PROTOCOL NEUTRAL MODELING OF SNMP

THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENT FOR THE AWARD OF THE DEGREE OF

MASTER OF ENGINEERING
(COMPUTER SCIENCE AND ENGINEERING)

TO
THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY
(DEEMED UNIVERSITY)
PATIALA – 147001

SUPERVISOR(s)
DR. RENU VIG
(Asstt. Prof., TTTI, Chandigarh)
DR. S. VERMA
(Asstt. Prof., IIITM, Gwalior)

SUBMITTED BY
SARABJEET SINGH BEDI
(ME(CSE) - 8003909)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
TECHNICAL TEACHERS' TRAINING INSTITUTE
SECTOR – 26, CHANDIGARH – 160019, INDIA

2001

CERTIFICATE

Certified that this Thesis report entitled "**PROTOCOL NEUTRAL MODELLING OF SNMP**" Submitted by **SARABJEET SINGH BEDI** in the partial fulfillment of the requirement, for the award of **Master of Engineering** (Computer Science) Degree of **Thapar Institute of Engineering and Technology, Patiala** is a record of student's own study carried under my supervision and guidance.

This thesis has not been submitted to any other university or institution for the award of any degree.

SUPERVISOR(s)

1.

2.

Prof. & Head,
Deptt. of Comp. Sc. and Engg.
T.T.T.I, Chandigarh

ACKNOWLEDGEMENT

The author is highly grateful to the Principal, Technical Teachers' Training Institute, Chandigarh for providing this opportunity to carryout the present thesis work.

The constant guidance and encouragement received from **Prof. V.P. Puri**, Professor and Head, Department of Computer Sc. And Engineering, TTTI, Chandigarh has been of a great help in carrying out the present work and acknowledged with reverential thanks.

The author would like to express a deep sense of gratitude and thanks profusely to **Dr. Renu Vig,** Asstt. Professor, Deptt. of CSE, TTTI, Chandigarh, who was the thesis supervisor. Without her wise counsel and able guidance, it would have been impossible to complete thesis in this manner.

The valuable advice, insightful suggestions and encouragement rendered by **Dr. S. Verma,** Asstt. Professor, IIITM, Gwalior is gratefully acknowledged.

The author thanks to **Mr. C. Ramakrishna,** Sr. Lecturer, Deptt. of CSE, and other staff members of Computer Science Department, TTTI, Chandigarh for their intellectual support throughout the course of stay at institute.

Finally, the author is thankful to parents, and all, whosoever has contributed in this thesis work.

(SARABJEET SINGH BEDI)

ABSTRACT

The presence of dissimilar network models and standards necessitate interoperability as a means of achieving ubiquitous connectivity and management.

The present work endeavors to overcome the limitation of Heterogeneous Network Management through Protocol Model especially in the context of SNMP. Instead of converting SNMP MIBs to any Object Oriented Model, model should be created in Unified Modeling Language (UML), separate from the protocol. The conversion between SNMP and OOM should be from the Protocol Neutral UML model as and when needed.

The primary focus of this work is to propose the network management model, which forms the basis for developing inter-working paradigms.

The aim of this research work is to demonstrate the feasibility of integration advanced information processing technologies and the fact that both SNMP and object oriented paradigm have achieved a significant acceptance.

The output is a graphical representation for Protocol Neutral Network Management Model, and SNMP Events model, which is designed using UML class diagram.

CONTENTS

LIST OF FIGURES

ACRONYMS

API	Application Programming Interfaces
ASN-1	Abstract syntax Notation
ATM	Asynchronous Transmission Mode
COM	Component Object Model
CORBA	Common Object Request Broker Architecture
DCOM	Distributed Component Object Model
DPE	Distributed Processing Environment
GDMO	Guideline for Managed Object
IDL	Interface Definition Language
IETF	Internet Engineering Task Force
IIMC	Inter Management Coexistence
IP	Internet Protocol
JIDM	Joint Inter Domain Management
MIB	Management Information Base
MIT	Management Information Tree.
MOC	Management Operation Centers
NMF	Network Management Forum's
NMS	Network Management Station
ODP	Open Distributed Processing
OID	Object Identifier
OMG	Object Management Group
OMT	Object Modeling Techniques

OOP	Object Oriented Paradigms
ORB	Object Request Broker
OSI	Open system Interface
PDU	Protocol Data Unit.
RFC	Request For Comment
SMI	Structure of Management Information
SNMP	Simple Network Management Protocol
TCP	Transmission Control Protocol
TMN	Telecommunication Management Network
TP	Transport Protocol (ISO)
UML	Unified Modeling Language

CHAPTER 1.

INTRODUCTION

1.1 OVERVIEW OF NETWORK MANAGEMENT CONCEPTS

Network Management is the ability to plan, configure and monitor a network in order to effectively utilize local and global network resources and to ensure reliable, secure operation of networks. Based on this definition, the scope of network management includes configuration, performance, fault, security and accounting management. These sub-areas have been specified as part of the OSI standards [7].

A *Network management Framework* encompasses the various components needed for network management. The Network Management Framework (Fig. 1.1) includes a Managing Entity or Manager, a Managed Entity or Agent, Managed Objects, Management Protocol and a Structure of Management Information. Managing entity and managed entity are also known as Management Entities. Individual network management models dictate the number of instances of each component.

On any network node the operational protocols (e.g. IP, TCP, X. 25) maintain variables (or real resources), which are used by the management entities. A projection of these variables as seen by the management framework is termed *managed Objects*. The management framework has access to the managed objects only. Any operation requested on a managed object is translated into a suitable operation on the corresponding real resource. A *Management Information Base* (MIB) is a repository of managed objects. A MIB is defined using a set of structural rules (schema) called *Structure of Management Information (SMI)*. Management objects have widely varying capabilities and characteristics depending on the management model. Ideally, anything that needs to be managed should be represented as a

managed object. However, in a particular model the structure or definition of management information may restrict the capabilities of managed objects. Such restrictions may result in only a subset of the real resources being represented as managed objects.

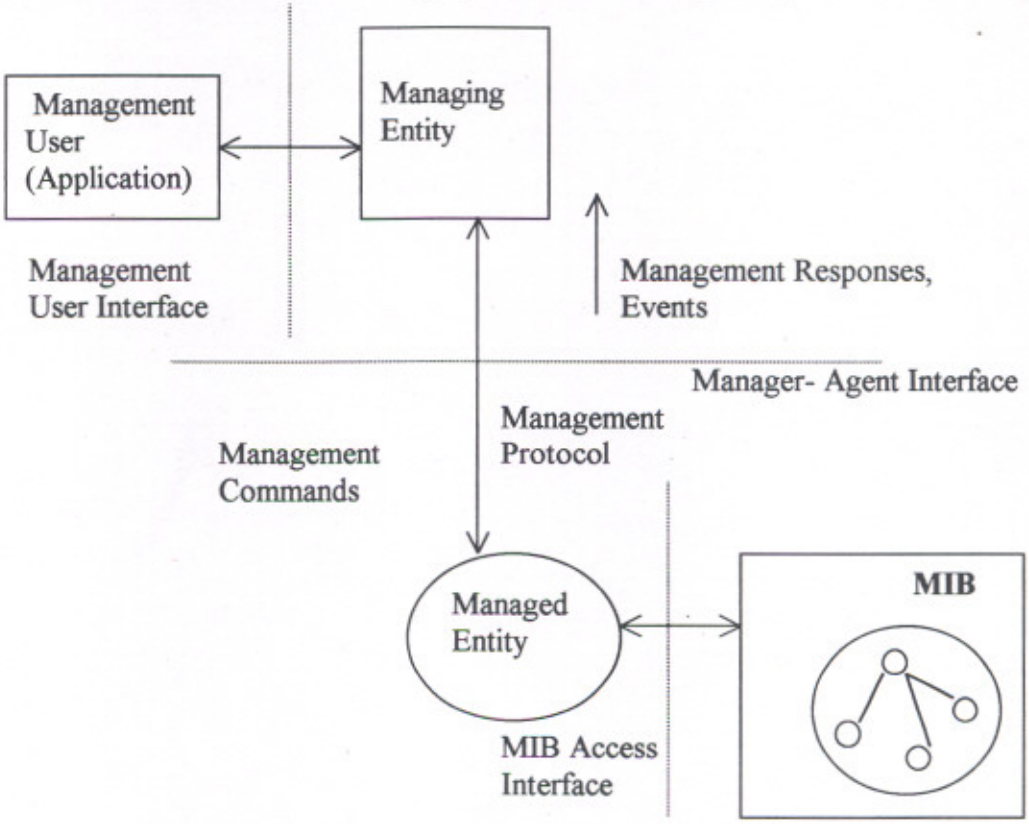


Fig 1.1 "Network Management Framework"

A *Managing Entity* provides a set of services to a Management User to control, monitor and configure a network. The managing entity passes management commands received from a user to one or more managed entities and acts upon responses or abnormal event reports received from managed entities. A Managed Entity carries out the commands received from the managing entity that performs certain operations on managed objects. A managed entity supports interfaces with the operational protocols and the MIB to access real resources through managed objects. A managed entity is also responsible for forwarding abnormal events to the manager. The bi-directional flow of information between a managing entity

and a managed entity is governed by a set of rules known as the Management Protocol. Both Manager and Agent support one or more management protocols at least one of, which is common between them.

A network management framework also defines the interfaces between components. These interfaces are shown in Fig. 1.1 and are described below:

User-Manager Interface: User management commands, command completion status and event reports are exchanged across this interface. Information flowing across this interface forms the highest level of abstraction of management information.

Manager-Agent Interface: This interface provides the rules governing the manager-agent interactions. The data (PDUs) flowing across this interface is governed by the management protocol used. Information (PDUs) exchanged across this interface includes transformed user commands and responses and event reports from the agent.

Agent –MIB Interface: The agent accesses the managed objects across the agent-MIB interface. Requests for operations on managed object flow across this interface from agent to MIB and responses to such operations are returned in the reverse direction.

A *Management User* is any application that uses the services provided by a management framework to perform network management functions. A management service is a facility that is made available to a management user by a management framework provider across the user-management framework interface. Management services include but are not restricted to: management object access and modification. MIB traversal to locate managed objects, event reporting and secure management information transfer.

A Management Model is a specific instance of the management framework. The framework specifies the components needed, whereas the model specifies the overall functional aspects

of the components and details of their interactions. A management model may be object oriented or non-object oriented.

Examples of Management Models are the Internet Management Model and the OSI management model. Fig. 1.2 shows that OSI and Internet Network Management Framework. In the case of the Internet Management Model, the managing entity is the Network Management Station (NMS), the managed entity is the agent, the management protocol is simple Network Management Protocol (SNMP) [15], SMI is defined in [22] and the MIB is MIB – II [21]. The resources modeled as management objects are those maintained by TCP, IP, ICMP etc. The OSI model is an example of an object-oriented model. The managing entity is an OSI Manager, the managed entity is an OSI Agent, and the MIB is specified using Guidelines for the Definition of Management Objects (GDMO) [8].

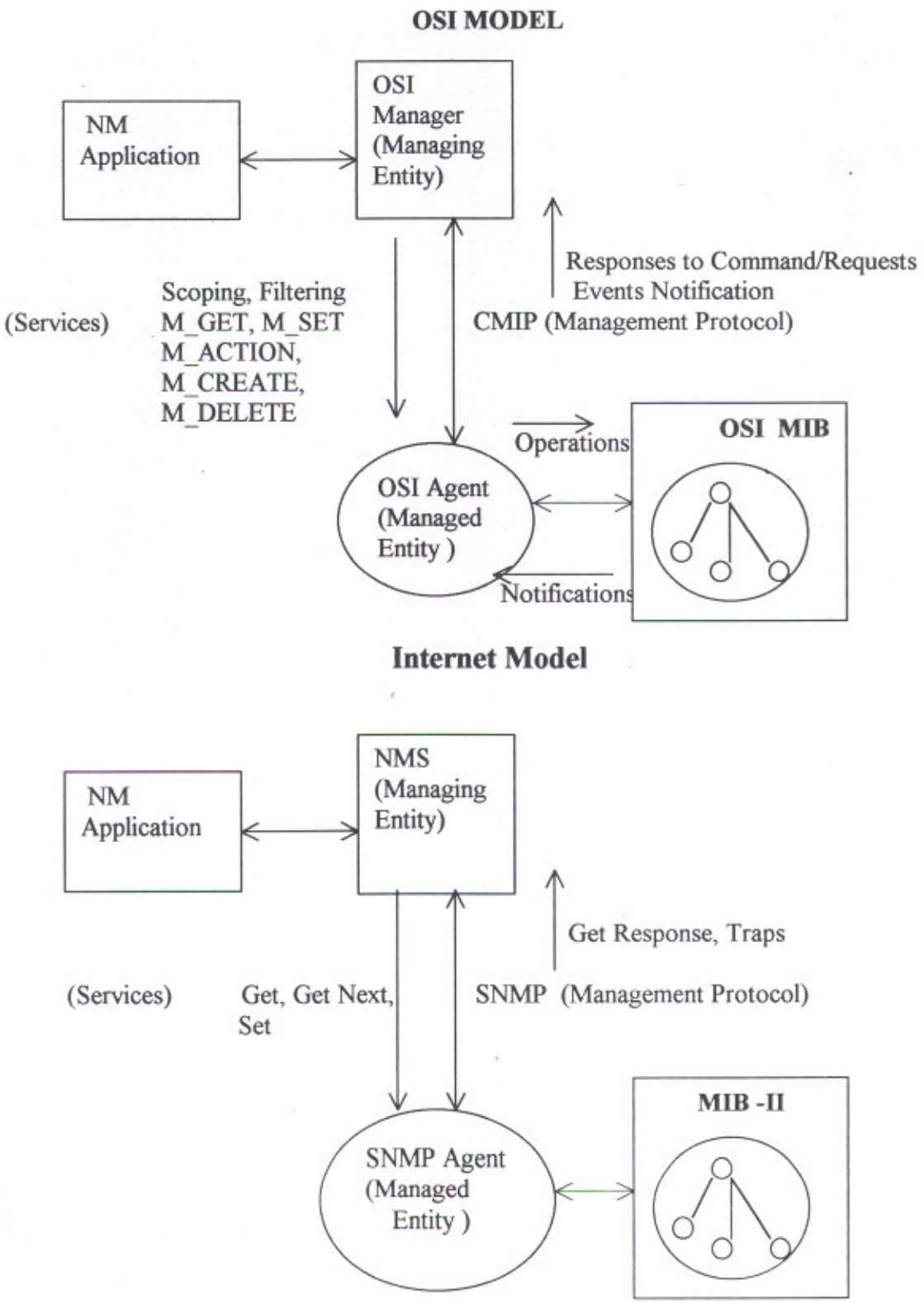


Fig.1.2 "Superimposition of Internet and OSI entities on the management framework"

1.2 SNMP MIBS AND OBJECT ORIENTED MODELING

The Internet-standard Network Management Framework is based on the notion of universal resource deployment. This may be alternatively stated by the fundamental axiom: "The impact of adding network management to managed nodes must be minimal, reflecting a lowest common denominator" [23]. Which can be summed up by the two words "simple" and "implement able".

In contrast the OSI standardization process attempts to achieve an all-encompassing framework, to meet any future management requirements. Since object oriented standardization is a self-perpetuating process a great deal of thought was initially placed into the underlying object oriented model so as to allow for the planned continual expansion. OSI can be summed up by the two expressions "complex yet powerful" and "relatively difficult to implement, relatively easy to extend."

