resources older than 3 weeks’’ whether the Resources in question are files, database records or user account details. Finally, it would be useful for an administrator to be shielded from unnecessary details as to whether the repository for backup data is half-inch magnetic tape, QIC-150 cartridge, or CD-WORM cassettes. A natural consequence of these arrangements is that it becomes desirable to encapsulate the details of inventory, source, and repository within well-defined interfaces.

So, how does this relate the management reference model as shown in Figure 5-1? In the backup system we have the following:

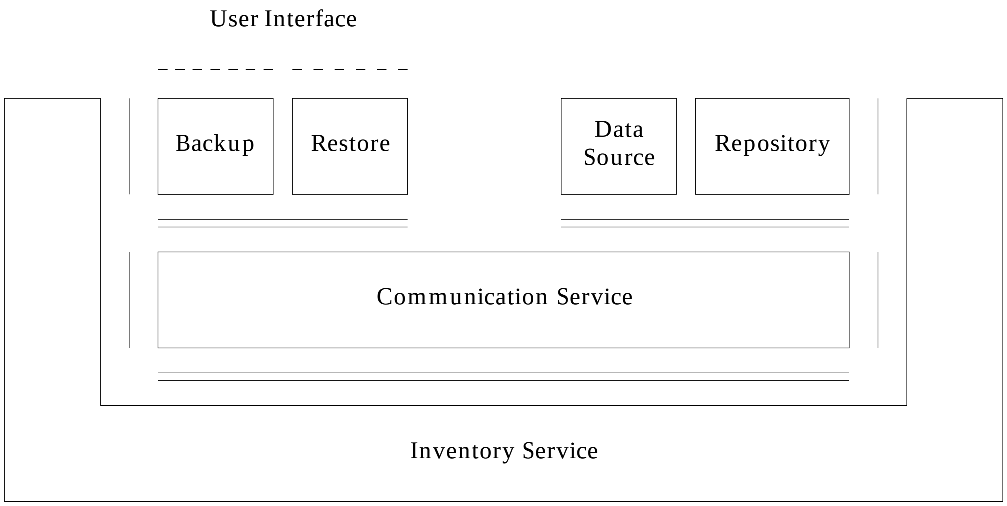

Figure 5-2 Application of the Reference Model

Within the context of the Reference Model, the backup and restore interfaces are Managers, a well-defined and encapsulated inventory service is an Application Service, and the data source and repository are Managed Objects.

The representation of the Inventory Service in the diagram illustrates the use of the Inventory Service. It is probable that such a service would also require to be managed, in which case the Inventory Service would appear in the Reference Model as a Managed Object. Such duality of purpose is well supported by the object-oriented techniques used to specify the Reference Model.

5.2 Modelling Resources as Managed Objects

There is considerable power in being able to define new Managed Objects to represent completely new Resources that represent the synthesis of Resources represented by other Managed Objects, thus providing different aggregations perhaps more suited to the needs of human users.

For example, consider a print room with the following equipment: two intelligent printers, a microcomputer serving as a print controller, all connected to a local area network drops.

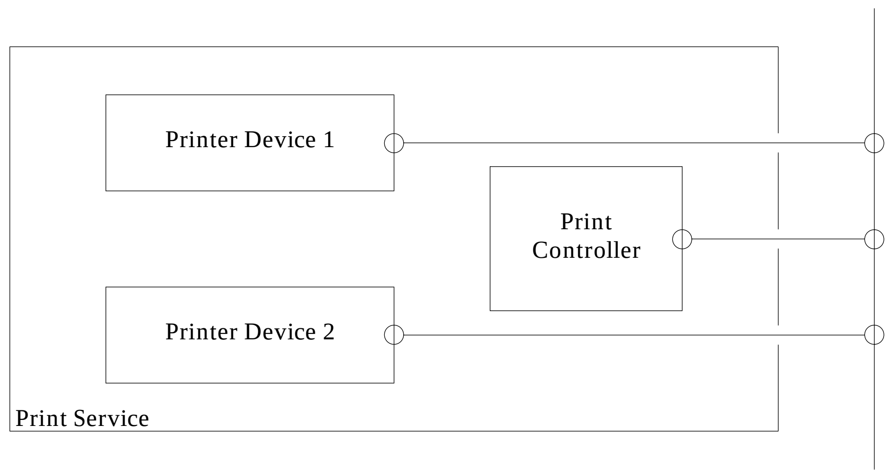

Figure 5-3 Components of a Print Room Service

The printers provide management access to a few attributes such as number of print jobs processed (since reset), number of queued jobs, names and sizes of the queued jobs and the currently active job. The print controller might offer a LAN service that accepts print requests and then queues the print jobs to the printers according to certain policies.

Ignoring the network for the moment, it is not difficult to imagine each of the printers and the controller being encapsulated as Managed Objects for the purpose of management. However, one may consider a new virtual entity representing the print service as implemented by the devices in the print room. Such an object is virtual in that there is no single object that is implemented in the network. However, from a Manager's perspective there is a print service and there is an obvious desire to be able to manage this service as though it existed as a single Managed Object. Such an aggregation can be defined as a new Managed Object with its own methods. The new methods would be implemented by invoking the appropriate methods against the real Resources, and synthesize the results according to the aggregation representing the complete print service.

Note the positioning of the Managed Object in the basic reference model in Figure 4-2. The example print service Managed Object would be accessed by the Manager using the communications service and other services.

Chapter 6

Meeting the Goals

This chapter reviews the way in which the Systems Management Reference Model addresses the goals and requirements defined in Section 1.2.

6.1 Portability

Portability is the ability to create software that is portable in source code form between systems from different vendors. The provision of XSM-conformant interfaces is the means by which this is achieved.

6.1.1 Scope of Portability

The scope of XSM interfaces is limited to management aspects. Thus XSM is a component of the wider environment required to achieve portability. Other components are provided by the X/Open Common Applications Environment (CAE).

6.1.2 Extent of Portability

Resources	Resources may or may not be portable. XSM does not facilitate the portability of Resources.
Managed Objects	If the interface between the Managed Object and the Resource conforms to some *de jure* or *de facto* standard, then a Managed Object using such an interface will be portable to systems on which the Resource provides such an interface. Where a Resource is portable, but does not provide a standard interface for a Managed Object, then the combination of Resource and Managed Object is portable.
Managers	XSM provides interfaces that enhance the portability of Managers. Use of non-standard interfaces by Managers will, of course, render them less portable.

6.1.3 Managed Objects

The Resources that a Manager can manage and a Managed Object can support are determined by the software implementing those roles. XSM also specifies that there are certain Managed Objects which can always be assumed to be present in an XSM-conformant environment. The existence of these objects is an aid to software portability.

6.2 Interoperability

Interoperability is the ability of systems and components from different vendors to share and exchange management information. This extends beyond connectivity, since it requires a common understanding of the significance of the information.

