2002

A Generic Network and System Management Framework

Knahl, Martin Hans

http://hdl.handle.net/10026.1/1824

http://dx.doi.org/10.24382/4020
University of Plymouth

A Generic Network and System Management Framework

by

Martin Hans Knahl

A thesis submitted to the University of Plymouth

in partial fulfilment for the degree of

Doctor of Philosophy

School of Computing

Faculty of Technology

September 2002

A Generic Network and System Management Framework

by

Martin Hans Knahl

A thesis submitted to the University of Plymouth

in partial fulfilment for the degree of

Doctor of Philosophy

School of Computing

Faculty of Technology

September 2002

Abstract

A Generic Network and System Management Framework

Martin Hans Knahl

Networks and distributed systems have formed the basis of an ongoing communications revolution that has led to the genesis of a wide variety of services. The constantly increasing size and complexity of these systems does not come without problems. In some organisations, the deployment of Information Technology has reached a state where the benefits from downsizing and rightsizing by adding new services are undermined by the effort required to keep the system running.

Management of networks and distributed systems in general has a straightforward goal: to provide a productive environment in which work can be performed effectively. The work required for management should be a small fraction of the total effort. Most IT systems are still managed in an ad hoc style without any carefully elaborated plan. In such an environment the success of management decisions depends totally on the qualification and knowledge of the administrator.

The thesis provides an analysis of the state of the art in the area of Network and System Management and identifies the key requirements that must be addressed for the provisioning of Integrated Management Services. These include the integration of the different management related aspects (i.e. integration of heterogeneous Network, System and Service Management).

The thesis then proposes a new framework, INSMware, for the provision of Management Services. It provides a fundamental basis for the realisation of a new approach to Network and System Management. It is argued that Management Systems can be derived from a set of pre-fabricated and reusable Building Blocks that break up the required functionality into a number of separate entities rather than being developed from scratch. It proposes a high-level logical model in order to accommodate the range of requirements and environments applicable to Integrated Network and System Management that can be used as a reference model.

A development methodology is introduced that reflects principles of the proposed approach, and provides guidelines to structure the analysis, design and implementation phases of a management system. The INSMware approach can further be combined with the componentware paradigm for the implementation of the management system. Based on these principles, a prototype for the management of SNMP systems has been implemented using industry standard middleware technologies. It is argued that development of a management system based on Componentware principles can offer a number of benefits. INSMware Components may be re-used and system solutions will become more modular and thereby easier to construct and maintain.

Table of Contents

List of Figures

List of Tables

Acknowledgements

Thanks to my supervisors Dr. Colin Stockel, Prof Peter Sanders, Prof. Dr. Udo Bleimann, Dr. Steven Furnell and Dr. Benn Lines for guiding me.

Thanks to everyone in Darmstadt and Plymouth helping, supporting and inspiring me throughout the period of my PhD project for accompanying me.

Love to my family for being with me.

Author's Declaration

At no time during the registration for the degree of Doctor of Philosophy has the author been registered for any other University award.

This study was financed by the author with support from the Network Research Group at the University of Plymouth, Distributed Applications Laboratory at the University of Applied Sciences Darmstadt and PanDacom GmbH.

Relevant seminars and conferences were attended at which work was presented. External institutions were visited for consultation and research purposes. Several papers were published in the course of this research project.

The author declares that this work has not been previously submitted as an exercise for a degree at this or any other University, and that, unless otherwise stated, it is my own work.

Signed Martin Knahl

Date 20 12 12

Chapter 1
Introduction

'Where shall I begin, please your Majesty?' he asked. 'Begin at the beginning,' the king said, very gravely, 'and go on till you come to the end; then stop'.

Lewis Carroll, Alice's Adventures in Wonderland.

The word *architect* usually refers to a "*designer of buildings and large structures who prepares plans and supervises construction*" and *architecture* is generally understood to be the "*art or science of designing and constructing buildings*" [Hawkins1992]. According to well-established practice, the structures are analysed to gain an understanding of their components, the functions of these parts and their relationships with one another. The knowledge is then used to design bigger or better buildings. The same process is used in the study and construction of telecommunication networks. Network architecture is a description of a network's component parts, their functions and the relationships between them that form the basis for integrated services.

However, there is a major difference between these two examples of architecture: the scale and scope of the architecture. A building generally has a more or less well defined scope and function and is generally conceived and built according to a single plan. Data and voice communication networks on the other hand - public voice networks, local data networks, local voice networks and other network structures - cover and link the world like a web [Norris2000]. At the same time their definition and construction proceeds in a fragmentary manner with contributions from a large number of independent organisations. Yet, when viewed from a wide enough perspective, the overall communication network appears to be a single entity aiming to provide instantaneous communication services between individuals and organisations. To meet the user requirement for ever increasing capacity and to enable integrated services, the network operators and service providers will rely on integrated management procedures to control the infrastructures and to provide comprehensive communication services [Lewis2002]. A number of concepts and technologies exist for the management of communication networks [King2000] [Patel2002]. However, research suggests that a number of key issues have not been addressed and solved sufficiently [Doman2001] [Hauck2001] [Jander1999] [Lewis2000].

The aim of this research is to investigate the area of Integrated Network and System Management to establish the current state of the art. To further address the issues identified by the research, a new framework for the management of communication networks is proposed.

1.1 Motivation

Networks and distributed systems are now on everyone's agenda. Electronic information exchange and its multimedia applications have started a revolution in many working places and form a basis of the *global village*. The constantly increasing size and complexity does not come without problems. In some organisations the data processing system has reached a state, where the benefits from downsizing and rightsizing by adding new components and services is negated by the effort required to keep the system running.

Management in general has a straightforward goal: to provide an environment in which work can be performed effectively [Magretta2002]. This is true for the management of companies as well as for the management of computer networks. In both environments should the work required for management be a small fraction of the total effort. In order to achieve this, management in both areas requires a plan or strategy [Mayerl2001]. The plan describes the desired goals and defines how these goals should be reached. Business management has been a research target for centuries resulting in a multitude of theories and models [Magretta2002]. Most computer systems are still managed in an ad hoc style without any carefully elaborated plan. In such an environment the success of management decisions depends totally on the qualification and knowledge of the administrator.

The advent of new networking technologies, resulting in heterogeneous networking infrastructures in Local Area Networks (LANs) and Wide Area Networks (WANs) and of distributed systems environments like Common Object Request Broker Architecture (CORBA) have revealed the limitations of conventional Network and System Management strategies [Dornan2001] [Hauck2001] [Jander1999] [Lewis2000]. Autonomy of components in such an open distributed system allows local management decisions. However, due to the cooperation such decisions could affect other components, possibly residing at a remote location. A thorough evaluation of the actual state of potentially affected parts of the overall system is therefore mandatory before any management activity is performed. The currently available tools for network and system management do not provide an appropriate view on the integrated system.

Another problematic constraint is the low degree of automation for management tasks. Typically, administrators of large distributed systems spend a significant part of their time for routine tasks (e.g. finding and fixing problems) and urgent emergency repairs (e.g. exchange of faulty networking equipment). The 80/20 rule can also be applied to Network and System Management (i.e. 80% of the time is spent looking for faults and 20% of the time is spent to fix these faults) [Oppenheimer1999]. The remaining fraction of time can be used for demanding tasks such as strategic planning or system optimisation. A significant improvement of this situation requires the automation of routine tasks and an automated solution for typical problems. The effect of such automation would be that administrators could spend more time for research, planning and optimisation, resulting in an improved overall system. The increased share of demanding tasks would also make system administration and maintenance more attractive (e.g. to people with higher qualification).

In addition to size and complexity, the administrators of distributed systems are confronted with an increasing heterogeneity. One lies in the heterogeneity of the system components involved, another in the specialisation of the available management tools [Dornan2001] [Jander1999] [Oppenheimer1999]. The heterogeneity of system components including hardware, operating systems, communication facilities and protocols seems unavoidable but the management tools could be evolved. Nowadays, most management tools are limited in scope and in the number of supported platform types (e.g. configuration of specific network elements such as routers or software distribution for Personal Computers). Due to differences in the specification and implementation of management commands, the activation of the same management tasks requires different commands if executed on different platforms [King2000]. To avoid these problems, the administrator should be able to state his management requests on an abstract level, shielding differences in the command syntax from the human.

1.2 Research Summary

The full title of this research is "A Generic Network and System Management Framework".

In addressing the above, the primary aim is to examine how management for heterogeneous networks and services has evolved from numerous technical and theoretical influences to the current state of the art. The research considers how this technique is impacting on the telecommunication world and leads into a new research area.

The research, initially being driven by the emergence of the Asynchronous Transfer Mode (ATM) Technology and its management procedures, provides an analysis of how networking technologies and services and Network and System Management development have evolved to the current state of the art. In order to present this work in context, another major thread of the work looks at the ever changing real world industry environment - the development to the seemingly integrated communication environment offering the diverse services that exist today. The research outlines the principles for optimised management procedures and provides a novel framework, methodology and prototypical implementation of an Integrated Network and System Management Architecture.

The research draws heavily on experiences and insights gained in research projects with the University of Applied Sciences Darmstadt (or "Fachhochschule Darmstadt" in German) and industry projects with the company PanDacom GmbH, Germany. This work was exclusively based in the area of Integrated Network and System Management, Network Planning and Network Implementation. The experience of working on industrial projects is drawn upon to demonstrate "real-world" examples of the theory discussed at a number of levels and for an in-depth knowledge of the state of the art. Further activities at the University of Applied Sciences Darmstadt and the University of Plymouth included the supervision of final year projects in the field of INSM and ATM Networks, teaching of final year undergraduate students (including modules in the field of Integrated Network and System Management) and an active participation in the Distributed Application Systems Research Group and the Network Research Group (in the areas of networking and distributed systems).

Ongoing work in the research area and various experiences lead to the development of a new management approach. This has drawn greatly upon both theoretical survey work and practical industry and research projects, providing invaluable information related to the research as a whole.

1.3 Research Objective

As already indicated in this introduction, the majority of management tasks performed today are routine work. The routine work mainly follows the management cycle that a manager monitors and controls a resource (or a group of resources) of the overall IT infrastructure. The managed system is monitored and a decision about an appropriate management activity is made based on the retrieved information. Most decisions are made by human administrators and bear a high risk for mistakes and often result in problematic solutions [Oppenheimer1999] [Norris2000]. Automation and distribution of management tasks can improve this situation. It should be stressed that the objective of this research is not to replace the administrator by any kind of machinery, but instead to increase the comprehension of the human administrator for the complex process of integrated management.

The delegation of management procedures to an automated manager enables a more abstract overall view to the human manager [Gheti1997] [Emanual1994]. Therefore, the human administrator is less involved in low-level management activities but spends more time in the high level activities for planning and improvement. Instead of monitoring simple values with low semantic content, an Integrated Management Architecture should incorporate management procedures that provide a much clearer picture of the overall situation. Instead of initiating mainly low level control actions, the system and network administrator should be supported by an automated management to perform the appropriate control actions.

The wide diversity and rapidly changing nature of today's communication infrastructures and services in which Integrated Network and System Management must be performed means that a common and accepted architecture for the provision of Integrated Network and System Management is currently not available [Dornan2001] [King2000] [Lewis2002] [Patel2002]. A model to describe an integrated management system based on an extensible logical framework encompassing the different relevant domains and technologies is required.

Following this analysis, the main objective of this thesis, namely to provide a comprehensive concept for Integrated Network and System Management, can be refined into three objectives:

(i) Provision of an overall management framework;

(ii) Identification of a development methodology and realisation principles for the development of a system that follows the conceptual model;

(iii) Prototypical implementation of an Integrated Network and System Management Architecture.

Within the proposed INSMware (Integrated Network and System Managementware) Framework, a common approach to model a Management Architecture based on Building Blocks will provide developers with the means to deal with the complexity and heterogeneity of Integrated Network and System Management. To enable this, a common modelling approach must further provide a development methodology and realisation principles for INSM developers.

This thesis includes the design of a generic Management Architecture based on INSMware conceptual principles. The architecture can be applied to the different aspects related to Integrated Network and System Management. In order to validate the architecture, a prototypical implementation based on the architecture has been developed and applied to test and evaluate the workability of the proposed concepts. As the research interest lies mainly in Network Management, the implementation takes issues and requirements of this field into account. Nevertheless this work does not provide an implementation for general purpose network management system that covers all aspects of managing network systems. It implements the industry standard Simple Network Management (SNMP) protocol family as it is in widespread use in current networking devices. Although the prototypical implementation described here is fully functional, rather general implementation issues have been covered.

1.4 Thesis Structure

The thesis structure is broken into 9 chapters.

Chapter 1 provides an introduction to the research programme and the thesis.

Chapter 2 examines the challenges for Integrated Network and System Management and Integrated Service Networking. The theoretical research looks at how and why the networking environments have evolved, the driving forces behind this evolution and the underpinning role of high-speed networks in the information society. It further illustrates the development and impact of networking technologies such as Asynchronous Transfer Mode (ATM). The chapter continues to look at how communication services have evolved as the networking environment has developed. The chapter finishes by discussing the requirements for Integrated Network and System Management.

Chapter 3 examines the state of the art in Integrated Network and System Management. The research looks at the requirements for Integrated Network and System Management from a theoretical point of view (i.e. literature review) and a practical point of view (i.e. requirements gathered in industry projects). The chapter continues to look at how INSM development can be seen to have evolved as the Information Technology (IT) environment has developed and identifies critical aspects and relevant concepts. The chapter discusses how INSM has evolved from numerous sources to today's state of the art and identifies current limitations (i.e. standardisation of relevant technologies, management architectures and frameworks, integration of Network and System Management.).

Chapter 4 describes generic principles for optimised Management Services drawing upon the identified management challenges. Several approaches to integrate Management Systems are examined. It concludes with a set of architectural and operational requirements for a Management System.

Chapter 5 proposes a new management approach to overcome the identified limitations and restrictions. The INSMware Framework is based on the notion of Building Blocks to break up the required functionality into a number of separated entities. A high-level view is presented that identifies and describes the different domains. The structural composition, interfaces and capabilities of the conceptual design are considered. The design and deployment environment to provide the means to develop a system without compromising its composite and comprehensive nature is also commented upon.

Chapter 6 provides a discussion regarding the realisation of the INSMware conceptual principles and management provision scenarios. A development methodology for the realisation of Integrated Network and System Management systems is introduced. The benefits and critical aspects given by the INSMware approach are commented on with respect to key aspects for the realisation of Management Systems.

Chapter 7 and Chapter 8 discuss the design and prototypical implementation of the INSMware prototype. The prototype exploits Componentware technologies for the realisation of the conceptual principles. The prototype provides a demonstration platform of an open and scalable system. It utilises the Simple Network Management Protocol (SNMP) for the integration of Managed Objects.

Chapter 9 concludes the thesis and summarise achievements and limitations of the research programme.

Figure 1.1 illustrates the different research work packages and its relations within the overall project.

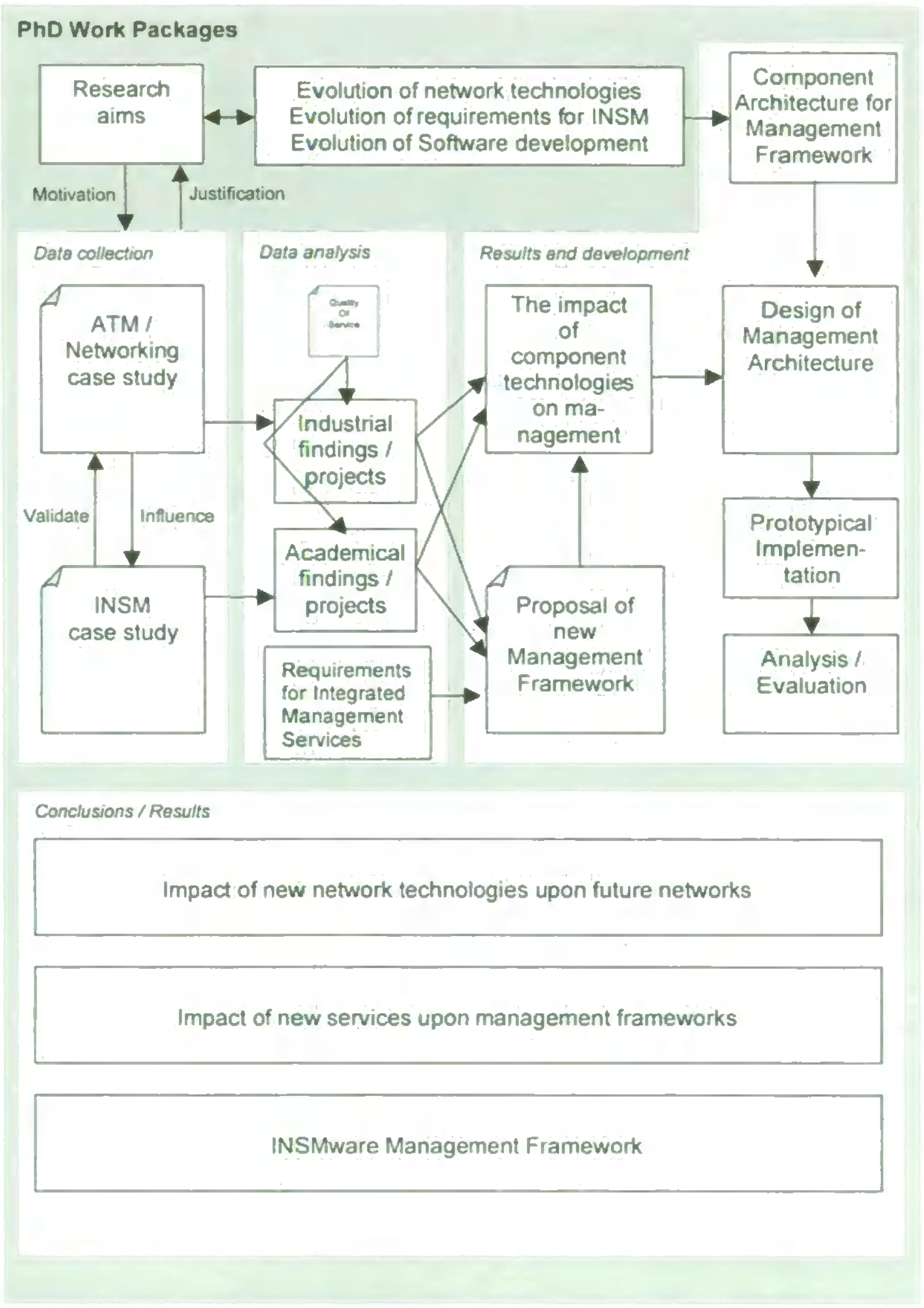

PhD Work Packages
Research aims
Evolution of network technologies
Evolution of requirements for INSM
Evolution of Software development
Component Architecture for Management Framework
Motivation
Justification
Data collection
Data analysis
Results and development
ATM / Networking case study
Quality Of Service
Industrial findings / projects
The impact of component technologies on management
Design of Management Architecture
Validate
Influence
Academical findings / projects
Prototypical Implementation
INSM case study
Proposal of new Management Framework
Requirements for Integrated Management Services
Analysis / Evaluation
Conclusions / Results
Impact of new network technologies upon future networks
Impact of new services upon management frameworks
INSMware Management Framework

Figure 1.1: PhD Work Packages

Chapter 2

Heterogeneous Networking and the need for Integrated Management

This chapter analyses the development of the IT world towards today's state and is looking at future trends to provide a context for the overall research. It discusses the current pressures on Information Technology systems and networks. The evolution of these systems is commented on with respect to the systems present, their capabilities and moves towards their integration. The problems associated with the current situation are also discussed.

A service model for network architectures is introduced to provide a structured view of distributed systems and the required management services. The chapter then provides an overview and analysis of the history and development of communications technology to the current state. It then examines the impact that the evolution of Information Technology and communication technologies have and reviews Information Technology and communications services development. Finally, the chapter provides a discussion of Management services and outlines a solution to the current problems of management provision, in the form of integrated management services.

2.1 A Service Model for Network Architectures

In order to provide a structured view of the functions of distributed systems and the required Management Services, a service model for network architectures derived from Patel can be used to illustrate the different layers involved in communication services [Patel2002]. It can also be used to further isolate the different aspects and functions of Management Services (see also Figure 3.1). The functions and services provided by network architectures are diverse and are used in a variety of different implementations. Typically, network architectures cover only a subset of the overall set of functionality. However a global and generic model is required in order to analyse required services and to outline areas of research and development [Patel2002].

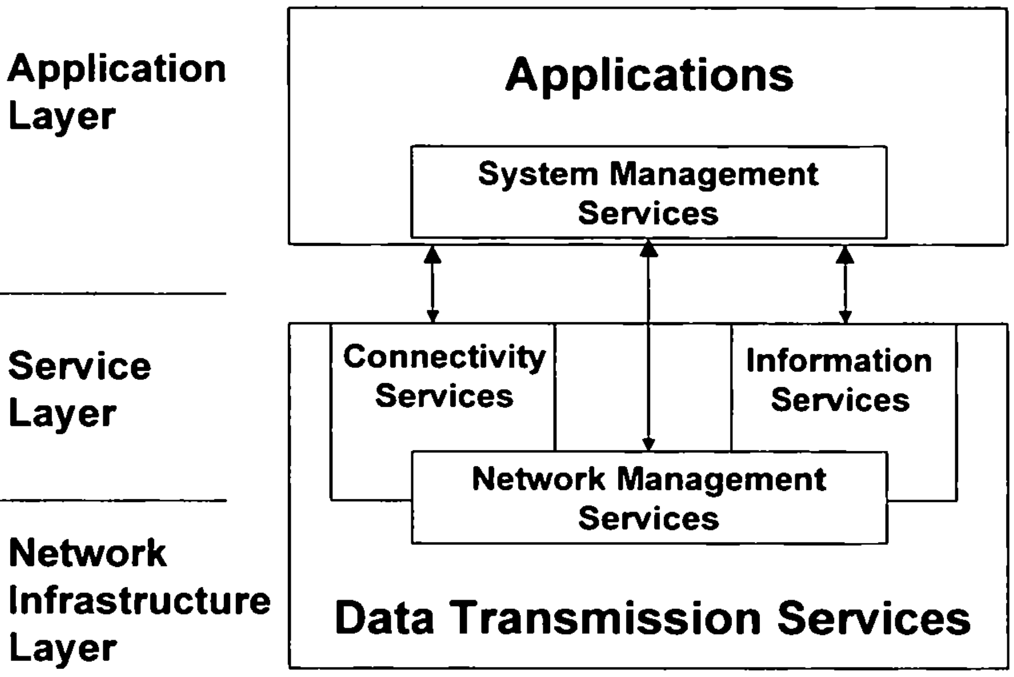

Application Layer
Service Layer
Network Infrastructure Layer
Applications
System Management Services
Connectivity Services
Information Services
Network Management Services
Data Transmission Services

Figure 2.1: Service Model for Network Architectures

The bottom layer of the three-layered model illustrated in Figure 2.1 is the Network Infrastructure Layer. This layer provides the data transmission services through networks. Data in this context refers to all types of data that may occur in a distributed environment (e.g. data, voice, multimedia). Between the Network Infrastructure Layer and the Application Layer is the Service Layer. The Service Layer is acting as an access layer for the applications that hides the complexity and heterogeneity from the applications and it provides applications with required information and control of the underlying network to ensure the correct operation of the various communication services. The top layer is the Application Layer that typically consists of end-user applications. These applications usually act as tools that support a user in the utilisation of a particular communication service. The following services can be found in the general model:

- Data Transmission Services are provided by the Network Infrastructure Layer and are concerned with the transmission of data through networks (e.g. LAN or WAN networks). Examples for activities in the Network Infrastructure Layer are routing and switching, congestion control and data transmission.