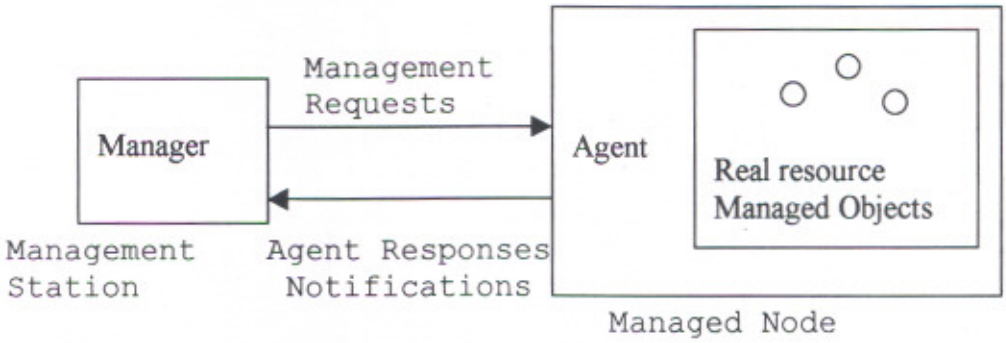


Fig. 1.3 "The Manager/Agent Model"

If we consider the manager/agent model shown in Fig. 1.3, then under SNMP the burden of management would be placed firmly on the management station, with only minimal impact on the more numerous managed nodes. Under OSI a more significant load is placed on the agents due to a greater expectation of the capabilities of managed nodes. Both camps set out with the same overall aim of achieving the effective management of heterogeneous

resources. One took a pragmatic approach and achieved exceptional market acceptance, the other attempts to provide a complete solution at the expense of its complexity. The future may prove that the simple foundations of the SNMP model cannot be continually built upon to meet the management needs of tomorrow's global open systems. Hence, **a neutral, reusable model, preferably an object oriented generic model is desired** which can cater to any legacy, in vogue, upcoming or future network management philosophy.

Management Information:

Each agent provides a management view of their underlying logical and physical resources, such as transport connections and power supplies, to the managing applications. Managed Objects provide an abstract view of these real resources. The Managed Object data is held in a management database called the **Management Information Base (MIB)**. Both SNMP and Object Oriented Paradigm (OOP) define schemata for the description of Managed Object MIB data, namely the Structure of Management Information **(SMI)** and the Guidelines for the Definition of Managed Objects **(GDMO)**, respectively [11].

The object oriented information model permits the refinement of existing Managed Object templates via inheritance see Fig. 1.4. Refinement may occur due to an increase in the capabilities of a given Managed Object, perhaps due to the evolution in the technology of the underlying resource. The OOP based model must apart from other intrinsic properties supports allomorphism, which facilitates the management of a given object as if it was an instance of any of the object classes in its inheritance hierarchy, thus permitting managing applications that have been coded to an earlier version of the information model, to continue to exercise control.

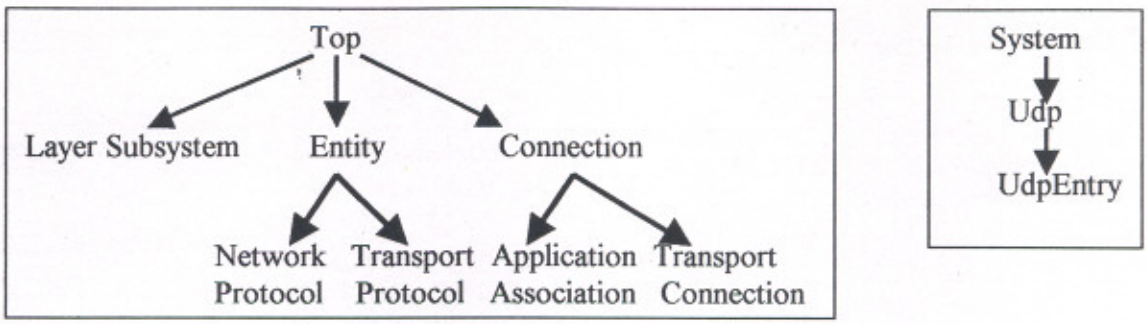


Fig. 1.4 "Example Inheritance and Containment Hierarchies"

The aggregation relationships between managed objects, such as "kind-of" and "part-of", are described by Name Binding containment descriptions. These containment descriptions yield a Managed Object instance hierarchy, which is termed the **Management Information Tree (MIT)** see Fig. 1.4. The MIT facilitates globally unique instance naming via Distinguished Names.

SNMP's information model is just object based and does not have full object oriented properties, is simpler than its OOP counterpart so as to reduce the complexity of the agent implementations. SNMP objects represent single, atomic data elements that may be read or written to in order to effect the operation of the associated resource. The SNMP SMI permits the variables to be aggregated into list and tables but there is no mechanism provided by SNMP to enable the manager to operate on them as a whole. Object identifiers are used, some would say misused, to achieve object instance naming, see Fig. 1.5. The syntaxes that each MIB variable may hold are a very much-reduced subset of the unlimited syntaxes that are permitted by the object oriented model.

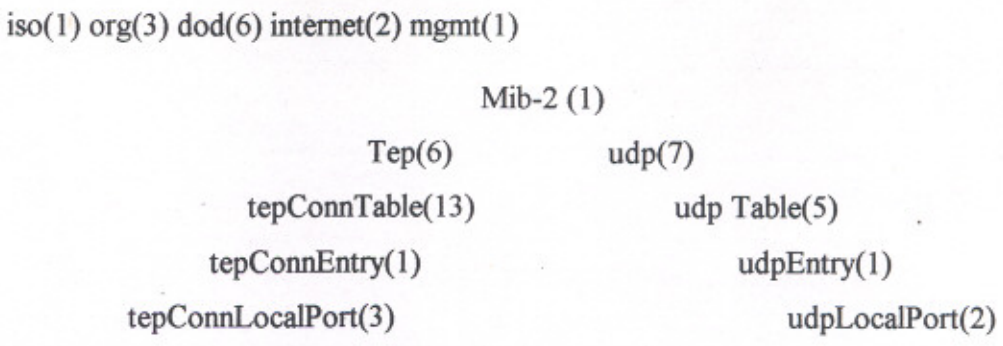


Fig. 1.5 "An Internet Management MIB Object Identifier Instance naming tree"

Protocol Operations:

OOP makes a distinction between the service offered by a layer and the underlying protocol that achieves those services, whilst SNMP makes no such distinction. A Common object passing services and protocol respectively defines OOP management's services and protocol. In placing the emphasis for Manager/Agent communications between asynchronous interrupt-driven and polling based approaches, SNMP selected "trap-directed polling", whilst OOP management adopted an event driven approach.

Upon an extraordinary event the SNMP agent emits a simple Trap notification to its manager, which must then initiate MIB polling to ascertain the full nature of the problem. Since Traps are both simple and unacknowledged their generation places only a small burden on the managed node. The manager may still need to poll important attributes periodically if the set of supported Traps are not sufficient to indicate the occurrence of all important error conditions.

Object passing supports extremely expressive and optionally acknowledged event reports, to the managing application, thus removing the need for any additional polling. The onus is placed on the OSI agent to inform the manager of significant events. The requirement for simplicity extends to the number and complexity of SNMPv1 protocol operations, compared with their OOP counterparts.

Object passing operations may include specifications for "scope" and "filter" so that the operation may be applied to some subset of the agent's managed objects. Scoping selects a sub-tree from the agent's MIT and filtering specifies a criteria, such as "those routing entries with a valid status", to select from the scoped objects.

Object passing operations are provided to retrieve and update attribute values. Since the usage of scoping and filtering means that the number of responses to an **Object-Get** (which are received in linked replies) will not necessarily be known when the request is sent, an **Object-Cancel-Get** operation is provided to prevent the possibility of the manager being over loaded. **Object-Create** and **Object-Delete** cause the creation and deletion of managed objects. **Object-Action** facilitates the execution of any supported imperative command such as running diagnostic tests.

SNMPv1 supports the retrieval of management information via Get and Get-Next primitives, the latter facilitating MIB traversal. Retrieval responses are limited to a single packet, which ensures that the manager will not be overloaded with response data, at the expense of requiring multiple retrieval requests to traverse an entire table. The Set primitive is used to update MIB objects, which, via side effects, achieves the control of imperative actions and the creation or deletion of table entries.

Transport Mappings:

Although SNMP is transport protocol independent, the connectionless User Datagram Protocol is the principal transport for SNMP. The "end-to-end" [16] argument makes a very strong case for leaving the selection of aspects such as the transport protocol to the application level, since only the application (in this case management) has a complete appreciation of its transport requirements. By selecting a connectionless protocol such as

UDP the management implementation is free to produce its own timeout and retransmission mechanisms.

At times of **network congestion** the SNMP implementation can then configure an appropriate level of retransmissions to increase the chances of successful management when the network itself is failing. The application can more readily determine when some form of out-of-bands communication is essential. This approach requires that each SNMP implementation must attempt to produce its own transport mechanism that will not end up accentuating any problems of network congestion.

1.3 PROTOCOL NEUTRAL MODELING

Definition: "The process of describing a system abstractly using a standard methodology or language independent of the technology employed to implement the system".

Dissecting:

- Abstractly (e.g. graphical notation)
- System (IT TMN, ATM machine)
- Standard methodology (e.g. G.851.1, OMT)
- Language (e.g. UML)
- Technology (GDMO, CORBA, COM, ...)

Instead of converting from SNMP to Object Oriented Paradigms (OOP) create models in UML first, separate from the protocol, and then convert to SNMP or OOP from a neutral form UML. When needed convert any UML object to SNMP or OOP as needed.

So Protocol Neutral Modeling is the specification of models that are independent for their implementation target protocols.

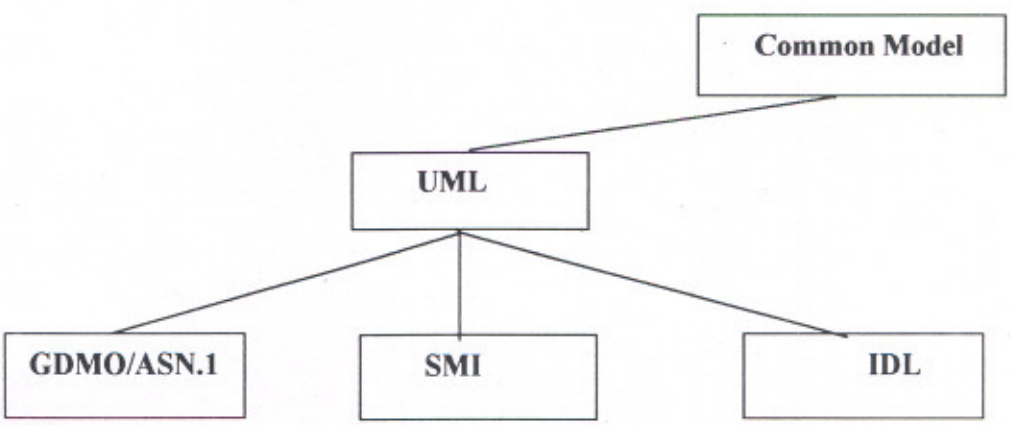


Fig. 1.6 "Protocol Neutrality"

Why Do We Model (Importance):

- Provide a structure for problem solving
- Experiment to explore multiple solutions.
- Reduce time to find problem solution.
- Decrease development cost.
- Mange the risk of mistakes.
- Compatible management protocol.

Advantages of Protocol Neutrality:

- Promote separation of software concerns.
- Increase implementation flexibility.
- Facilitate reuse.

Advantages of Protocol Neutral Model In UML:

- UML is an OMG standard.
- Strong at specification.
- Strong at dynamic exchange
- The artifacts of software system and their interfaces.

1.4 SUMMARY

The ultimate goal of computer networks is universal connectivity. In order to achieve this goal network all over the world must be connected together. More importantly, these connected networks must work together in a cooperative manner to satisfy the end user needs. Ironically, the existence of multiple standards prevents the attainment of this goal. The concept of **protocol neutral modeling** provides a possible solution to this problem.

The growth of user needs and advances in networking and computing technology has given rise to increasingly large networks. Performance and reliability have become major concerns. Networks are now too complex to be easily managed by human managers alone. Network Management aids the task of human managers in achieving their complex, sometimes conflicting, goals by providing facilities for monitoring and controlling the operation of the network and its components.

Research in Network Management has made rapid progress over the past few years. Nevertheless, different network management models exist, with each model conforming to a different set of standards. The need of the day is to make these models intemperate so that artificial barriers that impede efficient network management are broken.

Whether driven by technological merit, simplicity of development or Government profiles, considerable investments have been made and will continue to be made into the provision of network management solutions based on the two dominant management protocol architectures, namely SNMPv1 and Object Oriented model like OSI and CORBA based models. They exist together so they must be made to co-exist, so as **to achieve global inter-working across heterogeneous platforms in the management domain.**

It is the contention of the present work that co-existence can most readily be achieved by selecting a semantically rich reference model as the basis for this inter-working. Such an approach can then be readily extended to encompass coming up technologies such as CORBA [OMG91] or DCOM, together with architectures that have not yet bridged the synaptic gap in the collective minds of standards bodies and manufacturers' consortia.

The collaborative work of the Network Management Forum's (NMF) ISO/CCITT and Internet Management Coexistence (IMC) activities has provided a sound basis to achieving co-existence through automated application level gateways. **Through out this thesis we shall use the terms "gateway" to indicate the automated translation of information and protocol models, which have been modeled in UML,** so as to achieve the representation of management objects defined under one proprietary paradigm (in TMN terms) under that of an universal model.

The rest of the thesis is organized as follows.

Chapter 2. Consist some background information, the brief of OO Paradigm, and Management Information Model. ***Chapter 3.*** Describe the integrate approach of existing SNMP and OOP models with detail aspect of Protocol Neutral Modeling. ***Chapter 4.*** First motivate why a UML is important, and then describe how conceptual MIB models can be represented as a variation of the UML class diagram. This chapter also briefly presents an example UML diagram. ***Chapter 5.*** Conclude the thesis work with present industrial trends and possible future directions.

CHAPTER 2.

REVIEW OF LITERATURE

2.1 THE MANAGER-AGENT MODEL

Object Oriented and SNMP (Internet management) have adopted the manager-agent paradigm. Managed objects at different levels of abstraction model manageable resources. Management interfaces can be thought of as **"exported"** by applications in agent roles and **"imported"** by applications in manager roles. Manager applications access managed objects across interfaces in order to implement management policies. Standardization affects the way in which management information is modeled and carried across systems, leaving deliberately unspecified aspects of their internal structure. The manager-agent model is shown in Fig. 2.1. Note that manager and agent applications contain other internal objects that support the implementation of relevant functionality. Since these are not visible externally, they are depicted with empty circles.