6.2.1 Communications

XSM, in defining a Communications Service to be used for management purposes, provides the means for management information to be exchanged between systems. This provides the basic capability for a Manager to interact with a Managed Object.

6.2.2 Management Interactions

XSM, in defining (or making reference to) particular standards for management interactions, will greatly improve the interaction between Managers and Resources.

6.2.3 Managed Object Definition

XSM provides guidance on how Resources should be expressed as Managed Objects, and further specifies how the definition of Managed Objects is to be stated. This therefore provides a means by which a Manager on one system can understand the definition of a particular Managed Object on any given system.

6.2.4 Managed Object Compatibility

Object-oriented techniques used for defining Managed Objects allow the refinement of Managed Objects whilst still providing their management according to their original definition. This aids interoperability, since it removes the need for a Manager to always understand the most up-to-date Managed Object definition for any given Resource.

6.3 Transparency

The use of a model based on object-oriented technology, and in which managed Resources are represented by Managed Objects provides considerable support for transparency. Several aspects of transparency are summarised below:

Access Transparency
: Access transparency enables interworking across heterogeneous computer architectures and programming languages.

Failure Transparency
: Failure transparency masks the failure and possible recovery of objects.

Federation Transparency
: Federation transparency hides the boundaries between different naming schemes and domains.

Location Transparency
: Location transparency provides the capability for Managers to be able manage a Resource without needing to be aware of its location.

Migration Transparency
: Migration transparency shields a Manager from the fact that a Resource may migrate from one node to another within the distributed system.

Replication Transparency
: Replication transparency hides the fact that what appears to be a single Resource may in fact be replicated for reasons of performance or redundancy.

Transaction Transparency
: Transaction transparency hides the implementation of transactional semantics.

These many aspects of transparency all serve to simplify the task of developing management applications. Particularly in the domain of management, it is important that transparency be selective, as there will inevitably be occasions when it is necessary to be aware of the precise location or implementation of a Resource. These occasions will normally arise at times of serious failure of aspects of the infrastructure. An example is the failure of the name service. The location of the name server must be known in order to re-start the service.

6.4 Extensibility

Extensibility is the ability to extend the management system and to customise it to implement differing management policies. The key areas of this are the introduction of new Resources to manage and the introduction of new ways to manage them. In addition, new services and support for new protocols may be provided.

6.4.1 Managed Objects

The way in which Resources are specified as Managed Objects is one key aspect of the ability to extend the manageability of the Resource. From the rules laid down for the definition of Managed Objects, their definition can be extended by the definition of further objects that are refinements of the original objects. These new objects may be capable of being used as if they conformed to the original definition, this capability being dependent on the implementation of the Managed Object or the managing software. In this way Managers that know of the original definition would continue to be able to manage Resources even when the definition of the Managed Objects has been enhanced to allow other Managers to manage the Resource in some other way.

6.4.2 Composite Management Functions

The Reference Model provides a mechanism by which it is possible to extend the capabilities of the management system. Extension in this way need not be connected with the definition of additional Managed Objects, but can represent a different way of using existing Managed Objects. This could be used, for example, to provide varying styles of interface that might facilitate the porting of management software developed for other operating environments.

6.4.3 Layers of Management

XSM provides that Managers can make use not only of the services provided as part of an XSM-compliant environment, but also the facilities provided by other Managers. Such a capability is, of course, dependent on the ability of a particular Manager to provide facilities in a way that is suitable for such use, but is fully catered for in XSM. It is hence possible to envisage the provision of management capability with increasing sophistication, with users free to choose the level at which they wish to manage their systems. At the same time they will be able, at a later date, to use more sophisticated management as it becomes available or the need for it is appreciated.

6.4.4 Variety of Managers

The availability of Managers that can manage Resources modelled as Managed Objects is not limited by XSM but only by the ingenuity of suppliers to provide such Managers. The way in which an Administrator wishes to see and identify Resources is dependent on the perception of that Administrator, both of the Resource and their role. Applications will become available that conform to these perceptions, enabling particular management policies to be implemented.

6.4.5 Proliferation of Objects

The ability of a Manager to manage a multiplicity of objects is not inherently constrained by XSM. This applies both to the number of classes that can be managed and also the number of instances of Managed Objects. With respect to the proliferation of classes, it should be noted that the addition of new classes does not mean that existing Managers will no longer be able to manage an object because its Managed Object model has been enhanced. Location transparency provides a means by which a Manager can be assisted in managing large numbers of instances of Managed Objects, since location information does not need to be obtained and retained by the Manager.

6.5 Robustness

Robustness is the ability of the management system to provide the necessary levels of security and reliability.

6.5.1 Security

XSM will address the provision of security in the relationship between the Manager and the Managed Object, by the provision of suitable services. This ensures that the Manager can be assured that the Managed Object with which it is communicating has the identity it claims to have, and that the communications with it cannot be corrupted. Likewise, a Managed Object can ensure that it only accepts requests from a Manager who has the authority to make such a request, and that it will supply information only to Managers that have the appropriate level of authority.

Hence, provision of authorisation and authentication services for the Manager and Managed Object comes within the scope of XSM. However, ensuring that the Manager and Managed Object use the security functions (mandatory security) is not covered by XSM, being the province of local procedures and/or other standards.

6.5.2 Consistency

As has been stated, a Managed Object may interact with many different Managers and XSM does not define how, where several Managers manage a single Managed Object, consistency of the Managed Object as viewed by a particular Manager is achieved. The interaction between two or more Managers which would be necessary to ensure consistency, is outside the scope of XSM.

6.5.3 Reliability

XSM provides for both confirmed and unconfirmed management interactions. When confirmed interactions are used, the initiator can be informed of the status of the requests it has initiated. Using this facility, Managers and Managed Objects can be assured of the success of operations they have requested, so improving the reliability of the management system.

OMG Mapping

A.1 OMG Object Model

The Object Management Group (OMG) is a non-profit international trade association formed to promote new interoperable software solutions to reduce the cost of software development. The OMG has defined an object reference model and architectural framework with supporting detailed interface specifications. The object model and architecture is defined in the Object Management Architecture Guide (see reference **OMAG**) and the OMG Object Model (see reference **OMGOM**) and the interface specifications are defined in the Common Object Request Broker Architecture[2] (see reference **CORBA**).

The OMG object reference model identifies and characterises the components, interfaces, and protocols that compose OMG's object management architecture (OMA). The reference model addresses how objects make and receive requests and responses, the basic operations that must be provided for every object, and the interfaces that provide common facilities useful in many applications. The OMA supports the object architecture described in Figure A-1, defining an infrastructure with an object request broker (OMG's implementation of the Communications Service) and object services.