- Connectivity Services offered by the Service Layer provide high-level access to data-transmission and management functions of the underlying infrastructure. They act as Application Programming Interfaces (APIs) hiding the complexity of the underlying infrastructure to the end-user through the use of a generic high-level interface (e.g. to set up a connection to a video server could be a single instruction in an application whereas it will involve a number of lower layer tasks such as connection establishments and switching requests).

- Information Services offered by the Service Layer support users in the delivery of data over the network (e.g. functions regarding location, searching or storage of data) and the provision of required high-level information about the underlying infrastructure.

- Management Services are concerned with overall management of distributed Software and Hardware and must be implemented on all layers of the proposed model and include Network Management Services and System Management Services. Network Management Services are incorporated in the Network Infrastructure Layer and the Service Layer and control the operation of the network infrastructure and the communication services. System Management Services are incorporated in the Application Layer to control the user applications and processes that are usually independent from the underlying network infrastructure.

Furthermore, a hierarchy of Service Layers may exist in which each other's Services and Network Infrastructures are being used (e.g. end-user network and Internet Service Provider network). Today's management standards and solutions in the areas of Network and System Management lack an integration of management functions that can be found in the different layers of the Service Model for Network Architectures (see Chapter 3.4, Chapter 3.5 and Chapter 4.2). The overall aim and contribution of this research is to provide a concept and reference model for the provision of Integrated Network and System Management.

The INSMware Framework as introduced in Chapter 5 of this thesis provides a new management approach for the provision of Integrated Management Services. It describes the structural concepts of the new management approach and their relationships in a manner independent of any implementation technology. It offers an essential supporting structure and can be used as a reference model to describe the provision of Management Services [Hawkins1992]. The core structural concept is the Building Block, and systems are built from assemblies of Building Blocks.

The INSMware development methodology and realisation principles identify the processes and principles to implement Building Blocks and to assemble systems that conform to the proposed Management Approach (see Chapter 6). Since the beginnings of Network and System Management research and development, a number of different technologies tried to provide a common platform for building distributed management solutions [Hegering1999] [King2000]. Since the requirements associated with INSMware represent a large scale distributed system it is valid to consider its mapping onto Componentware and distributed system technologies [Dornan2001] [Hofmann2000] [Lewis2000]. The most relevant enabling technologies include OMG's CORBA [OMG1998] and Microsoft's DCOM [Eddon1998]. The Componentware approach together with distributed system technologies enables the implementation and integration of INSMware Building Blocks. INSMware is conceived as a distributed management framework that consists of Building Blocks that use the distribution services of a generic Building Block Interaction Broker.

The INSMware prototype design and implementation addresses how the concepts can be applied using current technologies (see Chapter 7 and Chapter 8). It also provides a repository for reusable Software Components that result when the proposed Management Approach is applied to a particular application domain (e.g. management of SNMP based networks).

2.2 The Evolution of Communication Services

"According to rabid netheads making a call over the Internet is inherently cheaper than using a conventional phone, thanks to the Net's superior technology. A nice idea, but too bad it's fantasy .... And it sacrifices reliability for low cost. But telephone companies are headed for extinction anyway. With the amount of data traffic growing almost a hundred times faster than voice traffic, the telephone network will soon be no more important than the Telex network." [Wired1997]

Until relatively recently, telecommunications networks have been mainly driven by a single, major requirement: the delivery of voice-based services. If any other service type (i.e. digital information) needed to be transported over the Wide Area Network (WAN), it had to be converted into the appropriate form (e.g. using a modem). However, the service revolution, in particular the Internet, has placed a far greater demand for data communication across the network, which is still growing rapidly [Chorafas1997]. Telecommunications companies realised that they had to provide for a multi-service network, with the biggest requirement coming not from their traditional market, but from the new IT market, which brought with it demands for greater capacity and faster delivery. Indeed the changes taking place in the development and implementation of telecommunications arose directly from the software boom, the driving forces being the limitations of the existing infrastructure that could not cope with the increasing volume of data traffic and the required speed of delivery and to the fact that the revenue gains of telecommunications companies were not proportional to the increase in data traffic.

The development of transmission technologies for very high transmission rates such as ATM have provided a common foundation in the local network and the long-distance traffic areas. Standardisation in transport systems such as the Internet is a contributory factor, as is the merging of data communication and classical telecommunications. The new technologies are enabling and promoting the flexible configuration of communication structures. The network becomes more than just a connection structure. The network thereby becomes like a market, like an independent system offering services that can be flexibly and individually configured and negotiated in dynamic communication relationships. Individual communication and mass communication begins to merge and classical point-to-point communication is supplemented by point-to-multipoint and multipoint-to-multipoint communication. General communication and information services enable new distribution and conferencing services (e.g. Virtual LANs, Virtual Private Networks, Virtual Workgroups or Intelligent Networks [Thomas2000]).

Therefore, providers of telecommunication and networking standards, services and equipment started to develop new ways and technologies to meet these increasing demands, as well as looking at ways which they could get some control over sources of this new traffic. In the mid-nineties of the twentieth century, ATM emerged as a promising technology for the provision of multi-service network infrastructures. However, as well as developing new network technologies, it was also necessary for the industry to develop new skills to integrate and control the heterogeneous network infrastructures and its services leading to the development of software based management systems. A traditionally hardware-centric engineering industry had to learn how to write software.

One might assume that the best way to develop services and applications (i.e. software) for this new domain would be to look at the IT industry, which has been producing software for many years. However, there seems to be an assumption in the telecommunications industry that their software is not the same as other software [McCauley1997] [Patel2002]. While a lot of the Management Services operate at a very low level or dealing with specific communication related tasks, they still essentially perform the same task of acquiring, transforming and presenting information as any other software. Take for example the requirement for a network router to reconfigure a network route. A user (or another piece of software) *informs* the software controlling the router that the router should re-route, the controller software *transforms* this request into something that the hardware router understands (i.e. digital signals) and *presents* this transformed information to the router. The router then *informs* the controller software that the re-routing was either a success or a failure. The controller software *transforms* the information into something the user will understand and then *presents* the information to the user.

However, with this attitude it seems that the telecommunication and networking industry prefers to take its own experience in hardware design and development, and apply it to their management models that are intended to provide a basis for the development of Management Services. For example, the initial specifications of the Telecommunication Information Networking Architecture (TINA), though based on a distributed and object-oriented approach, bears little resemblance to established software technologies in its development guideline documentation [TINA1994]. This *cultural clash* between IT and telecommunications and networking is very much ongoing, and has been widely reported [McCauley1997] [Chorafas1997].

2.3 A review of IT and Communication Services development

It usually takes a few years before the opportunities offered by services that have already been introduced and are available today are fully exploited and are widely integrated into operating and business processes. These services include basic Internet services such as WWW and E-Mail as well as more enhanced services such as videoconferencing. Networked systems support distributed cooperative work approaches such as workflow management systems, Small Office / Home Office (SOHO) access and application sharing. Universality does not stand only for uniform exchange formats but also for uniform service access interfaces as well as for standardised name and address concepts. This includes merging all the relevant data, such as all addresses, into a universal data repository. Users of co-operative distributed systems should have as much transparency as possible when determining which technical subsystem of an IT structure is providing their communication services. They should have the option to select or be able to change to the service that is most suitable for each respective work package in the business process.

2.3.1 Organisational Forms and Networks

Distributed IT services support and enable the implementation of flexible organisational forms, and in fact are a prerequisite for the provision of required services [Oppenheimer1999] [Norris2000]. Examples are decentralisation and outsourcing of IT services, Intranets and Extranets, division of labour over physical distances and virtual workgroups, just-in-time supplier relationships, workflow management or market globalisation. On the other hand, the IT itself is becoming more and more an organisation critical factor, i.e. the successful running of an organisation is dependent on the availability and operation of the underlying IT services.

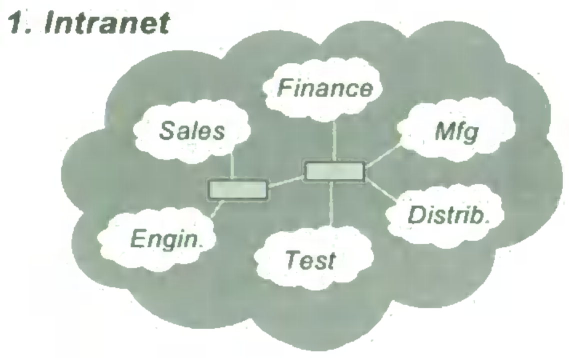


Figure 2.2: Intranet

Corporate networks, as illustrated in Figure 2.2, span over the different departments of an organisation and typically consist of an heterogeneous mix of components and technologies from different vendors. Most networks today are physically separated into a voice network and a data network. An Intranet is typically referred to as the internal Internet, i.e. the internal provisioning of Internet services.

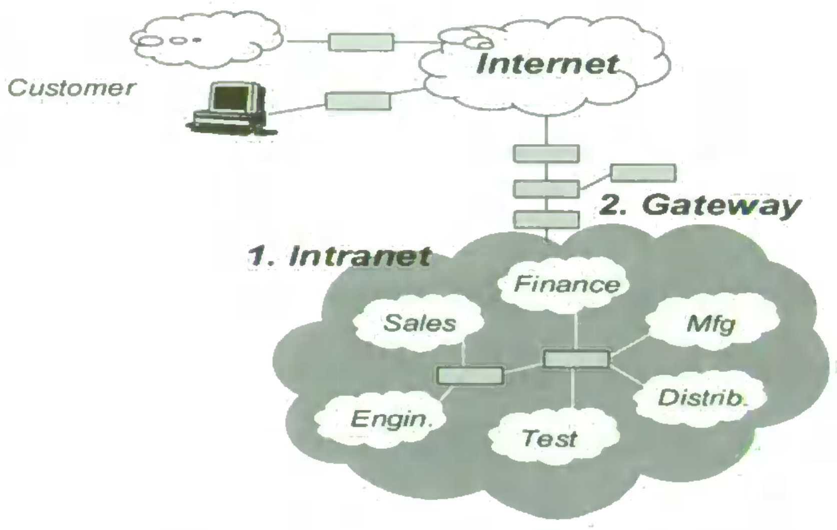


Figure 2.3: Remote access through Internet

Remote users connect to the internal network of an organisation through another provider's network (i.e. LAN and WAN internetworking is required). A remote user can be an employee working from a location that is not within the organisation's network or a customer that connects to a service of the organisation (e.g. looking up the Web site of this organisation). In Figure 2.3, the remote access to the Intranet is provided via the Internet.

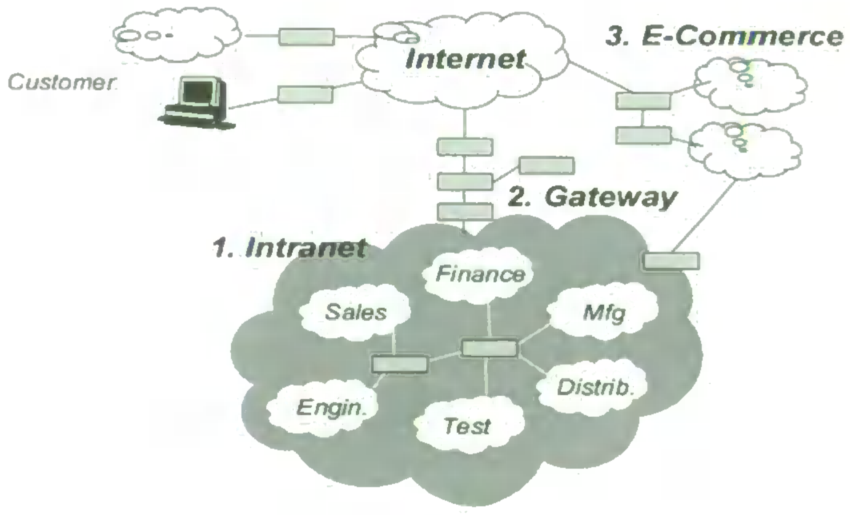


Figure 2.4: Integration of E-Commerce

The integration of E-Commerce typically requires the connection of further external Service-Providers such as banks or credit card companies [Kalakota1999]. The nature of E-Commerce data (e.g. credit-card numbers) is highly sensitive and therefore adds further security requirements.

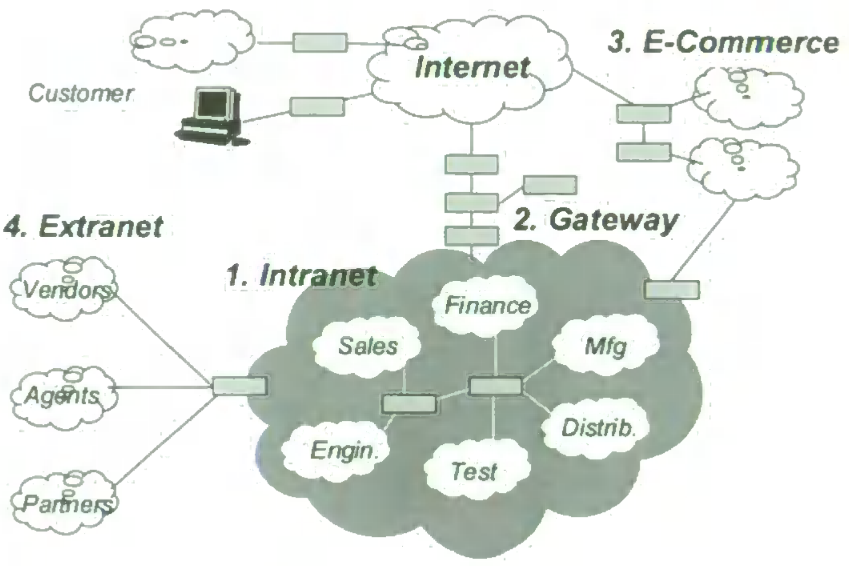


Figure 2.5: Integration of Extranets

The integration of Extranets requires the connection of further external networks (e.g. to connect to vendors or partners). The nature of the relationship with the Extranets determines the requirements on the network (e.g. security requirements and / or high capacity requirements).

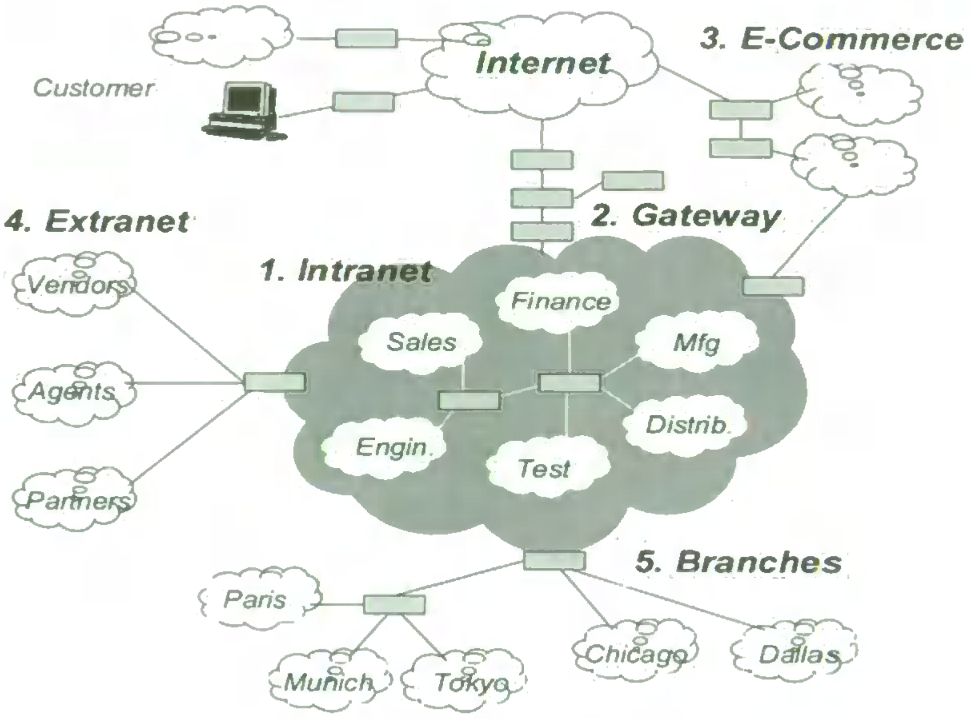


Figure 2.6: Distributed Model

Globalisation and the trend to integrate communication infrastructures leads to the integration of the various branches into the organisation's network to create the distributed model as illustrated in Figure 2.6.

Corporate Networks or *Enterprise Networks* are organisation wide unifying communication structures that form the basis for distributed IT services. Not only the corporate network but the entire co-operative distributed IT infrastructure complete with its services and distributed applications provided by must be subordinated to a preferably integrated management architecture. They require the following [Oppenheimer1999] [Norris2000]:

- A flexible, future-oriented cabling structure in the local area that is preferably not dependent on any particular network technology and can meet bandwidth requirements of new services and increasing traffic volumes;

- A uniform protocol structure for the transport system to ensure interconnectivity and interoperability between different networks and end-systems and public data networks;
- Integration of data and voice networks and associated services;
- Availability of comprehensive, organisation-wide communication services, service access interfaces, name and address spaces;
- Protected gateway points to the Internet, other public networks and other organisational networks using integrated security services and concepts;
- Uniform management concept.

2.3.2 Multimedia Applications

It is important to determine the architecture and implementation of future communication networks to get an idea about communication services and applications that can be expected (and vice versa). Multimedia has had a strong impact in the field of telecommunication since the 1990s [Fluckiger1995] [Froitzheim1997]. It is rather difficult to define multimedia precisely because the term multimedia has different meanings in different areas. The word *multimedia* is composed of two parts: the prefix *multi* and the root m*edia*. *Multi* means 'numerous'. *Media* in this context refers to various types of information such as data, images, audio and video. The following definition reflects the characteristics of multimedia systems:

> *"In the area of communication systems and computer networks, multimedia is the field concerned with the computer-controlled integration of text, graphics, still images, audio (including all types of sound, such as, speech, music, etc.), video and any other medium where every type of information can be represented, stored, processed and transmitted digitally." [Fluckiger1995]*

Another definition in literature is as follows:

"A multimedia system is characterised by the computer controlled, integrated generation, manipulation, representation, storage and communication of independent information, which are coded in at least a continuous and a discrete media." [Steinmetz1993]

Since the 1990s, information technology has been developing from a rather restricted end-user perspective towards a variety in form of media, from voice and paper to today's telephone and computer based communication and video/audio transmission. There has been a dramatic increase in the range of media used to convey information. On the other hand, from a system perspective, information technology is seen as a combination of computing technology and communications infrastructures. The most noticeable trend is integration of media. The variety of media types is an important feature of modern information systems. Meanwhile, in order to deal with the variety, integration is a critical concern. From this perspective, multimedia can be seen as a combination of variety of media types and integration of information and communication processing. This is just the most important feature of modern telecommunication systems. A multimedia system is then one which allows end users to share, communicate and process various forms of information in an integrated manner.

There are many ways to classify media. Common classifications are based on physical formats and media relationships with time. In this work, we classify multimedia applications with their requirements on the networking and management architecture. Time dependent multimedia applications (also referred to as *continuous media*) such as voice and video, where the presentation requires a continuous playing as time passes, and time independent information items such as text, graphics and images (also referred to as discrete media) have extensive requirements on the operational behaviour of the network and therefore require comprehensive control and management services.

A multimedia application, in this context, is the use for a definite purpose of multimedia communication systems capable of offering well-defined functions to end-users. The multimedia application involves rather wide application areas, in particular, the areas where it is typical to combine various information sources. Users may exchange multimedia information over broadband networks, for example multimedia E-Mail or videoconferencing. On the other hand, users communicate with a remote system to access, receive or interact with multimedia information retrieval. Typical examples for multimedia information retrieval are Video-On-Demand, TV programme distribution or Distance Learning. These new applications result in new requirements for the design of communication networks at all levels, from communication infrastructure and protocols, operating systems to platforms supporting these new applications due to the new requirements of multimedia applications. These requirements include user-oriented design of multimedia systems, integration of a wide range of service types with guarantee of different Quality Of Service, real-time performance and flexible and dynamic management [Bai1999] [Leopold1995]. There are many research activities in the above research areas dealing with provisioning of multimedia services [Chan1998] [Foitzheim97] [Hong1999]. The identification of these new application areas that are evolving in parallel with the rapid advancement of integrated services communication technology. There are many areas of application where the introduction of multimedia can enhance the utility of existing systems (*application pull*). Similarly, there are many new areas of multimedia applications made possible by the emergence of new communication and IT technologies (*technology push*).

> *"For distributed multimedia systems, it is essential that Quality of Service be guaranteed system-wide, including end systems, communication systems and networks. Although many researchers have addressed issues of QoS Management, little attention has so far been paid to the QoS management services in distributed multimedia services and applications." [Hong1999]*

The widespread use of distributed multimedia applications is setting forth a new set of challenges in networking, including management of network resources in order to enable these services through meeting the required Quality of Service. As user become more familiar with multimedia services, QoS must be approached from the users perspective as well as from the system or network perspective which has mainly been addressed so far. The user must be given the opportunity to define the requirements. These must then be translated using integrated Management Services into parameters for the underlying systems and networks, satisfying the users needs.

2.3.3 Case study: ATM Implementations

During the practical part of the research, various projects with PanDacom and projects at the University of Applied Science Darmstadt enabled the author to participate on several ATM implementations. In the mid nineties, ATM was widely regarded as the most relevant and promising networking technology to enable high-capacity and multimedia end-to-end services [Korostoff1998] [McDysan1999]. Hence it became very popular and relevant in both academia and industry and a lot of effort (i.e. emphasis in research community and money) went into ATM research.

2.3.3.1 Telecommunications Development towards B-ISDN/ATM

> *"An ISDN is a network ... that provides end-to-end digital connectivity to support a wide range of services, including voice and non-voice services, to which users have access by a limited set of standard multipurpose user-network interfaces." [ITU1984]*

The idea of Integrated Services Digital Network (ISDN) was presented in the early 1980s. In 1984, the CCITT (now ITU-T) adopted the *I* series recommendations dealing with the ISDN matters. ISDN Networks have been implemented over the last decade and offer various services for the users. The highest bit rate that the ISDN can offer to the user is about 2 Mbit/s (2048 kbit/s). Services such as the connection of Local Area Networks (LANs) or services such as video conferencing require considerably higher bit rates. Therefore, the conception and realisation of a Broadband ISDN (B-ISDN) was desirable. The ITU-T Recommendation I.113 defines B-ISDN as *"a service or system requiring transmission channels capable of supporting rates greater than the ISDN primary rate"* [ITU1993]. This definition of B-ISDN does not indicate anything about its technical concept. Whereas this definition was settled from the beginning, the technical concept for B-ISDN only emerged after long and controversial discussion within the standardisation bodies and is still an ongoing process [McDysan1999]. This reflects the differing backgrounds and intentions of the participants [ATMForum1998] [DATACOM1997] [Korostoff1998]. The original driving concept of B-ISDN has been to enable high-speed transmission of voice and non-voice services and to define new broadband user-network interfaces, for both the LAN and WAN environments. ATM is the transmission technology used in B-ISDN. ATM network implementations, especially in the LAN environments, do not always necessarily strictly follow ITU-T rules mainly due to cost considerations, lack of standard conforming equipment or because the ATM Forum, an industry Forum that is a key player concerning the ATM specifications, has specified alternative solutions.

2.3.3.2 ATM Implementations

The most common strategy for ATM implementation is to replace existing backbone technologies in enterprise networks with ATM Switches. Mainly Routers are replaced by ATM Switches that offer more capacity and scalability than native technologies that are used in enterprise network backbones such as FDDI [Joch2000]. In this scenario, ATM must internetwork with existing technologies and must perform adaptation such as LAN-Emulation (LANE) or Multiprotocol-Over-ATM (MPOA) [McDysan1999].

