NISTIR 5735

Distributed Systems: Survey of Open Management Approaches

Joseph Hungate
Geraldina Fernandes

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Computer Systems Laboratory
Distributed Systems Engineering
Gaithersburg, MD 20899

NIST

NISTIR 5735

Distributed Systems: Survey of Open Management Approaches

Joseph Hungate
Geraldina Fernandes

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Computer Systems Laboratory
Distributed Systems Engineering
Gaithersburg, MD 20899

Draft 1.0
September 21, 1995

U.S. DEPARTMENT OF COMMERCE
Ronald H. Brown, Secretary

TECHNOLOGY ADMINISTRATION
Mary L. Good, Under Secretary for Technology

NATIONAL INSTITUTE OF STANDARDS
AND TECHNOLOGY
Arati Prabhakar, Director

TABLE OF CONTENTS

1 Introduction

Networks and distributed systems are becoming critical for the working of many enterprises. Many organizations are becoming dependent on distributed computing systems. These are expected to contribute to the financial and operational well-being of the organizations who utilize them.

Traditionally, tools necessary to perform effective system management were inherent to proprietary operating systems and dealt with user and resource allocation and administration. With the introduction of local area networks (LANs), distributed computing environments started to develop. System management tools were enhanced with network management facilities, but rarely in an integrated fashion. However, these environments were still composed of single-vendor products.

In a very short time technology has become incredibly diverse, providing cheaper and more sophisticated information technology (IT) products. With the incorporation of emerging technologies into existing distributed computing environments, single-vendor systems are giving way to multi-vendor, heterogeneous distributed systems. The integrated management of these systems has become nearly impossible. Some components are not managed at all; they are just monitored whether they are in service or not. Those that are managed are each managed differently, usually by independent systems that need to be administered and monitored separately. Administration of the many management systems, and monitoring of many non-managed components, is done by a small army of people using ad hoc management mechanisms.

With systems becoming larger and more complex, manual management is not sufficiently coping and the associated costs are rising rapidly. Technology should be able to provide automatic management, releasing people from non-productive tasks. However, the way distributed environments have been constructed, by many vendors working independently, means that the management systems do not speak the same language or use the same terms. 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. Also, different aspects of the user's environment (e.g., interaction with mail and printing systems) are managed using different interfaces.

Management services, providing mechanisms to monitor and control a great diversity of components and user interactions with these components, and an integrated approach to assure consistency are just now being addressed by standard development organizations and user consortia.

This report gives an overview of some existing approaches proposed by different organizations: IEEE POSIX Working Group P1003.0, X/Open, ISO/IEC, and the Network Management Forum (NMF).

2 POSIX OSE

The Institute of Electrical and Electronics Engineers (IEEE) Portable Operating System Interface (POSIX) Working Group P1003.0 describes an Open System Environment (OSE) Reference Model (OSE/RM), which provides a framework for describing open system concepts and defining a terminology that can generally be agreed upon by all interested parties. The OSE/RM spans the gap between system requirements specification and design of a specific system that uses IT[1], allowing future information system users (who state requirements) and providers (who provide the system to fulfill those requirements) to understand each other.

The OSE/RM provides a basis for specification of standards to enable portability and interoperability of applications, data and people.

Distributed system management is just now being addressed, particularly for the heterogeneous system. The goal of systems and network management standards is to enable the provision of portable and interoperable management technology[1]. The existence of standard user-level interfaces to management functionality also provides portability of system and network administrators.

A standards program for management of OSE-based distributed systems needs to address many diverse elements. The elements of this process include requirements, framework and specifications. Requirements analysis is the first step in determining which services are needed. Once requirements are documented, the second step is to establish a viable framework. The framework needs to provide an integrated infrastructure and includes reference model(s) and architecture(s), service definitions, requirements, metrics, as well as methodologies for applying them. Appropriate standards and publicly available specifications would then be selected to meet the requirements and populate the framework.

2.1 Objectives

Portability and interoperability of applications, data and people are major objectives of P1003.0. The approach to management of OSE-based distributed systems should also satisfy those same objectives, providing:

- Application Portability at the Source Code level,
- System Interoperability,
- User Portability,
- Accommodation of Standards,
- Accommodation of new Information System Technology,
- Application Platform Scalability,
- Distributed System Scalability,
- Implementation Transparency, and
- User's Functional Requirements.

Additionally, management of OSE-based distributed systems should provide:

- An integrated approach to distributed systems management -- essential to achieve interworking between different management domain areas.
- A consistent user interface, avoiding pointless differences in the interfaces presented to the user for management functions.
- The ability for the entire system to be manageable under different domain relationships, ranging from a purely centralized approach to loose federations.
- Support for interoperability among diverse resources.
- Effective automation.

The distributed systems management domain can be thought of as the integration of five distinct, supporting domain areas which reflect the system entities identified in the OSE Reference Model[2]:

- Application Software Management,
- Application Platform Management,
- Human/Technology Interface Management,
- Information Management, and
- Communication Management.

2.2 Service Requirements

A requirement is defined as the user need for a service at an interface. The requirements for management of OSE-based distributed systems[2], can be drawn from the requirements of the five supporting domains stated in the previous section. Services for all domains need to be integrated to provide a comprehensive set of distributed systems management services.

Some requirements are similar across the five domains. The goals are similar, only the system entities involved, and the underlying mechanisms provided to meet those goals are different Thus, all five management domains include the following requirements:

- State Management: a mechanism is needed for managing the state of the system entity involved in each management domain. This need includes system initialization to a pre-determined state (startup), and the ability to suspend operation , synchronize and shutdown.

- User and Group Identification: there is a need to establish identity of a user/group by appropriate authentication means prior to interaction with the system.

- Configuration Control: management of system configuration is needed and includes the ability to view the current configuration, to modify it and to tune the system.

- Service Access Control: the system must be protected from unauthorized access or hostile modification through appropriate security measures at the component level where modification is affected.

- Usage Management and Cost Allocation: system utilization/cost data per user must be collected, stored and analyzed, in order to determine what users are doing and what resources are being consumed by that use.

- Performance Management: this service is needed and it includes measurement and collection of tunable performance parameters, adjustment of system resources to improve performance, redistribution of workload to optimize resource usage, and evaluation of the results of adjustment or redistribution.

- Fault Management: this service is critical and it includes the prevention, isolation, notification, diagnosis, and correction of fault conditions.

- Security Management: this service is needed and it includes security measures utilized in IT resources to insure system integrity. Other aspects of security, such as physical and personnel factors, are clearly outside the scope of this paper.

Specific details in each domain for some of these requirements will be described next, as well as additional requirements particular to each domain.