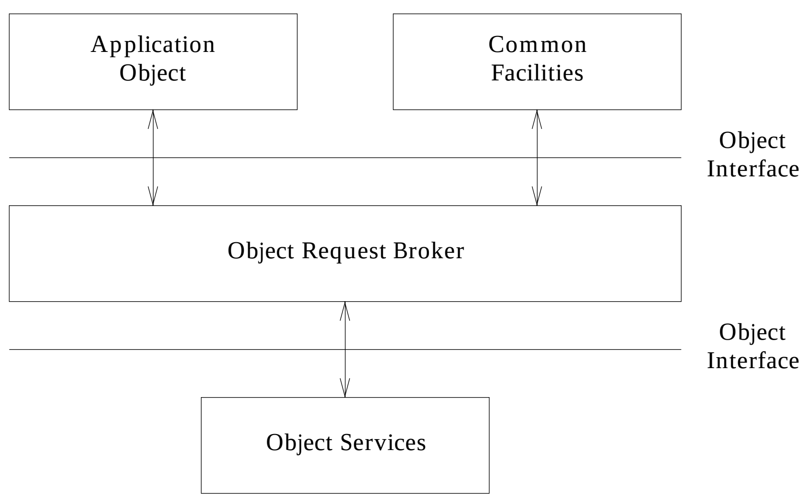

Figure A-1 OMG Object Model

---

2. The description in this Appendix is based on the stated direction of CORBA 2.0 and OMG Object Services which are still in the process of definition.

The three categories of objects (Object Services, Common Facilities, Application Objects) reflect a partitioning in terms of functions from most basic and common to application specific. Note that objects can issue as well as process requests (act as clients as well as servers). Thus application objects can provide services for other application objects, can access common facilities and object services objects; common facilities can use object services as building blocks, and so forth.

The OMG reference model and architecture defines the following major components:

Object Request Broker

The ORB provides the mechanisms by which objects transparently make and receive requests and responses. It is intended to provide interoperability between applications on different machines in heterogeneous distributed environments. An object request (and its associated response) is the fundamental interaction mechanism. A request names an object, an operation and includes zero or more parameter values, any of which may be object handles identifying specific objects. The ORB delivers the request to the appropriate object implementation server and causes a method representing the operation to be executed.

Object Services

Object services provides basic operations for the logical modeling and physical storage of objects. It defines a set of intrinsic or root operations that all classes of objects should implement or inherit. Object services operations are made available through the ORB (that is, ORB compliant interfaces are defined for each object service). The operations supplied by object services are typically used as the building blocks for extended functionality provided by Common Facilities. The operations that object services can provide include life cycle, naming, persistence, event, security, relationships, transactions, and concurrency control.

Common Facilities

Common Facilities provide optional, extended services that are useful for many applications. They are made available through ORB compliant object interfaces. Their purpose is to reduce the effort needed to build OMG compliant applications through reusability. Examples of common facilities include cataloguing and browsing objects, error reporting, help facilities, object querying facilities, and user profiles.

Application Objects

Application objects correspond to the traditional notion of an application; that is, individual related sets of functionality that are implemented using the OMG architecture. Such an application consists of a set of interworking OMG compliant objects. These objects communicate using the ORB and make use of the objects comprising the Common Facilities and Object Services.

A.2 OMG Object Request Broker

The OMG Object Request Broker (ORB) is that component of the object management architecture which is responsible for accepting a client object request, finding the object implementation for the request, preparing the object implementation to receive the request, and communicating the data making up the request and response. Figure A-2 shows the architecture and components of the ORB.

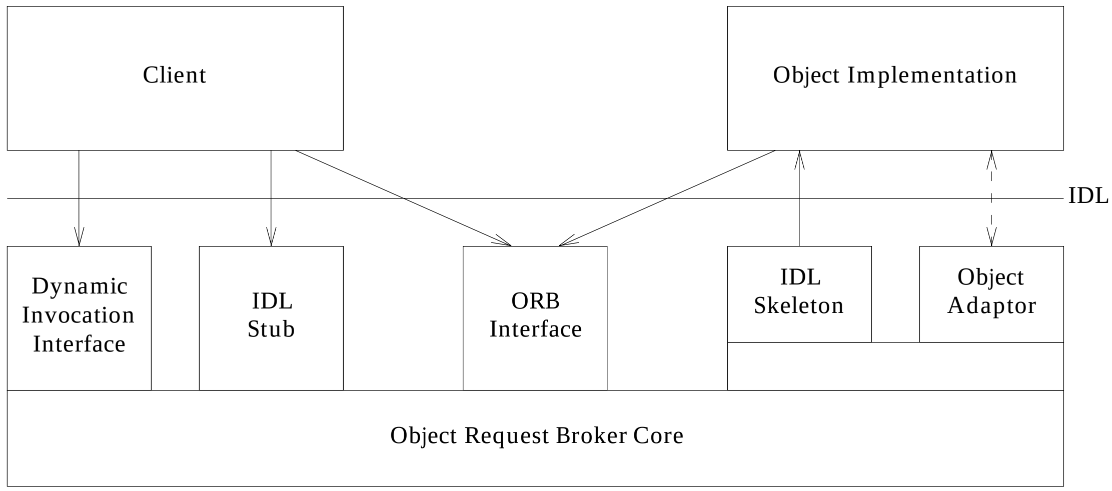

Figure A-2 OMG Object Request Broker

The ORB is composed of the following components:

ORB Core
: The ORB Core provides the basic representation of objects and communications of requests. It provides transparent object location for the client making the request given an object reference consistent with the ORB implementation.

Object References
: An Object Reference is the information needed to specify an object within an ORB. Clients and object implementations are insulated from object reference representation through the use of an opaque object referent consistent with the client or implementation language mapping. An object reference is an object handle in the OMG model.

IDL Interface Definition Language
: IDL defines the types of objects by specifying their interfaces in terms of data types, operations, parameters, and exceptions. It describes a conceptual framework for executing operations on objects. Most ORB implementations supply IDL compilers, which generate the stub routines, skeleton routines, and argument encoding and decoding routines needed to execute requests on the object interface. These routines are expressed in a specific programming language, or language binding.

Programming Language Binding
Different programming languages will access CORBA objects in different ways. CORBA defines a language mapping for each supported programming language, which includes language-specific data types and procedural interfaces to access objects through the ORB.

Client Stubs
Client stubs are programs usually generated by IDL compilers with a specific language binding and are used by client programs to make object requests to the ORB. Client stubs are private to a particular ORB implementation and specific to a particular interface definition.

Dynamic Invocation Interface
The dynamic invocation interface (DII) is an interface to the ORB that allows the dynamic construction of object invocations. The client builds up the object request by specifying the object to be invoked, the operation to be performed, and the set of parameters for the operation. The DII serves as an alternative client interface to client stubs.

Implementation Skeleton
Skeletons are programs usually generated by IDL compilers with a specific language binding and are used by object adaptors to make *up-calls* to the object implementation in the object server. That is, the object implementation writes routines to conform to the skeleton interfaces and the ORB calls them through the skeleton.