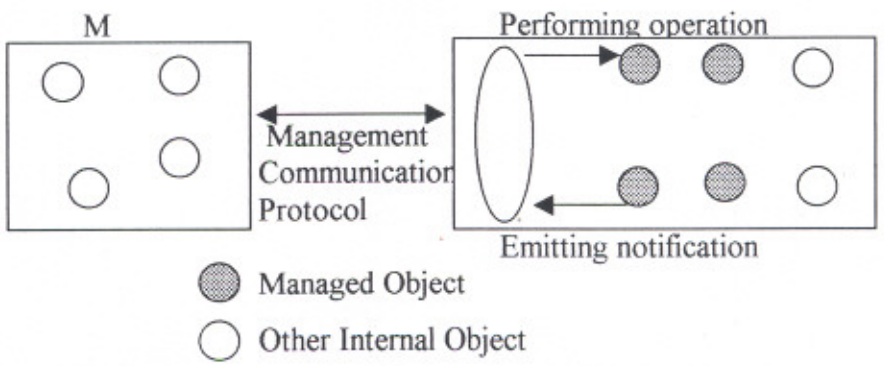


Fig. 2.1 "The Manager-Agent Model"

The manager and agent roles are not fixed and management applications may act in both roles. This is the case in hierarchical management architectures such as the Telecommunications Management Network (TMN) [9].

Management applications may act in dual manager-agent roles, in either peer-to-peer or hierarchical relationships. Fig. 2.2 shows three types of management organization according to the manager-agent model: centralized, flat and hierarchical.

The centralized model is best exemplified by SNMPv1 Management Operation Centers (MOCs). The flat model reflects the evolution of SNMPv1 to SNMPv2, with "manager-to-manager" capabilities. Finally, the hierarchical model is best exemplified by the TMN, which uses OO management as its base technology.

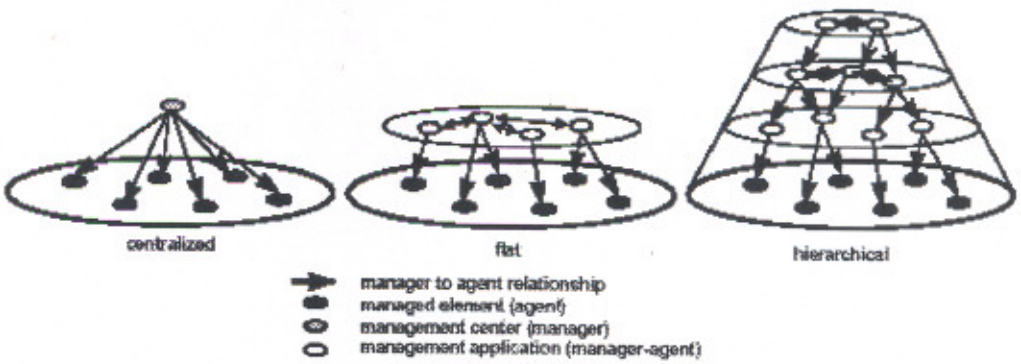


Fig. 2.2 "Models of Management Organization"

2.2 OBJECT ORIENTED PARADIGMS (OOP) - OSI AND CORBA

ISO/ITU-T Open Distributed Processing (ODP) [10] is a general framework for specifying and building distributed systems. The Object Management Group (OMG) Common Object Request Broker Architecture [CORBA] can be seen as its pragmatic counterpart. **While SNMP and OSI management are *communications* frameworks,** ODP/OMG CORBA target a *programmatic* interface between objects in client or server roles and the underlying support environment, that is the Object Request Broker (ORB). Server objects are accessed through interfaces on which operations are invoked by client objects in a location transparent fashion. Choices made by the Internet Engineering Task Force (IETF) and

ISO/ITU-T on the one side and OMG on the other side reflect their different pre-occupations: management communications for the former and distributed software systems for the latter.

The difference in approach is sometimes referred to as "vertical" versus "horizontal" interfaces. Vertical interfaces standardize communications interactions between systems. The horizontal approach standardizes Application Programming Interfaces (APIs), which are used to "plug" application objects on the global supporting infrastructure. The latter is also referred to as the Distributed Processing Environment (DPE) and encapsulates the underlying network, hiding heterogeneity and providing various transparencies. The ODP / OMG CORBA model is shown in Fig. 2.3 [2].

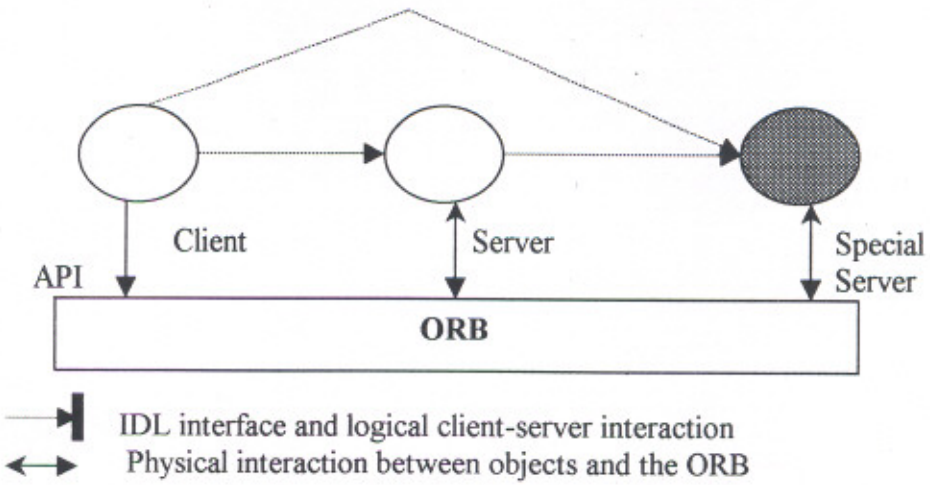


Fig. 2.3 "The OMG CORBA Model"

The OMG CORBA paradigm is that of a client-server, with distribution provided through the ORB. The unit of distribution is the single object as opposed to the OSI and Internet object cluster that is visible across an interface. Client and server CORBA objects communicate through the ORB, whose services are accessed through standard APIs.

Interoperability is achieved through the formal specification of server interfaces, the ORB APIs and the underlying inter-ORB protocols. One key difference to OSI and Internet management is that the object model and APIs have been addressed first, while the underlying protocols may be replaced. Of course, interoperability dictates an agreed protocol but the rest of the framework is not heavily dependent on it. The key benefit is portability of objects across different CORBA implementations due to the standard ORB APIs and the various transparencies that are (or will be) supported by the latter. Note that communication aspects are "hidden" inside the ORB and, as such, are not shown in Fig. 2.3. While OMG CORBA is a general distributed systems framework, its object-oriented nature and the fact that management systems are composed of interacting objects suggest that it could also be used for management.

2.3 BACKGROUND INFORMATION: ASN.1

Some necessary background information concerns the OSI Abstract Syntax Notation 1 (ASN.1) [12] language. This is an abstract "network" data structuring language that supports simple and constructed types. A particularly important ASN.1 type is the Object Identifier (OID), which expresses a sequence of nonnegative integers on a global registration tree. OIDs are registered by standards bodies, for example ISO, ITUT and IETF, and are used instead of friendly string names to avoid ambiguities. For example, the OSI *objectClass* attribute name is registered as {joint-iso-ccitt(2) ms(9) smi(3) part2(2) attribute(7) objectClass(65)}. ASN.1 is used in both the OSI management and SNMP frameworks to specify the management protocol packets and the structure of managed object information, for example attributes, operation parameters / results and notification information.

The ASN.1 Primitive Types: The ASN.1 primitive types are listed in Table 2.1. These types are built into the language and from the building blocks for more complex types. The names of these types are reserved words, and, like all ASN.1 reserve words, are always written in upper class letters [1].

PRIMITIVE TYPES	MEANING
INTEGER	Arbitrary length integer
BOOLEAN	TRUE or FALSE
BIT STRING	List of 0 or more bits
OCTET STRING	List of 0 or more bytes
ANY	Union of al types
NULL	No types at all
OBJECT IDENTIFIER	OBJECT NAME (E.G. A LIBRARY)

Table 2.1 "The ASN.1 Primitive Types."

The ASN.1 Constructors: The primitive types can be combined to build more complex types. Table 2.2 shows the five principal constructors used in ASN.1 for this purpose.

CONSTRUCTORS	MEANING
SEQUENCE	Ordered list of various types
SEQUENCE OF	Ordered list of a single type, like an array
SET	Unordered collection of various types
SET OF	Unordered collection of simple types
CHOICE	Any one type taken from a given list

Table 2.2 "The Principal ASN.1 Constructor"

2.4 MANAGEMENT INFORMATION MODEL

A management framework and associated technology should be applicable to network, service, system and distributed application management. In addition, the applications and support infrastructure that constitute the management system should also be manageable. The ideal information model must cope easily with management information related to such a variety of management targets. At the same time, it must impose a measure of uniformity on the structure of management information so that it is possible to devise a set of generic management operations that are applicable in all management contexts.

The original motivation for developing the three frameworks was different: network management for SNMP; network and service management for OSI; and distributed application operation and management for OMG CORBA. As a result, the relevant information models, despite exhibiting similarities, have important differences.

In the case of SNMP, the information model is often referred to as Object-based, with classes and inheritance regarded as unnecessary complications and thus deemed undesirable. All three models are quite general and, despite their differences, anything that can be modeled in one can also be modeled in another.

However, such specific translations would not be of use/reuse as any progress in SNMP or the OO Paradigms or any other development would entail a revision of the translation gateway. **The key issue is the existence of greater expressive power offered by a protocol neutral modeling. The answer is an affirmative and the language is UML** (Unified Modeling Language) results to the better abstraction of management information and is worth the price of the additional complexity whether the greater expressive power one

offers results to the better abstraction of management information and is worth the price of the additional complexity.

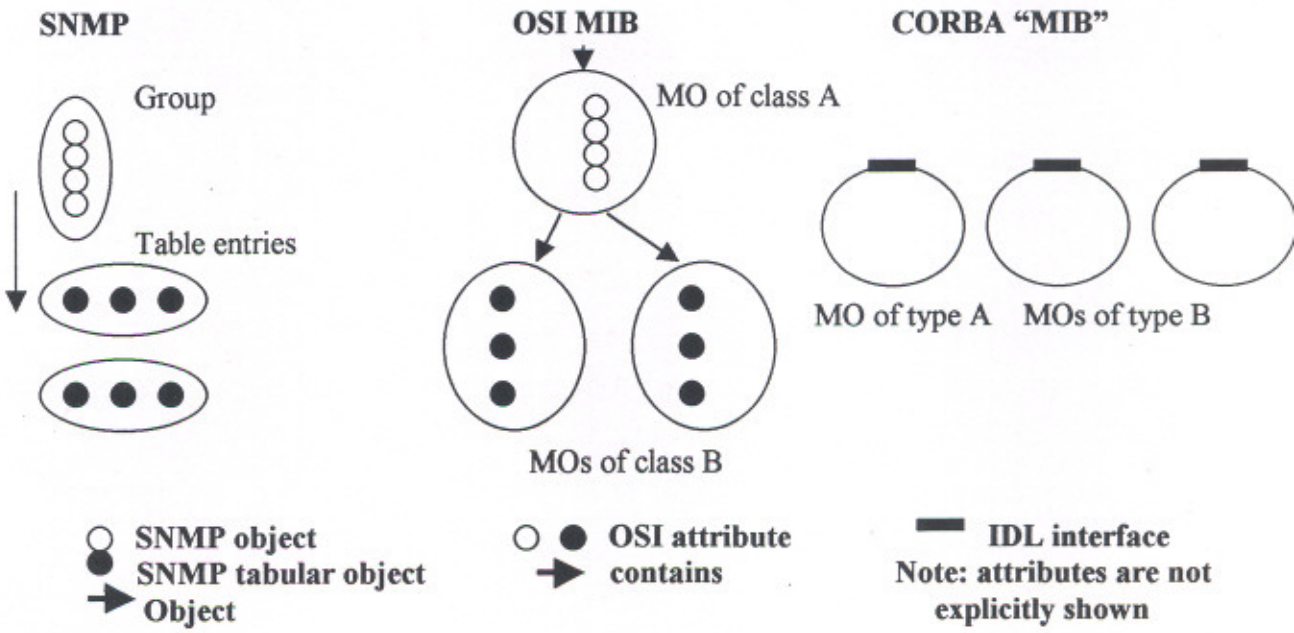


Fig. 2.4 "SNMP, OSI and CORBA Management Information Base"

Fig. 2.4 shows a pictorial view of the notion of objects in the three frameworks and their collective view as a **Management Information Base (MIB)** across a management interface.

2.5 SNMP INFORMATION MODEL

SNMP information modeling principles are collectively referred to as the Structure of Management Information (SMI) and are specified in [20] for SNMPv1 and in [14] for SNMPv2, the latter being an extension of the SNMPv1 model.

The basic building block of an SNMP MIB is the *object*. Objects belong to a particular *object type* and have *values*. According to the SNMP SMI, object values are restricted to a very small set of allowed syntaxes, resulting ultimately in the basic ASN.1 types INTEGER, OCTET STRING and OBJECT IDENTIFIER. Other *application-wide* types such as Counter, Gauge, Network Address and Time ticks must resolve to either integer or string scalar types. The only constructed type allowed is a simple two-dimensional table, consisting of elements of the previous primitive types.

Great emphasis is given to the fact that the allowable syntaxes are few, simple and scalar. The key advantage claimed for this approach is that object values carried across a network must be encoded in a "network-standard" way. Encoding and decoding can be computationally expensive, especially if the syntaxes involved are complex. Because SNMP uses only a small fixed set of syntaxes, it is possible to hand-code the encoding and decoding software in an optimal fashion. Thus, the SNMP SMI is analogous to a computer language that has a small set of basic types together with two-dimensional arrays.

The SNMP SMI defines a notation for specifying the properties of new object types, an ASN.1 *macro* called OBJECT-TYPE, while ASN.1 is used to specify the object syntaxes and the tabular structure. SNMP object types can be either single or multiple instanced, with multiple instanced objects allowed only in tables.

SNMP object and ASN.1 Table

SNMP objects are similar to OO attributes while there is no notion of a "composite" object-boundary that encapsulates a number of scalar objects modeling a manageable entity. The only composite relationship relates objects to tables.

A table has rows (also referred to as table entries or records), with each row represented by a *SEQUENCE* ASN.1 type that contains a statically defined number of objects. The table itself is modeled as *SEQUENCE OF* ASN.1 type with respect to the rows or entries, allowing an arbitrary number of those to be dynamically instantiated. A table thus resembles a "dynamic array of records". Tables may grow to arbitrary length but must always be of fixed width. A further rule restricts the syntaxes used within a row to be "scalar"; **thus one cannot define tables within tables.**

[*Note also that tables and table entries are only conceptual composite objects in SNMP: only the individual scalar objects that constitute a table entry are accessible through the management protocol.]*

Let us examine the use of **this modeling framework through an example**.

A typical single-instanced set of objects are, for example, those modeling aspects of a connection-oriented transport protocol entity such as the ISO Transport Protocol (TP) and Internet Transmission Control Protocol (TCP). Such objects will cover the number of current and previous connections, the number of incoming and outgoing unsuccessful connection requests, the number of transport packets sent, received and retransmitted and the number of various protocol related errors. All these will have to be separate objects, loosely related through a "transport protocol" group.

[Note that this model does not support multiple instances of a transport protocol per node. If the latter was necessary, a table of transport protocol entries would be needed, but ***we will overlook this restriction for simplicity****].*

The group will also comprise transport connections, which are multiple-instanced objects and have to be modeled through a table. A tpConnTable may be defined as a SEQUENCE OF tpConnEntry, with tpConnEntry being a SEQUENCE of objects modeling various aspects of the connection, such as source and destination access points, the connection state, and so on. Fig. 2.4 shows objects of a single-instanced group, for example tp group, and two table entries, for example tpConnEntry. Note that both the group and table entries are depicted with dotted lines since there is no notion of a composite object boundary in the SNMP information model.