2.2.1 Application Software Management

Application software includes programs, data, documentation and embedded training.

The requirements for managing application software include:

- State Management: a mechanism is needed for initializing the application software. In addition to startup, the ability to suspend, synchronize, or shutdown is also required.

- User and Group Identification: a user must be identified prior to interaction with the application software.

- Configuration Control: from the application software perspective, management of

system configuration includes configuration of memory and CPU allowances and processor allocation.

- Service Access Control: for the application software this includes enforcing resource authorization by authentication on invocation.

- Usage Management and Cost Allocation: this requirement calls for a common measure of usage for application software is memory/CPU utilization over time.

- Fault Management: Common application software faults might include abnormal termination, memory access violation, or excess cpu utilization.

Application software management requirements also include:

- Application Software Installation, Distribution, and License Control: this requirement calls for the addresses the installation and distribution of software, including new releases of operating systems software and the moderation and management of licensed software usage.

2.2.2 Application Platform Management

POSIX defines an application platform as a set of resources, including hardware and software, that support the services on which application software runs. These services are provided through a set of defined interfaces.

The requirements for managing the application platform include:

- State Management: a mechanism is needed for initializing the system or components of the system. Also required is the ability to suspend, synchronize, and shutdown. Shutdown of all or part of the system may be necessary for maintenance, security or component upgrade reasons.

- User and Group Identification: a user must be identified prior to establishing a session on an application platform.

- Configuration Control: from the application platform perspective, management of system configuration might include configuration of memory and CPU cycle allocations.

- Service Access Control: for the application platform this includes enforcing resource authorization by authentication on session establishment or utilization of services at key interfaces.

- Usage Management and Cost Allocation: a common measure of usage is needed for

application platform is MIPS (Million-Instructions Per Second).

- Fault Management: this requirement needs to deal with common application platform faults due to such factors as out of range environmental conditions, circuit failure or power fluctuations.

Application platform management requirements also include:

- Print Management: needs the functionality to initiate, stop, and manage print jobs, queues and their associated output device.

2.2.3 Human/Computer Interface (HCI) Management

Human Computer Interface (HCI) services provide a consistent way for people who develop, administer, and use a system to gain access to applications programs, operating systems, and various systems utilities. The HCI services address client-server operations, object definition and management, window management and dialog support.

The requirements for managing the HCI include:

- State Management: a mechanism is needed for initializing an information appliance. In addition to startup, the ability to suspend, synchronize, and shutdown is also required.

- User and Group Identification: a user must be identified prior to establishing interaction through an information technology component.

- Configuration Control: in human/computer interaction, management of system configuration needs to include configuration of variable options available from information appliances.

- Service Access Control: for HCI this needs to include enforcing resource authorization by authentication on information appliance activation.

- Usage Management and Cost Allocation: needs a common measure of usage for the human/computer interface such as the duration of use for an IT resource.

2.2.4 Information Management

The fundamental objectives of information management are to maximize the value, quality, and usability of information resources and to coordinate requirements and design across the entire

organization. Information management is concerned with information from the user view and functional standpoint. Information Management includes data modeling, development of policies and standards for use and management of data, training users and coordination of other data activities with system designers and users.

The requirements for managing information include:

- State Management: a mechanism is needed to initialize, suspend, synchronize, and shutdown the information storage components of the system.
- User and Group Identification: a user must be identified prior to accessing information storage.
- Configuration Control: in information management, this requirement includes manipulation of media options and their availability.
- Service Access Control: this information management requirement includes enforcing resource authorization by authentication on access.
- Usage Management and Cost Allocation: a common measure of usage (CMU) is needed; for information management a CMU is bytes of data stored over time.
- Fault Management: to fulfill this requirement, services are needed to deal with common information storage faults might be due to bad blocks or exhausted capacity.

Information management requirements also include:

- Database Management: provides mechanisms to accurately represent the meanings and relationships of the information items to be managed. These mechanisms will provide a standard way of representing data.
- File System Management and Content Integrity Checks: in a distributed system environment files no longer reside locally. A distributed file system management application must allow access to, and manipulation of, files across the system. Support for the distribution and integrity of data must be provided.

2.2.5 Communication Management

Communication services provide functionality to allow two different parts of a system, or two distinct systems, to invoke services from one another. These services include data transfer protocols, services of the communications infrastructure, and services to manage data transfer and communications.

The requirements for managing communication services include:

- State Management: a mechanism is needed to initialize the system or components of the system. Also required is the ability to suspend, synchronize, and shutdown all or part of the system.

- User and Group Identification: a user must be identified prior to establishing communication.

- Configuration Control: the ability to configure network components is needed in communication services.

- Service Access Control: for communication services this includes enforcing resource authorization by authentication upon usage.

- Usage Management and Cost Allocation: a common measure of usage (CMU) is needed; for communication services a CMU is bandwidth.

2.3 Reference Model

POSIX has defined a reference model for systems management[2] consistent with the POSIX OSE Reference Model[2]. The reference model for systems management is shown in figure 1.