Object Adaptors
The object adaptor is the primary way object implementations access services provided by the ORB. A few object adaptors will be implemented that are appropriate for specific kinds of objects. Services provided through an object adaptor includes generation and interpretation of object references, object invocations, object and implementation activation and deactivation, mapping object references to implementations, and registration of implementations.

ORB Interface
The ORB interface provides a few operations common for all ORB implementations, and are thus implemented directly by the ORB core.

Interface Repository
The Interface Repository is an ORB service that provides IDL information about OMG compliant objects in a form available at run time. The interface repository may be used by the ORB to perform requests, or by a browser application to form object requests dynamically.

Implementation Repository
The Implementation Repository contains information that allows the ORB to locate and activate implementations of objects. Installation of object implementations and control of policies related to the activation and execution of object implementations are typically done through operations on the implementation repository.

Figure A-3 shows the relationship between the ORB components used to create client and object implementations.

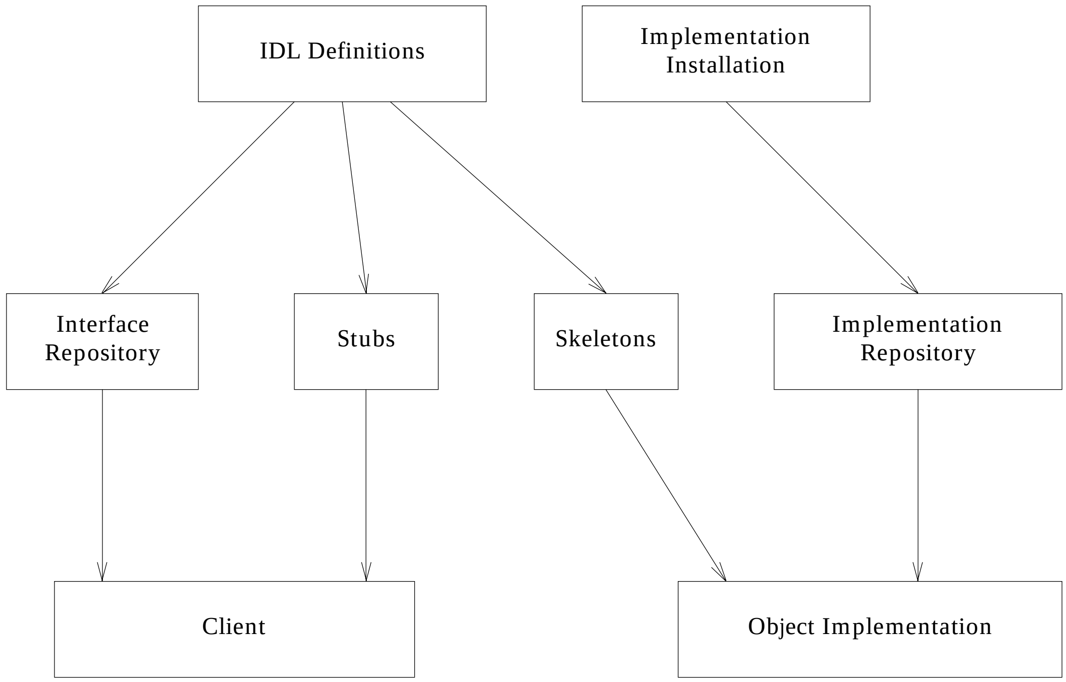

Figure A-3 OMG Interface and Implementation Repositories

A.3 Realisation of the Reference Model

The reference model supports distribution of management functionality and objects. The object request mechanism provided by the ORB's communications service invokes operations on objects independently of their location on the systems within the network. Information associated with an object reference is used by the ORB to determine the system upon which the object implementation is to be executed. ORB process management mechanisms can automatically instantiate an appropriate server process on that system and call the activation functions needed to make the object state available. Object implementation code is then called within the server process to respond to the object request (the exact mechanism for calling a method depends upon the IDL language binding used to implement the object implementation code).

Because a single object request mechanism is used by the ORB for both ''local'' and ''remote'' request invocations for all ORB compliant objects, the management application can be distributed at many levels. For example, the application client code and task-oriented management functions could be located on the client's system, while the Managed Objects are located on a remote system. Implementations of the Management Services and object services could be executed on yet other systems. It is possible that the task-oriented management function objects can be executed on systems remote from the management application's client code, thus supporting high level management functionality "closer to" the managed Resources. Figure A-4 shows the distributed management architecture using the OMG based system management reference model.

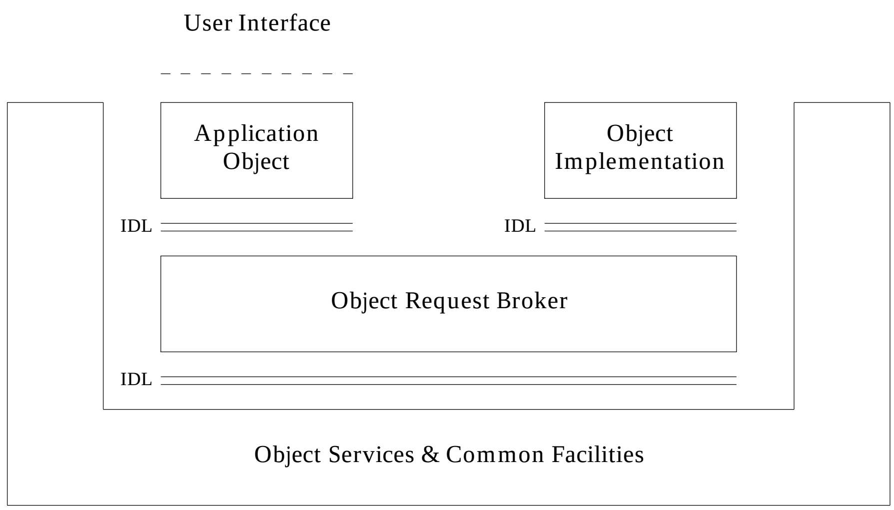

Figure A-4 OMG Mapping to the Reference Model

ISO/CCITT and Internet Management Mapping

This Appendix describes the mapping between the Reference Model and the ISO / CCITT and Internet Management models.

As part of XSM, X / Open has defined a Management Protocol API (XMP) (see reference **XMP**) which provide consistent access to both the ISO/CCITT and Internet Management Services.

B.1 ISO/CCITT Management

The primary goal of ISO / CCITT Management is to provide the capability to manage networks implemented using the OSI protocol specifications. In addition, it is also intended to be extensible to allow management of and interaction with a wide range of non-OSI Resources and management systems.

OSI Management is described in a large number of standards, including the OSI Management Framework, ISO / IEC 7498-4, the Systems Management Overview, ISO / IEC 10040, and the Management Information Model, ISO / IEC 10165-1. Managed Objects are defined using the Guidelines for the Definition of Managed Objects (GDMO), ISO / IEC 10165-4. The OSI management protocol is defined in the Common Management Information Service (CMIS) Definition, ISO / IEC 9595, and the Common Management Information Protocol (CMIP), ISO / IEC 9596-1.