When objects are instantiated, they must be named so that they can be addressed unambiguously.

SNMP framework:

The SNMP framework uses object identifiers for naming. Every *object type* has a registration OID, while the object type OID, suffixed by a part that identifies uniquely that instance, identifies an object instance.

For nontabular objects any suffix would do, so the minimal *.0* is used. For example, *tpCurrentConnections.0* is the instance of the object type *tpCurrentConnections*. In the case of multiple-instanced tabular objects such as the *tpConnState of the tpConnEntry,* the suffix needs to signify the table entry. The latter can be constructed from the values of one or more objects of that entry that constitute the "key", as specified in the relevant OBJECTTYPE template for that table entry.

In our example, the values of the *tpConnSrcAddr* and *tpConnDestAddr* objects may be used as they uniquely identify each connection.

For example, *tpConnState.123.456* is the instance of the object type *tpConnState*, corresponding to the connection with source address 123 and destination address 456.

[Note: that when object values are strings as opposed to integers, they need to be converted to object identifier suffixes: the SNMP SMI specifies rules for this conversion].

The SNMP naming architecture exhibits a very tight coupling between object type and instance identifiers. The problem with it is that instances of a particular object type can only appear in one particular place in the registration tree. This means that one cannot define generic object types to be used in several contexts.

For example, a common object for representing the state as perceived by a managed resource is the *operationalState*. In the SNMP framework, it is not possible to define such a generic object but specific objects have to be defined for every particular context, for example *tpOperationalState*.

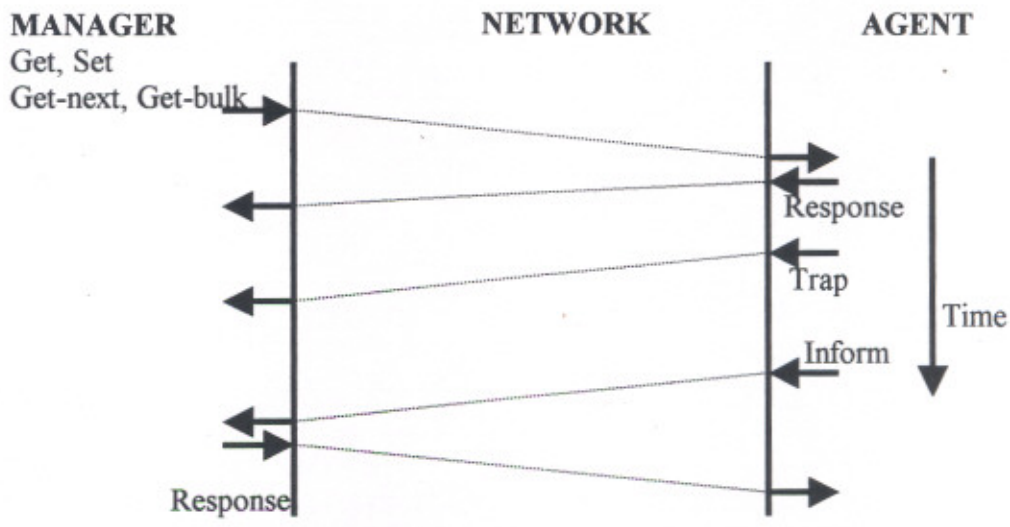
An additional problem is that because this is not the intended use of OIDs, it has been found that most SNMP implementations spend a lot of processing time scanning through object names, trying to separate type and instance information.

Finally, **object names formed through OID suffixes are not natural.**

SNMP operations:

SNMP objects accept only *Get* and *Set* operations. The Set operation can be performed only to objects that have a *read-write* or *read-create* access level according to the object specification. Access to objects with Get and Set operations is subject to the access control policy across a management interface.

The possible interactions using SNMP operations are shown in Fig. 2.5.



Note1: Inform is only allowed for dual agent-manager entities
Note2: Get-bulk and inform have been added in SNMPv2

Fig. 2.5 "The SNMP Protocol Interaction"

SNMP objects support neither *Create* nor *Delete* operations explicitly. Create and delete semantics are implicitly supported however for multiple-instanced objects, that is table entries or rows.

In SNMPv1, row creation may be requested through a Set operation for an entry that is currently not in the table by setting the values of all the row objects. This type of Set operation is interpreted as table entry creation; however, this behavior is not mandated and implementations may behave differently.

In addition, the SNMP protocol limits the maximum Protocol Data Unit (PDU) or packet size, so it might not be possible to pass values for all the row objects! SNMPv2 remedies this defect and uses an elaborate interaction scheme to ensure atomicity of row creation.

This is supported through the use of a *RowStatus* object which must be present in any table that allows row creation. Creation can follow either a *createAndGo* or *createAndWait* protocol, according to the value of row Status. The former is similar to the SNMPv1 creation style and restricted by the maximum PDU size. In the latter, the values of the row status object progress successively through *notReady -> notInService -> active* while the row is being created through more than one Set requests.

Row deletion is achieved through a Set operation that sets the value of row status to *destroy*.

Although all this works, it is certainly not simple. In fact, a high price has been paid in order to avoid introducing Create and Delete object operations through separate protocol primitives.

In both the OO models, objects may accept arbitrary actions that operate on the object boundary. ***Given the fact that SNMP objects are essentially attributes compared to OO objects, imperative actions with arguments and results are meaningless.***

Nevertheless, actions are necessary and may be modeled in SNMP by objects that support the arguments and results of an action. An "action" may be emulated by a Set request, followed possibly by a Get request to retrieve the results.

For example, a "reboot" action may be modeled by a Boolean *rebootState* object whose value is set to *true* (in this case, there is no action result). This type of emulation is not very elegant and may result in complex interactions and an awkward object model for imperative commands with complex "argument and result" parameters.

In SNMPv1, agent applications may emit notifications called *traps* associated with the SNMP protocol rather than a MIB specification. These are supposed to be used only for a small and pre-defined set of events (*warmStart*, *coldStart*, *linkUp*, *linkDown*, *authenticationFailure*).

However, some MIB designers (notably those of the Remote Monitoring MIB) have extended the concept and have provided notations that allow MIB designers to specify resource-specific notifications and the information, which should be included in them.

A notation of this type has now been included in the SNMPv2 SMI [14].

CHAPTER 3.

SNMP AND OO PARADIGM - INTEGRATION APPROACH

3.1 INTER-WORKING

In this section, we examine inter-working aspects for the three different technologies. We look first at inter-working between OSI and Internet management in the TMN context, where SNMP-capable network elements may need to be managed by TMN applications.

3.1.1 OOP and SNMP: Translation

Inter-working between OSI management and SNMP is mostly necessary to manage SNMP-capable network elements in a TMN fashion. This is particularly common for ATM equipment, as relevant SNMP information models have been available for some time before the relevant TMN recommendation, resulting in a number of SNMP-capable ATM elements in the market place. In transmission technologies such as SDH, the situation has been the reverse with early Q-compliant available elements. In general though, it is expected that in the short to medium term i.e. for the next few years, it will be necessary to manage SNMP-capable elements from a TMN environment.

Inter-working between OOP and SNMP has been a subject that led to a lot of research trying to bridge the two worlds. Various solutions have been proposed, all of which can be classified into two broad categories:

i. integration in the manager end; and

ii. integration in the agent end of the manager-agent model.

The integration in the manager end means that one accepts the diversity in the supported technology by managed elements and tries to provide element management applications that

understand the different underlying information models and access mechanisms. **This approach which has been developed in the present thesis using UML and we shall refer to it as the protocol neutral or the UML** approach and in this applications will need to understand both the OO management and SNMP management models. Such an approach has been envisaged in this study as stated previously.

The UML based approach would uniform access to both OOP and SNMP services. In such an approach, it is difficult to conceal which model the managing application deals with since both the underlying information models and access mechanisms have important differences. For example, the nature of objects and the naming schemes are different in the two models. This is also the case with respect to the supported communication and access paradigms. Of course, this does not mean that such an approach is not feasible but simply that integration cannot be seamless, increasing significantly the development effort and investment for dual manager applications, if the PROTOCOL NEUTRAL MODEL is not powerful or neutral. The dual stack manager approach is depicted in the left part of Fig. 3.1.

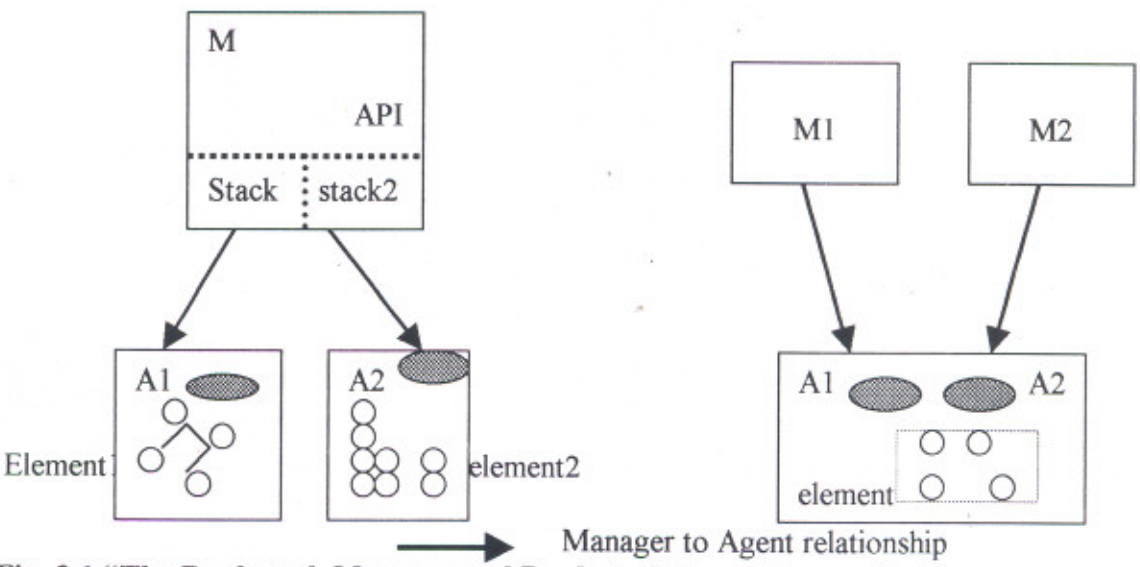


Fig. 3.1 "The Dual-stack Manager and Dual-stack Agent Approaches"

Integration by agents can again be classified into two broad categories:

i. The dual agent approach; and

ii. the application gateway approach.

In our approach based on UML we have two agents that exist for every managed element, both an SNMP and an OSI (or any other) one. The information models will be semantically similar, which implies that the associated "real resource" aspects could be the same. As such, investment in providing such agents could be reduced if a modular approach were followed.

In the latter, managed objects are realized in a "model/protocol independent" fashion and are associated with both SNMP and OOP access methods. This approach is depicted in the right part of Fig. 3.1, in which the objects are model independent, with different views presented through the two agents. Despite the fact that this approach is technically feasible and attractive, it requires heavy investment and additional resources in managed elements. As such, no products support this type of functionality to date [5].

The work has been simplified in our work by the following observation.

The protocol neutral conversion, which resides between SNMP and any other framework, relies on the simple observation that the SNMP structure of management information is a pure subset of the OSI one. SNMP objects are equivalent to OSI attributes, groups are mapped to classes and table entries become separate classes.

The *specific translation gateway* approach provides a solution for integrating the two frameworks. In this, an application acts as a gateway (*proxy* or *adapter* are two other terms often used) for one or more agents of the other framework, exporting "converted" information models and providing service conversion from one access method to the other.

The conversion between the information models can be performed either manually or automatically.

Manual conversion means that human heuristics may be applied to result in an "elegant" model. In many cases, the target model for that technology may already exist, in which case a gateway should simply map one to the other.

Automatic conversion means that a well-defined set of rules exists and can be used to automate translation of any MIB specification from one model to the other. As a result of automatic conversion rules, the dynamic interaction translation and subsequently application gateways may be automated.

But such an approach is never scalable, the conversion is usually unidirectional: it can only be bi-directional if the two information frameworks are equally powerful and expressive. This is not the case with OSI management and SNMP, GDMO being much more powerful than the SNMPv1/v2 SMI. As such, there may be automatic conversion of an SNMP information model to the equivalent GDMO one but not vice versa. Human intervention is required for mappings in the opposite direction. For example, there is no deterministic method for emulating a CMIS/P *Action* through an SNMP *Set*.

Typically, every time a new element with an "unknown" MIB needs to be adapted for, an off-line procedure is involved to let the gateway "know" of this MIB through suitable translators/compilers. The latter generate run-time support in a data-driven fashion so that the gateway logic does not need to be altered. The specific translation gateway is shown in Fig. 3.2.

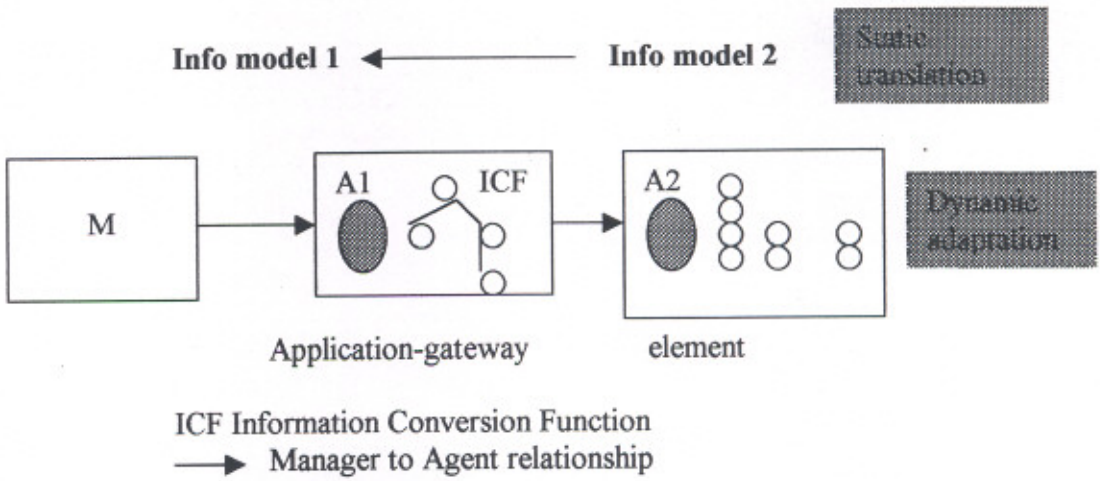


Fig. 3.2 "The Application-gateway approach"

One more key drawback is that the resulting information model does not exploit the object-oriented aspects of the OO Models: inheritance is only two-level (every class inherits only from top) while containment is also fairly "flat". Furthermore, the resulting information model needs to be standardized in order to be considered a standard interface in TMN terms.

The important aspect of the protocol neutral approach is that investment is rather small compared to the end result, which is OO manageability of any SNMPv1/v2 capable element. The key benefit OO management brings to the SNMP world is event-driven management through the Systems Management Functions [6].

For example, metric and summarization functions together with event reporting and logging may be used to provide sophisticated management capabilities, eliminating polling in the local environment between the gateway and the proxy SNMP agent.

3.1.2 Protocol Translation And Service Emulation:

If management models that need to interoperate support different management protocols, there is a need to (bi-directional) map the protocols in order to ensure flow of management information between the manager and the agent. Such a mapping is achieved using a technique called protocol translation. It is a systematic, deterministic, bi-directional mapping of PDUs. The translation does not involve functional mapping of management services. For protocol translation to be complete each PDU in one protocol must be mapped onto one or more PDUs in the other protocol along with associated parameters and error codes.