One example of a project involving an ATM implementation has been realised at Rodenstock in Munich (Germany) where Bay Networks (now Nortel Networks) Centillion ATM Switches have been implemented into the backbone in 1997. Experiences with the Rodenstock network have been made through a Network and System Management project which was designed and implemented by the author. The management environment is based on a Network Management Platform (HP OpenView) that is used to integrate additional Management Applications (LanOptics StarView for Hub Management, Bay Networks Optivity Internetwork for Router Management). The ATM infrastructure has been integrated into the management system through integration of other applications (e.g. Bay Networks Optivity LAN for Switch and ATM management) into the management platform. However, HP OpenView is mainly designed for TCP/IP Innternetworks and does not offer integrated Management views for TCP/IP and ATM networks (e.g. separate maps for the TCP/IP network and the ATM network).

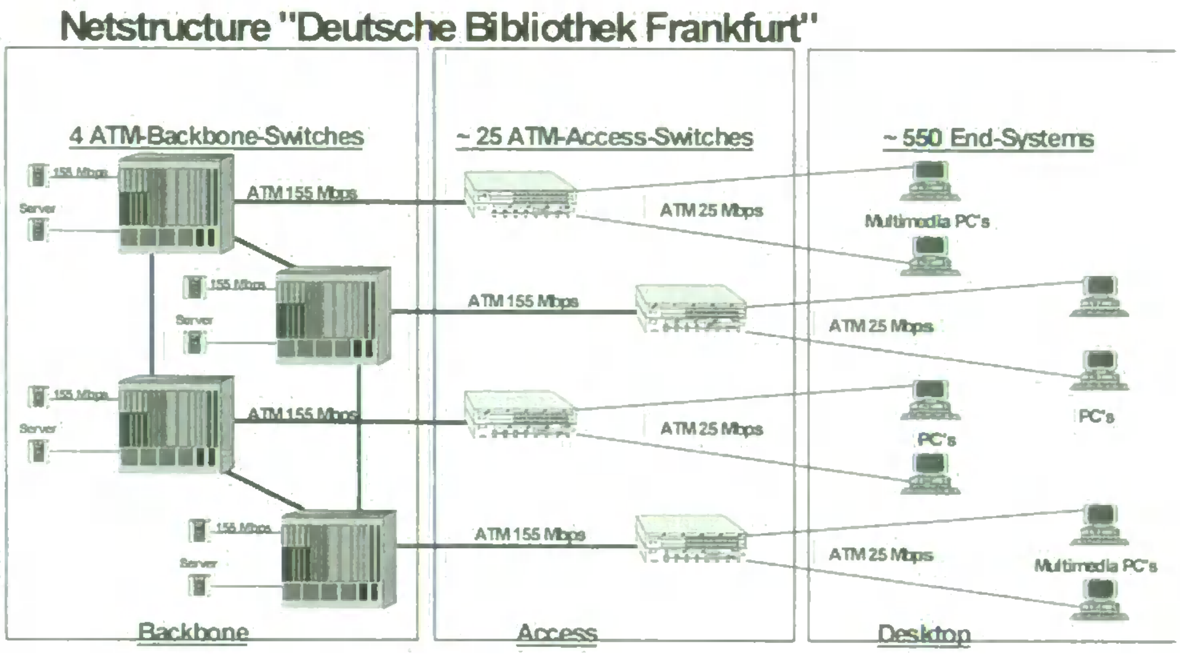


Figure 2.7: ATM Network "Die Deutsche Bibliothek"

Another strategy is to build pure ATM networks. This is most suitable for new networking infrastructures. One example of a pure ATM Network is "Die Deutsche Bibliothek" in Frankfurt am Main (the German counterpart of the British Library) where the author participated in the design and the implementation stages of the ATM network and its management solution. Figure 2.7 shows the architecture of the designed ATM solution. The architecture is divided into a Backbone, Access and Desktop area and is based on Top-Down Network Design principles [Oppenheimer1999].

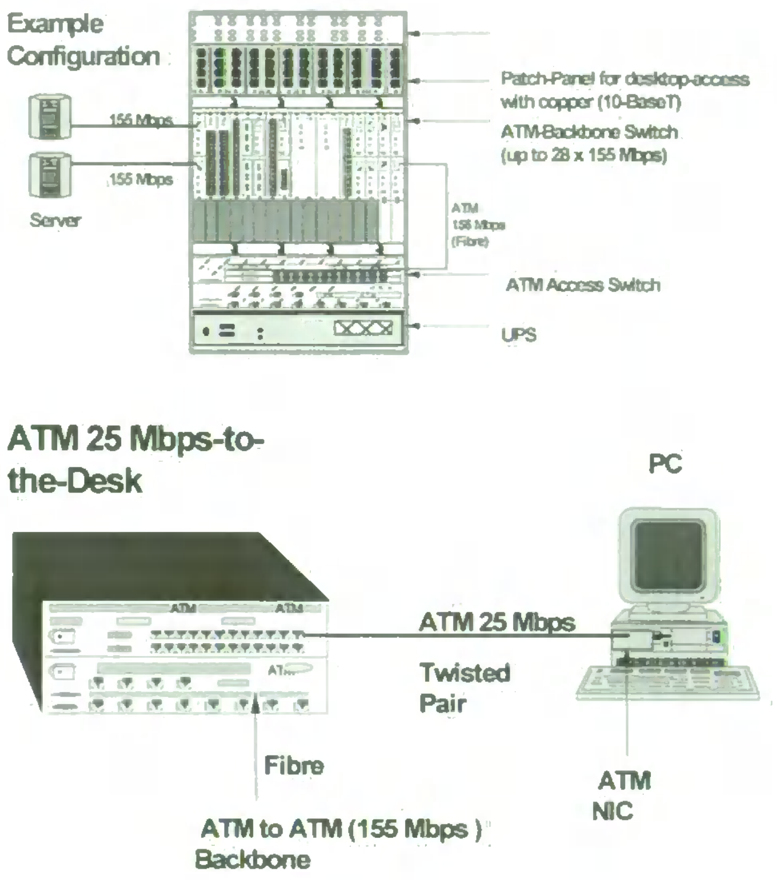


Figure 2.8: ATM Switch configurations

Figure 2.8 shows example configurations for ATM Backbone and ATM/LAN Switches as configured in the ATM network of 'Die Deutsche Bibliothek'. Fibre cabling is implemented in the backbone whereas cheaper copper cabling is used to connect the end-stations. One application of the ATM network is to bring Multimedia services to the workplaces and service stations in the library halls that typically consist of industry standard PC's. A number of lessons were learned over the different stages of the network. First of all, early adopters of new technologies are potentially faced with immature standards and immature products. In the case of 'Die Deutsche Bibliothek', the initially implemented IBM equipment did not provide the required functionality (even though the product specification sometimes claimed that it would) and needed to be replaced. Furthermore, changing standards may require to replace equipment as it can not be upgraded or it fails to interoperate with other products. Another problem encountered and lesson learned is addressing heterogeneity. It was intended to minimise heterogeneity through the implementation of a pure ATM network. However, a number of products did not support ATM (e.g. networked printers and some Unix workstations). Therefore, ATM and non-ATM needed to work together, thus requiring complex internetworking. Furthermore, the management of the environment proved difficult and did not satisfy the user's requirements. The provided software was initially developed for IP networks and regarded ATM as a 'add-on' technology with non optimal support.

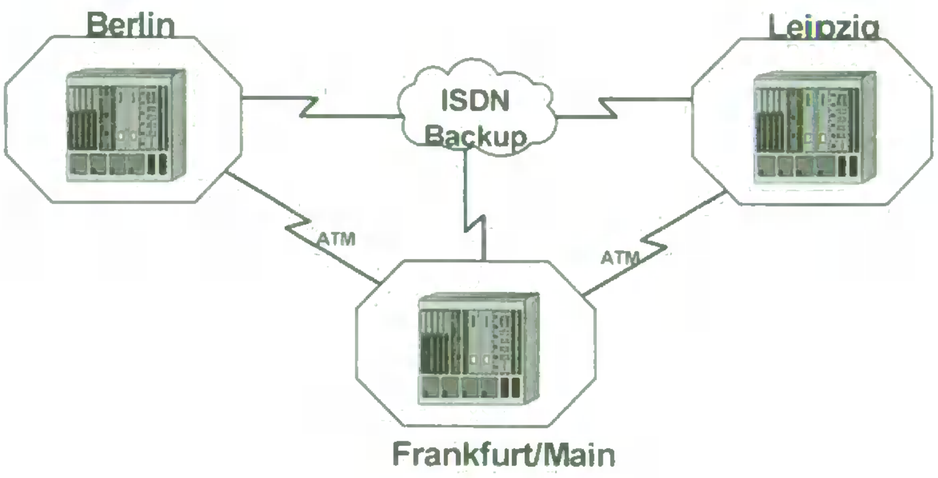


Figure 2.9: LAN/WAN Integration

Figure 2.9 shows the planned (and partly realised) WAN links to other locations of the "Deutsche Bibliothek". However, the relatively high prices of ATM services and the high prices of ATM/WAN equipment led to the customer decision to implement (Narrowband-) ISDN links (up to 2.048 Mbit/s) to the other locations.

2.3.3.3 ATM Management Requirements

ATM technology imposes new requirements and challenges on Network Management [Cekro1997] [Willets1996] [Wiltfang1999]. For this, it is necessary to identify and highlight ATM specific management issues and to discuss the current state in Network Management (see also Chapter 3).

Currently, Network Management is mainly concerned with the support of configuration management, network monitoring and isolation of faults. However, having in mind the possibility of controlling all the services and all elements in the network, the goal should be to integrate services to enable operation and maintenance and to prevent faults.

The necessity of a new role of ATM management has been recognised by previous research [Willets1996] [Bhagavath1997]. Examples for early ATM Management research include an architecture for the integration of performance management and resource control [Pathak1994]. It has been established that it is necessary for management systems to quickly determine and execute corrective actions and to provide Management Services in response to new technologies such as ATM and enhance communication services. [Wiltfang1999] provides a collection of the various requirements upon the management of ATM Networks. It proposes the usage of SNMP as a Management protocol and proposes an architecture to integrate ATM Management. However, it does not consider the usage of a distributed management architecture to integrate ATM and non ATM Management Services.

ATM incorporates new challenges for Management Services and procedures [Knahl1998] [Wiltfang1999]. One challenge is how to juggle with different service requests, in order to determine the best compromise between save use of the network (e.g. by allocating peak rates to guarantee network availability) and efficient use of the capacity (e.g. by assuming that users will not start using the bandwidth at once). Another challenge is how to integrate the management tasks of the existing infrastructure with new ATM technology into the existing management environment. Lessons learnt in industry projects include, that products used in the industrial implementations typically do not offer the functionality to manage services throughout the network. Beside that, there are no ATM management implementations that allow the monitoring of network services throughout the whole network. The management applications implement the management of individual ATM devices, such as ATM Switches or ATM-To-LAN Switches rather then end-to-end management [Cekro1997] [Wiltfang1999].

Most ATM management applications use a device view for management purposes and allow the administrator to monitor device status and activity, and to configure these devices. In this case, certain values of management parameters (i.e. link load, discarded cells on a connection) are considered critical and the management station receives an alarm when the values are exceeded. This usually leads to a manual reconfiguration of the affected system by the responsible administrator. However, it must be established which values are critical for an ATM service and for the overall network health, how to change the presented parameters and to determine whether it is necessary to change a network configuration. It is common that by the time alarms are received, problems have already arisen. Hence a more proactive strategy is required. So, in attempting to relieve the Network Manager of these decisions the key questions to answer are, which parameters have to be monitored, which values must be considered as critical, which actions must be performed after the critical values have been exceeded or errors have occurred [Wiltfang1999]. It is not possible to answer these questions when relying on standard equipment (i.e. the equipment itself can not determine which values are critical with respect to optimal resource allocation for a particular end-to-end service). For this reason it is necessary to integrate Network Management tasks to support the quality of the overall network services. The question is which circumstances disturb an ATM service (or an ATM connection) and what helps to prevent this. Therefore, an integrated view for the definition of management tasks has to be provided.

2.3.3.4 The Future in the Networking and Telecommunication World

ATM field trials that implement new services, applications of research institutes and in industry provide valuable experience and knowledge for the further development of ATM and other networking technologies. However, there are still many options for the evolution from current narrowband networks to a future broadband infrastructure. It has also been questioned whether there is a future for ATM networks in the LAN and in the WAN given the rather limited market acceptance that ATM has received and the emergence of alternative technologies (e.g. Gigabit and 10 Gigabit Ethernet for Enterprise Networks) [Oppenheimer1999] [Nolle1997] [Norris2000].

A likely scenario for the near future is an ATM backbone network infrastructure in the WAN serving as access platform for business customers ATM networks and transport data as well as other services such as voice and video. Even though ATM service sales right now are much smaller than Frame Relay service sales [DATACOM2001], ATM is still regarded as a "*growing force in the world of ATM access*" (but it is certainly not the main driving force in the networking and telecommunications market) [Clark2000]. It can be assumed that existing and new network and service providers (e.g. in Europe after deregulation of the telecommunications market or in developing countries where there is no large basis of installed Frame Relay) are likely to continue to implement ATM based backbones capable of integrating voice and data traffic and that ATM remains a significant force [Clark2000] [Nolle2000]. On the other hand, Enterprise Network operators that deployed ATM as the backbone technology for their organisation's network to provide more capacity in the 'bottleneck' of the network and to have an upgrade path towards further ATM integration in the LAN as well as in the WAN are likely to keep their infrastructure in place for the foreseeable future [Clark2000] [Joch2000]. Workgroups or single user and server-systems that require high-capacity and multimedia services will be (and already are in some cases) typical ATM users. Numbers for ATM Networking Equipment sales such as ATM Enterprise Switches show a large growth rate but are still relatively small compared to other networking equipment sales figures [DATACOM1997] [DATACOM2001]. The biggest share of the LAN market is taken up by 10 Mbit/s, Fast and Gigabit Ethernet technology for the Access, Distribution and Core Layers of current networks [DATACOM1997] [DATACOM2001] [Oppenheimer1999].

This shows that ATM will be more and more important as a high capacity and multi-service networking technology, but it is quite unlikely that it will be the only networking technology in the LAN as well as in the WAN [Nolle2000]. ATM will coexist with other networking services such as Frame Relay in the WAN market and technologies such as Ethernet or Gigabit-Ethernet in the LAN market. Furthermore, the relevance of IP might lead to the operation of LAN and WAN backbones purely based on IP services (i.e. making IP routers the core Building Blocks of the LAN and WAN Backbone) [DATACOM2001].

All technologies and industries develop and mature over time. Networking cannot possibly be an exception even though it has certainly developed rapidly in a relatively short time. Even the 'legacy' 10 Mbps Ethernet networks or old Mainframe systems are little more than vibrant middle aged performers; ATM, the Internet and the PC are still teenagers; services such as multimedia World Wide Web (WWW) transactions with a guaranteed Quality of Service (QoS) are toddlers – when compared to most other engineering or technical artefacts. In many ways, the capabilities of the communications industry have advanced faster than our ability to harness them. There may be plenty of drive and energy behind the advances in computers and telecommunications, but has there been the time to acquire the wisdom and in depth understanding of other engineering disciplines? It took biologists and physicists hundreds of years to hone their skills.

2.3.4 End-to-End Service Provision

The type of network operation and control that have been envisaged and proved effective for ATM networks do not directly carry over to a connectionless, packet-based internetwork architecture such as Internet Protocol (IP) based networks. These architectures do not rely on end-to-end virtual channels that can be managed in terms of capacity and variability of bit rates. Instead, the key to effective performance management in packet-based internetwork environments are packet switches, routers and its routing capabilities. The new requirements and challenges for these environments are:

- To have the processing capacity to move data (e.g. IP packets or ATM cells) through the network (e.g. through the routers and / or switches) at high rates to provide the required high network capacity;
- To meet the Quality of Service (QoS) requirements of new applications (e.g. Multimedia Applications);

- To have sufficient knowledge of the networked configuration and the condition of the overall network to route in an appropriate way to a given class of traffic and to compensate for congestion and failure of the network;
- To employ a scheme for exchanging network information with other networks that is effective and does not excessively contribute to the overall network traffic.

The first point has to be addressed in the domains of Hardware (e. g. processor) and Operating system design and is not addressed by this research. However, the remaining points are vital for the integration of new services and are addressed by this research.

2.3.4.1 End-To-End Quality

Quality Classes can be defined in terms of network performance parameters at the edges of the provider's network. For a given Quality Class and for a specific User Domain as illustrated in Figure 2.10, a corresponding End-to-End Quality must be achieved that can be evaluated both subjectively (through user tests) and objectively (by measurements made on the application).

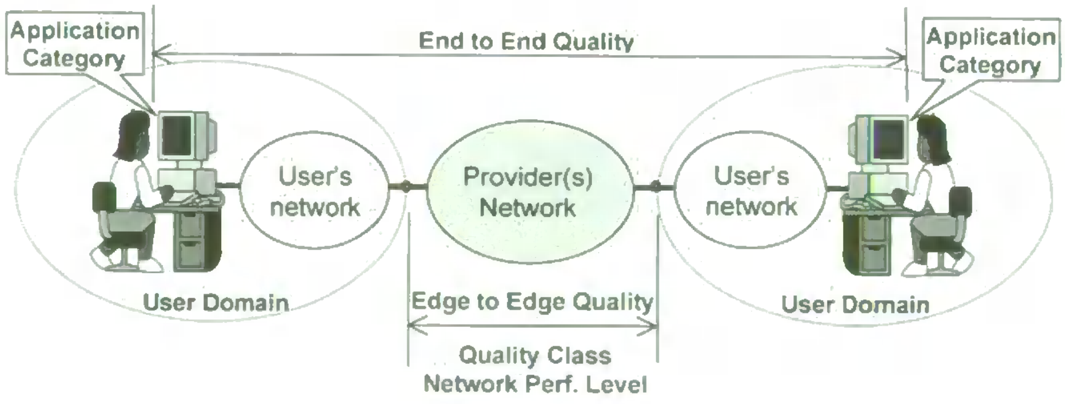


Figure 2.10: End-to-End Quality

- An *Application Category* is a set of applications characterised by a set of common basic performance requirements (e.g. real-time, interactivity, isochrony, reliability, etc.) and related performance constraints.

- *End-to-End Quality* is the communication quality (both subjectively perceived and objectively measured) at the application level.

- *Edge-to-Edge Quality* is the communication quality (objectively measured) from the ingress point of one provider's network to the egress point of that network.

- A *Network Performance Level* is a set of target values (or range of values) and related guarantees for a set of network performance parameters to be provided at the edges of the provider network. A Network Performance Level is the part of the edge to edge quality that is guaranteed by the provider.

Service Level Agreements (SLAs) are formal negotiated agreements that help to identify expectations, clarify responsibilities and facilitate communication between a service provider and its customer [Bhoj2001] [Karten1998] [Langer1998]. In a typical customer / provider relationship, a customer demands (and pays for) specified services and for a Quality of Service (QoS) that are defined in the SLAs. To enable (and prove) the fulfilment of those SLAs it is necessary for the provider to have Management Services that can monitor and control the service status.

2.3.4.2 QoS Frameworks

Several QoS Frameworks have been proposed and exist on the market. An early contribution to the field of QoS driven architectures is the ISO QoS framework [ISO1995]. The ISO framework broadly defines terminology and concepts for QoS and provides a model that identifies objects of interest to QoS. The QoS associated to objects and their interactions is described through a set of QoS characteristics. Key ISO framework concepts include the following:

- QoS requirements, which are realised through QoS management and maintenance entities;

- QoS characteristics, which represent the fundamental measures for QoS that have to be managed;
- QoS categories, which represent policies governing a group of QoS requirements specific to a particular environment;
- QoS management functions which can be combined in various ways and applied to various QoS characteristics in order to meet QoS requirements.

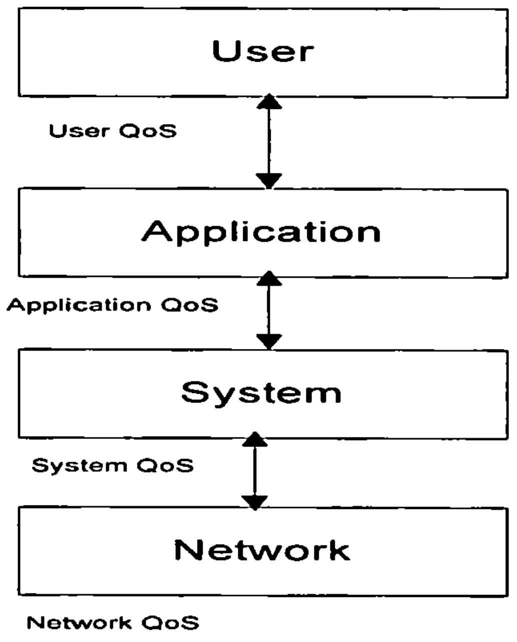


Figure 2.11: Layered QoS Framework

The ISO QoS framework is made up of layer-specific and system-wide types of management entities that attempt to meet QoS requirements by monitoring, maintaining and controlling end-to-end QoS (see Figure 2.11). The task of the policy control function is to determine the behaviour that applies to a specific layer of an open system. The policy control function models any priority actions that must be performed to control the operation of a layer. The definition of a particular policy is layer specific and therefore can not be generalised. Policy may include aspects of time-critical applications, security or resource control. The role of the QoS control function is to determine, select and configure the appropriate control entities to meet layer specific QoS goals. The system QoS control function combines the system-wide capabilities to tune the performance of protocol entities and to modify the capability of systems via OSI System Management. The OSI System Management interface is supported by the systems manager, which provides a standard interface to monitor, control and manage end-systems. The system policy controls function interacts with each layer–specific policy control function to provide an overall selection of QoS functions and facilities.

A number of other frameworks and architectures have been proposed or been implemented by academia and industry. Current work is based on one of the earlier proposed specifications or incorporates ideas from several frameworks.

An early project based at IBMs European Networking Centre in Heidelberg, Germany, has developed an integrated QoS model – The Heidelberg QoS Model - as part of the HeiProject as early as 1991 that provides guarantees in the end-systems and network [Hehmann1991]. The communications architecture includes an infrastructure that provides media mapping and media scaling. The network supports QoS based routing and QoS filtering based on IBM protocol implementations. The key to providing end-to-end guarantees is HeiRat (german for Marriage) which comprises a QoS management scheme that includes QoS negotiations, QoS calculations, admission control and resource scheduling. The HeiRAT scheduling policy is a rate-monotonic scheme whereby the priority of a thread performing protocol processing is proportional to the message rate accepted.

The QoS-Architecture (QoS-A) is a layered architecture of services and mechanisms for the management of QoS and the control of continuous media flows in multiservice networks [Campbell1993]. The QoS Management Architecture (QoSMA) follows the vertical approach and uses QoS objects, which must be particularised for each of the 4 layers of the architecture and external entities – user, service provider and network provider – which represent and incorporate the parties that provide the multimedia services [Alfano1995].

Quality of Service for CORBA Objects outlines a framework for constructing adaptable applications by introducing concepts for QoS of object access and providing mechanisms which can be used to integrate these concepts into emerging applications [Zinky1997]. The initial focus of this work has been on adapting CORBA objects to manage network communication resources. However, an extension of the CORBA functional IDL with QoS parameters based on a QoS Description Language (QDL) has been proposed. Here, the QoS and usage specifications are on the software object level, not the communication level. It provides mechanisms for measuring and enforcing QoS agreements and for dispatching handlers when the agreements are violated to help distributed applications to be more predictable and adaptive even when end-to-end guarantees cannot be provided.

Another contribution is the development of a Quality Assurance Language (QuAL) for the specification of QoS constraints on underlying computing and communication platforms [Florissi1994]. QuAL is intended to ease the management of QoS for distributed multimedia computing and communication applications and provides language-level abstractions for the specification, negotiation and management of communication and processing QoS constraints.