B.1.1 Basic Management Communication

The basic model for OSI management communications is shown in Figure B-1.

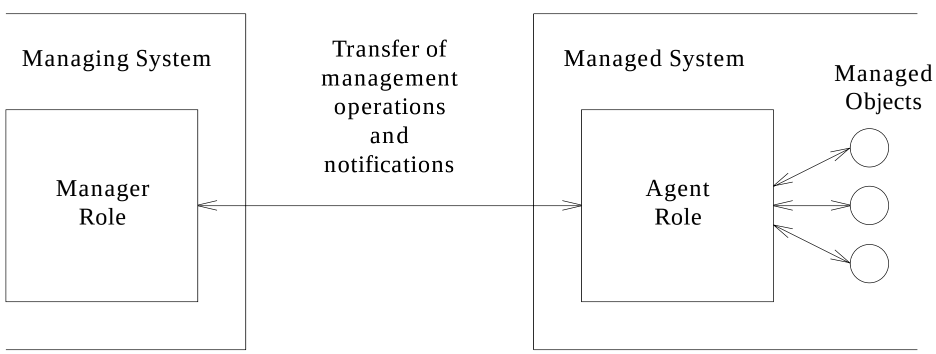

Figure B-1 Basic Model for Management Communications

The figure shows a Manager in a managing system communicating with a managed system containing a number of Managed Objects. The managed system contains an agent, which is responsible for implementing the functionality needed to effect the operations on the Managed Objects requested by the Manager and communicated by the protocol.

Information flowing between the components of the system is divided into two categories, management operations performed on the Managed Objects, and operation results and unsolicited notifications originated by the Managed Objects.

The agent process provides access to the Managed Objects. It may also be responsible for the local logging and distribution of event reports that are generated when particular events occur in the managed system.

Information is transferred using the CMIP protocol, which is an OSI application layer protocol specifically intended for the purpose of transferring information concerning Managed Objects. CMIP provides the CMIS service, which includes the following:

- management operation requests from managing systems to managed systems,
- results and confirmations arising from management operation requests, and
- notifications, in the form of event reports, from managed systems to managing systems, and confirmations returned by the managing system, where appropriate.

The OSI management model is defined in terms of a single management interaction. However, a system may act as a managed system in one interaction and as a managing system in another. Thus an agent may in its turn act as a Manager to other agents, thus providing the "cascading" capability described in the Reference Model.

B.1.2 System Management Functions

In additions to the elements identified so far, there is a further set of specifications called Systems Management Functions (SMF). These specifications are intended to define common facilities that can be applied to particular Managed Objects corresponding to different Resources. The SMFs include the following:

- mechanisms for controlling access to Managed Objects
- mechanisms for controlling the distribution of events
- common formats for reporting alarms
- common formats for reporting status
- mechanisms for invoking and controlling remote test execution

The Systems Management Functions are defined in a multipart standard, ISO/IEC 10164. In addition, a collection of generic definitions is contained in the Definition of Management Information (DMI), ISO/IEC 10165-2.

B.1.3 Realisation of the Reference Model

The OSI management model is defined in terms of the interaction between the managing and the managed system. It provides for management communications between the software entities on both systems and defines a set of common management functions that provide Management Services.

The components of the OSI management model provide all the elements described in the Reference Model. They allow for the implementation of the necessary functionality to be provided at the most appropriate place in the management system, and specifically provide for the "cascading" of management to allow composite management functions to be implemented. Figure B-2 shows the mapping of the OSI components onto the Reference Model.

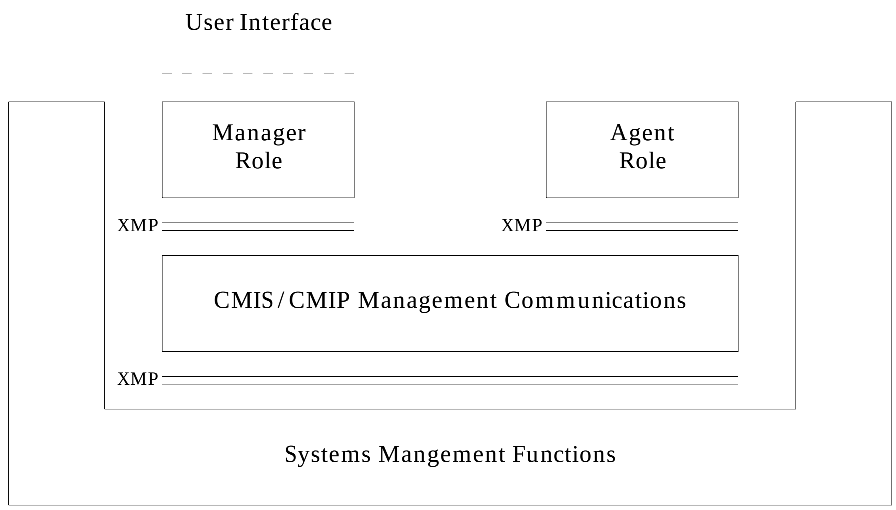

Figure B-2 OSI Mapping to the Reference Model

B.2 Internet Management

The primary goal of Internet Management is to provide the capability to manage TCP/IP-based networks (that is, ‘‘internets’’). Over time, Internet Management has also come to be used for management of and interaction with a wide range of non-network resouces and management systems.

Internet Management is described in a large number of Request For Comments (RFCs) published by the Internet Engineering Task Force (IETF). Two versions of the Internet Management framework have been defined.

- Version 1 of the Simple Network Management Protocol (SNMPv1) is defined by RFC 1157. The corresponding Structure of Management Information is defined by RFC 1155, and the Concise MIB format for defining objects is defined by RFC 1212.
- Version 2 of the Simple Network Management Protocol (SNMPv2) is defined by RFC 1448. An overview of the SNMPv2 Framework is defined by RFC 1441. The corresponding Structure of Management Information is defined by RFC 1442, and includes an extended format for defining objects.

B.2.1 Basic Management Communication

The basic model for Internet management communications is shown in Figure B-3.

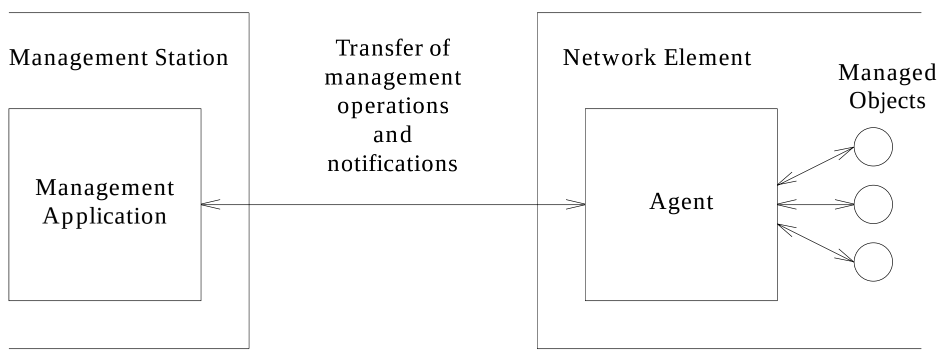

Figure B-3 Basic Model for Management Communications

The figure shows a management application in a management station communicating with a network element (sometimes also called a managed device) containing a number of Managed Objects collected together in a Management Information Base (MIB). The network element contains an agent, which has access to Managed Objects as needed to carry out requests made by the management station and communicated by the protocol.