Fig. 3.3 "Shows the mapping between OOP and SNMP"

Service emulation is adopted as a solution when services supported by one management model are different from those of the others. Service emulation is defined as the mapping of services offered by one model into services offered by another model.

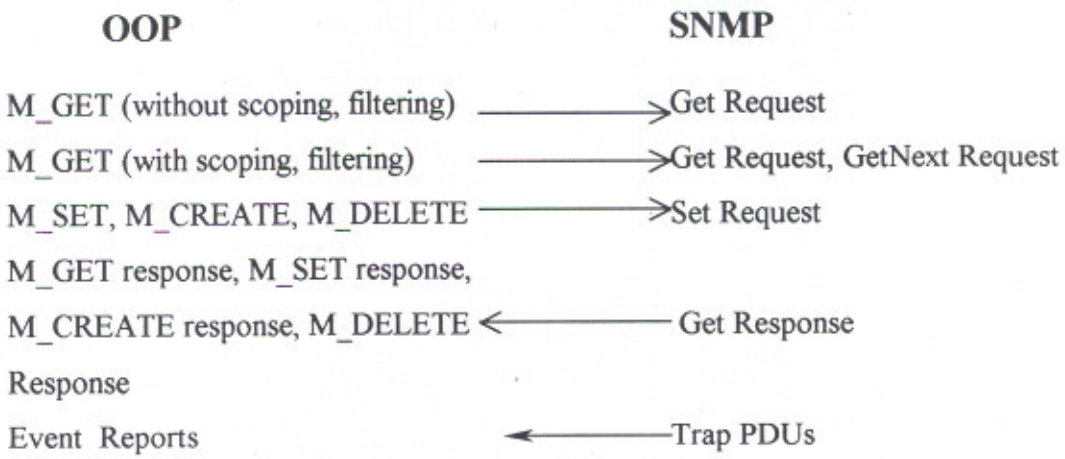
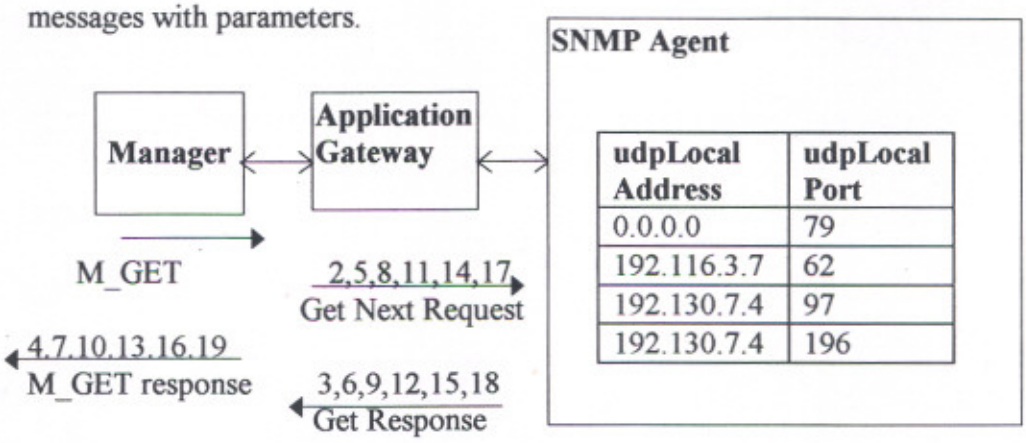


Fig. 3.3 "Protocol Translation"

The primary need for services emulation is to inter-network management models supporting services that are not one-to-one correspondent. While Neutral Protocol Translation ties to perform mapping at a PDU level, service emulation attempts to map a service (one or more PDUs) in one management model to a services (one to more PDUs) in another management

model. Fig. 3.4 shows a M_GET request with full sub-tree scooping to retrieve the udp group of objects maintained by the SNMP agent. This figure also shows the flow the messages with parameters.



1. M_GET(InvokeId=44, MOC=udp, MOI={udp Name="0"}, Scope=FullSubtree, Filter=NULL, Access=None, Synch=BestEffort, Attrs=NULL)

2. GetRequest(RequesteId=71, ErrorStatus =0, ErrorIndex=0, VarBind={(udpInDatagrams.0,0), (udpNoPorts,0,0), (udpInErrors,0,0), (udpOutDatagrams0,0)})

3. GetResponse(RequesteId=71, ErrorStatus =0, ErrorIndex=0, VarBind={(udpInDatagrams.0,1249), (udpNoPorts,0,8), (udpInErrors,0,23), (udpOutDatagrams0,2593)})

4. M_GET response(InvokeId=44, LinkId=44, MOC=udp, MOI= {udpName="0"},...., Attrs ={udpInDatagrams=1249, udpNoPorts=8, udpInErrors=23, udpOutDatagrams=2593})

5. GetNextRequest((RequesteId=72, ErrorStatus =0, ErrorIndex=0, VarBind={(udpLocal Address,0), (udpLocfalPort,0)})

6. GetResponse((RequestId=72, ErrorStatus =0, ErrorIndex=0, VarBind={(udpLocal Address,0.0.0.0.79.,0.0.0.0), (udpLocfalPort,0.0.0.0.79,79)})

7. M_GET response(InvokeId=44, LinkId=44, MOC=udpEntry, MOI={udpName="0"@udpEntryName="0.0.0.0;79"},-----, Attrs={(udpLocal Address,0.0.0.0.), (udpLocfalPort=79)})

8. GetNextRequest((RequestId=73, ErrorStatus =0, ErrorIndex=0, VarBind={(udpLocal Address,0.0.0.0.79.0), (udpLocfalPort,0.0.0.0.79.0)})

9. GetResponse((RequestId=73, ErrorStatus =0, ErrorIndex=0, VarBind={(udpLocal Address.192.116.3.7.62,192.116.3.7), (udpLocfalPort,192.116.3.7.62.62)})

10. M_GET response(InvokeId=44, LinkId=44, MOC=udpEntry, MOI={udpName="0"@udpEntryName="192.116.3.7;62"},-----, Attrs={(udpLocal Address, "192.116.3.7), (udpLocalPort=62)})

:

:

11. GetNextRequest(....,VarBind{(udpLocalAddress.192.130.7.4.196.0), (udpLocalPort.192.130.7.4.196.0)})

12. GetResponse (..,VarBind(udpLocalPort,0.0.0.0.79.79),egpInMsgs,0.24)})

13. M_GET response(InvokeId=44, response indicating completion)

Fig. 3.4 "Service Emulation"

3.2 INTEGRATION PURPOSES

A large network management environment may use different management technologies such as OS and SNMP in different subsystems. Integration is thus required in order to communication betwee these subsystems.

The integration's goal is also to strengthen the existing solutions by introducing technologie which offer better features in certain functional areas.

For example, we distributed technology to provide the management system with a distribute environment. This introduction requires the integration of the distributed technology with th existing management solutions.

It is also useful to provide a domain-independent (or protocol-independent) applicatio development and deployment environment. In a large telecommunication network, it is commo that different network equipment and network technologies (ATM, SDH, SONET, WDM, Fram Relay, etc.) are used. They are normally managed by different systems using different managemen

technologies (CMIP, TL1, SNMP, CORBA, Java). It is always desirable but hard to develop network management systems supporting transparent end-to-end applications from the individual underlying management technologies. The problem becomes more acute when the migration of these technologies takes place. In many cases, the integration forces the change in the management applications-a painful process that should be avoided at any cost.

Finally, technology integration also helps migrating the legacy systems to more advanced technology frame works for application development and deployment. Integration is also a common mechanism to allow legacy systems to be used in a newly developed system to reduce the development cost.

In network and service management, perhaps the most important purpose of the integration is to achieve the following two management scenarios:

- Domain (or protocol) independence managers and applications.
- Service and network management integration.

3.3 INTEGRATION APPRAOCHES

As discussed above there are different approaches to integration, and each for a different purpose. We do not intend to discuss all integration methods; rather we concentrate on the processes to achieve the integration. Given the prominence of the SNMP technology in traditional network framework and the recent industry acceptance of OO models, the integration between these two is used here as an example.

There are two approaches that can be used to support integration.

- Specific Translation Application Gateway approaches.
- Protocol Neutral (Abstract object deification) approach.

3.3.1 GATEWAY APPRAOCH

A gateway is a piece of software, which sits in the intersection between two different environments with different object models see Fig. 3.5. The gateways behave like a mediation device to translate object definitions and functions from one environment to another. One ways of interfacing SNMP – and OO based management environment. The main functionality of the gateway is to perform the protocol translation between systems developed in these two different technologies.

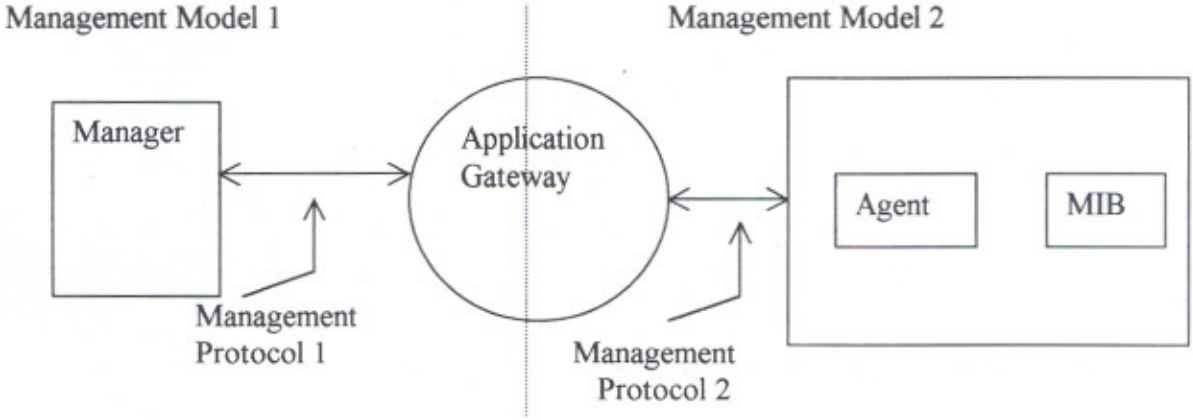


Fig. 3.5 "Gateway Approach"

There are two typical scenarios of using such a gateway is SNMP-OO network management systems:

- OO manager and SNMP agent.
- SNMP manager and OO agent.

In order to achieve communications between a manager and the agent for the purpose of exchange management information and functions, agreements must be made at two levels.

At the Object Definition Level – Object Mapping – The development of the SNMP has result in vast amount of object definitions in the forms of mibs. This has further led to the development and deployment of many applications based on these object specifications. In the case of translation gateways, each of these objects needs to be mapped into the specific interface object of the translated framework. These mappings would include the object definitions such as attribute type and values, operations on the objects, notifications emitted by the objects and the behaviors of the objects.

At the communication protocol level: Operations requested sent by the manager using the manager protocol need to be translated at the gateways into the agent protocol being passed to the agent and vice versa. Moreover, the confirmation of the translations has to be sent back to the respective manager/agent protocols.

3.3.2 PROTOCOL NEUTRAL APPROACH

Different form the one-to-one object mapping based translation gateway approach; the Protocol Neutral approach (or Abstract Mapping approach) addresses the integration problem from a different viewpoint. In many real-world object-modeling exercises, different object granularity is used for different modeling concepts at different level of abstraction.

For example:

(i) ***When modeling a network element,*** such as a switch, all the details of the switch need to be modeled which may include cards, ports, and other internal details. These details are useful for operations related to the management of that switch.

(ii) ***When modeling a sub network,*** the internal details of each individual switch are no longer important. Instead, only the access points of switches and the connections between switches are important (these form the so-called the network view of network elements).

(iii) ***When modeling a global network*** containing many sub networks, the features of the subetworks, such as the types of sub network (in terms of the transport protocol supported by the sub networks, e.g. ATM and SDH), the termination points of each sub network, the connections between sub networks are important. The internal connections within a sub network become less important for a global network manager.

These observations state an obvious concept that in a layer network structure it is important to allow different levels of object abstraction to be defined for different layers. This is particularly important to support the management of large-scale network. The object

specification at a higher level should view as a simplification and generalization (with more general concepts) of the object specifications at the lower level. Object specifications at the lower level should be viewed as the specialization and the instantiation (with more details and specifics) of the objects at the higher level. This is also true for the differences between the network service management layer and network management layer. The service management layer is more interested in abstract services supported by underlying networks while the network management layer is more interested in network details, which form the basis for the services.

Thus, each higher management layer is seen as a further Object abstraction and simplification and the previous layer. The protocol neutral approach is basing on this principle. It treats each level of object abstraction as an integration point and class diagrams are made in UML for each level of abstraction going from a macro coarse grain view, which hides the details on the micro fine grains view, which brings out the details. Thus, the approach has also specific design goals. The ability to manage a large network at a higher abstraction level with more abstract concepts. The ability to retrieve to details when required.

One of the most important purposes of the technology integrations to achieve application internetworking. The protocol neutral modeling approach describes a process of a more coherent and technology independent object modeling and the implementation of different portions of the object model using different technologies to achieve integration. The value addition of this approach is manifested by a process of object abstraction, or a semantics enrichment, through the integration, no just the protocol translation. This part of the approach primarily deals with the scenario of vertical integration. The scenario of

application internetworking is also achieved and is an important part of this approach. It addresses the horizontal integration of different operation systems with different technologies and different object models.

A typical example of this integration is the internetworking between different service providers' operation support systems like SNMP, OSI, DCOM and CORBA based models.

It is neither necessary nor possible for every service providers to accept the same technology and same object model and process model to implement their respective systems. It is, however, crucial for different service providers to exchange application information between their systems for the support of integrated, one-stop-shopping customer requests. In the new deregulated environment, this is important even just for the purpose and to the extent of fulfilling regulatory obligations.

At a glance, this type of integration goes beyond the normal solution development life-cycle management. However, how one's operation systems behave to the external influences and interactions is every system designer's responsibility, and it has to be governed by the same coherent management strategy. The protocol neutral approach typically requires a common interface between two operation systems to be defined and agreed. This interface agreement provides the basis for the integration. However, this interface agreement has to have the following features:

- ***It needs to be technology and protocol neutral:*** As the target operation systems are likely to be developed using different technologies and by different suppliers at different stages, it is reasonable to suggest that the technology neutral approach to this interface specification has a chance to receive more acceptance. This technology neutrality does go beyond the notation used in the modeling process. It includes and

focuses on the features and functions of the modeling language and framework (UML). For instance, a GDMO modeling technique implies a specific naming convention, and an object invocation through management entity, etc., .whereas in COBRA, the functions in all services imply that they are supplied as external services in the computational architecture. The technology neutral approach should not assume the use of these implementation specific features.

- ***It needs to be generic***: The interface specification has to be generic to fully encapsulate the specific design features and functions of different systems. It should only cover the most general functions two systems are required to interwork. This also helps reduce the integration cost for the integrating operation system, as a general rule, a subsystem supporting a very general interface is much easier to be replaced than a subsystem supporting a very specific interface.
- ***It needs to be simple***: The interface specification should be defined as a minimum set of specifications only to exchange information for internetworking purposes. It does not have to he an operational system in its own right. The simplicity criterion of the interface specification is crucial to the success of achieving interworking.

UML satisfies all the above characteristics and has been adopted in the present work for modeling of the SNMP mibs in the protocol neutral fashion.