Two Architectural Models for Internet Protocol QoS have been proposed in the Internet Engineering Task Force: The Integrated Services (IntServ) and the Differentiated Service (DiffServ) architectures [Xiao1999]. A fundamental difference between the two is that the IntServ model includes the definition of a signalling mechanism and an admission control framework and seems preferable for the development of IP QoS in a real life network [Salsano2002]. The early work by the Integrated Services (IntServ) Group [Braden1994] of the IETF was a significant contribution to provide controlled QoS for multimedia applications over an Internet Protocol based internetwork. The group has defined a comprehensive architecture and QoS framework used to specify the functions of internetwork system elements and functionalities [Xiao1999]. The behaviour of elements such as routers, subnetworks and end-systems is captured as a set of services, some or all of which are offered by the elements. Each element is QoS aware and supports interfaces required by the service definition. The concatenation of service elements along an end-to-end data path provides an overall statement of end-to-end QoS. IntServ currently offers the following optional services: Guaranteed service, controlled load-service and best effort service [Wroclawski1997a] [Wroclawski1997b] [Shenker97] [Xiao1999]. The flows in the IntServ architecture are characterised by the traffic specification which looks at the traffic pattern a flow expects to exhibit and a service request specification which regards the QoS a flow desires from service elements. Current research is further looking at ways to manage and control IP DiffServ Networks [Flegkas2002] [Salsano2002].

2.3.5 Impact of new software technology

> *"There is a widening gap between ambitions and achievements in software engineering. The gap appears in several dimensions: between promises to users and performance achieved by software, between what seems to be ultimately possible and what is achievable now and between estimates of software costs and expenditures. The gap is arising at a time when the consequences of software failure in all its aspects are becoming increasingly serious." [Naur1976]*
>
> *"The average software development project overshoots its schedule by half; larger projects generally do worse. And three-quarters of all large systems are "operating failures" that either do not function as intended or are not used at all." [Gibbs1994]*
>
> *"Software Components are clearly not just another fad – the use of components is a law of nature in a mature engineering discipline." [Szyperski1998]*

The above quotes outline the dilemma of software development and software implementations in general and refer to the often-quoted *software crisis* (i.e. the continual failure of software providers and developers to meet requirements with respect to quality, availability and cost). Nearly 20 years passed between the first quote taken from a 1976 NATO Conference on Software Engineering, and the second from an article in the Scientific American in 1994, but the key problems in the broad context of Software development remained. At the same time, new development approaches are constantly arising that promise to solve the current problems and limitations (e.g. Componentware). This situation is also affecting the provision of communication services and the development of Integrated Network and System Management systems, that are themselves complex software applications developed to provide management of networking Hardware and services [Patel2002].

The Network and System Management development community is therefore faced with a potential dilemma. The rapidly changing nature of the communication networks and services imposes wide-ranging and volatile immediate requirements on Network and System Management applications whilst holding up the development of standards that could simplify problems of interoperability and encourage standardised solutions. Meanwhile, the lack of a common framework for the development of management systems leads to the development of proprietary systems. These proprietary systems then hold-up the development of common approaches. Even though this situation is not different to any industry sector that relies on software, the highly distributed and interconnected nature of communication service provision means that manageability and interoperability are key requirements for the industry.

This issue is actually a compound of a number of different problems, which together make up the overall predicament that software demand exceeds software supply (i.e. the productivity of software developers cannot keep pace with the demands on their services) [Gibbs1994] [Phippen2001]. Furthermore, software project management generally falls short on cost and time estimates and software quality is sometimes less than adequate.

The development of open programming environments, client-server concepts and object-oriented distributed systems results in the availability of applications and system services based on distributed system networks, adding further flexibility as well as further complexity to the software development and deployment process. Distributed systems allow the data, processing and visualisation modules of a distributed application to be separated logically and these modules to be distributed on a function-dedicated and usage-dependent basis to appropriate physically separate systems. If standardised distributed software concepts and protocols are used, then an appropriate customisation and evolution using even heterogeneous computer systems is possible. Networked systems even enable the formation of virtual computing structures over physical distances. An example is the use of large numbers of networked workstations as virtual parallel systems for tasks requiring extensive computing and storage space rather than the usage of monolithic mainframe systems. However, while the problems have remained the same, the domain in which software exists and functions – the *software environment* - has not and is constantly changing.

- Software is written on top of systems. The more powerful the underlying hardware and operating systems, the more complex the software can become and the more it can potentially achieve. As the hardware industry is also constantly progressing, additional requirements are placed upon software to exploit the new potential of the hardware.

- Meeting the increasingly complex requirements of users has led to a continuous growth in the scale of both the implemented software and manpower used to develop it. With this increase in scale come far greater management and design challenges for software development projects.

- The requirements for service and application provisioning in the software environment is continuously broadening and new areas have requirements that may not suit existing development techniques. For example, when event-driven systems were first coming into being, the then ubiquitous procedural development techniques were unsuitable, and new tools had to be produced.

- The explosion in the use of distributed environments, such as the Internet, demonstrates the massive impact that standards can have upon the software environment. Without agreed standard technologies (e.g. TCP/IP, HTTP or HTML for the Internet), this boom could not have happened. LAN based application development was reasonably straightforward - the software systems could utilise whatever network protocols were chosen for the LAN. However, to achieve highly distributed application is extremely difficult, as one could not guarantee network interconnectivity among a distributed collection of computers. Developing and deploying services and applications within distributed environment means that a server application must be virtually guaranteed to work with another client, and vice versa. As a result of this, mainstream software development has once again moved into a new area. Industry analysis indicates that the majority of new software systems will be developed as distributed applications, mainly based on World Wide Web services [Forrester1999] [Microsoft1998].

Therefore, it can be argued that rather than developing and refining maturing technologies, the software development world has emphasised on trends resulting in the ever-expanding heterogeneity of the software environment. Development approaches are usually based on evolutionary or reactionary considerations (as opposed to radical considerations) and benefits are therefore small.

Supporters of Software Components and Componentware portray it as the development technology that will finally realise the potential for software reuse and move software engineering into an industrial practice [Szyperski1998] [PCWeek1997]. Component-orientation can be considered to be the foundation of any mature engineering practice. Often quoted comparisons between electronic or mechanical engineering and software engineering, highlight the fact that an electronic engineer does not start every circuit board from scratch, they determine the functionality required and the components they need to achieve this [Cox1990]. The same is true of a mechanical engineer building, for example, a gearbox. They would not consider designing new gears, building the new gears and then incorporating them into a gearbox. They would select existing parts to achieve their aims.

However, the same cannot be said of software engineering. The vast majority of software projects still practise rather limited reuse, and the concept of achieving requirements through the construction of pre-existing or third party components seems to go against the ethos of a large part of the software development community (which seems to be of the opinion "if you didn't write it, don't use it") [Phippen2001]. The outlined Componentware approach to provide a framework through the selection of existing Software Components (rather than to build management solutions from scratch) can achieve this aim. The research outlined the principles for the implementation of Building Blocks in the form of Software Components and demonstrated its principle advantages. However, it merely illustrated the principles rather than proving a solution build on 'Off-the-Shelf' Building Blocks, as 'traditional' development was required to build the required Software Components. Future focus of this research should further emphasise generality (i.e. provide solutions based on industry-wide accepted mechanisms based on 'Off-the-Shelf' Building Blocks).

Common system development problems can be addressed by the development community working on management solutions by establishing a common development framework and methodology. It is widely recognised in Network and System Management that the lack of a coherent development approach has often resulted in proprietary management solutions that provide little integration with other management solutions and limited adaptation to specific management requirements [Furley1997] [Lewis2000]. A development framework for Network and System Management is an organised set of architectural and methodological guidelines for the development of a management system that provides the required functionality of the envisaged problem domain. The development Framework must provide architectural guidelines that help the developers to identify the requirements and to divide the domain into smaller blocks and methodological guidelines for the development. It must further comprise the common underlying IT functionality required to build management solutions.

2.3.6 Impact of the Internet

"With the recent advent of new network services like the Internet as well as multimedia applications, and the rapid spread of wireless telephony, we can expect future communication environments to become extremely complicated and changeable, where various types of networks, terminals, applications and methods to use services will be intermixed. Therefore, the development of adaptive and self-organising communication systems will become increasingly important for the achievement of greater capabilities of services and a higher network efficiency." [Kosuga1999]

The architecture of today's Internet is built on a collection of a small number of core backbone providers called Network Service Providers (NSPs). Each backbone consists of a packet-switching network running over high-speed lines and a boundary of high performance routers. The NSPs share traffic at exchange points. Local and regional Internet Service Providers (ISPs) maintain their own communication facilities of routers and lines to connect with the customers and the NSPs. They contract with backbone providers to carry their traffic over the Internet.

Enterprise networks that are geographically distributed, owned and operated by businesses (e.g. world-wide operating companies such as McDonalds or IBM) or other organisations (e.g. world-wide operating organisations such as the United Nations) typically consist of a number of LAN networks that are interconnected by leased lines, public or private packet-switching networks or a mix of both. These networks, in turn, have one or more access points into the Internet, typically routers or access switches.

The traffic that the Internet and these enterprise networks must carry continues to grow and change [EC1999] [Wang1998]. The demand generated by widespread traditional database applications such as file transfer, E-Mail or Telnet proved to be sufficient to challenge these systems. However, the driving factors nowadays are the heavy use of the World Wide Web, which demands real-time response, and the increasing use of voice, image and even video over internetwork architectures. Furthermore, technologies such as Voice over IP (VoIP) are adding new requirements into the network infrastructures.

The performance problems and architectural limitations are most openly seen on the Internet itself [Clark1997]. As the Internet evolves into a multi-service infrastructure, there is a growing need to support more enhanced services than the traditional Internet services (e.g. FTP or E-Mail). The current Internet is based on the so called 'best effort' model where all packets are treated equally and the network tries its best to achieve reliable data transfer. Although this simple model is easy to implement, it has a number of undesirable consequences when the Internet is evolving towards a multi service network with heterogeneous traffic and diverse QoS requirements. The flat rate charging structure attached to the best effort model has undoubtedly contributed to the growing problem of congestion on the Internet. Since the network resources are equally shared by all users, the Internet tends to suffer from the explosion of the number of Internet users and from greedy users, grabbing as much resources as possible, leading to an unstable system. The widespread lack of Integrated Management Services (e.g. end-to-end bandwidth management, delay guarantees) in the current Internet also prevents ISPs from creating flexible packages to meet the different need of their customers [Wang1998].

As the Internet evolves into a global commercial infrastructure, there is a growing need to support more enhanced services than the traditional best-effort services. To address this issue, there are new efforts in the IETF and the research community to develop new, integrated management and service models (e.g. Differentiated Services Model [Flegkas2002] [Salsano2002]). While the number of Internet users keeps growing, the architecture of the Internet and the associated Management Services must scale with the increasing number of users and the exploding amount of traffic without altering the current Internet paradigm much to guarantee market acceptance. Furthermore, the integration of Internet services into legacy end-systems (e.g. the integration of Internet services into current mobile phones through Wireless Application Protocol or into 3rd Generation Mobile Networks) will add further challenges and complexity to the management of the overall network services.

Unfortunately, recent advances have not spawned much work in pricing yet [Fan1999] [Redmond2000]. This is due to the complexity of economic elements involved and the fact that the shape of the services of the Internet is still not fully defined. Current pricing practices are very different on the Internet and the Telephone Networks. Calls using the existing telephone networks typically incur a usage price which is depending on time, distance and destination, whereas most Internet Service Providers use flat rate pricing (users pay an access capacity dependent connection fee or they pay a certain rate for a specified amount of data) which is an important factor in stimulating traffic growth and the development of applications. Its major advantage is simplicity leading to lower network operating costs. A disadvantage is its inherent unfairness in which a light user has to pay as much as a heavy user. Furthermore, in this scheme there is no restraint to prevent users from reserving excessive bandwidth, exacerbating the state of network congestion. Apparently, flat rate pricing is no longer tenable in an environment of rapid commercial growth, rising demand for high bandwidth and rising need for service differentiation.

Electronic Commerce (E-Commerce) provides the capability of buying and selling products and information on the Internet and other online services (e.g. Mobile Phone Networks, where M-Commerce would be the referring acronym) [Kalakota1999]. E-Commerce is about building better relationships among customers, products and suppliers. Among its enablers are traditional tools and techniques, like Electronic Mail and WWW, while at the same time designers are eager to take advantage of new technologies, like mobile systems and intelligent agents. Online auctions, networked catalogues, portals, brokers are concepts, practices and tools under constant development in the E-Commerce arsenal [Aaron1999]. Quality and security are key issues, which can hardly be overestimated [Prahalad1999]. Everybody has seen poor Web page designs and defective payment facilities even in the largest companies and organisations (e.g. the author's first order with the Internet bookshop "amazon.de" was never billed to my credit card account). Since numerous businesses are now in their first attempt to enter the E-Commerce realm, quality and security are two particularly important requirements, which can easily be sacrificed to cost cuts. E-Commerce services are being adopted in various areas and growing rapidly [EC1999].

2.3.7 Summary

It has been asserted that the history and development of telecommunications and networking service engineering has revealed the need to define and provide services independently of the underlying communications technology (i.e. 'the network') in a manner analogous to the hardware independence compilers afforded early by software engineers [Mac1994]. Many industry groups and standards bodies can be seen to have acted in the light of this demand, even if it rarely explicitly stated. A great number of public and private networks have emerged offering new services, particularly the importance of the Internet services is playing a major role. Regarding this development, it becomes obvious that the key of future networks will be the service provisioning. Different technologies will enable these services; there will not be 'universal' technology from the desktop to the WAN but a heterogeneous mix of technologies, required to inter-operate with each other. Hence, it may become even more complicated to layer communication systems through multiple, iterated tunnelling layers (e.g. IP over ATM). Internetworking between organisations and end-users have led to a linking of Intranets to Extranets. There is a technical demand for the formation of groups through the creation of virtual networks, which are based on the existing network structures of companies and transit networks of service providers. These will offer new technological and cost-effective options for communication links to external business partners. This means that new demands are being placed on the management of these networks in the combined LAN / WAN environment and that there is a need for a definition of quality standards for the different management functional areas.

Distributed enterprise applications and services are gaining in company-critical significance (e.g. business applications such as SAP R/3), thereby increasing requirements such as availability and security. Furthermore, time sensitive services (e.g. videoconferencing) are becoming more and more popular. Security sensitive services (e.g. electronic commerce or home banking) are gaining acceptance. The whole network will become a big marketplace. Hence, the quality of the services will be dominated by factors such as availability and security. It should be mentioned that individual IT providers typically do not have complete freedom in establishing security policies and implementing security measures because the legal frameworks of different countries can also have a considerable bearing on how systems are used. The requirement to set up and administer the information basis (e.g. to check the consistency of links or to filter unauthorised data), results in extensive management challenges, especially in the backdrop of a rapid growth in the use of communication services.

2.4 Specification of Management Aspects

Since the management of distributed systems is still an evolving research area, many terms are used ambiguously. The generic concepts for the management of distributed systems are outlined and are then formalised in definitions, that will be used throughout this thesis. The following definitions are used to avoid any ambiguity.

2.4.1 Generic Management Areas

Not only the network but the entire distributed IT infrastructure complete with the communication services and distributed applications provided must be controlled and handled by a preferably integrated management structure [Chen2000] [Knahl1999]. This affects both, Local Area Networks (LANs) and Wide Area Networks (WANs). By *management* in this context, all measures are meant that ensure the effective and efficient operation of a system with its resources in accordance with defined goals. Management is therefore responsible for providing and maintaining the services and applications of the distributed system.

Definition 2.1: Management

Any activity with the purpose of ensuring the effective and efficient operation of a system with its resources in accordance with defined goals.

Viewed in general terms, the management of distributed systems encompasses personnel and procedures (i.e. business processes) as well as the technical systems and affects different levels of objects that are managed. The emphasis of *Network Management* is on the management of communication services and active and passive (cabling) network components, whereas *System Management* relates to the resources of end systems and applications.

Definition 2.2: Network Management

Management of communication services and components.

Definition 2.3 : System Management

Management of systems including software and peripheral devices that make use of the communication infrastructure. This includes the management of monolithic as well as distributed applications and applications available on a dispersed basis.

However, if a service is a general communication or system service, then the distinction between Network and System Management can be unclear. If services are implemented by distributed applications (e.g. WWW or Mail services, video conferencing or value-added services in voice communication) and offered to the internal or external users of an organisation, then these services may require the adaptation and reconfiguration of the network infrastructure (e.g. to provide a higher network capacity for a user that requires a new services). Furthermore, these services must be managed so that they can be processed, provisioned and administered in a customer-oriented way with respect to possible Service Level Agreements (SLAs) [Langer1998] [LewisL1999]. This means that a *service management* level is to be inserted above network and System Management.

Definition 2.4: Service Management

Management of End-To-End Services incorporating the various aspects of Network Management and System Management related to a Service.

Objects of Network Management include networking equipment such as switches and routers and protocol entities as well as communication lines. Objects of System Management include Server and End-Systems, file systems, software modules and user management. Examples of System Management Services include software distribution and license control, alarm forwarding and state management, where the last two are also services of Network Management. Examples of objects of service management include performance reports, trouble tickets and service level agreements.

The overall management of an organisation combines the tasks of financial, personnel, technology and production management from the standpoint of corporate wide aspects (areas of business, business processes) and establishes the concept of *policies* in respect to the IT infrastructure, operating processes and the associated services. The policies of enterprise management produce the conditions for creating the management layers below it [Sloman1995] [Sloman2002] [Wies1995] [Stone2001].

Literature on Network and System Management uses the term policy quite differently, as the following different quotations illustrate:

> *"Management policies are derived from the goals of management and related business strategies to define the desired behaviour of distributed, heterogeneous systems, nework and applications." [Wies1995].*
>
> *"Policy is one aspect of information, which influences the behaviour of objects within the system, they define relationships between managers and managed objects." [Sloman1995]*
>
> *"Policies are the plan of an organisation to meet its goals." [Langsfort1993].*

Wies uses the term policy to define what has to be accomplished not how it is done. Langsfort and Moffett use the term policy to define the goals that should be reached by the distributed system and allow the allocation of the resources to achieve the goals. All cited authors agree on the fact, that a policy should influence the behaviour of objects in the system. But due to different viewpoints and different intentions about the usage of policies, they disagree on the question whether a policy should define the way in which the desired behaviour can be achieved. On a very abstract level it is desirable to describe the intended behaviour only, without defining how it can be achieved and realised. On the other side, Network and System Management is performed with commands and procedures. An unambiguous description of how to achieve the goal is desirable for human managers and required for any automation [Stone2001]. Based on these considerations, the meaning of the general term policy as given in Definition 2.5 will be used throughout this thesis.

> *Definition 2.5: Management Policy*
>
> *A Management Policy defines the desired behaviour of a system or of system components.*

A major problem with the formalisation of management knowledge is the great semantic gap between the demands of system users on a very abstract layer (e.g. given in plain text) and the formal description of appropriate procedures in terms of executable commands. Koch defines a model for the stepwise refinement of policies from an informal strategic level to a formalised operational level based on a three level policy hierarchy [Koch1996]. A *requirement policy* defines abstract goals and management requirements. These requirements are refined into *goal-oriented policies*, defining goals and appropriate strategies to define them. The *operational policies* are expressed by use of a formal Policy Definition Language (PDL) and they can be interpreted automatically by the management system. Lupu provides a framework for the incorporation of management policies into distributed management systems [Lupu1998].

Different aspects play a role in differentiating distributed systems environments from one another: Objectives (Services and service quality), communication characteristics, underlying communication topologies and protocols and the organisational and operational structure that exists for the operators and users of the distributed system. Furthermore, the complexity of management in concrete IT structures is determined by the number and the heterogeneity of the objects being managed, physical distribution, the number of participating organisational units, the depth of service integration and the number of networks involved. This leads to the conclusion that there cannot be only one management solution and only one management system to suit all situations and that a scenario-specific customising process is essential.

Through the distribution of the objects and services that are being managed and the different areas that are involved, management itself is represented as a distributed service. Network and System Management in large networked systems therefore have to be implemented as distributed systems themselves. The goal must be an integrated management, that uses standardised concepts for universal management, permits an integral approach to the different management aspects and requirements (i.e. integration of network and System Management), includes organisational aspects (e.g. domain concept), supports heterogeneous systems and offers open interfaces to the users and developers of the management system. In an integrated management architecture, all management relevant system components co-operate on the basis of shared interoperability standards, open interfaces and established management technologies. An aim of this research is to analyse existing methods and to provide a new approach for integrated Management Services.

2.4.2 Generic Management Concepts

The following definitions are used throughout the thesis to avoid any ambiguity regarding generic management concepts.

> *Definition 2.6: Management Task*
>
> *Any command or method invocation issued with the purpose of a management related activity is called a Management Task.*

The term Management Task is not restricted to commands that will change the behaviour of an entity. Reading the value of parameters is a Management Task as is the notification about changes in a system. Usually, a set of Management Tasks must be performed in a predefined order for a certain management requirement.

> *"Service ... a meaningful set of capabilities provided by an existing or intended set of systems to all who utilise it." [Tina1997]*

The scope of this quite general definition can be consolidated so that a service can be regarded as a *'meaningful set of capabilities'* provided to the users of a system. In practical and economical terms different services correspond to different environments and different markets, so while the *'meaningful set of capabilities'* might be similar, they may be applied differently. Accordingly, a Management Service can be regarded as a meaningful set of Management Tasks.

> *Definition 2.7: Management Service*
>
> *A coherent set of Management Tasks, created to provide a solution to a management related problem is called a Management Service.*

The most abstract generic management models simply define two different entities, a manager and managed resources, where the manager monitors and controls the resources [Langsfort1993]. The manager is interested in the management functions of the resources only, internal representation and application functionality are of no concern. This suggests the use of an object model for the definition of an management architecture. An object definition provided by Wegner will be employed.

> *"Objects are collections of operations that share a state. The operations determine the messages (calls) to which the object can respond, while the shared state is hidden from the outside world and is accessible only to the object's operations." [Wegner1990]*

Using this basic definition of an object, a managed object will be defined as an abstract model of a resource by encapsulating system resources for management purpose. The state of an object can change only as result of an internal action or as result of an interaction with another object through a communication link established between interfaces. An interface specifies the signature of operations and types of attributes provided by the management object [Sloman1993]. A management object may provide several interfaces with management functionality. Two different partners participate in a management activity, described by the following definitions:

Definition 2.8: Managed Object

An object which is the target of a Management Task.

Definition 2.9: Manager

An object that initiates, controls, maintains and analyses Management Services.

A Managed Object is the view of a resource that is subject to management, such as a layer entity, a connection or an item of physical communications equipment. Thus, a Managed Object is the abstraction of such a resource that represents its properties as seen by (and for the purpose of) management by the Manager. An essential part of the implementation of a Managed Object is the relationship between these properties and the operational behaviour of the resource. Part of the implementation of a managed object is the specification of the set of management tasks that can be performed upon it and the effect that these management tasks have upon the Managed Object and its attributes. The specification may also specify the effect, if any, on related Managed Objects. The execution of a Management Task may also be conditional upon the state of the Managed Object or its attributes. An essential part of the implementation of a Management Task is the set of possible ways in which it can fail. Managed Objects may also initiate Management Tasks (e.g. emit notifications that contain information concerning the occurrence of an event associated with the Managed Object such as Traps in Simple Network Management Protocol entities).

The basic difference between a Manager and a Managed Object is that a Manager has a degree of authority to ask for information and to perform Management Services. Depending on the task being executed, an entity may act as Manager and Managed Object at different times.

Based on the state of a system, Management Events are created that notify the Manager about system states or critical situations.

Definition 2.10: Management Event

A Management Event indicates status and changes in the state of a Managed Object.