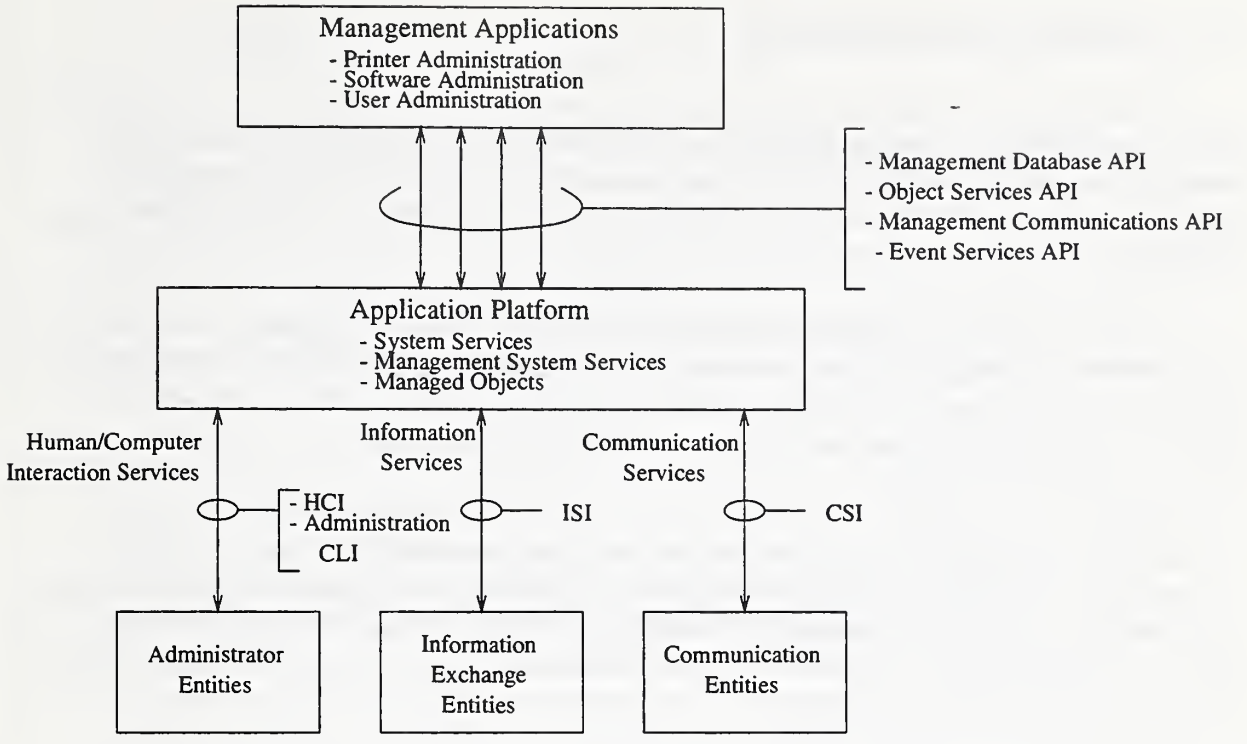

Figure 1 POSIX OSE Reference Model for System Management

A management application provides software application entities to support management tasks in specific areas of management functionality. Application Entities make use of services provided by the application platform.

In addition to the normal platform functionality, management components within the platform provide management system services. Managed objects are included in the application platform as abstract representations of resources that are to be managed. This abstraction allows different resources to be managed in a uniform manner.

The external environment, as described for the systems management reference model, includes administrators, who use management applications via a command line interface or a graphical interface (facilitating administrator portability), and mechanisms for exchanging information outside the model (e.g., via external networks and transportable media).

The POSIX OSE standards for some management services (i.e., print, software distribution and user/group management) already exist. Appropriate standards and emerging standards developed by other organizations, will be included in POSIX OSE. The development of an approach for management of OSE-based distributed systems is still in an early stage. At present most work is being done on service requirements definition.

3 X/Open Systems Management

X/Open is an open systems organization with the primary aim of bringing to users greater value from computing, through the practical implementation of open systems[3]. The X/Open strategy is to combine existing and emerging standards into a comprehensive, integrated, high-value and usable system environment referred to as the Common Application Environment (CAE).

The CAE covers the standards above the hardware level needed to support open systems. The CAE provides portability and interoperability of applications, and allows users to move between systems with a minimum of retraining. Components of CAE include APIs, to enhance portability of application programs at the source code level, and protocols to enhance interoperability of applications.

The X/Open Systems Management Programme (XSM) addresses distributed systems management. The primary requirement of XSM is to develop management software that allows an administrator to manage a network of heterogeneous systems as a single logical system. The XSM is concerned with the definition of those interfaces necessary for portable implementation of distributed management systems.

3.1 Requirements

The XSM is intended to achieve several high-level requirements:

- Portability: the ability to create software that is portable at the source code level (i.e., can easily be moved between systems from different vendors). The XSM provides interfaces to enhance portability of managed objects and managers. The scope of XSM interfaces is limited to management aspects. Other components to achieve portability are provided by the CAE.

- Interoperability: the ability of systems and components from different vendors to share and exchange information. Systems and components need to have connectivity and a common understanding of the significance of information. The XSM provides this property by defining:

    - communications service for management purposes, which provides the means for management information to be exchanged between systems;
    - standards for management interactions;
    - managed objects using object-oriented techniques, therefore allowing the refinement of managed objects while providing for their management based on their original definition. This approach provides sufficient abstraction so a manager does not need to have knowledge of the recently refined managed object definition. Guidance is also given on how resources should be expressed as

managed objects, so that a manager on one system can understand the definition of a managed object on any given system.

- Transparency: simplifies the task of developing management applications, hiding distribution details. There are different aspects of transparency, access, failure, location, migration, replication, and transaction transparency. Transparency should be selective, particularly in the domain of management as there will be occasions when it is necessary to be aware of the precise location or implementation of a resource (e.g., in case of failures). The XSM provides support for transparency by using a model based on object-oriented techniques, in which managed resources are represented by managed objects.

- Extensibility: the ability to extend the management system and to customize it to implement different management policies. Extensibility means to allow the introduction of new resources to manage and new ways to manage them. This property is provided by resources being specified as managed objects. Managed objects definitions may be extended by the definition of new objects that are refinements of the original objects.

- Robustness: the ability of the management system to provide necessary levels of integrity, security and reliability. The XSM will address the provision of security in the relationship between manager and managed object, by the provision of suitable services (e.g., authorization and authentication services). The XSM does not define how consistency of a managed object as viewed by a particular manager is achieved when several managers manage the same managed object. Reliability is addressed by XSM providing both confirmed and unconfirmed management interactions.

Additionally, the following requirements relate to the interfaces that will be provided to access the management functionality:

- Ease of use: the services and APIs should be simple to use.

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

The management functional areas to be covered, corresponding to real end-user requirements, include:

- Network Backup and Restore,
- Performance Management,
- Accounting Management,
- Software Management, and
- Printer Management.

3.2 Reference Model

The X/Open Systems Management Reference Model (XRM) has been developed as part of the XSM to meet the high-level requirements listed in the previous section.

The XRM's primary goal is to provide solutions for distributed systems management. The XRM describes the components and architecture necessary to build a comprehensive distributed systems management environment. Based on the use of object-oriented specification techniques, the XRM describes the environment in which distributed systems management can be performed without requiring the use of particular technologies. The XRM does not give a detailed description of its various components; addresses only general properties. Required management interfaces are identified but not defined.

Figure 2 shows the basic components of the XRM and their relationships.