3.4 EVALUATION OF APPROACHES

Each of these integration approaches discussed here. The benefits of the IDL-DGMO (specific translation) gateway are manifested in the following ways.

- The concept of this gateway approach is easily acceptable. Industry's initial criteria technology for acceptance of a new technology or a new protocol is that it will provide at least the same functionality as the existing ones.
- There is an immediate high demand for interoperability between different network management components developed using COBRA technology and OSI technology. There is a real need for a COBRA manager to work with an OSI agent, or vice versa. With the support of this tool, it becomes possible for the managers and agents to be developed separately using different technologies.
- With the work conducted by consortia (such as TNT, X/Open and OMG), standard mapping algorithms are starting to mature. This will assist the development and deployment of this mapping approach. The solution of such a mapping is reasonably clear.
- The commercial environments for this mapping exist today. These include various commercial ORBs and network management platforms that support communication.
- The perceived benefit of JIDM approach is that it provides tools for COBRA developers to get IDL definition for all currently defined managed objects. There is a saving on the object modeling exercise.

While the **Protocol Neutral approach** has the same value of a standard object-oriented process and methodology. It manifests the paramount importance of a properly defined object model for the network management problems in an implementation technology neutral way. That is, the network management problem modeling needs to be performed before any implementation strategies are considered. In the scenario of SNMP mibs, the approach simply states that the primitive object model SNMP uses is not going to be dictated by the underlying implementation protocol like TCP/IP. It wills purely described from the general network management requirements and these requirements exist before the technology integration requirements.

To support the right technology for the right domain, the ***protocol neutral approach offers a better use of object oriented technologies***. Given the major difference between different methodologies, creating specific translation gateways for each pair would be very difficult and inefficient.

CHAPTER 4.

UML TRANSLATION – RESULTS And DISCUSSION

4.1 UML BASIC CONCEPTS

The Unified Modeling Language **(UML)** is a language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is called a modeling language, not a method. The modeling language is the (mainly graphical) notation that methods use to express design.

A modeling language must include:

- **Model elements** — fundamental modeling concepts and semantics.
- **Notation** — visual rendering of model elements.
- **Guidelines** — idioms of usage within the trade.

In the face of increasingly complex systems, visualization and modeling become essential. The UML is a well-defined and widely accepted response to that need. It is the visual modeling language of choice for building object-oriented and component-based systems.

It is practical to represent the semantics of UML model elements according to the formalisms of UML – itself, the process is known as meta modeling.

Meta model: A model that describes model elements.

Meta modeling: The recursive modeling of model elements from themselves.

Common Elements: Elements are the building blocks of UML, and comprise model elements and visual elements.

Model Elements: Represent abstraction of the system being modeled.

Visual Elements: Provide textual a graphical projections that facilitate the manipulation for model elements.

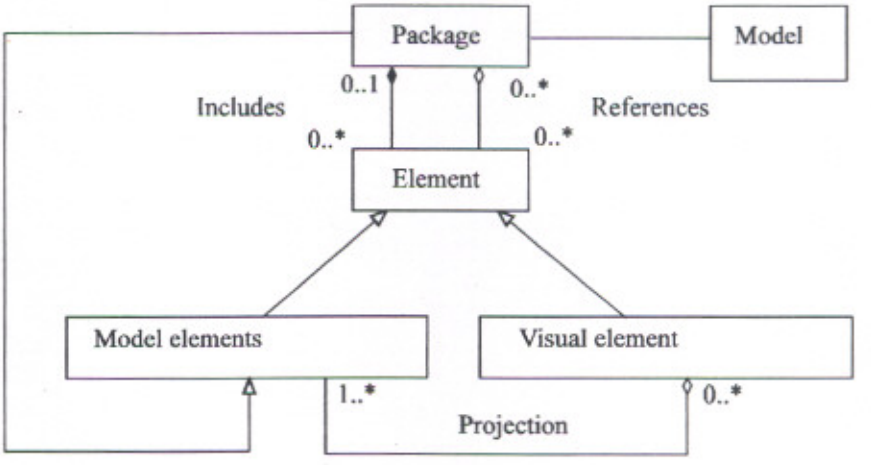


Fig 4.1 "Representation of the two large families of elements that from the content of models."

Common Mechanism

UML defines a small number of common mechanisms that ensure the notation's conceptual integrity. These common mechanisms comprise with following [25].

i. ***Stereotypes***: An extension of the semantics of a meta-model element. It permits users to add new model element classes on top of the kernel predefined by UML.

0Examples of predefined UML stereotypes are <<uses>> and <<extends>>.

ii. ***Tagged Value***: A tagged value is a (name, value) pair that describes a property of a model element.

Properties allow the extension of meta-model element attributes.

iii. ***Notes***: A note is a comment attached to one a more model elements. By default, it does not carry semantic content.

iv. ***Constraints***: A constraint is any kind of semantic relationship between model elements. UML does not specify a particular system for constraints.

Packages: Packages provide a general mechanism for partitioning models and grouping mechanism for partitioning models and grouping model elements. Package in represented graphically by a folder.

Example : Zoo :: Kangaroo

Kangaroo class as defined in the Zoo package.

Goals of the UML

The primary goals in the design of the UML were as follows:

1. Provide users with a ready-to-use, expressive visual modeling language so they can develop and exchange meaningful models.
2. Provide extensibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development processes.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of the OO tools market.
6. Support higher-level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

The Importance Of Modeling

Developing a model for an industrial-strength software system prior to its construction or renovation is as essential as having a blueprint for a building. Good models are essential for communication among project teams and to assure architectural soundness. As the complexity of systems increases, so does the importance of good modeling techniques. There are many additional factors of a project's success, but having a rigorous modeling language standard is essential.

Modeling Tool Used For The UML

Standardizing a language is necessarily the foundation for tools and process. The primary goal of the OMG was to enable tool interoperability. However, tools and their interoperability are very dependent on a solid semantic and notation definition, such as the UML provides. The UML defines a semantic metamodel, not a tool interface, storage, or run-time model, although these should be fairly close to one another.

Fortunately, technology comes to the reuse. A number of tools are available to help to create UML models.

- **Rational Rose**
- **SELECT Enterprise**
- **Visual UML**

For present work we used Rational Rose 98 (a service pack)

The common features of Rational Rose are as follows:

- Allows "rubber-band" diagramming: Create a link between two elements and it adjusts accordingly when we drag those elements around the screen.
- Allows at least some code generation, producing code stubs from models.

- Rational Rose also has extensive tools and dialog boxes for editing.

Rational Rose modeling tool has a following set of capabilities.

Flexibility: User must be able to use one diagram type's icon in another to form a hybrid diagram. State icons can be helpful in the sequence diagram.

The Use Case diagram's actor icon can be helpful in a number of other diagrams.

In addition ***to importing*** icons from diagram-type to diagram-type, it is better option, to have ability to import clip art and use it as graphic stereotypes in diagram. Perhaps some frequently used clips could be built into the tool [17].

A ***systematic semi-animated*** tutorial would be extremely helpful in said modeling tool.

4.2 PROTOCOL NEUTRAL NETWORK MANAGEMENT MODEL – UML REPRESENTAION

Object Oriented design encourages modularity at one level by the specification of self-contained objects. Coarser grained modularity is supported through the package construct, which allows the design to group related system components into logical modules.

The major **benefits of modularity** are improved **maintainability** and ease of **understanding.**

The protocol neutral network management model is composed of following packages as shown in figure 4.2.

- **Protocol Neutral Monitor Package.**
- **Protocol Neutral Network Package.**
- **Protocol Neutral Agent Package.**
- **Protocol Neutral MIB Package.**
- **Protocol Neutral Utility Package.**
- **Protocol Neutral User Interface Package.**

All the packages except PN_UI (Protocol Neutral User Interface) package in Fig. 4.2 contains the logic of the PN_NM, while the interface of user are described in PN_UI package.

The separation of PN_UI package from rest of the package is to minimizing the coupling between two modules.

The dependency relationship in Fig. 4.2 shows that the logic of the system does not depend on the user interface, but that the interface does depend on the logic, which allows the user interface to vary without affecting the application logic.

PROTOCOL NEUTRAL NETWORK MANAGEMENT MODEL

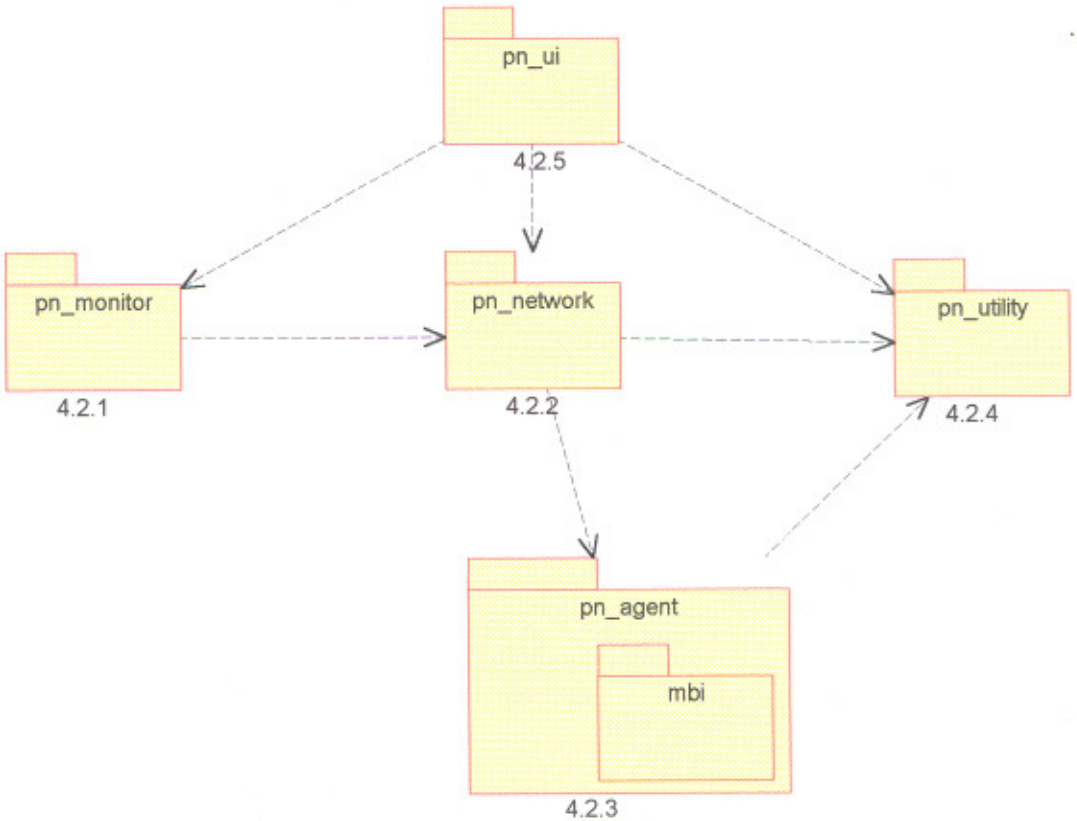


Fig. 4.2 "Protocol Neutral Network Manegment Package Diagram "

4.2.1 Monitor Package Class Diagram:

AIM: To represent the classes and objects used in periodically monitoring some management information.

Monitor Class: It performs the actual monitoring, and is an independent, generic value monitoring utility class.

It is designed to monitor any property (attribute) from any property source by provided a neutral monitoring interface and being loosely coupled to the information provider.

In Protocol Neutrality for interactive user interface: the monitor uses the standard (JAVA) event model to notify any registered listener that it has recorded a new value. The Client, use the addchangelistener and removechangelistener to respectively register or unregistered to receive these notifications.

Monitrable Interface is used to provide the getcurrentvalue operation.

Monitor Period Object: It is a container for a number of records, each of which is stored in a Monitor Entry Object. The monitor keeps a list of the values it has recorded over a period of time in a Monitor Period Object.

Monitor Entry: Each Monitor Entry object records the time the record was made, and the value of the property at that time.

Example: Clients request the Monitor Period object from the monitor, than use this to access the sequence of records it contains for that period.

The benefit of this design is that it allows code, which uses the monitor to vary without affecting the monitor implementation. So in protocol neutral model the monitor and its associated classes could be treated as a reusable monitoring component, and applied in other situations requiring periodic monitoring.

4.2.2 Network Package Class Diagram:

AIM: To represent the network entities in terms of Protocol Neutral _ Network management system.

Network Management Structure: Managed nodes are part of subnet (aggregation) and subnets are part of network map (aggregation) as shown in Fig. 4.3

Operation: get property operation has two arguments, which determine what property is being requested. This operation must be independent of any specific underlying Network Management, so it has two arguments, which specify the

(i) Category of management information being requested (categoryid).

(ii) Specific property (specificid).

The protocol neutral network management model also allows clients to request all *management information* within a particular category by requesting all as the specificid.

Defining *management information* in its own terms the request to retrieve a property to be independent of the actual protocol used to carry out the request, each of which defines it own way of categorizing and referring to attributes.

Managed Node Class: Represent a network node, which is to be managed.

It's role to participate in the network map and to implement a caching policy as a performance enhancement. So to improve the performance of the software, the managed node class provides this facility by maintaining a cache of information.

Cache of Information: *The managed node object check the request (which is made to retrieve the information) for information.* If that information is held in its cache, then this can be returned to the client. Else a request is made and returned normally and the new value is added to the cache.

4.2.3 Agent Package Class Diagram:

AIM: To represent the different agents, and handles communicating with those agents.

[In this model it is specific to SNMP agent].

Fundamental Agent Interface (FIAgent): The interface for classes representing agents (heterogeneous). All of the types of agent implement the Fundamental Agent Interface (FIAgent), which client should use to access the agent object. So the commonality and transparent access to the agent is achieved, while the client is not connected with the actual type of the agent object it is using, simply that it implement the FIAgent interface and therefore supports the standard get property operation.

So the process is simplified, if a new type of agent is added in future.

Fundamental Agent Repository (FRAgent): Define the mechanism for creating the various subclasses of the FIAgent class (which represent remote agent).

To increased transparency, modularity and maintainability clients should use the FRAgent for object creation.

Features: Provide commonality and transparent access to the agent.

Example: A client request an agent retrieve a feature using the getfeture operation, passing as a parameter a key specifying the feature to be fetched.

Mib: It is a private package in agent package.

It contains code for resolving SNMP OIDs and stub code for each entry in the MIB.

[Using any compiler tool (mibgen/libimax) generates the code].

4.2.4 Utility Package Class Diagram:

It is a collection of classes and objects that are not specific to Protocol Neutral Network Management, but simply provide useful utility functions to other subsystems.

AIM: To represent a table data structure, with labeled rows and columns and cells containing the actual data.

Internally, the table stores data in rows. Each row is stored as a Hash Table, and values within that row are keyed according to the name of the columns they are in. The set of rows is also stored in a hash table (i.e. data is stored as a hash table of hash table).