From a management viewpoint, the state of the system changes significantly if some kind of Management Task is required. This is not necessarily a corrective activity that intends to restore the former state of the managed object. If, for example, a timing event triggers the start of a backup procedure, the resulting management activity will obviously not try to reset the system clock. A management event can result from different types of Management Procedures such as monitoring of the system, policy activities and commands issued by human administrators. Monitoring is the most important source of information about the actual state of the management system.

2.5 Integration of Management Services

Distributed networked systems offer flexibility for a great freedom of scope in rightsizing and downsizing for all different kinds of organisational and operational structures and information processing scenarios. In some circumstances, this can substantially increase effectiveness in the workplace (e.g. IT supported word processing rather than typing machines) and of business processes (e.g. after integration of enterprise business applications such as SAP R/3). The price to be paid is that the organisations rely on the operation of the IT infrastructure and services. Another price to be paid is complex management that should be as transparent to the user as possible. Management is only controllable and affordable if the underlying concepts are structured, flexible and based on an integrated approach.

End users of distributed systems are interested only in results and not in the methods used. They want applications and services to meet the requirements (e.g. reliability or security) and the agreed quality (e.g. performance). Service Level Agreements (SLAs) may serve as the basis for specifying the requirements of these technical management processes. From the standpoint of business management, technical management encompasses only a part of the total management pyramid .

Most real management systems do not operate at all the layers of the Service Model for Network Architectures (see Figure 3.1), although this is the ultimate goal from the standpoint of integrated management. However, the economic success of management is primarily manifested in the top layer of the model with the lower layers acting as foundation and provider/enabler. One layer cannot be managed effectively if a lower level is not operating effectively, and so forth. Still, IT managers expect a management system to provide everything. They are wedged between the end users and business management for whom the system is essentially only a cost factor. The direct users of a management system are usually network and system administrators, help desks staff and network and IT planner. Higher management or those with responsibility for the management of the higher layers are usually interested only in cost and layer related benchmarks for monitoring and control that are obtainable through the tools of the lower layers. An integrated management concept is required to support all the groups and actors mentioned with appropriate, integrated services with respect to the organisation's business processes [Chen2000] [Hegering1999] [Strauss1995].

The following definition for Integrated Network and System Management will be used throughout this thesis:

Definition 2. 6: Integrated Network and System Management

Integrated Network and System Management refers to the linking of all management related aspects (i.e. integration of heterogeneous Network, System and Service Management).

The aim of this research is to propose the conceptual principles of a novel management framework and a methodology for Integrated Network and System Management development that meet the requirements for the provision of Management Services and overcome limitations of existing systems. The aim of the prototype is to utilise and demonstrate the basic principles of the proposed solution.

2.6 Summary

This chapter outlined heterogeneous networking and illustrated the challenges for Integrated Network and System Management. It then provided a closer look at the various terms and concepts and discussed the need for Integrated Network and System Management.

The next chapter moves on to discuss the current state of the art regarding Integrated Network and System Management and provides an analysis of the current research in this field. The thesis then moves on to introduce concepts to facilitate the development of a suitable Integrated Network and System Management framework.

Chapter 3

Integrated Network and System Management: The State of the Art and its Impacts

The reasons and economics of every existing network and every network investment are rather simple: a network operator needs to be capable of providing services to the users (e.g. enterprise network) or customer (e.g. carrier network). Some of these services are familiar and common to all network operators, while other services are new. Switched Networks, and the integration of various services (such as voice, data and multimedia) that high capacity networks such as ATM enable, are new to network operators and require new management procedures to provide them.

This chapter describes the requirements and frameworks for Integrated System and Network Management. Different approaches and solutions that have been examined in research and industry projects are described, leading to a comprehensive overview of existing concepts and research done in this area and the identification of open issues that remain.

3.1 Integration of Heterogeneous Network Management Services

After analysing published research and active collaboration in industry projects with several commercially available network and system management tools such as HP OpenView or IBM Tivoli, it can be concluded that existing approaches and current practise are not suitable for the visualisation, monitoring and control of heterogeneous, high-speed networks (e.g. ATM networks) [Gethie1997] [Knahl1998] [Ray1998]. While these tools provide basic features like network discovery and browsing of management related data (e.g. browsing of Management Information Bases), they have many weaknesses. A number of tools only allow status polling of the Managed Objects. Status polling helps to detect network component failures. However, detection of network component failures is a rather small portion of management requirements with respect to Service Management (e.g. it will not provide the manager with useful data about the behaviour of applications). There are some tools that do both status and performance polling. However, they keep these tasks separate thus making it difficult for the network manager to keep track of the huge amount of information and to identify critical faults and trends. Beside that, scalability is a major weakness of the commercially available tools. They were originally not designed to integrate different networking technologies such as ATM and IP and have different representations or different network maps for different technologies [Cekro1997] [Dornan2001] [Gheti1997]. For example, the management tools may redundantly poll devices that are part of more than one network map (devices that support more than one networking technology such as IP and ATM). This generates excessive management traffic when administering large networks. The architecture of the most widely used systems is designed to manage and control discrete entities with no knowledge of the overall network environment (apart from visualising the network in different network maps).

The isolated management of the different IT and communication islands leads to the requirement for the integration of heterogeneous network environments and their related management procedures (e.g. ATM and Ethernet management procedures) into native network and system management architectures. Other than the marketing that regularly implies the integrated nature of management products, standardisation and realisation of management frameworks and the industry realisations of management platforms, network and system management remains mainly a collection of single solutions aimed at handling specific areas.

Today the management standards, procedures and implementations for Local Area Networks (LAN) and Wide Area Networks (WAN) are substantially different [King2000] [Subranian2000]. Telecoms management of Carrier Networks is rigorous and mature because its services are charged for. On the other hand, IT management of Enterprise Networks is rather loose and immature and the users are usually not charged for the usage of services. There is no End-To-End Management for the interconnected systems that communicate via heterogeneous networks. Beside that, connections and applications handling traffic sensitive to delay require management procedures that are not required for asynchronous data applications. For example, ATM uses one technology for transmitting diverse traffic types and for a number of various applications (i.e. data, voice and multimedia) across LANs and WANs. Beside pure ATM networks, ATM is designed to operate as backbone technology or High-Speed Technology for interconnecting Server or workgroups with other networking technologies such as Ethernet or Token Ring. Integrated management procedures for the various networking technologies are required to operate and manage these networks in an efficient and practicable way and to ensure the different communication services and deliver an appropriate solution [Tsuchida1996].

Management Services for high-speed multimedia applications have to consider the concept of an integrated services network with both real-time and data services [Willets1996]. From the WAN point of view, a backbone from the telecom world based on ATM or SDH would provide all the required QoS levels for integrated services over the Wide Area Network. From a LAN point of view, existing Internet connections and corporate Intranets must carry real-time multimedia traffic (e.g. voice and video) in addition to data. Traditionally, the networking world has been divided along such lines. There has been a telecommunications network primarily designed for point-to-point communication (mainly telephony). Subsequently it has been adapted for the growing need of data communication (e. g. ISDN and B-ISDN or ATM). On the other side, there also exists a data networking world that creates LANs (and also WANs) primarily designed for the requirements of data communication. Although these two worlds are converging, an integrated management framework that unifies the various technologies and requirements has yet to be provided [Subranian2000]. Since the telecommunications world has always been very QoS focused, ATM is provisioned with mechanisms to provide different QoS levels. From the IP community the long term view is that services such as real-time voice and video can multiplex with existing traffic.

However, the QoS services and technologies have not been considered in-depth or implemented into the IP world. The current IP model, which is based on the Ipv4 protocol is flat, offering a classless, best-effort delivery service. However, the next generation of IP, IPv6, supports basic QoS functionalities (e.g. end-to-end capacity can be set aside) [Lee1998] which then needs to be integrated into the management systems. Further work into the area of end-to-end management for heterogeneous networking environments is required.

3.2 Management Disciplines and Services

The various disciplines of Network and System Management represent all actions taken to enable and guarantee the provision, maintenance and operations of the resources and services - either hardware or software - in a network. This includes the communication network as well as the server and the end-systems in a network.

ITU-T defined 5 management categories - Fault, Configuration, Accounting and Billing, Performance, Security Management (FACPS) - that define the different disciplines and requirements for the management of heterogeneous networking environments [ITU1992]. Required Management Services include configuration of network elements, monitoring and controlling of network elements, software distribution (e.g. software distribution to PCs), user management, security management (e.g. access control), service management (e.g. video) and so on. The complexity of the Management Services is related to the complexity of the environment to be managed. Different factors that influence the complexity of Network and System Management have been identified in various industry and research projects [Hegering1999] [Gheti1997] [Knahl1998]:

- The number and heterogeneous nature of managed resources (e.g. PCs, Unix Workstations, Router, Switches);

- The existence of distributed applications and services;
- The existence of different communication infrastructures and protocols (e.g. ATM, Ethernet, Frame Relay, IP, IPX);
- The characteristics of applications and their Quality of Service (QoS) requirements (e.g. videoconferencing requires a certain QoS for user satisfaction);
- The existence and characteristics of various organisational and administrative units involved (e.g. different units for PC operating and network operating);
- The overall deployment strategy in a given environment.

These factors influence the management strategy of the organisations that are planning and implementing a network and system management solution. They further establish the requirement for the integration of the different Management Services for Network Management and System Management into an Integrated Management Architecture [Gutschmidt1995] [Hegering1999] [Knahl1999]. Various standards that have been established over the last years and existing implementations for Network and System Management provide at least some integration and interoperability for the identified areas, yet the problem of integrated Management Services for these different disciplines and various components has not yet been solved [Knahl1999] [Lewis2002].

The general Information Technology (IT) strategy over the last years, and further developments planned in the near future, are changing the IT structure more and more from centralised mainframe based data processing towards distributed data processing, even though enterprise-critical mainframe based applications will still remain and coexist with new technologies. The distributed systems are based on heterogeneous Client/Server structures, and the trend is towards distributed software and application engineering with utilises middleware technologies for the interaction between the distributed software [Patel2002] [Rabhi2002]. The drawback of these technologies is that they potentially add even more complexity into the distributed environments and require additional Management Services [Rabhi2002].

The complexity of the described Management Services leads towards the conclusion that there cannot be one management standard or one management application that integrates and provides solutions for all the different areas. Typically, every organisation has a unique distributed environment and unique service and management requirements. Therefore it is always necessary to customise Management Services to suit the particular environment. This leads to the conclusion, that management itself must be seen as a distributed application. Based on the business goals and technical requirements of Network and System Management, it is necessary to select and apply management standards and tools and to define Management Services accordingly. Unlike the case of providing mainframe based IT services, today's infrastructure makes management complex and resource intensive and, therefore, a very expensive service of its own [Emanuel1994] [Knahl1999] [Lewis2002]. The familiar approach of using isolated management tools for managing separate IT areas should be replaced by an Integrated Network and System Management Architecture, based on existing standards and established concepts (e.g. open Application Programming Interfaces) for the integration and continuous extension of Management Services. Ideally, integrated management should support the concept of 'global' Management Services (i.e. end-to-end management). The building of Management Services must incorporate organisational aspects such as the existing business processes [Mayerl2001]. All these services have to provide support for heterogeneous systems and networking technologies (e.g. IP and ATM).

3.3 Management Approaches

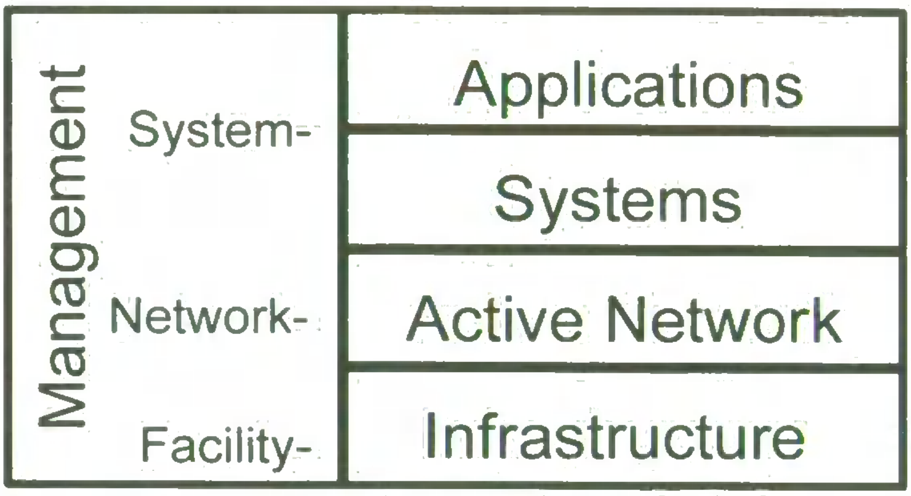


Figure 3.1: Management Areas

Figure 3.1 illustrates the areas of Network and System Management in a layered model similar to the OSI model (see also Figure 2.1). System Management deals with systems such as server or end-systems (e.g. PCs) as well as systems like printers, and is also concerned with the applications that run on top of these systems. Network Management contains the active Network Infrastructure, such as ATM Switches or Routers. Facility Management provides Management Services for the infrastructure (e.g. cabling plans) of a network. The direction in research and industry is to integrate the Management Tasks of these layers into one Integrated Network and System Management (INSM) Architecture to enable comprehensive and Integrated Management Services.

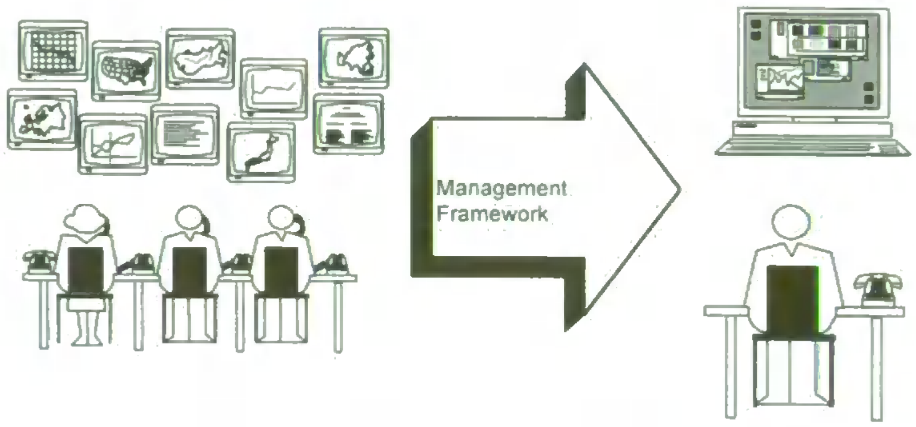


Figure 3.2: Consolidation of Management Services

A primary goal for the provision of Management Services is to have an Integrated Management system that allows the monitoring and management of the different products and various services (as opposed to a number of stand-alone tools) as illustrated in Figure 3.2. This would allow the network managers to use a single management system with a common Graphical User Interface and a common data repository. The Management Framework has to be distributed to reflect the distributed nature of the overall system and to enable distributed users to access the management system transparently. Furthermore, policies and rules must be implemented to control management and access rights between the different users (i.e. typically not every network manager is allowed to manage or even see all the objects in a network).

A number of management approaches have been proposed and established in the market, and criteria for the classification of Management Tools have been published [Hegering1999]. The objective of *Umbrella Management* is to abstract from the heterogeneity of the underlying management architectures by defining means of interoperability [Keller1999]. Most notably is the widespread use of generic management platforms to provide access to Managed Objects and the establishment of Service Level Agreements to specify service requirements and relationships between objects in a distributed environment [Gheti1997]. On the other hand, stand-alone tools can be used for isolated Management Tasks and Management Services without being integrated into an integrated management system (e.g. without integration into a management platform).

3.3.1 Provision of SLAs

In a typical relation between a network user and a network provider, a network user (i.e. a customer that demands and pays for a service) uses a provider for communication services with a specified Quality of Service (QoS) that are defined in the Service Level Agreements (SLAs) [Langer1998] [LewisL1999] [Bhoj2001]. Figure 3.3 illustrates the context and components of SLAs and the relation between the customer and the service provider [Abeck1999]. To enable and prove the fulfilment of those SLAs it is required for the provider to have Management Services that monitor and control the service status.

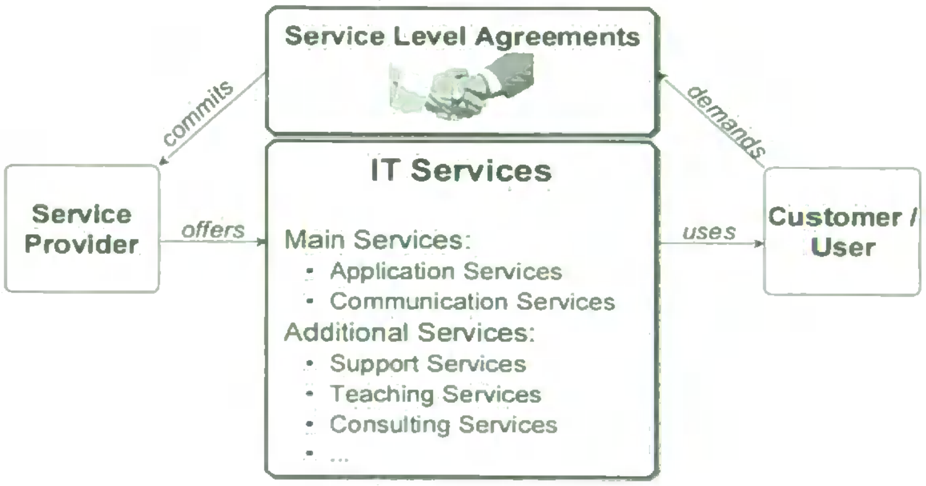


Figure 3.3: Service Level Agreements [Abeck1999]

The utilisation of Network and System Management and Service Level Management characterises services primarily from the perspective of a user or business, and secondarily as a function of several components including networks, devices and software applications that proved difficult to realise using existing approaches [Lewis2002]. In order to observe the quality of delivered services it is necessary to negotiate QoS parameters as well as modes for the measurement and evaluation of these parameters. QoS parameters must reflect the expectations of the customer and they are part of the SLAs between the customer and the provider. SLAs are for several purposes, e.g. if a service is not delivered with the specified quality the customer may obtain an appropriate reimbursement. Therefore, the SLAs and QoS parameters have to be supervised by the management system of the provider. Further, the provider is obliged by the SLA to report the compliance with agreed QoS parameters. The concept of Customer Service Management (CSM) specifies a management interface between customer and network provider that enables the customers to monitor and control their subscribed services [Langer1998].

Examples for SLAs customer/provider relationships include:

- Network provider (e.g. Deutsche Telekom, a network operator, who is acting as a service provider) and customer (e.g. PanDacom, a system integrator with branches all over Germany, who is using the services offered by this or another service provider);
- System Integrator (e.g. PanDacom as a system integrator that is offering remote Management Services based upon QoS parameters to its customers) and customer with service contract.

One example for such QoS parameters is the response time which implies the connectivity or the availability of a certain service where these parameters have to be measured and valued from the customer point of view when the customer uses a specified service, which requires parts of the management system to be installed at the customers side (e.g. intelligent agents or management gateway that report to the provider's management framework). This allows the monitoring and measurement of the services from the customers side, e.g. from an end-system to monitor end to end-connectivity.

SLAs will be modified in the course of time due to new requirements for services that demand modified or new QoS parameters. It must be possible to extend the management system with additional functionality, e.g. for the configuration and monitoring of a new QoS parameter. It is, therefore, important that the architecture is flexible and that it enables fast and cheap integration of these new services. For large scale, distributed networks it is also essential that the management system scales well. Furthermore, the variety of the different hardware and software in existing and future networks requires a high-degree of platform independence for the management system.

3.3.2 Manager of Managers

The *Manager of Managers* approach refers to a hierarchical architecture in which an integrated manager interfaces to lower-level element management systems [Kalyanasundaram1994] [Keller1999]. The Manager of Managers approach requires that the integrated manager can access functions of lower-level element management systems to provide Management Services. In practice today, only a few solutions exist and only a small management area can be addressed on the integrated manager approach [Keller1998] [Keller1999], so most functionality is obtained through access to stand-alone management applications. There are several strategies to achieve interoperability between systems, most notably are the Multi-Architectural Agents, Multi-Architectural Manager and the Multi-Architectural Gateway approaches [Kalyanasundaram1994] [Soukouti1997] [Keller1999]. In typical management scenarios, this approach requires multiple management systems at a time when integrated management is intended to reduce the number of management systems and management databases they use [Lewis2002].

3.3.3 Management Platforms

Another general approach is the *Management Platform* approach, in which different Management Services are provided in conjunction with a Management Platform that provides common Management Services [Gheti1997]. The idea for Management Platforms is that multiple Network and System Management Applications access a generic Management Platform through a standard set of Application Programming Interfaces (APIs) and use its services. The platform approach offers the possibility of a greater degree of multi-vendor integration and for easier integration of heterogeneous Management Services than the Manager of Managers architecture through the provision of open APIs to enable organisations to develop management applications for a generic Management Platform.

The following list provides an overview of generic aims and features of Management Platforms [Gheti1997] [Hegering1999]:

- Management Platforms are run in open distributed environments;
- Management Platforms provide a runtime environment for management applications;
- Management Platforms provide a development environment (i.e. can be expanded through 3rd Party suppliers);
- Management Platforms provide the means of integration of a heterogeneous underlying information model (i.e. different managed objects) and a functional integration (i.e. through the use of a platform database and shared platform services).

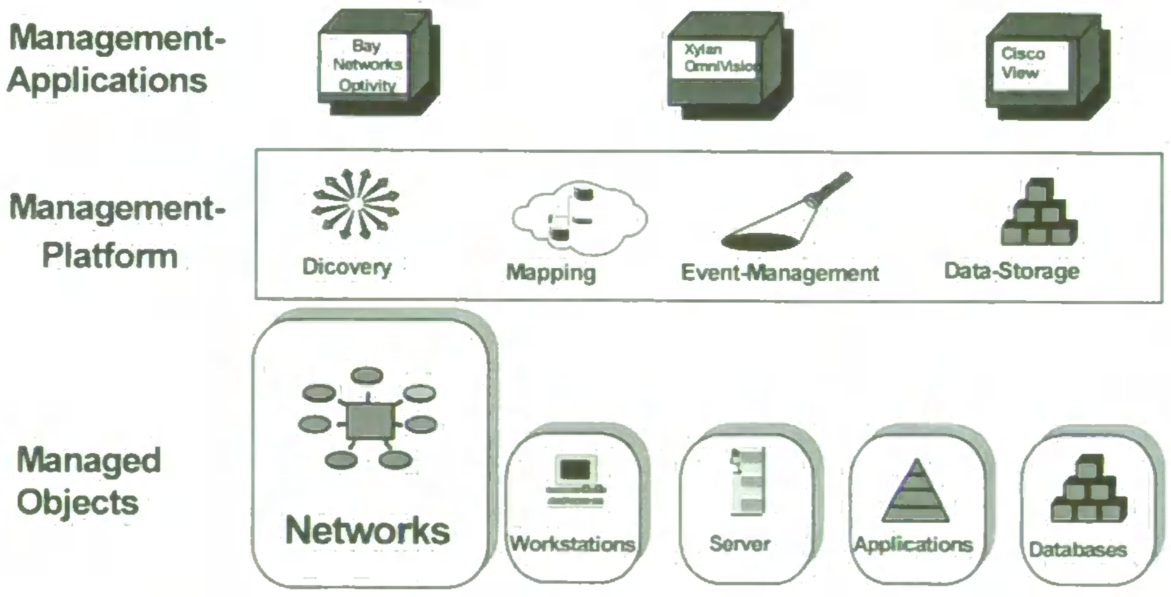


Figure 3.4: Management platform with multi-vendor applications

In the Management Platform approach, the idea is to have a single management system that can handle a heterogeneous network that provides a migration to complete IT-Management through integration of lower-level protocol standards (e.g. SNMP and CMIP) and higher-level API's (see Figure 3.4). Rather than providing a hierarchical architecture (like in the Manager of Managers architecture) the platform developer provides an open development environment in which multiple vendors can write software that shares common user views and a common data repository. If the platform offers a rich set of services and advanced capabilities such as object oriented user interface or a relational (or even object oriented) database and becomes an important system on the market, other organisations are likely to develop management applications to run under this platform [Ghita1997].