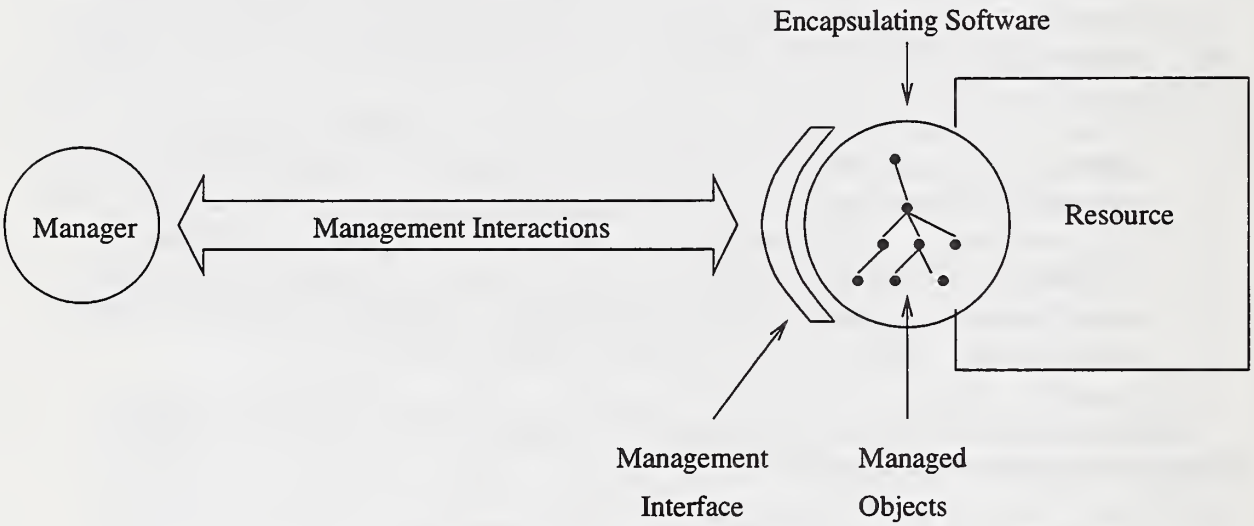

Figure 2 XRM components

A manager is the initiator of a management interaction. A manager is a software component that requests some operation to be performed by a managed resource. A manager may provide a user interface allowing the invocation of management tasks, or may invoke, or be invoked, by other managers. When invoked by another manager, a manager appears itself as a managed object. This is the concept of cascading (i.e., multiple layers of management interaction between the original initiator of the management request and the ultimate target resource(s) that are affected by it).

To enable the management of heterogeneous systems, the management view of a resource has to be isolated from the implementation of that 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 different.

This management view corresponds to managed objects. Their definition encapsulates the management characteristics of the resource and isolates these characteristics from the resource implementation. Some managed objects may not correspond to any real resource within the system, but rather correspond to an abstract element of functionality relevant to management of some other resource, thus providing the capability of cascading managers.

Resources are entities within a system or network of systems that require management. Resources can include physical entities (e.g., printers, routers) or logical entities (e.g., users, groups). Not all resources need to be managed.

Services provide the common facilities that must be provided by the XSM Support Environment in order to support distributed systems applications. Services are divided into three major categories:

- General Services: the normal services available to all applications.
- Management Services: provide the common management functionality available to all management applications. Management Service are built on common underlying system services and management specific services.
- Application Services: services specific to some particular functional area within the overall management problem space.

The basic reference model for management is shown in figure 3. The reference model describes the relationship between the three fundamental components -- managers, managed objects and services.

The communications service is specifically singled out for special treatment as this service provides all the functionality necessary to provide transparent communications between managers and managed objects. There is no direct access between manager and managed objects except via the communications service. In the case where manager and managed object are on the same system, the communications service may use local transport mechanisms, such as Interprocess Communication (IPC).

Two types of interfaces are addressed in the reference model. The interface to the communications service is an object-oriented interface, in which requests and responses are expressed in object terms. Non-object interfaces are also included because many services will be expressed in terms of traditional, functional interfaces, and mainly general services not specific to management.

X/Open describes mappings of different existing technologies to the XRM, namely the Object Management Group (OMG) mapping, the International Standards Organization (ISO)/CCITT

OSI Management mapping and the Internet Management mapping.

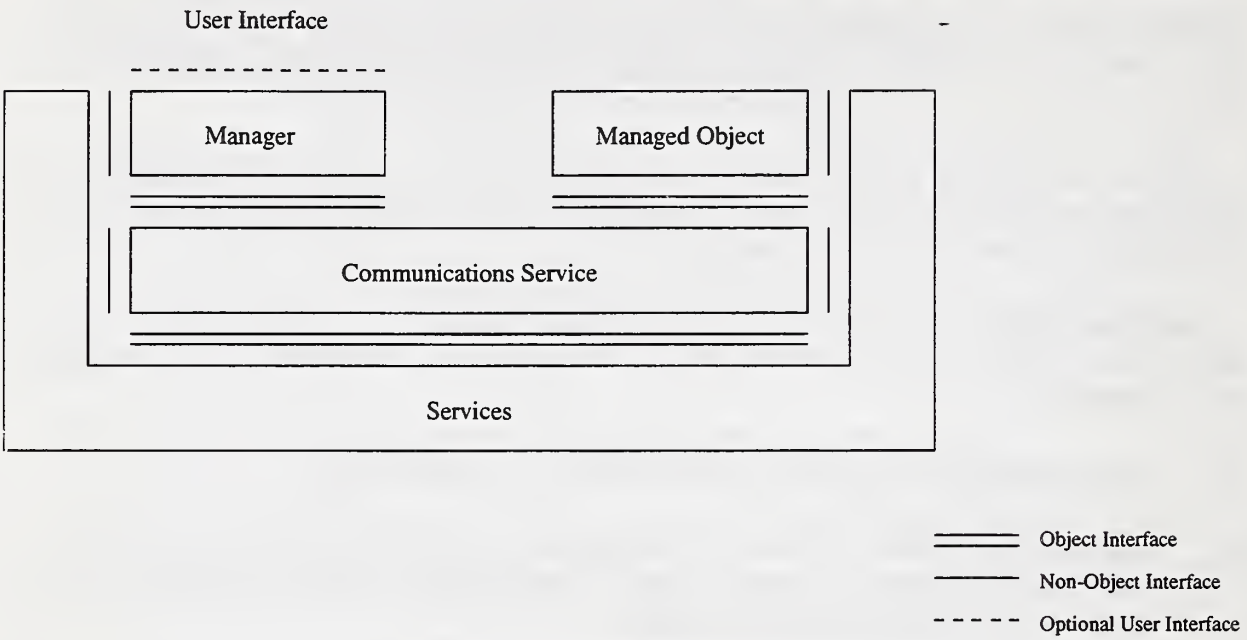

Figure 3 XRM Reference Model

4 NMF OMNIPoint

The OMNIPoint (Open Management Interoperability Point) program is an NMF (Network Management Forum) approach to the integrated management of networked information systems. OMNIPoint comprises a framework for managing networks and systems from the perspective of the services they deliver to end-users[4]. The ultimate goal of OMNIPoint is to achieve fully automated, flexible management systems that provide the enterprise using the information system the required levels of service.

OMNIPoint defines a complete infrastructure that enables management systems to interoperate and exchange information. OMNIPoint consists of a set of standards, implementation specifications, testing methods and tools, and object libraries to enable the development of interoperable management systems and applications. This framework enables the implementation of products and service quality while reducing costs, and provides the integration of multiple management technologies, in both computing and telecommunications.