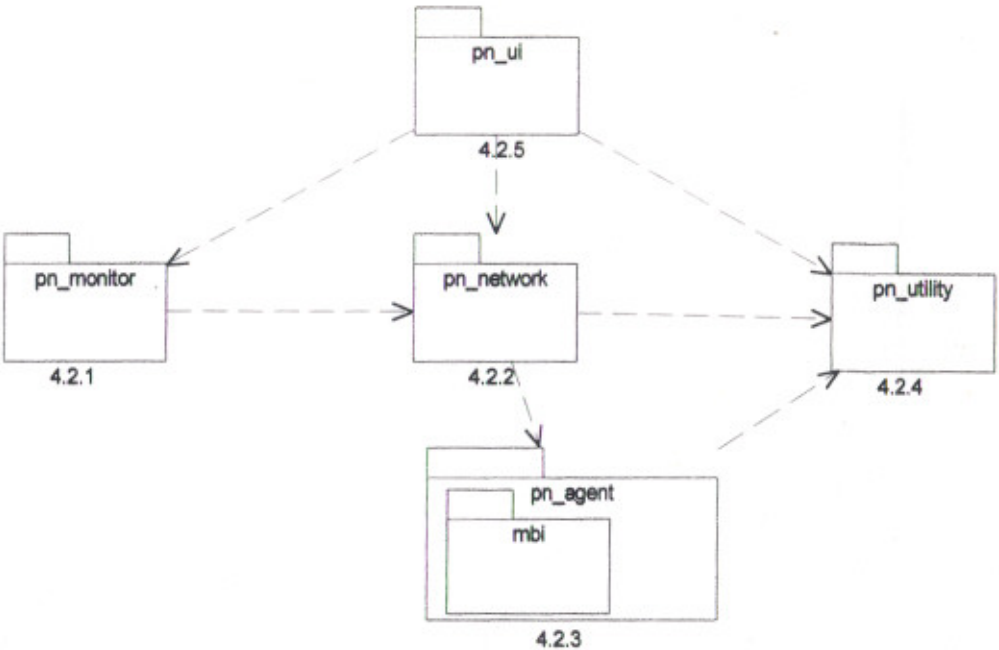
The table maintains two key sets for columns, one for "friendly" name and another for low-level name.

It is intended that internally the system will use the low-level names to refer to columns.

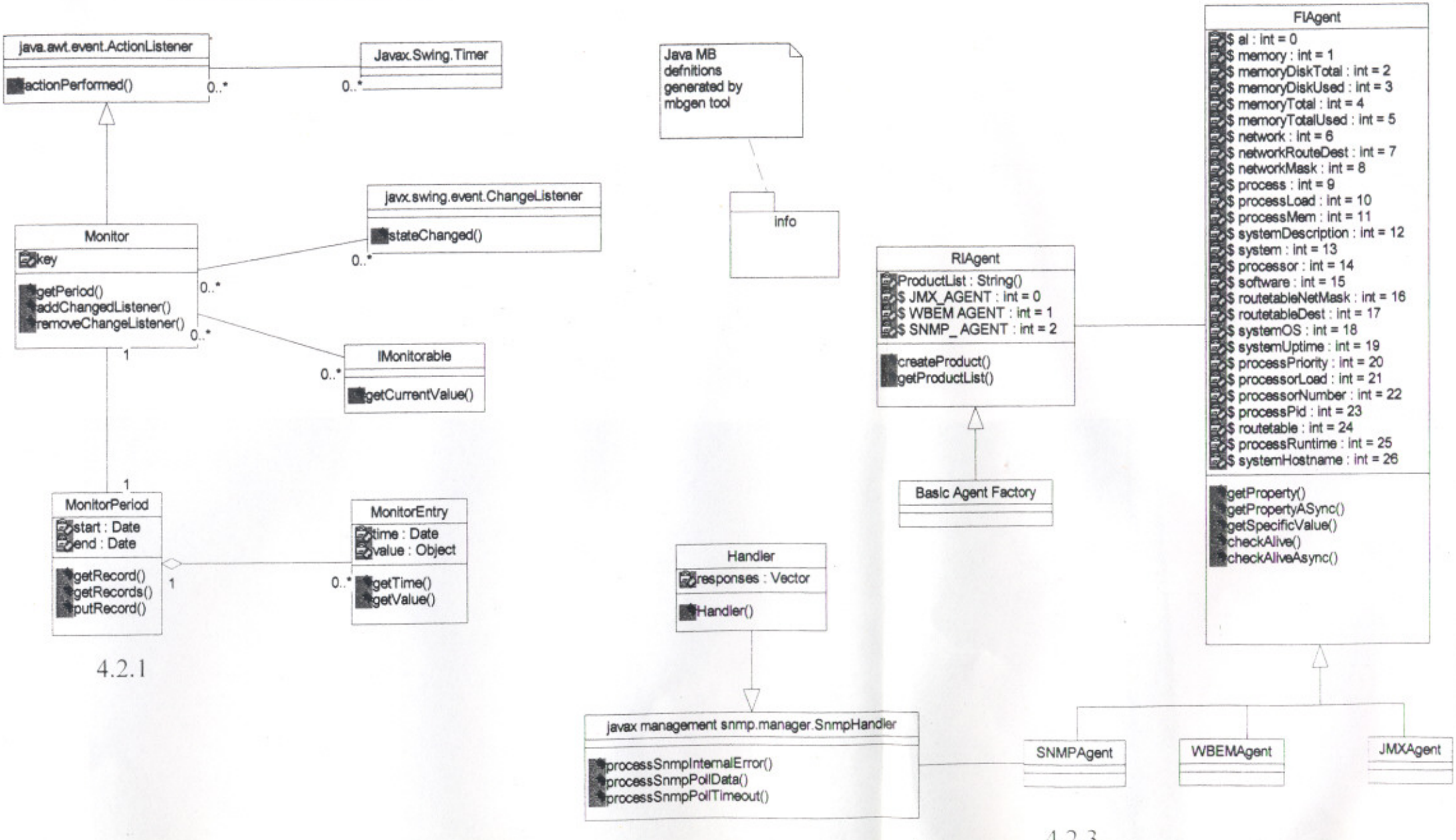
The "friendly" names are just labels for use by the user interface when displaying the information to the user.

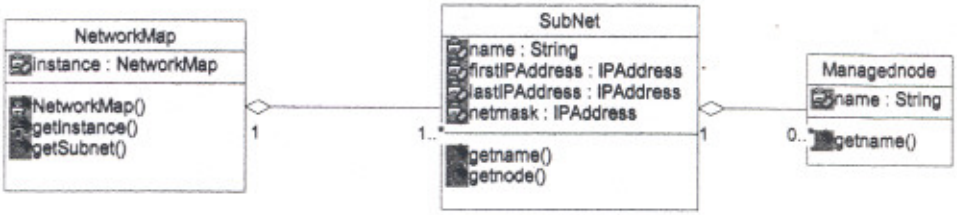
Table Class: Provides the required fundamental data access operations, including getting an entire row or column, getting a value in a specific cell.

It also provides some basic searching facilities.

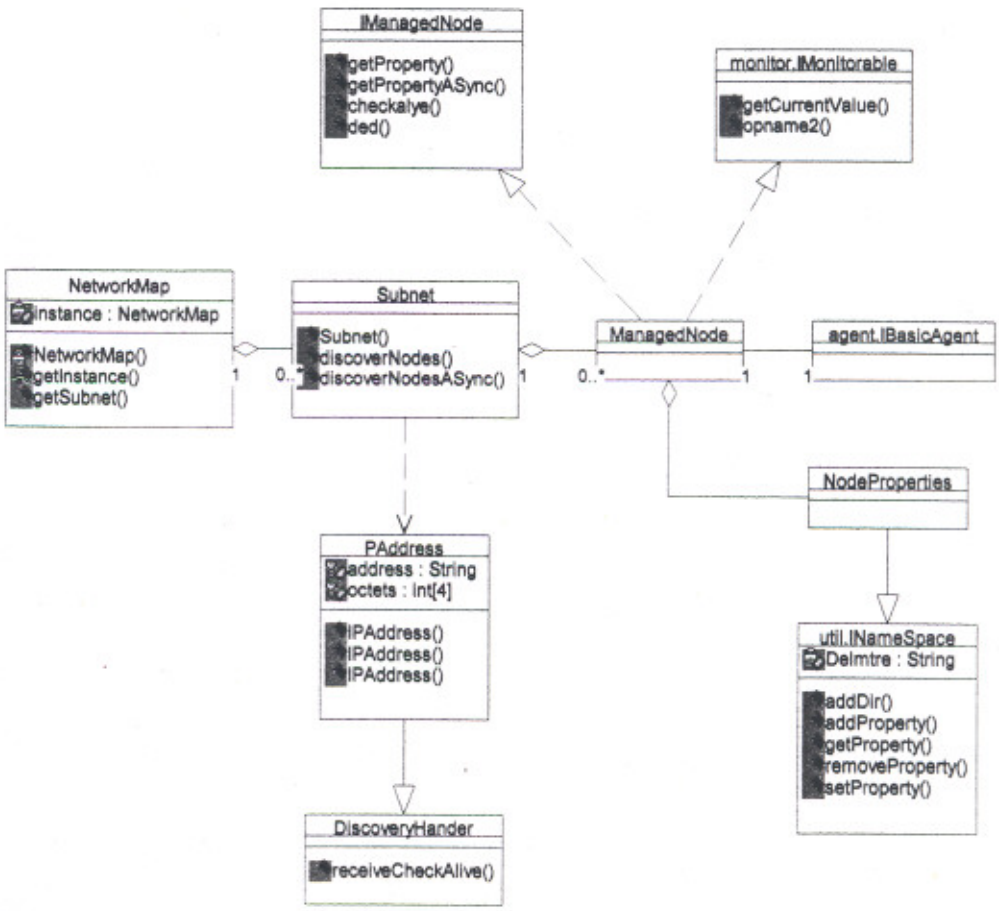

pn_ui
4.2.5
pn_monitor
4.2.1
pn_network
4.2.2
pn_utility
4.2.4
pn_agent
mbi
4.2.3

PROTOCOL NEUTRAL NETWORK MANAGEMENT MODEL - DETAIL UML REPRESENTATION

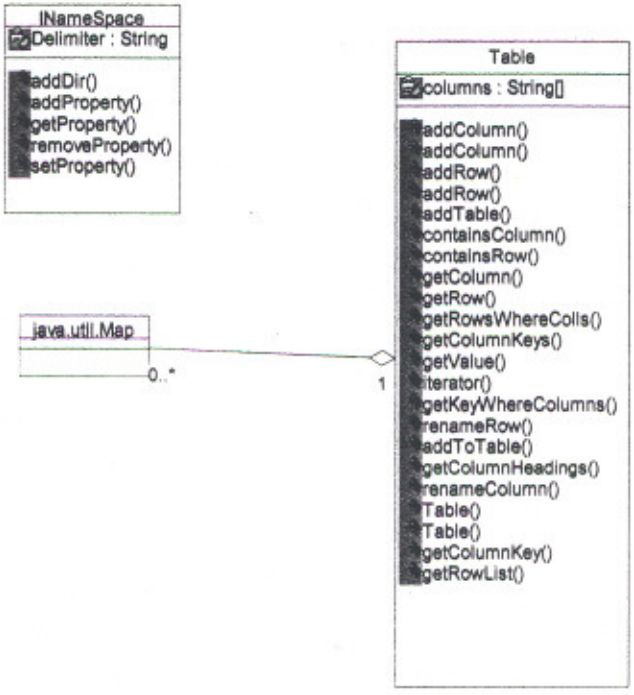




STRUCTURE OF NETWORK MAP



4.2.2


INameSpace
Delimiter : String
addDir()
addProperty()
getProperty()
removeProperty()
setProperty()
Table
columns : String[]
addColumn()
addColumn()
addRow()
addRow()
addTable()
containsColumn()
containsRow()
getColumn()
getRow()
getRowsWhereColIs()
getColumnKeys()
getValue()
Iterator()
getKeyWhereColumns()
renameRow()
addToTable()
getColumnHeadings()
renameColumn()
Table()
Table()
getColumnKey()
getRowList()
java.util.Map
0..*
1

4.2.4

4.3 CONCEPTUAL MIB MODEL – UML REPRESENTATION

The Unified Modeling Language (UML) [24, 4] has become the lingua franca for visualizing, specifying and documenting artifacts of software intensive systems. UML is very rich and features several diagram types to represent structural, behavioral and architectural aspects. Current **MIB** modules only specify structural aspects in a machine-readable format. The focus in this section of thesis is on structural aspects, which are usually visualized in UML class diagrams.

Fig. 4.4(i) shows a hand crafted UML class diagram visualizing the **conceptual model** underlying the IF-**MIB** module [18] and the IF-INVERTED-STACK-**MIB** module [19]*. There are some important details that distinguish the UML class diagram notation for SMI **MIB** modules shown in Fig. 4.4(i) from ordinary UML class diagrams:

1. The classes representing **MIB** definitions use the smi **mib** class stereotype in order to distinguish them from other UML classes.

2. There is a UML class for each **conceptual** row definition in a **MIB** module. The name of the UML class is taken from the row definition.

3. Scalars that are logically bound to a particular UML class are shown as under-lined class variables in the UML diagram.

4. Scalars that are not logically bound to a **conceptual** row are logically grouped into additional UML classes, which only contain class attributes.

5. Notifications whose required variables are attributes of a single UML class are shown as private operations for that class. The operation is marked private since it only conceptually exists as an internal interface to emit a notification when the corresponding event occurs.

**The diagram only visualizes those definitions that are current and ignores all deprecated or obsolete definitions.*

6. Class attributes that are used to uniquely identify a class instance are marked with the index UML property. The order of the instance identifier components is indicated by the order of the attributes in the UML class representation. Note that all instance-identifying attributes are listed, even if they are actually defined in other tables.

7. The standard UML visibility attributes have a slightly different meaning. Public visibility (indicated by a symbol) implies that the corresponding attribute is publicly accessible for reading and/or writing. Private visibility (indicated by a symbol) implies that the attribute is not directly accessible, e.g. because it is a not-accessible index component of a **MIB** table. *[Note that an attribute with private visibility can still be read indirectly by extracting the actual value from the instance identifier of another readable attribute of the same UML class].*

Fig. 4.4(i) shows several different relationships between the UML classes. The generalization between the ifEntry and the ifXEntry class represents a **MIB** table augmentation where every ifXEntry instance is bound to an ifEntry in-stance and vice versa. Table augmentations are frequently used in MIBs to extend an existing table definition. The association between ifRcvAddressEntry and ifEntry shows a **MIB** table extension where multiple ifRcvAddressEntry instances can relate to a single ifEntry instance. This is indicated by the cardinalities of the UML association.

The ifStackEntry is an association class. It represents properties of the "is stacked on" relationship between network interfaces. The association class itself has a class attribute (the **MIB** scalar ifStackLastChange), which indicates the time-stamp of the last change in an ifStackEntry instance. Finally, the association between an ifStackEntry and an ifInvStackEntry represents a **MIB** table reordering relationship where the not-accessible index components have been re-ordered. (Such reorder relationships enable management applications to read tables more efficiently and are thus an SNMP specific optimization).

Fig. 4.4(ii) shows a hand crafted UML class diagram for the **HOST-RESOURCES-MIB** [19]. This diagram clearly shows the generalization/specialization relationship between the hrDeviceEntry and the derived classes hrProcessorEntry, hrNetworkEntry, hrPrinterEntry, and hrDiskStorageEntry. The "implements" association links the hrNetworkEntry class to the ifEntry class of the IF-**MIB** shown in **Figure** 4.4(i).