With the platform approach many organisations can simultaneously work on Management Services that will work together under the same architecture. If the required applications and services are available for one platform system, then this particular platform is likely to become the first choice for the user. If users start buying a particular platform, then this platform will become established on the market and more applications will be developed because there is a good market for them (e.g. HP OpenView was regarded as the market leader in the mid & late 1990s for Network Management Platforms and still features prominently on the market, however nowadays with strong competition from Management Platforms such as CA Unicenter, Aprisma Spectrum and particularly IBM Tivoli [Jander1999] [Dornan2001] [NetworkMagazine2001]).

Figure 3.4 illustrates the generic services addressed by a typical industry implementation of a Management Platform. The practical part of this research included design and implementation of management platforms (e.g. HP OpenView Network Node Manager) with several 3rd party management applications (e.g. Nortel Optivity) in a number of industrial projects. Using leading Management Platforms ("leading" particularly in terms of market acceptance and vendor support), typical industry implementations lack standardised interfaces (e.g. for the development of 3rd party applications) and use proprietary mechanisms (e.g. proprietary databases). Furthermore, most 3rd party applications are not integrated with the generic management platform services (e.g. Nortel

Optivity uses its own database that is not connected to the platform's database) and sometimes not even provide a common user interface (e.g. Nortel Optivity and HP OpenView GUI provide different GUI concepts).

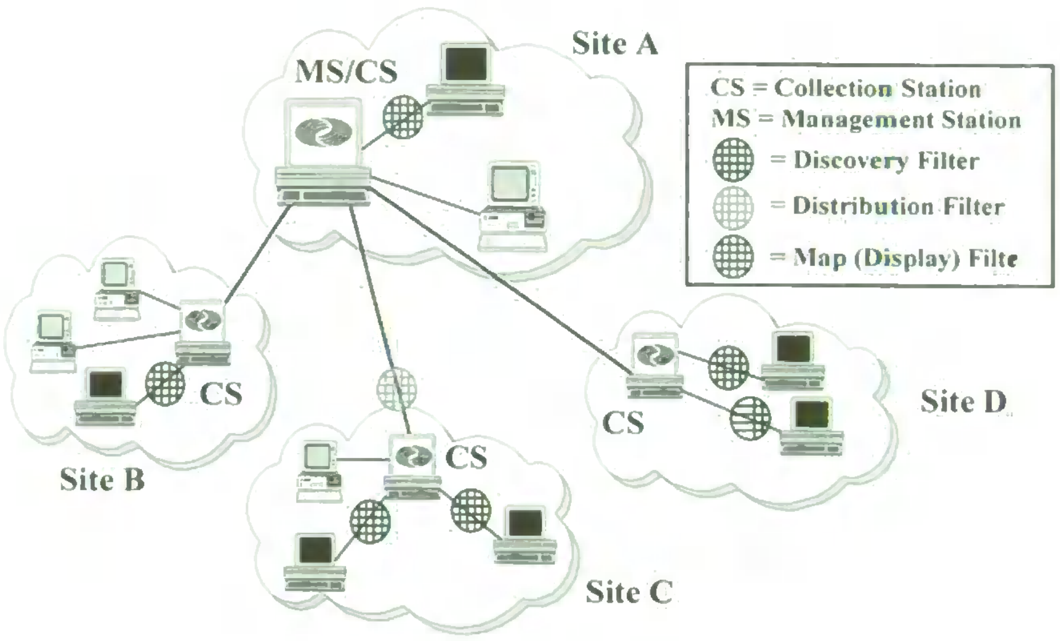


Figure 3.5: Distributed Management Platform

Leading industry Network and System Management Frameworks such as HP OpenView and IBM Tivoli allow the distribution of Management Platforms throughout the network. A hierarchy of Management Platforms can be designed and implemented, with full management functionality e.g. for remote branches that report to a central Management Platform. Distributed HP OpenView Network and System Management environments have been implemented at various industry projects. Figure 3.5 illustrates a distributed HP OpenView configuration that spans over 4 sites. Beside technical challenges (e.g. technical issues such as dimensioning of server and client machines, generic issues such as interoperability of management applications with the implemented management platforms), the design of the distributed environment itself (e.g. who is managing what, access permissions, data distribution, management of information and event distribution) is a complex and sensitive process [Hegering1999].

The Platform Approach has been widely implemented in today's Network and System Management environments. However, in practice, applications and therefore Management Services running on Management Platforms are not deeply integrated and are used alongside each other on a common user interface.

The dominant Management Platforms providers (e.g. Hewlett Packard, Tivoli and Computer Associates) use their own proprietary approaches and implementations to provide a distributed management infrastructure [Kreger2001]. Furthermore, these vendors not only compete regarding the Management Platform capabilities and infrastructures, but also on the set of products to be managed and additionally regarding the services for installation, customisation and maintenance [Hegering1999] [Jander1999] [Doman2001]. Simple Network Management Protocol is a de facto standard protocol for internetworking device management and is provided by the relevant Management Platforms. However, it has not widely been deployed for System and Application Management that is typically performed using proprietary (i.e. vendor specific) approaches.

Users of Management Platforms are quite often experiencing a gap between what has been promised by platform vendors and what has been provided [Jander1999] [Doman2001]. It may be assumed that Management Platforms easily customised to meet the requirement of a particular environment and will easy to install, maintain and operate. However, Management Platforms such as Tivoli, HP Open View or Spectrum usually require extensive customisation to implement environmental characteristics and to integrate 3rd Party Applications via proprietary Management Platform APIs [Jander1999] [Doman2001].

3.4 Network and System Management Frameworks

Several network and system management frameworks and standards have been defined by the various standardisation bodies (e.g. ITU-T) or by industry-driven consortiums, such as the ATM Forum or the TeleManagement Forum.

Early Management Architectures such as the Distributed Management Environment (DME) from the Open Software Foundation (OSF) define an extensive range of Management Services [Chappell1992]. However, DME lacks acceptance in research and industry implementations. Major management architectures and protocols in the area of network and system management are Internet Management based on the Simple Network Management Protocol (SNMP), mainly driven by the Internet Engineering Task Force (IETF) and TMN (Telecommunication Management Network), mainly driven by the ITU-T.

This section presents a brief overview and analysis of relevant management frameworks and networking concepts. Intelligent Network (IN), TMN and Telecommunication Information Networking Architecture (TINA) are frameworks that have originated from the ITU and the Service Provider (SP) community and reflect their requirements. On the other hand, the Internet Management Model (comprising of SNMP and other related standards) originates and has been developed by the IETF and is in widespread use in End-User networks, particularly in the LAN area.

TMN provides a generally accepted framework for unified management of telecommunication services and their underlying networks, by means of generic Management Services and related management interfaces [ITU1992]. TMN concepts may be applied for managing INs and their services (i.e. INs may represent a part of the whole telecommunications system which are managed by TMN-based systems). Several standard organisations, such as TINA [TINA-SA1997] or OMG [OMG1997], made contributions towards the integration of IN and TMN, TMN and Common Object Request Broker Architecture (CORBA) [OMG1995].

3.4.1 Internet Management

The IETF provides a set of network and system management standards. The most widely accepted contributions of the IETF are the SNMP related specifications that describe the asynchronous requests and responses for the exchange of management data between SNMP management objects [King2000]. SNMP was originally developed by the IETF to manage TCP/IP networks and network elements such as routers and hubs. The 'basic' SNMPv1 (i.e. SNMP version 1) is now in widespread use and the de-facto industry standard. Virtually all major vendors of end-systems, workstations and network devices such as router and switches offer SNMP support. The widespread acceptance of the SNMP framework has resulted in adoptions such as the use of SNMP over TCP/IP and other non TCP/IP protocol suites (e.g. Novell IPX) In addition, enhancements to the initial SNMP have been pursued in a number of directions (RMON, SNMPv2, SNMPv3) [Hegering1999] [King2000]. However, SNMPv1 is still the most widely used Management Protocol.

The SNMP entities are a SNMP Manager, a SNMP Agent and a Management Information Base (MIB). The SNMP Manager is typically network management software that based on the SNMP protocol. The SNMP Agent resides in a managed network element, such as a router or switch or in an end-system such as a PC, Server or Unix Workstation. The agent stores management information and processes SNMP requests from the SNMP Manager. The MIB is the database for management information that is stored in Abstract Syntax Notation Version 1 (ASN.1) format [ISO1987]. SNMPv1 provides a rather limited set of operations is a insecure protocol as the community strings (that controls access to the MIB) use clear text passwords [King2000]. The SNMPv2 standardisation process looked at a number of different SNMP message types with differing security features. However, the finally agreed version of SNMPv2 has not been able to make an impact on the market [Hegering1999]. Recent developments in the IETF are looking at SNMPv3 and the aspects of extended management capabilities [Katz1999] [King2000]. However, these developments currently lack industry acceptance and need to be further enhanced and specified.

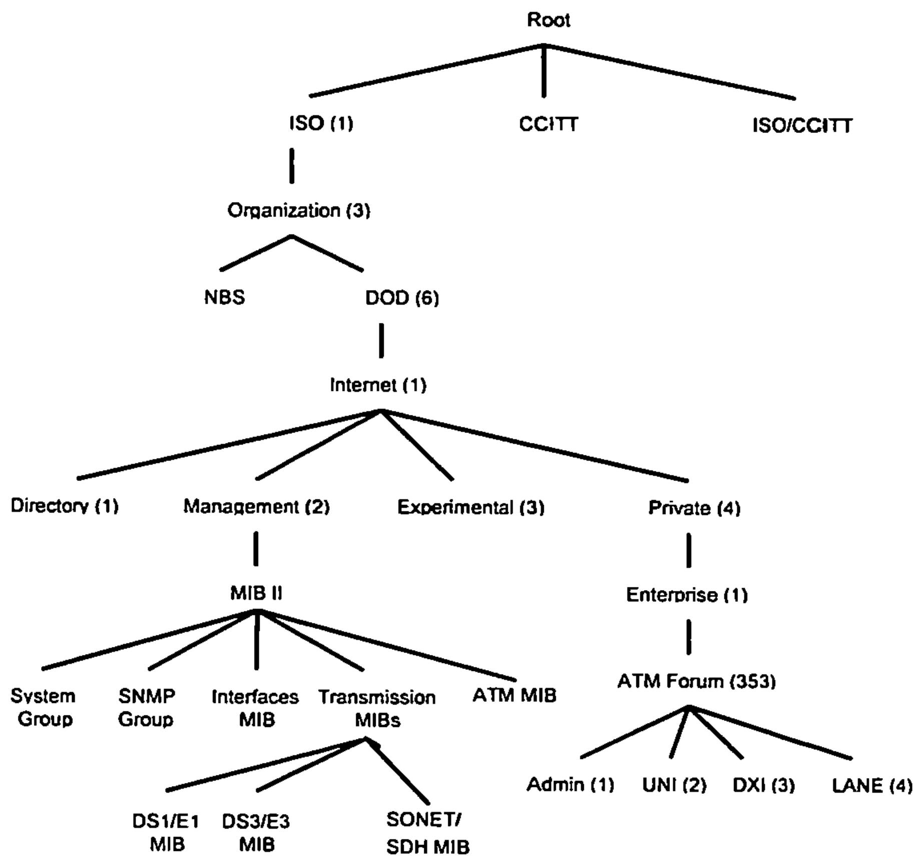


Figure 3.6: MIB Object Tree containing IETF and ATM-Forum Specifications

Figure 3.6 illustrates the MIB Object Tree containing IETF and ATM-Forum Specifications. The MIB Object Tree, also referred to as Object Identifier Tree, permits any object to be assigned a unique global identifier. The overall name space is organised like a tree-structure in which each tree node is assigned a name and number allowing the delegation of naming authority (e.g. allowing separate generic IETF and private vendor areas) [Hegring1999].

SNMP is a polling-based protocol. The manager sends a request for information to the agent periodically and the agent responds to that request. Beside that, an agent can send a trap (or alarm) to the manager to mark an important event, such as a critical network error. Even though SNMP is popular for monitoring, most network operators are not using it for device configuration [Feldman2001]. Reasons for this include lack of security, difficulty in scripting and debugging, and the desire to use a common configuration method regardless of the communication mechanism (e.g. console port or network access.)

There are two basic approaches to coexistence in a multi-lingual network (i.e. several SNMP versions exist): multi-lingual implementations and proxy implementations [Levi1999]. Multi-lingual implementations allow elements in a network to communicate with each other using an SNMP version which both elements support. This allows a multi-lingual implementation to communicate with any single-lingual implementation, regardless of the SNMP version supported by the single-lingual implementation.

Proxy implementations provide a mechanism for translating between SNMP messages using a third party network element. This allows network elements that support only a single, but different, SNMP version to communicate with each other. Proxy implementations are also useful for securing communications over an insecure link between two locally secure networks. A proxy implementation may also be used to enable communication between non-SNMP and SNMP entities, where the Proxy-implementation acts as a translator between the different environments. This is accomplished in a proxy forwarder application. The proxy forwarder performs translations on a Protocol Data Unit (PDU) in the various situations as required (e.g. if a SNMPv2 *GetBulkRequest*-PDU is received and must be forwarded using the SNMPv1 message version, the proxy forwarder would have to set the non-repeaters and the max-repetitions fields to 0 and would have to set the tag of the PDU to *GetNextRequest*-PDU). If no adoption is required, the proxy forwards a received SNMP-PDU without change. The major drawback with Proxy implementations is the added overhead regarding the processing (i.e. translation and transmission) of messages. Furthermore, the Proxy-System can become a single point of failure and a potential performance bottleneck.

The multi-lingual approach requires an entity to support multiple SNMP message versions. Typically this means supporting SNMPv1, SNMPv2c, and SNMPv3 message versions. This approach allows entities that support multiple SNMP message versions to coexist with and communicate with entities that support only a single SNMP message version. A command generator must select an appropriate message version when sending requests to another entity. One way to achieve this is to consult a local database to select the appropriate message version. In addition, a command generator could 'downgrade' sophisticated management operations such as SNMPv3 *GetBulk* requests to SNMPv1 *GetNext* requests when SNMPv1 is selected as the message version for an outgoing request. This can be done by simply changing the operation type to *GetNext*, ignoring any non-repeaters and maximum repetition values, and setting error-status and error-index to zero. A command responder must be able to deal with MIB instrumentation that is written using the different SNMP versions. SNMPv1 error codes can be used without any change when processing SNMPv2c or SNMPv3 messages.

3.4.2 The OSI Management Model

ITU-T Recommendation X.700 [ITU1989] and X.701 [ITU1992] recognised that management of a communications environment is an information processing application. Because the environment being managed is distributed, the individual components of the management activities are themselves distributed. Management applications perform the management activities in a distributed manner, by establishing associations between systems management application entities [Hegering1999] [King2000].

The interactions that take place between system management application entities are abstracted in terms of management operations and notifications issued by one entity to the other; these are communicated using systems Management Services and protocols. Management activities are effected through the manipulation of managed objects.

3.4.2.1 CMIS and CMIP

The interactions between the role of manager and agent respectively are realised through the exchange of management information. This communication is accomplished using OSI protocols. The generic OSI communication service for systems management is the Common Management

Information Service (CMIS) [ITU1991]. The systems Management Services have primitives for the communication of requests for the various types of management operations and primitives for the transfer of notification information [ITU1992b]. In this way, the systems Management Services mirror the exchange defined at the managed object boundary. The systems Management Services provide additional support for the selection of appropriate managed objects (e.g. through filtering of management related information).

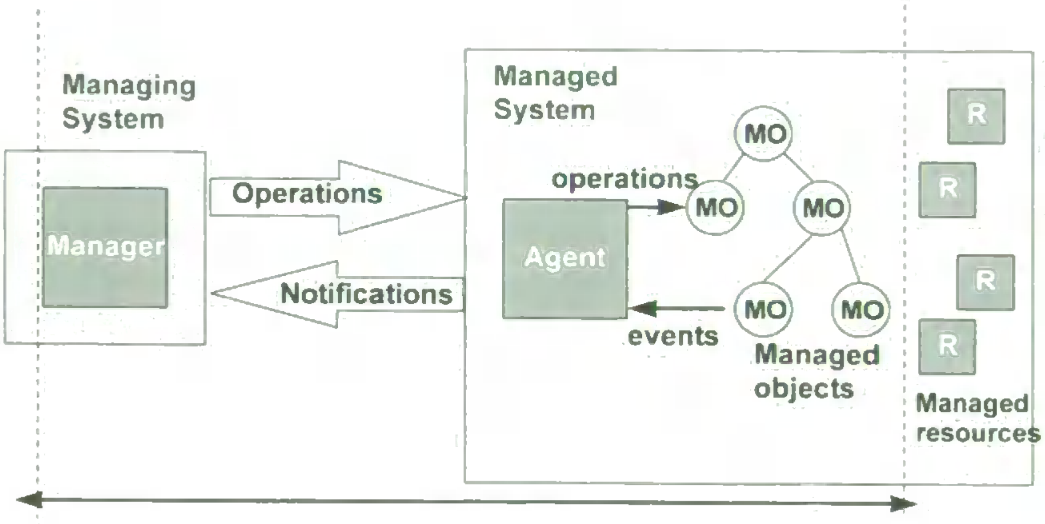


Figure 3.7: The OSI Management Model

Common Management Information Protocol (CMIP) is the protocol that provides the Common Management Information Service [ITU-CMIP1991]. It specifies procedures for the transmission of management information between application entities, the abstract syntax of the CMIP, procedures for the correct interpretation of protocol control information, and the conformance requirements for implementations [King2000].

3.4.2.2 Structure of Management Information (SMI)

An Object Oriented approach has been applied to the rather complex information model that is called Structure of Management Information (SMI) [Hegering1999]. The mechanisms and standards relating to management information fall into two groups: definitions of managed object classes covering their attributes, the management operations that can be performed on them, the notifications they may emit and the appropriate naming schemes so that managed objects and attributes can be identified [ITU1991], and generic concepts which support the definition of managed object classes and templates that can be imported into a variety of managed object class definitions, to support the consistent definition of attributes, notifications, and management operations including their parameters [ITU1992c] [ITU1992d].

These concepts provide a comprehensive set of guidelines for the provisioning of Management Services. However, there is a rather limited market acceptance of the OSI standards for LANs due to the complexity of the OSI Manager-Agent model [King2000]. It has been widely accepted by the telecommunications community, and was adopted for implementing the physical architecture in TMN and also including systems in the TMN Service Management Layer. The TeleManagement Forum has developed some service management related interface agreements using CMIP, and several research projects have attempted to implement integrated service management using CMIP [PREPARE1996] [Griffin1996]. There is, however, little evidence that CMIP has had widespread use within industrial management applications to date. Research experiences reveal that the difficulties encountered were closely related to the different CMIP Application Programming Interfaces that have been made available to the developers. These ranged from libraries used in Hewlett Packard's OpenView system to high-level C++ [Pavlou1994] and TCL/TK API's used in the OSIMIS platform [Pavlou1995]. The differing natures of these API's also precluded the reuse of code, though the TMF has since proposed an open C++ CMIP API [Chatt1997].

Interoperation between managers and agents implemented using the different protocols is possible using gateways for converting between SNMP and CMIP [Dassow1997] [McCarthy1995].

However, when accessing a CMIP agent via a SNMP Manager using such a Gateway, some of the protocol features of CMIP (that provides more functionality than SNMP) are lost.

3.4.3 Intelligent Network

The Intelligent Network (IN) is based on a service-independent telecommunications network [Magedanz1997] [Venieris1998]. Intelligence is taken out of the traditional telephone switch and placed in computer nodes that are distributed throughout the network. This provides the network operator with the means to develop and control services more efficiently. The aim of this framework is to provide mechanisms that enable capabilities to be rapidly introduced into the network. Once introduced, services are easily customised to meet individual customer's needs.

The IN functional model consists of three levels:

- The management and service creation level provides the means for creation, deployment and maintenance of the functions provided in the lower levels;
- The service control level stores the service logic and data;
- The service switching level encompasses the detection of IN calls.

Much of the research in the area of management for Intelligent Networks revealed the lack of a clear functional demarcation between service management and service control, and advocates the migration to a middleware based approach [Chapman1994]. While the importance of management aspects was acknowledged [Bauer1988], IN standardisation efforts largely neglected the operational aspects of service provisioning, failing to specify standardised functional components that could be used to manage IN services [Kockelmans1995] [Venieris1998]. Separate management support for IN services is usually considered within the context of the TMN management concept.

3.4.4 Telecommunications Management Network

The standardisation and adoption of the Telecommunications Management Network (TMN) is an ongoing process in different international standardisation organisations such as the International Telecommunications Union - Telecommunication Sector (ITU-T), European Telecommunications Standards Institute (ETSI) or industry driven groups such as the TeleManagement Forum [TMF1999] or the ATM Forum [Sidor1998]. TMN addresses the management of network elements and its framework services are designed for TMN Management platforms [ITU1992]. TMN was originally conceived as a separate physical network used to manage a telecommunications network [Scherer1988]. This concept was later relaxed to a logical separation [Shrewsbury1995] due to the costs involved with maintaining a distinct network [Glitho1995].

However, there is still a long way to go until there is a framework with a complete set of standards and functionality and for TMN to be integrated into new broadband environments [Yamura1997] [Manley1997] [Pavlou1998] [Lewis2002]. Beside that, existing TMN standards (e.g. Q3 based on CMIP) need to be introduced into existing networks [King2000] [Petermueller1996] [Patel2002].

3.4.4.1 TMN Implementation and Objectives

TMN based implementations occur mainly in Wide Area Networks. Practical research by the author (i.e. via industry projects with PanDacom) found that it is typically regarded as too complex for the management of enterprise networks. Network management platforms such as HP OpenView or IBM Tivoli usually provide TCP/IP Management via SNMP but have TMN interfaces to integrate the management of TMN objects via CMIP (Common Management Information Protocol).

TMN provides a managing network framework that is designed to be, despite its complex implementation, flexible, scalable, reliable and easy to enhance [ITU1992] [Manley1997]. TMN provides defined standard tasks and ways of doing network management tasks and communicating management information across networks. TMN allows processing to be distributed to appropriate levels for scalability, optimum performance and communication efficiency.

TMN principles are incorporated into a telecommunications network to send and receive information and to manage its resources. A telecommunication network is comprised of switching systems, circuits, terminals, etc. In TMN these resources are referred to as Network Elements (NEs). TMN enables communication between Operations Support Systems (OSS) and NEs.

TMN provides management functions for telecommunication networks and services, and offers communications between itself and the telecommunication networks, services and other TMNs. In this context a telecommunication network is assumed to consist of both digital and analogue telecommunications equipment and associated support equipment. A telecommunication service in this context consists of a range of capabilities provided to customers.

The basic concept behind a TMN is to provide an organised architecture to achieve the interconnection between various types of Operations Systems (OSs) and telecommunications equipment for the exchange of management information using an agreed architecture with standardised interfaces including protocols and messages [Petermueller1996]. In defining the concept, it is recognised that many organisations have a large infrastructure of OSs, networks and telecommunications equipment already in place that must be accommodated within the architecture. The principle of keeping the TMN logically distinct from the networks and services being managed introduces the prospect of distributing the TMN functionality for centralised or decentralised management implementations. This means that from a number of management systems, operators can perform management of a wide range of distributed equipment, networks and services.