4.1 Requirements

OMNIPoint provides a list of requirements targeted for use by developers of management systems, applications, and platforms in developing their own statement of requirements. It is not necessary for a product to support all the requirements to be OMNIPoint compliant.

4.1.1 System Interoperability

OMNIPoint defines mechanisms for ensuring interoperability between management systems using a set of standard services and protocols. The requirements necessary to achieve interoperability include the following requirements[5]:

- Interface between Management Systems
    - The offered management system must support ISO/CCITT Common Management Information Service (CMIS) services.
    - An approach to naming and exchanging management information must be supported.
    - An interface to nonconformant management systems or elements must be provided.
- General Management Services
    - The offered system must provide configuration, alarm, and event reporting and logging services.

- Managed Objects

  A particular network resource is represented within management systems by one or more managed objects.

  - The offered system must provide a framework for defining managed objects using managed object classes, instances, attributes, management operations, behavior, and notifications. The standard used by OMNIPoint is ISO/CCITT GDMO.

- Security of Management

  In order to prevent unauthorized access to network management systems, the system should provide access control.

  - Access control should restrict unauthorized access to the system via the interoperable interface (see section 4.2), and for authorized users access control should then restrict functions and the span of control.

  - The system should provide an audit trail and log commands entered by the operators.

  - The system should provide a means to authenticate the identity of any other system trying to access management information.

  - Security alarms for network management violations should be provided.

4.1.2 Management Application

OMNIPoint provides a range of application services that are prerequisites for the implementation of management applications, but OMNIPoint does not specify the applications themselves. However, OMNIPoint offers a range of application requirements as an aid to organizations wishing to develop management applications:

- Integrated Configuration Management

Configuration management is defined as the process of managing change in the business operation. This process includes the ability to initialize and close down particular resources, to collect information on demand about the current condition of systems, to obtain notification of significant changes, and to change the configuration of the components in the system.

Requirements to meet integrated configuration management include the following:

- It should be possible to enter data only once. This can be achieved either by having a common database, a centralized change management system or application, or a distributed approach by each of the involved management systems owning specific data elements that are within its management domain.

- Management systems should make transparent to network operators differences in language, terminology, and methods of identifying resources employed by different systems.

- Management systems should support the construction of a single inventory of all resources from which a network map can be developed.

- Distribution of configuration management should be supported so that operators at geographically separated locations can operate on the same configuration information.

- Integrated Fault Management and Trouble Ticketing

In a complex communication network with multiple components, one failure can generate multiple alarms from multiple systems. A mechanism should be provided to ensure that various management systems work together to present the network operator with answers instead of problems. Requirements to achieve this integration include.

- a mechanism to generate trouble ticket information automatically.

- a correlation among alarms from all management systems received by a single management application to identify the most likely source of the problem identified. The system must provide for the activation of predefined diagnostic and testing procedures to determine or verify where faults occur.

4.1.3 Management Platform Functional Requirements

In addition to the requirements specified for system interoperability, platform requirements specifically related to application portability and platform scalability must also be met. These requirements are concerned with the following.

- Application Programming Interfaces

The X/Open Management Protocol Application Programming Interface (XMP), also known as Communications Management API (CM-API), provides the capability to manage objects using ISO/CCITT CMIP and Internet SNMP. The system provider must identify the support offered for the XMP protocols.

- Management Protocol

  The system should support the Common Management Information Protocol (CMIP). The level of support provided for the Simple Network Management Protocol (SNMP) should also be identified.

- Management Information Database

  The requirements that provide a basis for managing the management information databases include:

  - a user-friendly data entry and editing mechanism that will allow the management information databases to be built and maintained, and
  - a database accessible using a standard query language.

- Operator Interface

  One of the primary objectives of the open management of complex, networked systems is to enable network operators to monitor and control systems without having to adjust to multiple operator interfaces.

  - The systems should provide the operator with a graphical user interface (GUI).
  - As the network grows, several operators may need to be on-line to the management applications concurrently. The system should provide a mechanism that allows multiple management domains and multiple management views to be defined.

4.1.4 Testing

The OMNIPoint testing program consists of three forms of testing: conformance, interoperability and acceptance. These forms are described below.

- Conformance Testing

  A product is conformant if it meets the requirements of an agreed-upon specification or international standard. Conformance testing is performed on a product by an appropriate test laboratory and results are documented in a conformance test report (CTR). The primary objective of conformance testing is to maximize the probability that different OMNIPoint implementations will interoperate.

- Interoperability Testing

Even when conformance to standards has been achieved, the functions performed may not meet user requirements due to different interpretations of the specifications by the developers of each system. Within the OMNIPoint program an interoperability framework is being developed that will lead to the introduction of testing services.

- Acceptance Testing

  This form of testing provides for specification of a process which verifies fulfillment of the requirements.

4.2 OMNIPoint Management Model

The development of OMNIPoint is based on the principle that if users and suppliers come together to identify the user's needs and a common approach to meeting those needs, the risks involved in developing and then implementing the resulting products is significantly reduced[4].

The main job of information systems is to deliver information services to clients. With the migration from centralized to distributed information environments, the infrastructure for delivery of services has dramatically increased in complexity and diversity. The requirement now is for interoperable management applications that provide a common view of all the resources within the infrastructure.

OMNIPoint specifies the basic infrastructure to implement open management of networked information systems. OMNIPoint does not define the actual management applications themselves. Its emphasis is on the point where two different management systems meet to exchange data -- known as the interoperable interface. Managed objects, representing management characteristics of the resources being managed, are made visible by one system to another across the interoperable interface. An agent is the management system that makes objects visible providing access to the resources represented by the managed objects. A manager is the management system that operates on managed objects in another system (i.e., a manager manages and controls the resources that the managed objects represent). Figure 4 illustrates the OMNIPoint management model.