Information flowing between the components of the system is divided into two categories, management operations performed on the Managed Objects, and operation results and unsolicited notifications originated by the Managed Objects.

The agent provides access to the Managed Objects. It may also be also responsible for detection of unsolicited notifications that are generated when significant events occur in the network element.

Information is transferred using the SNMP protocol, which is an application layer protocol specifically intended for the purpose of transferring information concerning Managed Objects. Internet Management RFCs do not specify an explicit service interface for SNMP. However, SNMP defines protocol data units which support operations and notifications that are conceptually very similar to those described previously for ISO / CCITT management. RFC 1449 further defines how SNMPv2 protocols can be used over a variety of transports, most often the connectionless UDP / IP. Finally, RFC 1452 defines methods for coexistence between versions 1 and 2 of Internet Management.

The Internet management model is defined in terms of a single management interaction. However, an SNMPv2 entity may act as a management station in one interaction and as an agent in another. Thus an SNMPv2 agent may in its turn act as a management station to other agents, providing the *cascading* capability described in the Reference Model. The Administrative Model defined by RFC 1445 and the Manager-to-Manager MIB defined by RFC 1451 describe this mode of operation for SNMPv2. No equivalent RFCs exist for SNMPv1.

B.2.2 Internet Management Security

In additions to the elements identified so far, there is a further set of RFCs which collectively define security services for SNMPv2. These RFCs include the following:

- an administrative model (RFC 1445),
- security protocols for authentication and privacy (RFC 1446), and
- mechanisms for management station control over security-related properties such as SNMPv2 parties, contexts, and access control lists (RFC 1447).

Except for the latter specification (RFC 1447), SNMPv2 RFCs do not specify any features equivalent to those defined by ISO / CCITT Systems Management Functions. However, RFC 1271 defines a Remote Network Monitoring (RMON) MIB which provides limited notification control analogous in some respects to the ISO / CCITT event management SMF.

B.2.3 SNMPv2 Management Information Extensions

In addition to the RFCs defined thus far, there is a further set of RFCs related to specification of MIBs for use with SNMPv2. These RFCs extend the management information specifications provided with SNMPv1 by adding the following features:

- textual conventions which are used to define reusable syntaxes that may appear in many MIBs (RFC 1443),
- conformance statements which are used to describe minimum required and actual implementation of MIBs (RFC 1444), and
- a MIB which defines Managed Objects that describe the behaviour of SNMPv2 entities (RFC 1450).

B.2.4 Realisation of the Reference Model

The Internet management model is defined in terms of the interaction between the management station and the network element, or management application and agent. It provides for management communications between the software entities on both systems and defines a set of related security services.

The components of the Internet management model provide all the elements described in the Reference Model. They allow for the implementation of the necessary functionality to be provided at the most appropriate place in the management system, and specifically SNMPv2 provides for the "cascading" of management to allow composite management functions to be implemented. Figure B-4 shows the mapping of the Internet management components onto the Reference Model.

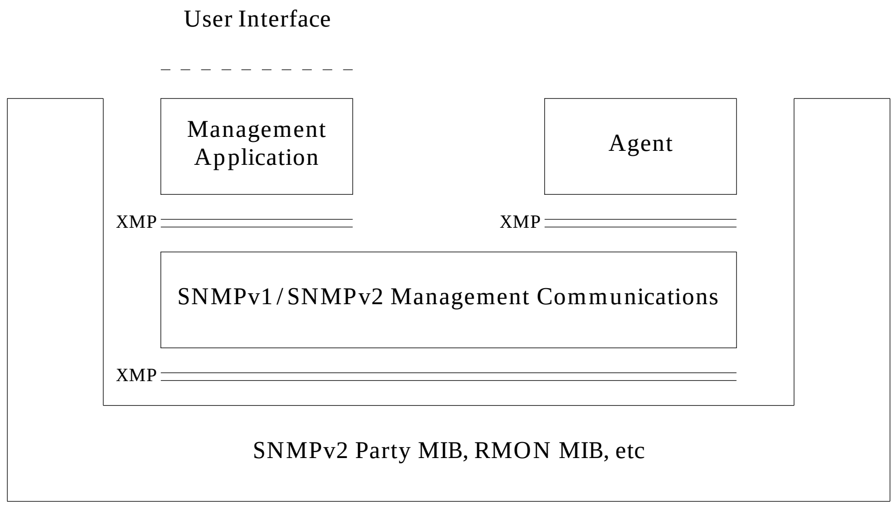

Figure B-4 Internet Mapping to the Reference Model

Note that Internet Management model does not explicitly provide for specification of common services as described in the Reference Model. However, service specifications can and do exist, represented as specialised MIBs such as the SNMPv2 Party MIB and the RMON MIB.

Interoperability between OMG and XMP

C.1 Overview

The following sections discuss different architectural approaches for providing integration and interoperability between an OMG based system management reference model implementation and an XMP based system management reference model implementation. There are two basic approaches to support this interoperation, including:

Parallel Frameworks	The managed Resources are encapsulated by an object from both the OMG based framework and the XMP based framework. Integration takes place ‘‘below’’ the object models.
Object Gateways	An object is defined in the OMG based framework which corresponds to or encapsulates one or more objects in the XMP based framework. This object implementation acts as a client or gateway to one or more XMP based objects, translating and forwarding object requests.

Although these approaches are expressed as mapping OMG object requests into XMP requests, the architectures can be symmetrical (XMP object agent programs can act as OMG clients making ORB compliant object requests; e.g., XMP based objects). In addition, these approaches, particularly that of Object Gateways, are suitable for implementing interoperability between XSM and legacy management systems.

The provision of services within the different models must also be addressed in order to achieve interoperability. The services summarised within the Reference Model identify the basic functionality that is required from those services. In order to have portability across the OMG and XMP environments, it is necessary to specify how the services relate to each other. To make portability work, the functionality and the interface to the services must be defined. To make interoperability work, where there are differences between the underlying services, a mapping between them is required.

C.2 Parallel Framework Interoperation

In this approach to interoperation between OMG based and XMP based versions of the XSM reference model, the managed Resource can be accessed either through the OMG framework or the XMP programming interfaces by making object requests on the Managed Object defined in each framework which encapsulates the same managed Resource state. Client programs would be written to use one framework; that is, the interoperation is through the Managed Object state itself. Each managed Resource would be described by both an IDL interface definition and an XOM class definition, and an object implementation provided.