Within the general TMN architecture, there are three basic aspects of the architecture that can be considered separately when planning and designing a TMN. These are:

1. TMN functional architecture;

2. TMN information architecture;

3. TMN physical architecture.

The functional architecture describes the appropriate distribution of functionality within the TMN to allow for the creation of function blocks from which a TMN of any complexity can be implemented. The definition of function blocks and reference points between function blocks leads to the requirements for the TMN recommended interface specifications. The information architecture, based on an object-oriented approach, gives the rationale for the application of Open Systems Interconnection (OSI) systems management principles to the TMN principles. The OSI systems management principles are mapped onto the TMN principles and are expanded to fit the TMN environment where necessary [Subranian2000]. The physical architecture describes realisable interfaces and examples of physical components that make up the TMN. See [ITU1992], [Hegring1999], [Pras1999] and [Subranian2000] for more information on TMN principles and architecture.

3.4.4.2 TMN Evaluation

A main driving force behind IN was to separate the provision of subscriber services from the provision of switches (i.e. Network Elements). On the other hand, a main driving force of TMN was to enhance Operation and Management Systems from the Management of Network Elements to the provision of higher level Management Services to facilitate telecommunication deregulation [Kockelmans1995] [Davison1995]. TMN, primarily established to facilitate element management, moved to provide service management on a similar basis. This was fraught with architectural problems, mainly due to a lack of a sufficient Distributed Processing Environment (DPE).

TMN lacks the concepts of application building blocks [Kockelmann95] and service creation [Appledorn93] [Kockelsmann95] as important software re-use concepts. Therefore it is was difficult to provide the management support needed for the rapid development of services

[Kockelmann95]. TMN lacks a component architecture that would allow software built to these specifications to be reused.

Service creation was also hindered by the reluctance to standardise the functionality of management nodes as it was thought that this would constraint product offerings [Glitho95]. Instead the standardisation concentrated on the definition of standard interfaces which would allow nodes to inter-operate as long as the protocols were specified to enable interaction. The reluctance to standardise TMN functionality was partly due to the selection of the complex OSI management standards as the basis for TMN interfaces [Glitho95] and therefore on the OSI management service and the OSI management Protocol CMIP, which effectively meant that the protocol dependence was imported into the TMN.

Recognising the lack of functional specifications, a number of different bodies subsequently produced several specifications. The Network Management Forum (NMF), now known as the TeleManagement Forum (TMF), provided specifications for testing, trouble and security management amongst others [Shrewsbury95]. The NETMAN project, part of the Research and Development in Advanced Communications in Europe (RACE) initiative, produced Common Functional Specifications for a number of areas (e.g. accounting management) [Netman94]. However, the lack of TMN functional standards has made it necessary for several TMN platform vendors to supply the functionality in proprietary ways [Davison99]. Recent TMN efforts have seen a reversion to the specification of a protocol independent framework for telecommunications management [Lewis2000] [Pavlou1998] [TMF-OM2000]. This indicates that the development of generic Management Services must be independent from a particular set of protocol and management interfaces, and should support the relevant IT technologies rather than using a telecommunications specific approach. However, the logical layer concept was found to be a useful principle for separation of management scope and responsibility and has been considered the most important concept of TMN [Pras1999].

3.4.5 TINA

The Telecommunication Information Networking Architecture - Consortium (TINA-C), which is no longer active, was founded in 1993 against the backdrop of what it thought were expensive and unnecessary separations between service operations, management and control and largely unsuccessful attempts at service creation by the TMN community [Appledorn1993]. The TINA Consortium defined an architecture for *"distributed telecommunications software applications"* [Berndt1999], divided into a DPE based computing architecture that supported a service architecture and a network architecture. TINA further adds to TMN a Distributed Processing Environment (DPE) [King2000].

3.4.5.1 TINA Implementation and Objectives

Telecommunication Information Networking Architecture (TINA) represents an integrated architecture for telecommunication networks and services. The TINA Consortium consisted of service providers, network operators and telecommunication vendors exploring the future evolution of telecommunication environments. The ultimate goal of TINA-C was the development of an environment comprising open distributed processing platforms positioned in telecommunication systems. Much of its work is based upon the Reference Model of Open Distributed Processing (RM-ODP) [ITUX901-1995]. The main focus lies on the definition of an abstract service management framework that should be powerful enough to address all relevant network technologies. Integration of IN, TMN and CORBA in TINA was envisaged to start with the substitution of a few IN components to save the network operator's investments. This means that the Service Switching Points (SSP) already deployed on a global basis in the exchanges, representing the biggest amount of IN related investment, would most likely be the last elements to be replaced (or TINArised). The first migration step could be in the Service Management System (SMS). Between the IN and TINA parts of the network, Adaption Units must be replaced. The units should support the communication between the existing IN and TINA.

3.4.5.2 TINA Evaluation

"Understanding all the documentation imposes a steep learning curve for parties interested in implementing TINA... " [McKinney1998]

"The details needed to implement TINA do not exist and the current documentation leaves room for interpretation.. " [McKinney1998])

"TINA has still not been deployed in an actual Internet user setting" *[McKinney1998].*

The above quotes illustrate the main dilemmas and difficulties that TINA-C encountered: complexity and rather limited market-acceptance. The support for 'legacy' systems is an important aspect in any industry, but especially in the networking and telecommunications related service industries. A denial of service for upgrades or new infrastructure (e.g. new switch) is not acceptable. There are few (if any) commercially deployed telecommunications services entirely based on TINA architecture as parts have been used, other parts adapted, but still others ignored. Acceptance and use by a wide range of vendors is a necessary precondition for the success of any software architecture and in this sense, the TINA architecture cannot be regarded a success. It is believed that the consortiums initial lack of openness (it was rather difficult for those outside the consortium to get hold of relevant and comprehensive information) and lack of interoperability considerations are partly to blame for TINA's limited acceptance on the market. At least part of TINA's lack of success can be attributed to the sheer quantity and complexity of the TINA architecture and its limited real-life utilisation particularly for connectionless services [McKinney1998].

When the TINA consortium initially specified a multi-service architecture, the idea of distributed processing was providing a new and insightful methodology for service provision and service management. However, in the timeframe of the TINA Consortium activities even the adoption of CORBA did not guarantee interoperability between applications written by different software developers. It was gradually realised that telecommunications software was not such a special case and should use 'mainstream' software engineering practices and products. With this in mind, the TINA Consortium's final efforts aimed at directing telecommunication specific extensions to

CORBA rather than designing and specifying yet another Distributed Processing Environment. However, experimental results suggest that the architecture's support for software reuse is far from ideal. The European ACTS (Advanced Communications Technologies and Services) project Prospect based a number of sample services on the TINA service architecture and found that while some software and specification reuse was possible, true component based re-use could not be achieved [Wade1998]. Further ways to specify and provide telecommunications management software were subsequently explored in a follow on ACTS project FlowThru and are not in favour of TINA concepts [Lewis 1999] [Donelly1998] [FlowThru2000].

3.4.6 OSF, OMG and JIDM Approaches

The Open Software Foundation's (OSF) Distributed Computing Environment (DCE) was an early distributed computing platform that was also considered for use in Integrated Network and System Management [Gaspoz1995]. The OSF also built on DCE in developing the Distributed Management Environment (DME) which aimed to bring together network and system management in an object-oriented manner. The use of DME was largely unsuccessful [Marcus1995], and the use of DCE has now largely been replaced by contemporary distribution technologies (e.g. CORBA).

Common Object Request Broker Architecture (CORBA) is an object model for distributed systems defined by the Object Management Group (OMG). The goal of Network Management based on CORBA is to reconcile the OSI and Internet Models using CORBA. Specifically it focuses on the management of OSI and SNMP resources using CORBA and vice versa. Assuming that future Network Management applications will be written based on CORBA, it will still be necessary to access network resources that are managed with existing technologies in order to preserve large investments.

Several approaches exists for CORBA-based management:

- Model alignment by identifying the core elements of the different models to be unified, omitting model features that may be difficult to align;

- Runtime mediation between implementations of the models by means of proxy applications;
- Use of notation mapping tools based upon translation algorithms to enable OSI and TCP/IP objects to be mapped into CORBA objects and vice versa.

The Joint Inter-Domain Management (JIDM) working group specifications covering specification translations have been adopted by the Object Management Group [JIDM2000]. These standards provide concepts and define mapping to enable interoperability between management systems based on different technologies, notably OSI management, SNMP, and OMG CORBA-based management frameworks. In addition, these tools permit technology from one domain to be used in other domains. These standards define how to map between the relevant object models in each domain, and how to build on this to provide mechanisms to handle protocol and behaviour conversions of the domain boundaries. Interworking between a particular pair of management reference models requires both a translation scheme between the different object models of both management reference models (Specification Translation), and a dynamic conversion mechanism between the protocols and behaviours in both domains (Interaction Translation). This then allows objects in one domain to be represented in the other domain, and the interactions can be governed by the domain of choice rather than by the domain in which the target object is.

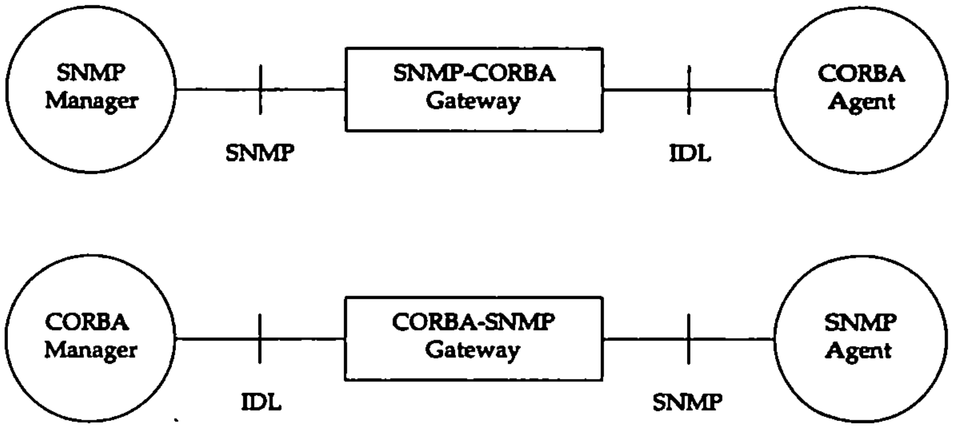


Figure 3.8: SNMP/CORBA Interoperability Scenarios

Figure 3.8 illustrates a set of scenarios between Internet management and CORBA. A CORBA Agent is an Agent which has its object definitions specified in CORBA IDL. A CORBA Manager is a Manager which has its object definitions defined in CORBA IDL. Using the scenarios described in this section, a CORBA Manager may interact with OSI, SNMP, and CORBA Agents. The CORBA/SNMP gateway makes the remote SNMP agents appear as CORBA server by extending the CORBA naming, event and interface repository object services. The gateway dynamically generates CORBA object references for the SNMP MIB table entries and converts the SNMP traps to CORBA events. The CORBA/SNMP gateway also dynamically converts method invoked on the object references for MIB entries to SNMP messages for remote agent. The application developers in CORBA domain need not know SNMP in order to access the MIB supported by the SNMP agents.

CORBA has also been advocated for telecommunications; network and system management [Chen2000] [Keller1998] [Patel2002] [Stringer1996]. As with manager-agent protocols, CORBA provides client-server interoperability between remote systems through the use of standardised protocols. In addition, CORBA supports the standardisation of APIs for performing remote procedure calls from clients to server objects. It therefore represents an open functional framework that could be applied to Integrated Network and System Management.

3.4.7 Summary and Synthesis

Management related standards and frameworks provide syntactic and semantic possibilities for the modelling of an Integrated Management System and for open architectures on a manufacturer and technology independent basis.

Internet Management based on the SNMP protocol is the de facto industry standard for Network Management Protocols and implementations in Local Area Networks. SNMPv3 is intended to be the next generation management protocol with enhanced management and security functionality but lacks industry support. Leading Network and System Management Systems and Platforms typically offer SNMP II interfaces but only a few vendors are offering SNMP II agents for their products. SNMPv3 has begun to be implemented on the manager's side, whereas only a few managed objects are equipped with SNMPv3 agents [SimpleTimes2000]. Other SNMP related standards include Remote Network Monitoring (RMON) and Remote Network Monitoring II (RMOM II), that define and specify more extensive Network Analysis, Performance and Error monitoring functions and extend the management area [Subramanian2000].

Network and System Management platforms based on SNMP provide vendor and product interoperable Management Services [Ghita1997]. Due to the fact that it was designed and primarily implemented as a management protocol for TCP/IP networks rather than other networking protocols and technologies, such as ATM, it cannot be seen as a universal management solution and does not suit all. However, the ATM Forum, the IETF and the Network Management Forum adopted the SNMP architecture and defined ATM MIBs for ATM Network Elements [Krishnan1998]. The ATM Forum specification of the Integrated Local Management Interface (ILMI) [ATMForumILMI96] specifies the communication of the ATM devices with implemented SNMP agents and the manager. The ILMI protocol is based on the SNMP definition without User Datagram Protocol (UDP) and IP addressing. A dedicated, permanently available ATM connection is used for sending encapsulated SNMP messages.

On the other hand, the range of inherent concepts in TMN go far beyond the concepts of Internet Management. One major drawback of TMN is its complexity that makes it quite expensive to implement, particularly for small and medium sized networks. Furthermore it can be argued, that the enormous costs for retrofitting the installed base of 'legacy' network equipment (non TMN management interfaces) and vendor specific management systems with non-compliant interfaces to a potential global TMN standard will prohibit its global acceptance for an unforeseeable period [Pathel2002] [Subramanian2000]. Glitho argues for what he calls Q-addition instead of Q-Adaption for a realistic introduction of more or less standard compliant network management systems [Glitho1996]. This approach is different to what the standard bodies recommend as it effectively means to add more or less compliant interfaces to proprietary management systems. Furthermore, standards that are being defined over such a long time (e.g. early TMN concepts originated in the early 1980s) do not consider recent innovations and improvements of the software industry.

The level of advocacy for the adoption of System Models such as CORBA for Integrated Network and System Management varies. TINA-C has suggested that its Distributed Processing Environment (DPE) can be implemented using CORBA, and therefore that CORBA may be used for the control and management of both networks and services. There is a widespread consensus in the Service Provider community that the investment in network element management, and to a lesser extent integrated management using CMIP and SNMP, will result in these technologies persisting in these roles [TMF1998]. This is reinforced by the features they provide for efficiently interacting with large numbers of network elements, which are not available in CORBA.

However there are strong arguments for CORBA and other distributed system technologies to be adopted for Integrated Network and System Management where SNMP does not provide the required functionality, CMIP does not have a widespread implemented base and were little or no SNMP and CMIP based legacy infrastructure has to be supported. These arguments include: easier integration with existing or future distributed systems, integration with business and workflow systems, greater likelihood of applicable applications emerging from the wider IT community, greater availability to development expertise and herewith a wider range of platforms at lower prices. The growing popularity of CORBA has prompted the investigation of CORBA to CMIP and CORBA to SNMP Gateways [Deri1997] [Keller1998], with standardised solutions now being available from the Joint Inter-Domain Management (JIDM) [JIDM2000] [Soukouti1997].

Another recent technology that has received much attention for its applicability to Integrated Network and System Management is the World Wide Web (as portable GUI and as a means of distributing applications and services). The relatively low cost of bandwidth and high acceptance of Internet services (e.g. WWW) make them an attractive option for management applications. Several methodologies and prototypes have been proposed for using WWW browsers to access management data [Hosoon2000] [Barillaud1997] and to incorporate WWW servers into Managed-

Objects [Choi2000]. However, industry acceptance of these concepts and corresponding models have been limited (e.g. rather limited acceptance of the Hyper Media Management Protocol that has been specified by the Web Based Enterprise Initiative). Work done by the discontinued Web Based Enterprise Initiative has been taken over by the Desktop Management Forum with the result that some major changes had been instigated. Work on HMMP was suspended and activities to map Common Information Model (CIM) to the Extensible Markup Language (XML) were instigated [Hegering1999]. CIM has evolved from the Hyper-Media Management Schema (HMMS) offering a non-proprietary meta-schema to describe management information [DMTF2000]. CIM aims to provide a common understanding of management data to simplify the integration of management information from varied sources into a common repository. The CIM is expressed in a technology neutral language called Managed Object Format (MOF) and can be mapped to XML [CIMXML1999]. Another XML related standardisation effort is currently going on in the ITU-T with the aim of publishing a series of network management recommendations based on the XML standard which will be based on a common telecommunications Markup Language (tML) [Mouritzen2001].

To summarise, it is highly unlikely that a single management framework and integration approach will provide a universal solution for the provisioning of an Integrated Network and System Management system. However, gateways between different appropriate technologies are feasible and are being integrated into management systems. Therefore it can also be concluded, that a development framework for Management Services must encompass a number of different technological platforms and the range of platforms that must be accommodated must be reflected in the development framework.

3.5 Provision of Integrated Management Services

In the previous sections the current state of the art in the area of Network and System Management standardisation and technology has been introduced. In this section, relevant industry and research initiatives that have further contributed to the research area will be presented. Furthermore, relevant development approaches and methodologies will be assessed.

3.5.1 Related Research

An interesting and influential early approach to distributed management is described by Goldszmidt [Goldszmidt1993]. The proposed model, called *'Management by Delegation'*, is based on elastic servers, whose functionality can be extended at execution time by delegating new functional procedures to them. A delegation protocol allows a manager to transfer programs, create process instances and control their execution. A health function provides efficient compression by combining managed data linearly to a single index of network state. Goldszmidt's main contribution was to establish the notion and concept of Distributed Management to achieve integrated management.

IDSM and SysMan [Sloman1995] were two early management ESPRIT funded projects that focused upon the management of distributed systems. They propose a domain system and the use of distributed Management Systems to implement Management Services based on the TMN architecture. It recognised the requirement for management to be distributed and looked at ways of achieving TMN based management. However, the focus of these projects was rather limited and concentrated on WAN service provisioning.

On the other hand, middleware and Software Component monitoring provides data about the run-time state of a set of Software Components [Hauck2001]. CORBA Assistant [Fraunhofer1996] is an early example framework used to monitor CORBA applications. Data such as server host utilisation and memory usage of Software Components can be monitored. This approach enables monitoring of distributed CORBA applications but does not integrate native network and system management. Another constraint is that Software Components have to be especially designed and implemented for the use with this architecture. Schade identifies the requirements for the management of distributed CORBA applications and outlines the usage of CORBA services for distributed applications management [Schade99]. However, he also focuses purely on distributed CORBA applications and does not provide solutions for the integration of other middleware technologies and for Network and System Management.

Hauck and Radisic further isolated four basic generic methodologies for application management [Hauck2000]: monitoring of network traffic, monitoring of system-level parameters, client-application monitoring and application wide monitoring. However, monitoring of network traffic [Apptitude2000] [Brownlee1999] and monitoring of system-level parameters cannot provide the required data (e.g. network traffic is not providing an indication of end-system specific states). Client-application monitoring can deliver user-oriented information, but only shows the client-view of the application and thus fails to provide a detailed enough picture for error-analysis. Application and Software Component management is an evolving research area and different approaches have been proposed (see also [Debussmann2002] [Hauck2001] [Kreger2001] [Ressmann2001b]. However, they are usually restricted to a particular technology domain and generally considered to suffer from high efforts posed on the application developer and are not in widespread use today in industry implementations.

Keller from the Munich Network Management Team proposed an integration of SNMP based Management systems and CORBA environments based on the extension of existing SNMP based platforms for managing CORBA compliant agents [Keller1998b]. The benefit of this Management-Platform specific approach lies in the reuse of existing platform code and can be considered as a stet towards CORBA capable management by providing interoperability between heterogeneous management architectures.

The ACTS project PREPARE [PREPARE1996] implemented TMN based multi-domain management systems. PREPARE's approach was based on administrative domains which were intended to enable service provider autonomy and encourage co-operation between service providers. This differed from other approaches where domains were technology based and served only to partition management responsibility within a provider. PREPARE introduced the concept of the 'Open Services Market', recognising the growing importance of software based services in the light of the European Commission's telecommunications service and 'Open Network Provision' directives [ECDirective1997] [ECDirective1990]. PREPARE itself concentrated on bearer services, such as Virtual Private Networks (VPNs).

The PROSPECT project [Wade1998] originated from PREPARE and concerned itself with the management of integrated services. It chose a business model identified by PREPARE in which one Service Provider could integrate multiple services to offer an integrated service to its customers. Based on this business model, PROSEPCT produced federated service management systems, composed from several service providers' management systems. Each management system was composed from service entities, including service components adopted from the TINA service architecture. Although seeking to adopt a TINA based approach to service management, at the time of the project inception many of the TINA standards were unpublished or not generally available. This necessitated an approach that used published TINA architectural concepts, but could not directly make use of unpublished TINA information model definitions. In the absence of a TINA Distributed Processing environment (DPE) implementation, PROSPECT took a pragmatic view and adopted the CORBA architecture as the basis for the creation of services that did not need the specialised communication facilities (e.g. IP based services).

The FlowThru project [Donnelly1998] [Lewis1999] arose from the combined interests of partners participating in several related projects, including PROSPECT. FlowThru aimed to reuse components from the earlier projects to show how different integration methodologies and technologies could be applied to create reusable telecommunication management components. FlowThru used the TeleManagement Forum's Operation Maps [TMF-OM2000] [TMF-

NMDOM1999] to identify business requirements for systems to be built using the different integration technologies and methodologies it wanted to validate. The FlowThru technical approach is heavily influenced by the industrial requirements captured in the Network Management Forum and provided interesting concrete implementation examples of management information flows within such a model. The architecture suffers from a number of limitations, but the most immediately apparent one is the lack of appreciable market success for the underlying TINA service architecture which this research is based on and for which services it would hope to account. The FORM project is a direct follow-up project from FlowThru (i.e. mainly the same partner institutions and a similar core team) drawing upon previous results and further enhancing FlowThru and TM Forum concepts [Wade2002].

3.5.2 Related Development Methodologies

Integrated Network and System Management is an area that is currently not well addressed by standards and design guidelines [Lewis2000] [Schmale1999]. The IETF SNMP family of standards focuses largely on TCP/IP related management provision, the TMN family of standards focusing largely on telecommunications network and network element problems. Two industrial bodies that have addressed integrated management development in some detail are the TeleManagement Forum (TMForum) and the Telecommunications Information Networking Architecture Consortium (TINA-C). Both these bodies, in addition to developing some specific management interfaces, have also provided some guidance on how Management Systems could be developed within their architectural frameworks. The TM Forum has suggested an approach to open telecommunications management interface development that draws heavily on current object-oriented analysis and design techniques. This is done with a framework of telecoms management business processes (termed by the Telecoms Operation Map or TOM) used to identify which management tasks should be analysed to develop industry interface agreements [TMF-OM2000] [TMF-NMDOM1999]. The TINA-C development approach was heavily based on the use of viewpoints specified in the ITU-Ts Open Distributed Processing (ODP) approach [ITUX901-1995]. This approach follows the concept of separation into service-specific and service-independent management sets, providing support for software-reuse when implemented in CORBA.

Both of these approaches have merits, the TM Forum one for its focus on addressing the real needs of industry through analysing business processes, and TINA-C's for its support of reuse. Ideally, a novel management system needs to combine the best of these approaches, to integrate management into the business processes based on a system design that can be built from reusable components. In particular, practitioners need to create operational management solutions for components and services that may be drawn from multiple origins.

To provide guidelines for the provisioning of telecommunication services, the TeleManagement Forum has further produced a set of requirements for telecommunications Building Blocks [TMF-TIM2000] The TM Forum specifications, which are derived from Telcordia's Operations Systems Computing Architecture (OSCA) / Information Networking Architecture (INA) framework [Telcordia1992b], places a set of strict design guidelines on Building Blocks (e.g. the requirement that a Building Block needs to exist in either the Human Interaction, Process Automation or the Enterprise Information Computing tiers) that potentially restrict the deployment of practical Building Blocks. The TM Forum also specifies the Building Block to be an atomic unit of deployment, management, distribution, security and interoperability [TMF-TIM2000].