The detailed functions to be performed by particular management applications are not defined by OMNIPoint. It defines the elements which must be implemented in order to achieve effective

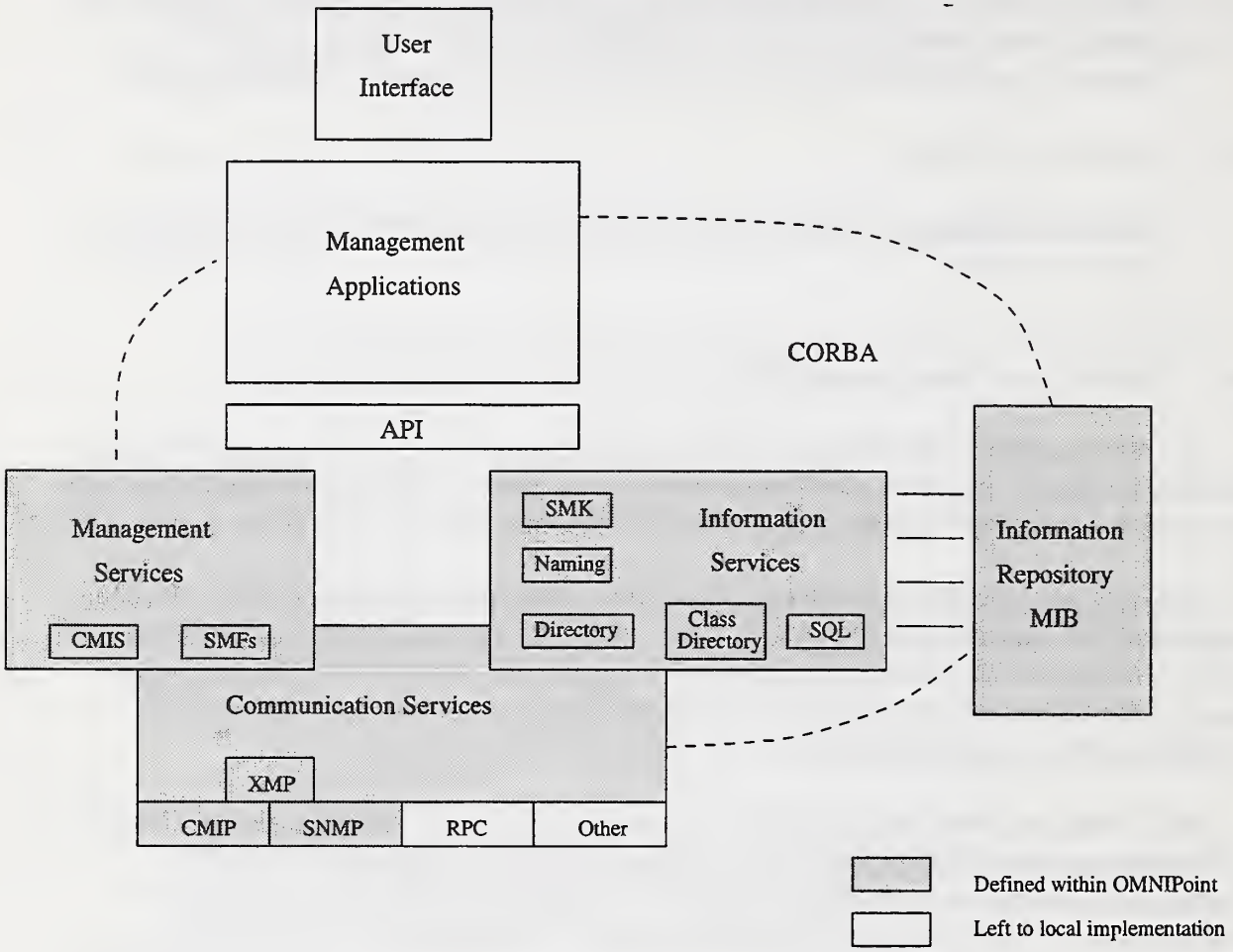

Figure 4 OMNIPoint Management Model

exchange of management information. The management model shows the logical relationships between these elements and management applications.

Management services are derived from ISO/CCITT CMIS (Common Management Information Services), a standard that defines the services required to exchange management, information such as requests to read object attributes or send event reports, together with a set of System Management Functions (SMF). The SMFs are described below.

- Configuration Management: using CMIS services, OMNIPoint defines functions that support basic configuration management. These functions depend on the role played by each system -- manager or agent.

- Fault Management: OMNIPoint defines functions that support fault management. These functions also depend on whether the system is a manager or an agent.

- Testing Management: provides the interface for a management system to request a remote management system to execute a test and return the results.

- Scheduling Management: provides a mechanism for defining and altering a timetable (schedule) for the triggering or operation of a function within a resource being managed over an interoperable interface.

- Trouble Management: provides procedures to initiate trouble reports as a result of a problem detected or reported. Trouble management allows for the monitoring of status from detection through resolution.

- Path Tracing: functions are provided to enable the identification of managed objects in a communications path.

- Security of Management: provides mechanisms to protect management applications and information, specifically access control, peer entity audit trail recording, and generation and logging of security alarms.

Information services define a common naming architecture to unambiguously identify the resources presented as managed objects, and shared management knowledge. Because interoperable management products are developed to achieve certain objectives, not all of these products will have the same capabilities nor behave the same way to achieve their objectives. A common understanding of management capabilities (or shared knowledge) must be established before two management products can interoperate. Each needs to know the other's capabilities in a given area in order to communicate effectively. OMNIPoint enables the automation of some aspects of this knowledge-sharing process.

The OMNIPoint management model includes a set of Application Programming Interfaces (APIs) that enhance portability features of implementations. These APIs are described below:

- X/Open XMP: provides the capability to manipulate managed objects by means of both ISO/CCITT CMIP and Internet SNMP.

- X/Open XOM: provides the capability (when implementing XMP) to manipulate abstract data, thus hiding complexity of their definition (ASN.1).

- CORBA: specifies interfaces from objects to an object request broker (ORB) and allows objects to be portable from one ORB implementation to another. Mapping between the interface definition language (IDL) and ASN.1 is still under study, resulting in the view that CORBA is as a strategy for future implementation.

Communication services are needed for a management system to be able to communicate

with another management system. OMNIPoint integrates two different standards: CMIP and SNMP; these are described below.

- CMIP: specifies the protocol required to exchange management information within an OSI environment.

- SNMP: specifies the protocol required to exchange management information between network elements and their management stations in an Internet environment.

Managed objects are the data representation of the resources to be managed by management applications. They supply management information to the applications and are defined according to ISO/CCITT GDMO. Managed objects information is stored in the Management Information Base (MIB).

5 Open Distributed Processing (ODP) Overview

The objective of Open Distributed Processing (ODP) standardization is to develop standards that allow the distribution of information processing services in an environment of heterogeneous information technology resources and multiple organizational domains[6].

The ODP standardization aims to build distributed systems which include openness, integration, flexibility and transparency. These properties are described below:

- Openness This property consists of portability (the ability to execute different system components on different processing nodes without modification) and interworking (the ability to support meaningful interactions between components, possibly residing in different systems).

- Integration This property allows the incorporation of various systems and resources (e.g., systems with different architectures, different resources with different performances) into a whole. Integration helps to deal with heterogeneity.