Figure C-1 shows the parallel framework architecture. Note that integration is achieved through the managed Resource itself; that is, it is likely that the OMG and XMP object implementations will access the same underlying data store to obtain and manage the Resource state, and will need to use a common consistency mechanism. A possible implementation in this environment is an agent program which exports both the OMG interface and the XMP interface; that is, a single object implementation. Another implementation would involve using a common data Manager which supported a multiple access consistency mechanism (such as a Relational DBMS).

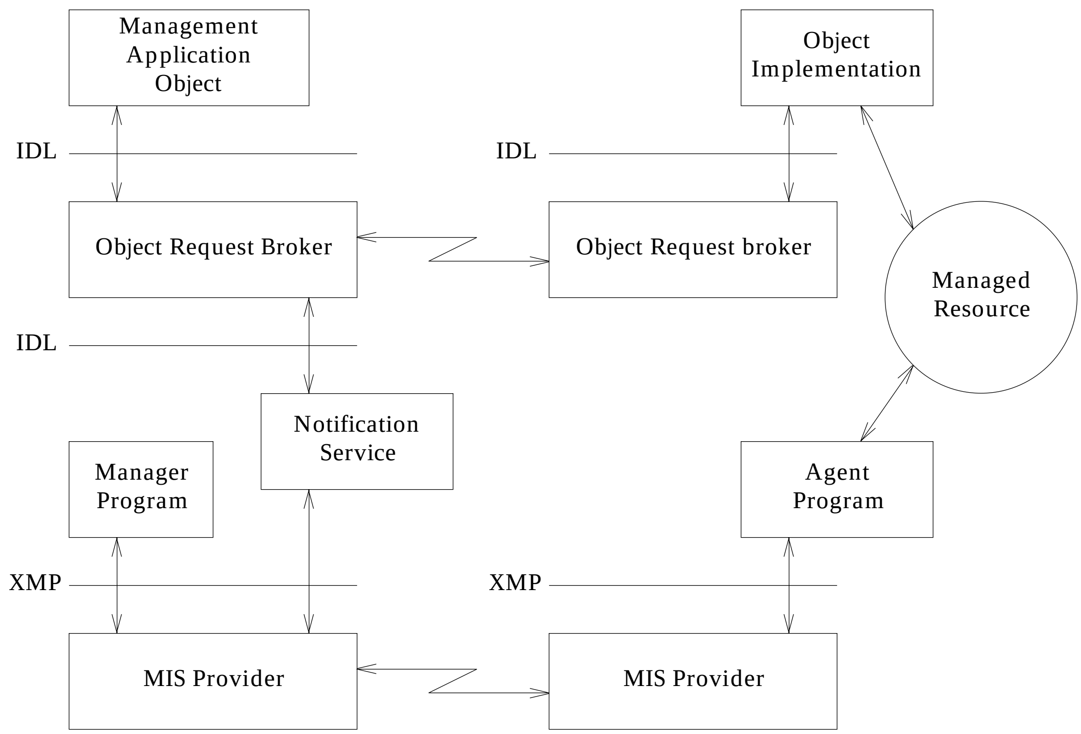

Figure C-1 Parallel Framework Interoperability

An event notification registry service is defined in the OMG framework to enable Management Tasks to register for and receive both OMG based events and XMP based notification event reports. The service would also act as a store and forward notification mechanism with appropriate notification grouping and filtering capabilities.

In practice, one would expect each management application to be implemented upon one framework that was most applicable; e.g., applications requiring a specific management application view might use the OMG based framework, and applications requiring a network management view might use the XMP based framework.

C.3 Object Gateways

In this approach to interoperation between OMG based and XMP based versions of the XSM reference model, special objects are used to encapsulate the object requests of one implementation into an object of the other implementation. Figure C-2 shows how OMG based objects can be used to translate and forward object requests to XMP based objects. Each of these special objects is a server in the OMG framework to a management client entity (a Management Task) and a client in the XMP framework acting as a Manager. This object forwards requests and returns responses between the OMG Management Task and the XMP Managed Object using the XMP programming interfaces. The gateway object is also responsible for initialising the XMP Manager environment, setting up the buffers and session object.

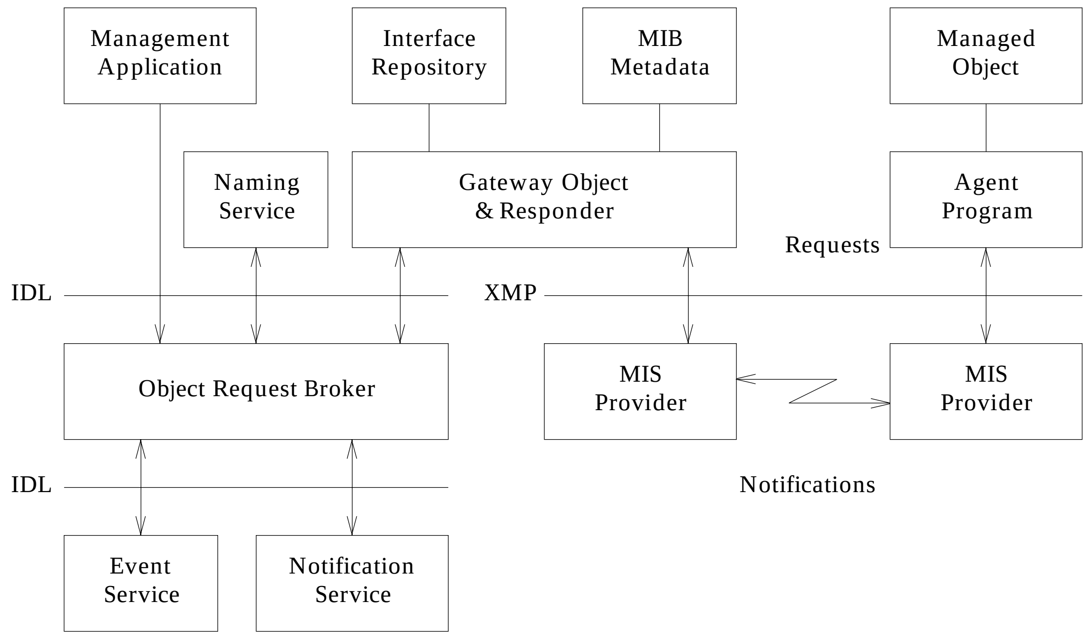

Figure C-2 OMG and XMP Object Interoperability

With a few conventions in defining the OMG based object interface in the IDL language, it would be possible to create an IDL interface that is isomorphic to the XOM class definition for the Managed Object. That is, the gateway object definition corresponds directly with the Managed Object interface definition in terms of public attributes, operations, and exceptions. The IDL defines the interface to the OMG based object, and the XOM class definition is used within the object implementation to forward the request and return the response. The translation from one interface implementation to the other is inherent in the gateway object's implementation. Such objects can use the OMG Naming service to translate Managed Object ''names'' to agent program Titles/Addresses, or use the XMP automatic location mechanism. They may also forward events from the corresponding XMP based object or objects.