CONCEPTUAL MIB MODEL - UML REPRESENTATION

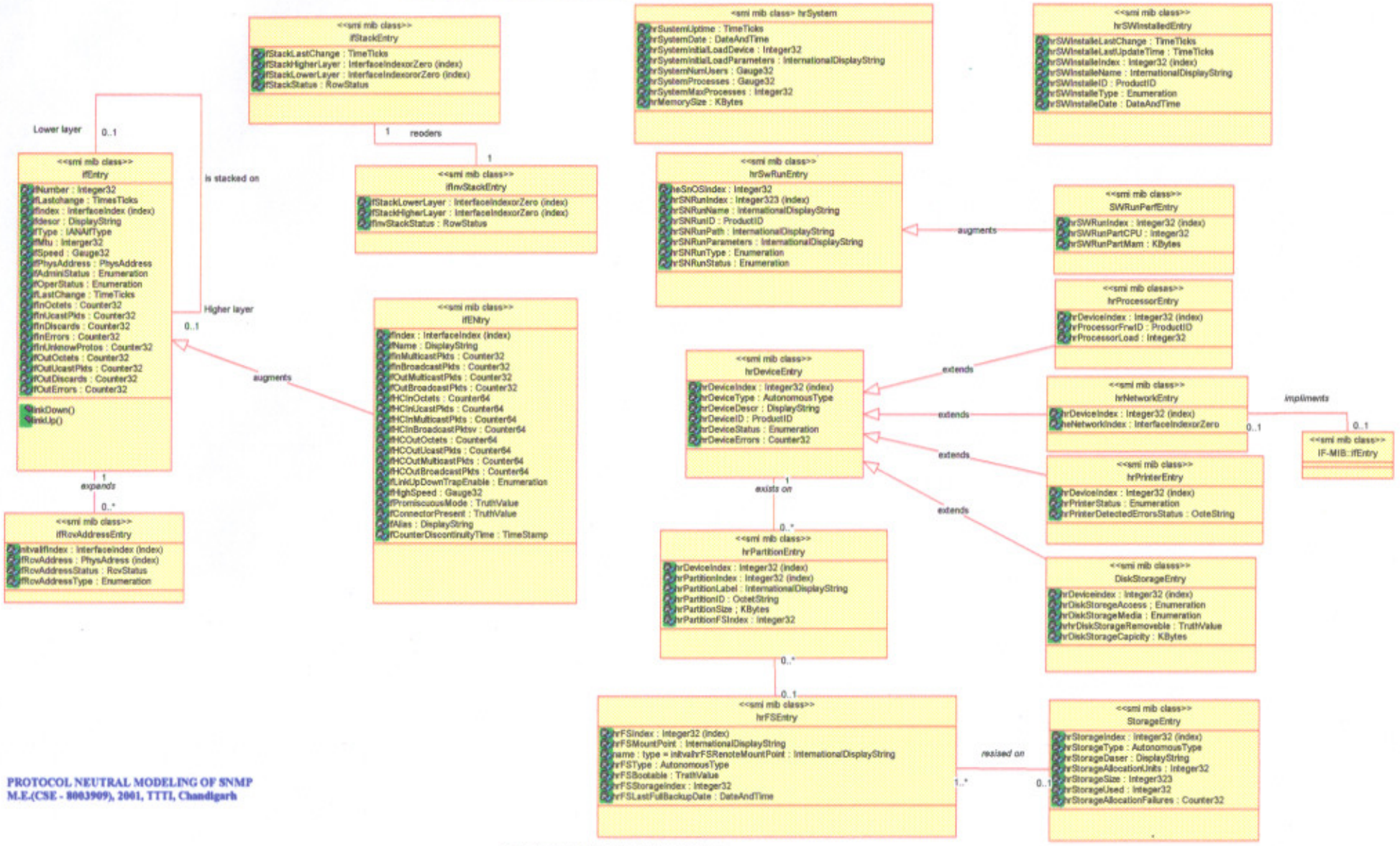
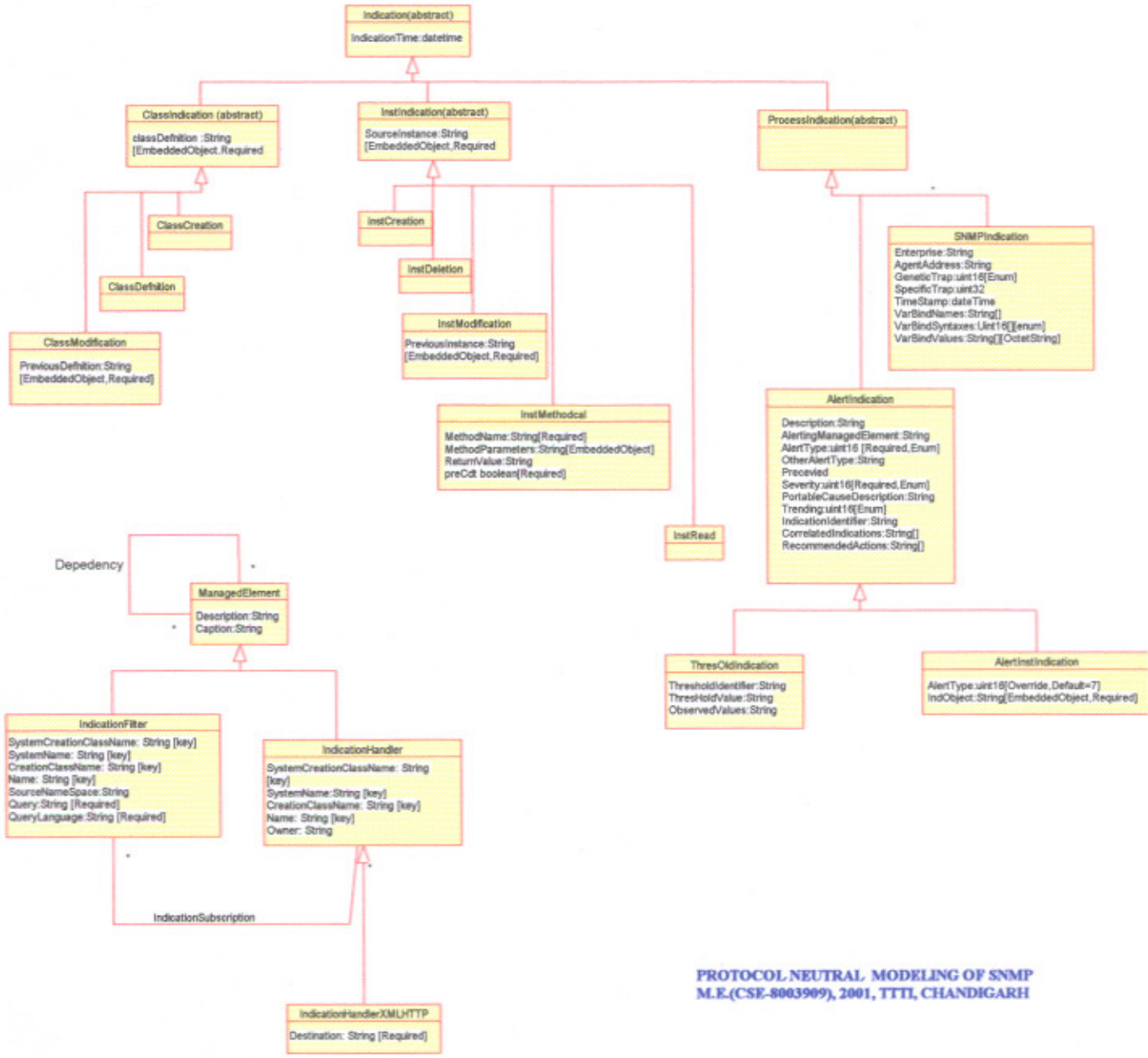


Fig. 4.4 "Conceptual MIB Model - UML Representation"

4.4 SNMP EVENT MODEL - UML REPRESENTATION



PROTOCOL NEUTRAL MODELING OF SNMP
M.E.(CSE-8003909), 2001, TTTI, CHANDIGARH

Fig. 4.5 "SNMP EVENT Model - UML Representation"

CONCLUSION

Network Management is rife with terminology. Even concepts are sometimes intermixed with specific model related terminology. Considering that network management is a fast paced research field, there is an urgent need to separate out the neutral concepts from the specific and there will always be necessity to achieve meaningful integration between diverse management paradigms.

This thesis work serves to fulfill this exact need. A consigns attempt has been made for the integration of Object Oriented based network management (OSI) and Internet network management (SNMP) in an automated manner.

The present work endeavors to overcome the limitation of Heterogeneous Network Management through *Neutral Protocol Model* especially in the context of SNMP. The work suggests that instead of converting SNMP MIBs to any Object Oriented Paradigm Model, model must be created in Unified Modeling Language (UML), separate from the protocol. The conversion between SNMP and OOP must be from the Neutral Protocol UML model as and when needed.

Although SNMP is quite an old protocol that has its limitations, it is still very important in network management sphere largely because it is so widely supported and used. Therefore supporting it in the artifact was an important feature that means it can be used to manage a large number of diverse network devices in a heterogeneous network.

The work describes how UML class diagrams can be utilized to document the conceptual model behind a MIB module and system design of protocol neutrality.

The output is a graphical representation for Protocol Neutral Network Management Model, and SNMP Events model, which is designed using UML class diagram.

FUTURE PROSPECTS

There is work to be done in automatic MIB translations, abstract MIB mappings. Additional areas that may need consideration are the **Web Based Network Management** and **Active Networks.**

Although the UML defines a precise language, it is not a barrier to future improvements in modeling concepts. Many leading-edge techniques have been addressed by OMG but they expect additional techniques to influence future versions. Many advanced techniques can be defined using UML as a base. The UML can be extended without redefining its core.

The UML, in its current form, is expected to be the basis for many tools, including those for visual modeling, simulation and development environments. **As interesting tool integrations are developed, implementation standards based on the UML** will become more available. The UML has integrated many disparate ideas, and this integration will accelerate the use of object-orientation and component-based development.

CHALLENGES TO UML

Most OMG members would find replacing CORBA IDL, which is used for specifying the distributed computing infrastructure, by UML too drastic a change. There are some areas where UML faces tough competition from CORBA. While the basic concepts of UML can

be quickly grasped, it takes much longer to learn the more advanced features, hence a steeper learning curve compared to CORBA or DCOM IDL. Also, because UML is a general-purpose language, it includes many standard elements, which have not yet been well defined. One suggestion is to remove these and replace them by customized profiles where necessary. Another problem is the lightweight extensibility and the current met modeling mania where everyone wants to apply this powerful new tool to every problem where often a lighter solution such as a modeling library would be perfectly adequate. The OMG needs to carefully decide which language extensions should be part of the kernel language and which should become separate profiles or model libraries in order to successfully face these challenges.

REFERENCES

[1] Andrew S. Tanenbaum, **"Computer Network"**, 3/e, PHI, New Delhi, 1996, (ISBN: 81-203-1165-5).

[2] An Overview of the Telecommunications Information Networking Architecture (TINA), TINA'95 Conference, Melbourne, Australia, 1995.

[3] David PerKins, Evan McGinnis, **"Understanding SNMP MIBs"**, Printice Hall, New Jersy, 1995 (ISBN: 0-13-437708-7).

[4] G. Booch, J. Rumbaugh, and I. Jacobsen. *The Unified Modeling Language User Guide,* Adisson Wesley, 1998.

[5] G. Chen and Q. Kong, Integrated Management Solution Architecture, 2000, Network Operations and Management Symposium, Paris, France, 2000.

[6] G. Chen, War and peace – How to avoid technology war? Journal of Network and Systems Management, Vol. 6, No. 3, September 1998.

[7] International Organization for Standardization. ISO IS 10040, Information processing systems - Open System Interconnection - Systems management overview, July 1991.

[8] International Organization for Standardization. ISO IS10165-4, Information Technology - Open Systems Interconnection - Management Information Services - Structure of Management Information Part 4: Guidelines for the Definition of managed Objects, July 1991.

[9] ITU-T Rec. M.3010, Principles for a Telecommunications Management Network (TMN), Study Group IV, 1996.

[10] ITU-T Rec. X.901, Information Technology - Open Distributed Processing - Basic Reference Model of Open Distributed Processing - Part 1: Overview, 1993.

[11] ITU X.722,Information Technology - Structure of Management Information: Guidelines For The Definition of Managed Objects, January 1992.

[12] ITU-T Rec. X.208, Specification of Abstract Syntax Notation One (ASN.1), 1988.

[13] James Rambugh, **"Object-Oriented Modeling And Design"**, fifth printing, PHI, New Delhi, 1998, (ISBN: 81-203-1046-2).

[14] J.Case, K.McCloghrie, M.Rose, S.Waldbusser, Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2), RFC1902, 1996.

[15] J.D.Case, M.S.Foder, M.L. Schoffastall, and C.Davin, **"Simple Network Management Protocol",** (RFC 1157), DDN Network Information Center, SRI International, May 1990.

[16] J.H.Saltzer, D.P.Reed and D.D.Clark, End-To-End Arguments in System Design, ACM Transactions on Computer Systems, Vol.2, No. 4, November 1984.

[17] Joseph Schmuller, **Teach Yourself UML**, 1/e, SAMS Publications, 1998, (ISBN: 81-7635-314-0).

[18] K. McCloghrie and F. Kastenholz. The Interfaces Group MIB. RFC 2863, June 2000.

[19] K. McCloghrie and G. Hanson. The Inverted Stack Table Extension to the Interfaces Group MIB. RFC 2864, June 2000.

[20] K.McCloghrie, M.Rose, Structure and Identification of Management Information for TCP/Ipbased Internets, RFC1155, 1990.

[21] M.T. Rose and K. McCloghire, **"Management Information Base for Network Management of TCP/IP based internets"**, (RFC 1213). DDN Network Information Center, SRI International, May 1990.

[22] M.T. Rose and K. McCloghire. **"Structure and Identification of Management Information for TCP/IP based Internets"**, (RFC 1155), DDN Network Information Center, SRI International, May 1990.

[23] M. Rose, The Simple Book, An Introduction to Management of TCP/IP. Based Internets, Prentice-Hall, 1991.

[24] Object Management Group. Unified Modeling Language Specification Version 1.3. Formal Specification, Object Management Group, March 2000.

[25] Pierre-Alain Muller, "**Instant UML"**, 1/e, Wrox Press, USA, 2000, (ISBN: 81-7366-175-8).

[26] "Wendy Boggs, Michael Boggs, **"Mastering UML with rational Rose"**, 1/e, BPB publication, New Delhi, 1999, (ISBN: 81-7656-143-6).

[27] William Stalling, **"SNMP, SNMPv2, SNMPv3, and RMON1 or 2"**, Adisson Wesley, 1998, (ISBN: 0-20-148534-6).

[28] Simple Network Management Protocol
(http://www.cisco.com/universal/cc/td/doc/cisintwk/ito_doc/snmp.htm)

[29] SNMP Research: SNMP v3 Research Paper
(http://www.snmp.com/snmpv3/v3white.htm)

[30] The Simple Web – Simple MIB – MIB Object lookup service
(http://www.snmp.cs.utweate.ul/ietf/mibs/oidd/help.htm)

[31] Network Management Laboratory
(http://www.sce.carleton.ca/netmanage)

[32] Inter-domain Management: Specification Translation
(http://www.opengroup.org/onlinepubs/8349099/chap06.htm)

[33] Object Oriented Concepts
(http://cookadlc.homepage.com/oo-cookadle.html)

[34] UML Overview
(http://nas.cl.uh.edu/helm/CS49319/umloverview/index.htm)

[35] UML Tool
(http://www.rational.com)

[36] Journal On UML
(http://www.jooopmag.com)
(http://www.openhorizon.co.uz/paper/osm.htm)