- Flexibility This property is geared to supporting a system's evolution. A system should be capable of facing run-time changes and be capable, for instance, of dynamic reconfiguration to accommodate these changes.

- Transparency This is a central requirement to facilitate distributed applications development. Transparency aims to hide distribution details from developers.

5.1 The Reference Model of ODP

The Reference Model of ODP (RM-ODP) provides a framework for the standardization of Open Distributed Processing. The RM-ODP creates an architecture within which support for distribution, interworking and portability can be integrated.

The purpose of the RM-ODP framework is to organize services within autonomous systems in order to facilitate interworking of software components distributed into progressively larger and larger systems.

The RM-ODP is organized into two main parts, described below.

- A descriptive model of distributed systems. This model takes an object-based modeling approach that provides a common ground for all of the ODP viewpoint specifications.

- A prescriptive model that describes the organization and structure of an open distributed system. This model is organized around the ODP five viewpoints. These viewpoints are: Enterprise, Information, Computational, Engineering, and Technology.

5.2 ODP Architecture

The prescriptive model[7] makes statements which must hold for a system to be considered as an ODP system. The model presents the ODP Architecture, which consists of a set of rules to define the structure of an ODP system and the interrelationships between its parts.

The Reference Model defines a framework comprising:

- viewpoints,
- a viewpoint language for each viewpoint,
- specifications of the functions required to support ODP systems, and
- transparency prescriptions showing how to use the ODP functions to achieve distribution transparency.

The architecture for ODP systems and the composition of functions is determined by the combination of the computational language, the engineering language and the transparency prescriptions.

5.3 ODP Management Functions

Resource management requires that it be possible to invoke management operations on individual services, the objects that contain them, the cluster that contains the object, the capsule that contains the cluster and the node that supports the capsule.

A cluster is a set of basic engineering objects (BEOs) associated with a cluster manager. Each BEO in a cluster can be bound to other BEOs in the same cluster, or in other clusters. Additionally, a BEO can have an interface supporting the object management function. This object interface is bound to the cluster manager. A BEO is also bound to the nucleus at an interface providing the node management function.

Each cluster manager in a capsule is bound to the capsule manager for that capsule. This structure is presented in figure 5.

The cluster management function can checkpoint, recover, migrate, deactivate and delete the cluster. The cluster management function may require the cluster manager to interact with the objects in the cluster via their object management interface. Cluster instantiation is performed by a capsule manager.

A capsule consists of: cluster(s), cluster managers (one for each cluster) and a capsule manager. The capsule management function instantiates clusters from a cluster template and can checkpoint, deactivate and delete a capsule. Capsule instantiation is performed by the nucleus.

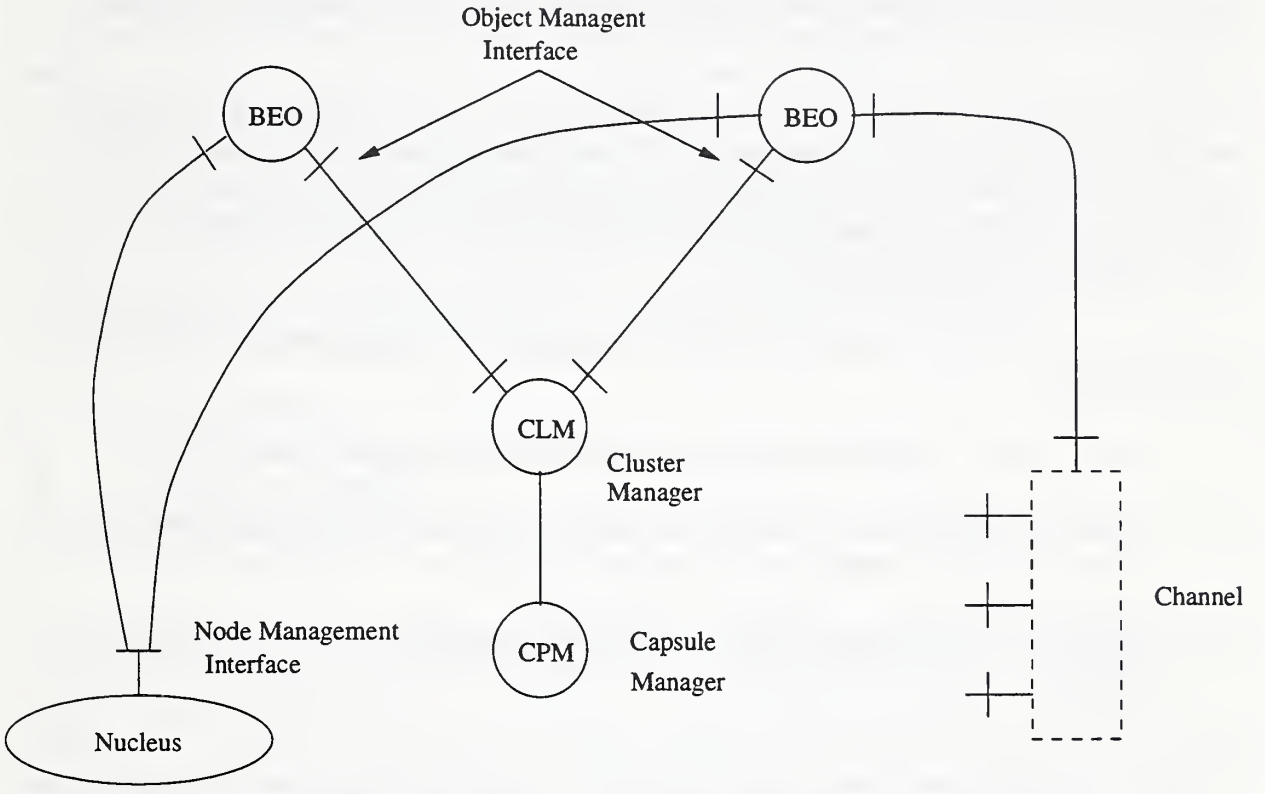

Figure 5 Structure supporting a basic engineering object

A node consists of one nucleus and a set of capsules. The nucleus provides a set of node management interfaces, one to each capsule within the node. The node management function provides thread management for objects which can provide spawning, forking and joining of activities defined in the computational language.

5.4 Management of ODP Systems

An ODP system consists of a number of ODP applications which make use of underlying support services. All of these services, as well as the ODP applications, need to be managed. Management in ODP is concerned with overall systems management, including application management and communication management.

A managed object is an entity to which a management policy applies and which behavior can

be monitored and changed by a manager. A managed object is an abstraction that represents the properties of data processing and data communications resources, for the purposes of management. A managed object could be a hardware or software component, or a collection of information (e.g., a data structure).