Benefits of the Object-Oriented Approach

The advantages of taking an object-oriented approach to systems management are:

- Synergy with standards activities. The major standards activities, *de jure* and *de facto*, as well as emerging work, are using this approach. Alignment with the standards bodies makes the eventual solution easier.
- Codified structure of management information. The structuring of management information as Managed Objects, with a clear distinction between the Managed Object representation and the actual Resource, provides for implementation-independence and interoperability.
- Subclassing or specialisation of objects. The ability to derive a new object definition from an existing object definition provides the ability to extend a system's function, building upon prior work.
- Generalisation of objects. The ability to represent a particular Resource as more than one type of object provides additional opportunities for interoperation between different systems.
- Encapsulation of data. Data encapsulation (making specific data accessible only through a well-defined, message-based interface) provides multiple benefits:
  - Integrity of the object is preserved. As the internal operations of the object are not exposed, the interactions with the object are controlled through its definition. The definition will not allow inclusion of implementation material, such as how operations are performed and how the appropriate consistency constraints are to be enforced.
  - Modularity is encouraged. As the objects are only accessed through their message-based interface, design priority is given to modular, object-centred implementations.
  - An existing application can be hidden behind interface software that allows it to be regarded as an object, thus allowing a simple means of integration that preserves existing investment.
- Message-based communications. A message-based communications scheme provides multiple benefits:
  - Location transparency is facilitated. The connection to other objects can establish pathways for the location-transparent exchange of messages.
  - Interoperability is facilitated. Through the use of standard protocols, messages can be reliably transmitted from one object to another.

Appendix E

Document History

The X/Open Systems Management Reference Model was originally published as a Snapshot in October 1991. This document represents the further development of the Reference Model, incorporating the further experience of the working group, and also builds on the widening implementation experience developed across the industry.

The original Snapshot demonstrated a high degree of OSI orientation. This demonstrated the fact that at the time the document was developed, OSI management technology represented the widest possible consensus as a basis for describing management functionality. The Snapshot also made reference to the expected emergence of the OMG CORBA technology as a key component of distributed systems management.

In the intervening period, this expectation has been demonstrated to be correct, and the updating of the Reference Model reflects this. The original development of the Snapshot had revealed that there were problems of terminology between the OSI and OMG technologies. Accordingly, the Reference Model has been re-stated in abstract terms, and the mapping onto these two technologies has been incorporated in the form of Appendices.

As might be expected, this change in approach has resulted in substantial changes to the original document. However, the focus and intent of the document remains unchanged.

Glossary

activation

Copying the persistent form of an object's methods and data into an executable address space to allow execution of methods.

administrator

A person who has responsibility for administering a network of systems; the person who initiates Management Tasks.

agent

An entity which performs an application service in response to a request from a Manager entity as described in the network management view of the reference model.

application object

Applications and their task-oriented components that are managed within an object-oriented system.

class

An implementation that can be instantiated to create multiple objects with the same behaviour. An object is an instance of a class.

client

An entity issuing a request for a service in the client server model.

common facilities

In the OMG object model, common facilities are objects which provide facilities useful in many applications.

communications mechanism

A means of transferring management information between entities. The precise method of transfer is unspecified, but specific instances of the Communication Mechanism may be defined. Transfer may be wholly within one system or may be between systems.

communications service

The interface to the Communications Mechanism. The Communications Service provides access to the Communications Mechanism in a way that is, as far as possible, independent of the actual implementation of the Communications Mechanism.

data store

A repository of information about Managed Objects. This may be a conceptual repository.

deactivation

Copying the object's data from an executable address space back to the object's persistent form.

event

An activity that takes place asynchronously within a managed Resource and is communicated to client entities.

filtering

Filtering entails the application of a set of tests to each member of the set of previously scoped Managed Objects to extract a subset of objects. Managed Object selection involves two phases: scoping and filtering.

IDL

The interface definition language defined by the Object Request Broker in the OMG object management architecture.

implementation

See Object Implementation.

Implementation Repository

The Implementation Repository contains information that allows the Object Request Broker in the OMG object management architecture to locate and activate implementations of objects.

inheritance

The construction of a definition by incremental modification of other definitions. Inheritance can apply to both object interfaces and object implementations.

Interface Definition Language

A language which describes the operations, parameters, and exceptions of the requests on an object. An interface definition language is used to define object interfaces.

Interface Repository

The Interface Repository is a service in the OMG object management architecture that provides persistent objects which represent the IDL information about OMG compliant objects in a form available at run time.

Managed Object

A Managed Object is an object-oriented encapsulation of a Resource that is subject to management.

Managed Resource

A Resource that is subject to management and is capable of being represented by a Managed Object.

management function

An encapsulation of functionality used to perform some aspect of management.

Management Task

An encapsulation of an administrator's view of a set of management functions. A Management Task corresponds to a human-oriented activity that must be performed in order to manage a system. Management tasks can be encapsulated as management objects.

Manager

A Manager is the initiator of a management interaction. It is a software component that requests some operation to be performed by a managed Resource.

method

The executable code in an object implementation that is executed to perform a requested service.

notification

Information about an event which is communicated to client entities that have registered to be informed about such events.

object

A combination of a state and a set of methods that explicitly embodies an abstraction characterised by the behaviour of relevant requests. An object encapsulates the state and operations which provide a service to a client.

object handle

A value that unambiguously identifies an object in an object-oriented system. In the OMG model, an object reference acts as the object handle. In the OSI model, a Distinguished Name acts as the object handle.

object implementation

A definition that provides the information needed to create an object, allowing the object to participate in providing an appropriate set of services in an object-oriented system. An object implementation typically includes a description of the data structure used to represent the object state and definitions of the methods that access that object state.

object interface

A description of a set of possible uses of an object. Specifically, an interface describes a set of potential requests in which an object can meaningfully participate. Object interfaces are often defined using an interface definition language.

object reference

An object handle in the OMG object management architecture.

Object Request Broker

An ORB is the implementation of the Communications Service in the OMG object management architecture. It provides the means by which OMG compliant objects make requests and receive responses.

object services

Object Services are objects in the OMG object management architecture which provide basic operations for the logical modeling and physical storage of objects.

object type

An object type is a well-defined type that represents a specific set of object interfaces and is satisfied by an object instance whose object implementation encapsulates the behaviour defined by those interfaces. In the OMG model, all objects are strongly typed.

operation

A service that can be requested.

operation signature

The definition of an operation, including the parameters and exceptions which make up that operation. An operation can be characterised by its signature in an object-oriented system.

request

A request is an action by a client to request a service from an object. A request has information associated with it, typically the specification of an operation and zero or more parameters.

server

An entity providing a response to a client request for a service. In an object-oriented system, servers are instances of object implementations activated in computer processes.

service

A computation that may be performed in response to a request from a client entity.

Index