Managers monitor the activities of managed objects, make management decisions based on that information and perform control actions on the managed objects, modifying the system's behavior.

In an ODP system, only an object can modify its own behavior. In order to realize ODP management applications, it is necessary to specify management interfaces supporting the management operations that can be performed on a managed object. Thus, a managed object, besides its normal functional interfaces, can provide more than one management interface reflecting different management views.

There are three types of interactions between managers and managed objects. These are described below.

- control actions: action directed by the manager to the managed object,
- requests for information: issued by the manager, these instructions result in a reply from the managed object sending the required information, and
- notifications: actions sent to the manager by the managed object to report faults or events.

Within the ODP environment there are multiple coexisting management views and boundaries of responsibility. To reflect these different views, *domains* provide a means of specifying boundaries of management responsibility and authority. All the managed objects in a domain are controlled under a common management policy. An important aspect of a management policy is to specify what management operations managers may perform on the objects they manage. Management policy and domain membership can be modified through interactions with a single object, called the domain coordinator managed object. This way managers do not need to interact individually with multiple managed objects within the environment.

Domains do not encapsulate the member objects; external objects may interact directly with an object in a domain. Domain relationships can be used to model management structures. It is necessary to permit different domains, with members in common, to coexist in order to reflect the different required views of management. Two domains are disjoint if they have no member objects in common. Two domains overlap if there are objects that are members of both domains. The necessary transformations in order to allow activities to cross from one domain to another take place at the domain boundaries.

6 Conclusion

The approaches presented are all intended to give a solution for distributed systems management. They have different objectives and requirements, but their primary goal is the same -- to provide a framework for managing a network of heterogeneous systems.

The POSIX OSE approach is the only one that tries to clearly differentiate objectives and requirements. Objectives are general characteristics the management framework must support, namely portability of applications and users, system interoperability, and scalability. Primarily, the framework is intended to meet the user's requirements. The first step to address management of OSE-based distributed systems is to analyze the requirements in order to determine which services are required. The POSIX OSE defines a requirement of what a user needs from a service at an interface. The POSIX OSE divides management into five domain areas, according to the system entities involved, and identifies the requirements for each domain. The OSE requirements for management of distributed systems may be included within what the ISO/CCITT OSI management model defines as management functional areas[8]:

- fault management,
- configuration and name management,
- performance management,
- accounting management, and
- security management.

The reference model presented by POSIX OSE for systems management is a very high-level model, only identifying the system entities involved: management applications, application platform, user, information interchange entities, and communication entities. Management is focused on the services expected at the interfaces between the application platform and the other system entities. These services are divided into: general services and management services. There is no reference to how these services are provided at the application platform boundaries. The OSE only refers to the existence of managed objects in the platform, but the OSE does not address how they are defined and organized.

Like POSIX OSE, the main goal of X/Open is to provide portability and interoperability of applications and users. To achieve that, X/Open tries to combine existing and emerging standards into a system environment. The XSM is also concerned with defining those interfaces necessary for portable implementation of distributed systems management.

X/Open does not distinguish between objectives and requirements. X/Open only define requirements, dividing them into high-level requirements and real end-user requirements. The high-level requirements may be compared to POSIX OSE objectives and the real end-user requirements to POSIX OSE requirements; however, XSM describes mechanisms used to meet high-level requirements. For instance, to provide interoperability, XSM includes a communications service that can be provided by ISO/CCITT CMIP or Internet SNMP, standards for management interactions (e.g., ISO/CCITT CMIS and SMFs), and managed objects defined

using ISO/CCITT GDMO.

The X/Open reference model for management of distributed systems (XRM) describes the components and architecture necessary to build a distributed system management environment. Unlike POSIX OSE, the XRM defines managers, agents and managed objects, based in object-oriented technology. The XRM does not give a detailed description of these components, referring only to general properties, relationships and the management interactions between those components. Management interfaces are also identified.

Similarly to POSIX OSE, services are divided into general services and management services. Furthermore, the XRM identifies an additional class of services -- application services which are specific to a particular management functional area.

In general, the management of an ODP system, similar to the OSE approach, is provided by specialized ODP applications. The ODP can distinguish the normal functionality of an object from its management functionality because objects in ODP are permitted more than one interface (i.e., interfaces can be regarded as "objects" that share state.). A typical use for this feature is in the specification of separate interfaces for the functional and management behavior of objects. An object would have one set of conformance reference points for functional behavior and another for management behavior. The management domain, encompassing the management behavior, can be re-mapped onto the other domains, since ODP assumes for the general case that an application will span organizational boundaries.

The NMF OMNIPoint provides a framework for managing networks and systems from the perspective of the services delivered to end-users. Like POSIX OSE, OMNIPoint is concerned about the interfaces to the management system where services are provided to users.

Like X/Open, OMNIPoint does not distinguish between objectives and requirements. Requirements are divided into: system interoperability requirements, management application requirements, management platform functional requirements and testing requirements. OMNIPoint is the only approach that refers to testing requirements. Like X/Open, OMNIPoint refers to mechanisms intended to meet the requirements defined. OMNIPoint includes XMP to provide the capability of managing objects using ISO/CCITT CMIP and Internet SNMP.

The OMNIPoint management model provides a range of application services, as does X/Open, however, does not specify applications themselves. None of the approaches specifies applications. OMNIPoint also defines managed objects, managers and agents giving more emphasis to interoperation, defining the elements which must be implemented in order to achieve effective exchange of management information and the relationships between them and management applications. OMNIPoint focuses on interoperable interfaces (i.e., the point where two different management systems meet to exchange data). Services are divided into management services, information services and communication services. This classification of services reflects OMNIPoint concern about interoperation among management systems.

7 References

1. ISO/IEC TR 14252 | IEEE 1003.0. Guide to the POSIX Open System Environment, 1995.

2. Requirements for Distributed Systems Management, Draft 1.1, January, 1995. Distributed Systems Management Working Group, Open System Environment Implementors' Workshop (OIW).

3. X/Open Company Ltd. Systems Management: Reference Model. X/Open Guide, August, 1993.

4. Network Management Forum. OMNIPoint Strategic Framework: A Service-based Approach to the Management of Networks and Systems. Technical Report 1993.

5. Network Management Forum. *Discovering OMNIPoint: A common Approach to the Integrated Management of Networked Systems.* PTR Prentice Hall, 1993.

6. ISO/IEC 10746-1. Open Distributed Processing - Reference Model - Part 1: Overview. ISO/IEC 10746-1, ITU-T X.901 (committee draft), July 1994.

7. ISO-10746-3. Open Distributed Processing - Reference Model - Part 3: Architecture. ISO/IEC 10746-3, ITU-T X.903 (committee draft), 1995.

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