3.5.3 Limitations

Although a trend towards abstraction and unification of different aspects regarding the provision of Management Services can be recognised [Schmale1999] [TMF-OM2000] [Wade2002], there are still a severe number of open aspects:

- Applicability to existing and upcoming real networks;
- Completeness in terms of functionality;
- Issues addressing abstraction;

- Issues addressing flexibility;
- Unrecognised realisation issues.

Unfortunately, research activities outside the standards bodies seem to work only on specific issues on the Network and System Management domain, and while a few activities can be found for some integration aspects none are addressing all relevant aspects of integration. As long as this situation remains unchanged, Network and System Management projects can only try to prepare the later unification by definition of architectures and models as flexible as possible and functionally complete in order to allow later adaptation to other systems. The following arguments highlight the problems within the area of Integrated Network and System Management development, research and standardisation:

- The development of research and official standards that will meet all the required demands will not coincide with the flexibility of already existing networks and their evolution over the foreseeable future.
- Frameworks defined over a long time period (e.g. TMN since the early 1980s) do not necessarily consider recent innovations and improvements of the software technology.
- Official standards in the area of Integrated Network and System Management are definitely coming too late. At the time when they become effective, working proprietary systems are already in place and due to the large amounts invested in installed base of hardware and software only low cost additions of standard compliant interfaces will be accepted.
- Research, standards and solutions are partly too specified and specialised to certain aspects and do not take advantage of a sufficiently high abstraction in order to address similar problems with a common solution (e.g. the technology dependency of current network management draft standards).

- There is no official committee defining a binding standard, but a number of concurrent activities are in progress without a clear distinction of working areas and rules to embody each other's results.

3.5.4 Synthesis

Based on the deficiencies of today's existing standards and solutions, it should be envisaged to tackle the openness of an Integrated Network and Systems Management approach in a different way. An INSM system must be robust against changes and flexible enough to react on changing network features and product specific restrictions. It cannot be assumed to reach the ability of supporting multi-vendor specific systems with only one predefined management approach onto them. This core of an INSM system should use stable standardised concepts where available and accepted in the IT industry (i.e. a de facto standard may be given preference to an official standard if it attracts wider acceptance). Proprietary concepts are useful if a common standard is not available or insufficient for the actual requirements. In the second case, the proprietary approach should be flexible enough to allow its mapping onto the standard concept if required.

Furthermore, fully Integrated Management Systems will not be possible if their basic modelling is over specified or depends on technology or product specifics in any way. Thus, the minimum commonality of all addressed networks and systems must be extracted and combined to a basic model with only few mandatory, but quite a number of optional, components. This model must not only be flexible enough to cover today's products but should leave enough room for vendor specific developments (mainly driven by cost saving aspects) and probable future developments.

Finally, the core INSM should have an open interface structure that can be extended with additional interfaces in order to interact with other systems. These interfaces might be proprietary, standard based or completely in compliance with an official standard.

3.6 Conclusions

As the requirements for networks and services keep changing at a rapid pace, so does the demand for management solutions and for the technologies to provide them. As a result, a continuous shift of focus for management research can be witnessed. Over the last several years, topics of primary interest have shifted from element management to Integrated Network and System Management, from management of single services to integrated management of multi-service environments, from managing relatively static systems to the management of highly adaptable environments.

This chapter illustrated the challenges for INSM, outlined the existing standards and technologies and provided a comprehensive view on Integrated Network and System Management. The next chapter will consider a number of different approaches for the provisioning of Integrated Management Services and derive a set of requirements to be fulfilled by a comprehensive Management System.

Chapter 4

Principles for Optimised Management Services

The previous chapters have identified the state of the art in networking and management technologies and its implications upon new management requirements. It has been identified that existing products, standards and technologies do not meet all current and future challenges. Therefore, the need for a new management framework and a number of different approaches will be outlined. This part of the thesis will highlight the requirements for Integrated Management Services and provide a catalogue of requirements to derive a new management framework.

4.1 Management Challenges: The Case of ATM

Every networking technology imposes particular requirements regarding its management. ATM can be regarded as an example technology as it imposes a set of specific and a set of generic management challenges.

Research into ATM deployment and management and lessons learnt in industry projects where ATM networks have been deployed showed that ATM requires comprehensive and sophisticated Management Services to enable its successful deployment [Cekro1997] [Knahl1998]. Furthermore, research showed that, in most cases, ATM will mainly be integrated into existing networking environments and coexist with other networking technologies rather than forming pure ATM networks that do not require internetworking with other technologies or protocols such as TCP/IP [Korostoff1998].

A number of key challenges can be identified that have to be considered and solved in a new and enhanced management solution [Knahl1998] [Wiltfang1999]:

- Management Services need to be integrated into an overall Network and System Management solution. This includes the management of the new application infrastructure such as video-servers and clients, as well as the management of the existing infrastructure.
- Integrated Network and System Management Architectures have to integrate ATM-specific services and have to allow an overall end-to-end view of the network.
- ATM Management Services have to enable End-To-End Management between communicating systems.
- Signalling and addressing between ATM and other networking protocols needs to allow easy internetworking. ATM will coexist with other protocols such as IP or Novell IPX.
- Signalling between ATM and other access technologies needs to allow easy internetworking. ATM will coexist with other access technologies such as Ethernet or Token Ring.
- ATM management will coexist with other Management Services and have to be integrated into existing environments.
- Traffic Management Services need to set-up, monitor and modify ATM service classes and enable specified QoS parameters. Multimedia traffic patterns are still unknown to a large extent but will affect Network Design and influence traffic management.
- Management for multimedia networks needs to exploit knowledge about typical usage patterns. The different applications and trends have to be monitored.

4.2 Aspects of Integrated Management Services

Research suggests that management standards such as SNMP and or existing management platforms like Tivoli or HP OpenView basically cannot handle distributed, end-to-end management of heterogeneous networks and their integrated services [Cekro1997] [Dornan2001] [Jander1999] [Hegering1999].

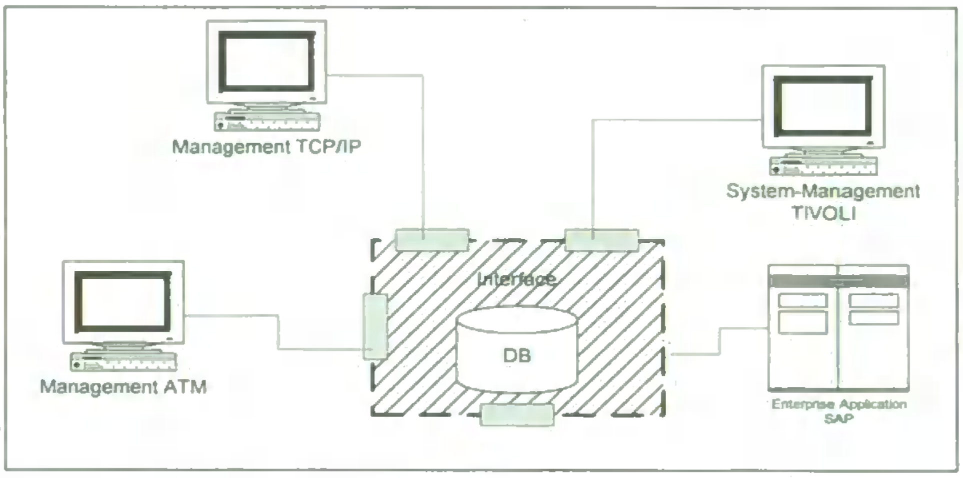


Figure 4.1: Integration of different Management Systems

The challenge for network operators is how to migrate from these restricted systems to an architecture that integrates the different management models (e.g. the ATM management model and the TCP/IP management model) and that will help them to meet the goal of providing end-to-end connectivity and management of users across LANs and WANs (see) [Pathak1994] [Gheti1997]. It has been found that this architecture presents problems and raises challenges in just about every area of network and system management [Cekro1997] [Jander1999].

Configuration management has to support the heterogeneous networking equipment. A mechanism should be available that allows the sharing of configuration management data between different network management applications. The network Management System must allow the set-up of end-to-end connections in response to user service requests and according to Quality of Service (QoS) parameters such as throughput and delay. The configuration services should incorporate a single view for easy browsing and changing of the various configuration parameters through a Graphical User Interface (GUI) for the different networking components.

The network Management System has to provide technology internetworking (e.g. tools to define routing tables to support the connections and to ensure that these tables in different switches and routers are synchronised). Existing tools and services do not provide network configuration that guarantees service provisioning involving heterogeneous networking technologies and heterogeneous networking equipment. Guaranteeing the desired service level for the users for the different service classes (e.g. real-time and non real-time services) and the various applications involves configuring optimal connection parameters on each involved internetworking device in the network. This in turn requires setting traffic priorities, choosing routes, and taking into account LAN as well as WAN connections availability and other factors that describe the current network state.

Today's High-Speed technologies such as Gigabit Ethernet or ATM and the requirement to handle different data types are adding another dimension to fault management, network monitoring and performance management [Wiltfang1999]. For a network manager to keep information on what is happening, the Management System must enable the collection of a wide variety of statistics, relating to class of service, fault, performance, and usage. Furthermore, carriers and end-user organisations require fault Management Services that provide network monitoring, fault detection and correlation, diagnostics, multiple views of the network to help administrators analyse and resolve fault conditions.

The integration of different technologies (e.g. the integration of a technology such as ATM into an existing Ethernet network infrastructure) complicates the visualisation of the network and the fault management procedures [Cekro1997]. Because ATM is more complex than conventional Time

Division Multiplexing (TDM) networks or LANs, network managers must have both a physical and logical (virtual) view of the network service. Complex network structures require dynamic visualisation tools instead of the static topology views of the physical network that are a feature of today's management platforms. Operators should be able to obtain different views focused on network problems and the systems should guide the operators in diagnosing and resolving a problem.

Another challenge lies within the Management of heterogeneous and diverse applications (i.e. applications vary greatly in terms of size, architecture, purpose and criticality and regarding architectural principles such as distributed or centralised applications [Hauck2001]. The introduction of distributed applications and the next generation of Web-based applications further complicates the Management of services based on these applications. A number of Application Management concepts are currently investigated and propagated [Debussmann2002] [Hauck2001] [Ressmann2001b] [Kreger2001]. The new end-to-end architecture of services and applications further challenges the requirements of cost allocation and billing [Redmond2000]. The complexity of these procedures in a heterogeneous network incorporating a number of services and LAN as well as WAN networks makes a powerful, reliable cost allocation and billing solution essential but complex. The billing system must give carriers and corporate administrators a way to identify the appropriate class of service for each connection and then measure usage. Usage-based billing may drive the cost of WAN services down (e.g. for ATM services) and may make the service affordable for end-users, for whom the opportunity to buy capacity according to actual usage rather than at flat rates. Furthermore, organisations with private networks need to determine how much traffic comes from one department and how much from another.

Complex networks require design tools to incorporate simulation and modelling exercises based upon real-world data capable of network modelling [Wiltfang1999]. These tools could run "what if" scenarios based on real-world data such as line cost, capacity availability and capacity utilisation, allowing customers to prototype new services, analyse how growth will affect the network and to control network planning.

4.3 Integration Approaches

Integration is one way of deploying multiple technologies to solve complex business problems [Chen2000]. Integration itself is not a goal. It is rather a vehicle to achieve business goals. In the area of Integrated Network and System Management, technology integration is required to achieve interoperability between different management domains [Chen2000] [Hegering1999] [Knahl1999] [Lewis2002]. A large network management environment may use different management technologies such as CMIP and SNMP in different subsystems. Integration is thus required to support the communication between these subsystems. Integration also helps to strengthen the existing solutions by introducing new technologies that offer better features in certain functional areas. For example, distributed technology can be introduced to provide the Management System with the means to distribute Management Services. This introduction requires the integration of the distributed technology with the existing management solutions.

It is also required to provide a domain and protocol independent management service development and deployment environment [Lewis2000]. It is common that different network equipment and network technologies from different vendors are used. They are normally managed by different systems using different management technologies. The problem becomes more acute when migration of these technologies takes place. In many cases, the migration forces the change in the management applications – a process that should be avoided. To make the Management Services independent from the underlying technologies, an integration layer is required which encapsulates all the underlying technologies and provides a domain independent environment for the applications.

Today's Network and System Management implementations based on management platforms enable management of systems and the networking elements, such as routers and switches, through the integration of additional Management Applications that implement different services and management disciplines (see Figure 4.2). Typically, management platforms communicate with the managed objects and provide generic core services such as discovery, mapping, event management and data storage. Management Applications on top of these platforms then provide specific Management Services [Hegering1999].

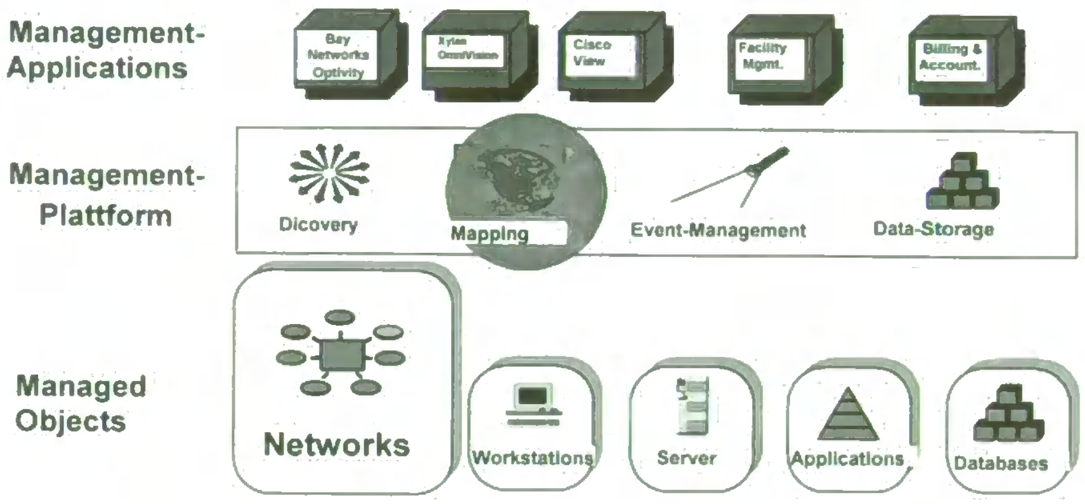


Figure 4.2: INSM Architecture

Management platforms furthermore provide access to the generic platform services via APIs. These APIs are used by 3rd party application developers to integrate Management Applications into the platform. These APIs could also be used to provide access between platforms for the provision of fully integrated Management Systems. Management Platforms are widely deployed and typically used in conjunction with 3rd Party Applications such as Cisco Works or Nortel Optivity [Jander1999] [Dornan2001].

The major network management standards as outlined in Chapter 3 include Internet Management utilising the SNMP protocol that is predominantly used in LAN environments and TMN based management in conjunction with the CMIP protocol. These standards and protocols and additional proprietary implementations that enable the management of the elements in the network have to be integrated in the system via open interfaces [Gheti1997]. Furthermore, there is the requirement to integrate future management protocols.

The most relevant network Management Systems are based on an architecture with a flexible database system to ensure data integrity and to access, manipulate and display data from various management applications and different technologies (e.g. for TCP/IP and ATM network elements). Hence, one strategy for integration is, to integrate the different databases that exist in a typical environment (e.g, environment outlined in Figure 4.3). This is also essential for presenting a consistent view of the overall network infrastructure and network elements [Cekro1997]. Furthermore, an integration of Management and non-Management Systems should be envisaged for the provision of fully Integrated Management Services.

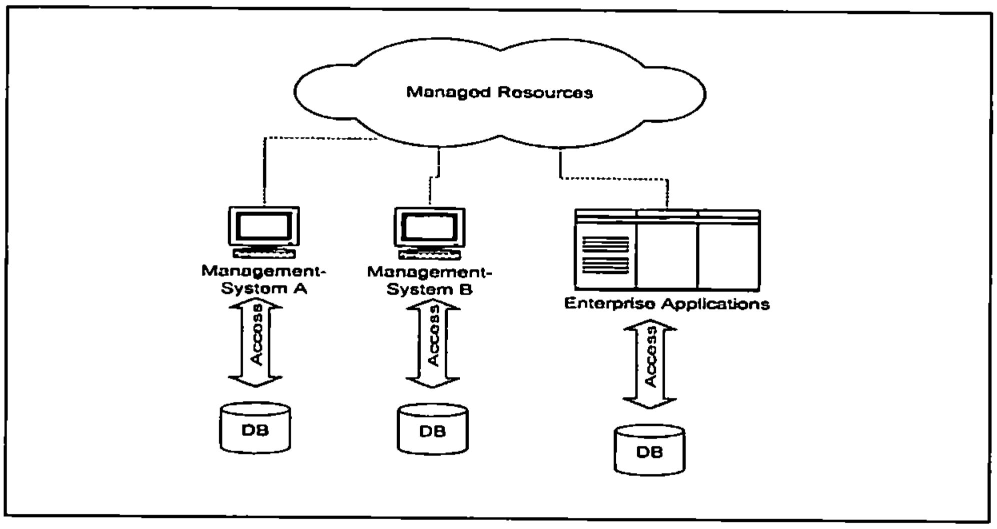


Figure 4.3: Management and Data Distribution

Standard protocols and standard interfaces from the network Management System will help ensure that the system can continually support changes and enhancements to the underlying hardware. The integration between the network protocol technology, the common distributed application technology and the user access technology is becoming the focal point in the management technology integration [Keller1998] [Knahl1998].

The first considered approach to provide interoperability between different Management Architectures and integration of Management Services are Multi-Architectural Agents (i.e. the Managed Object supports more than one management protocol). This is usually unfeasible as Agents in Managed Objects should not consume a large amount or resources and are often built into the firmware of the device. Hence, these Managed Objects (e.g. an SNMP agent of a hub) can neither be enhanced to support another management protocol nor should they introduce additional complexity.

Figure 4.4 outlines an the integration approach of two management platforms, Cabletron Spectrum (now Aprisma Spectrum) and HP OpenView. A research prototype showed, that industry standard platforms can be integrated on the database level using their proprietary data storage services [Bleimann1996]. Management data has been retrieved using the provided APIs from Spectrum's Inference Handler and the OpenView database services. However, this solution is far from being generic and remains limited to the specific development context [Bleimann1996]. It cannot be applied to other industrial management platforms (or even other versions of the used management platforms) without being redeveloped.

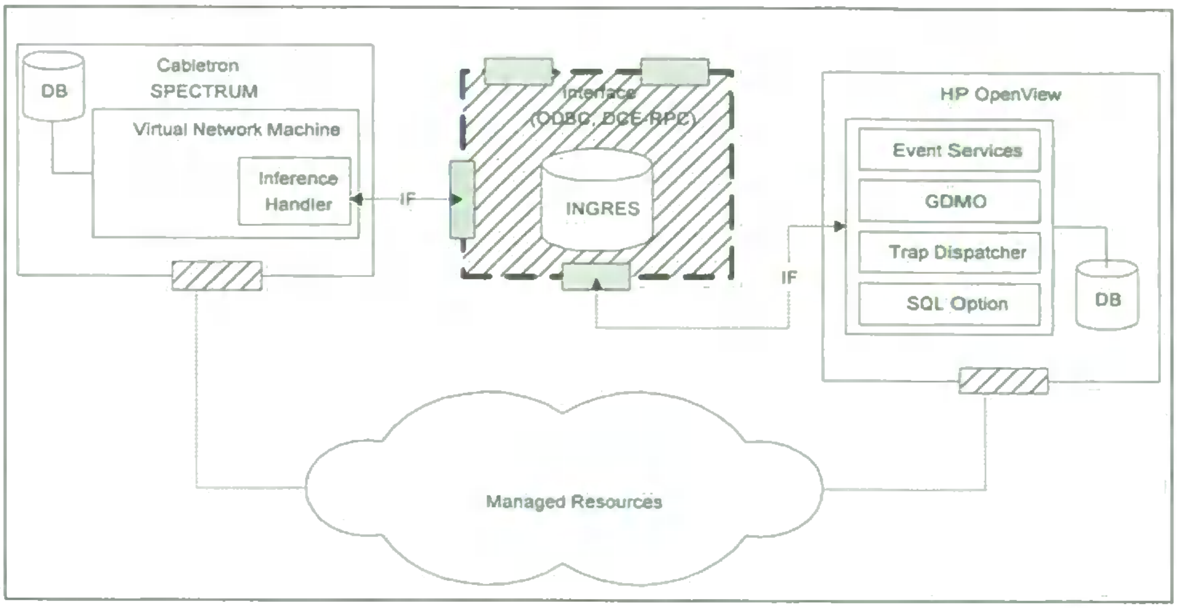


Figure 4.4: Integration with Management Platforms

Most networks consist of multiple locations and the network and system Management Systems are also distributed. Typically, enterprise networks span time zones and continents [Lewis2002] [Strauss1995]. Beside the standard requirements for integrated network and system management, different organisations have additional or even unique management requirements. One scenario could be that a central site wants to have the ability to manage a subset of Managed Objects (e.g. the central switches and routers at all sites), but gives each site the ability to manage a specified subset of Managed Objects (e.g. a particular functional or geographic domain) from that site. The Management System must have the flexibility to accommodate different access and control functionality. The integration of distributed Management Systems could be built on middleware architectures such as the Common Object Request Broker Architecture (CORBA). Distribution of management tasks has to allow the backup and recovery of all sites within the WAN. Figure 4.5 and Figure 4.6 illustrate different approaches for integrating Management Services as envisaged in the research of this project [Bleimann1996] [Knahl1998]. Another approach that has been applied by the author in industry projects is the distribution of Management Platforms. This has shown, that it is principally possible to distribute management functionality and to split Management activities and responsibilities (e.g. with respect to geographic domains) and to reduce the overall management related traffic that must be exchanged (e.g. management traffic between domains). However, careful consideration is required for such a system with respect to design and configuration. The main limitation of this approach is, that it is limited to a particular Management Platform (i.e. not suitable to integrate Management Platforms from different vendors).

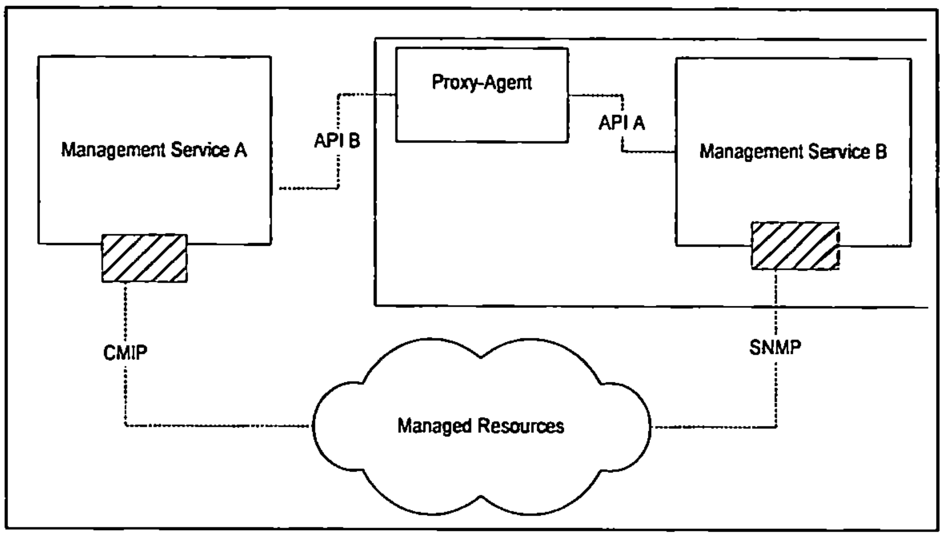


Figure 4.5: Proxy Integration of different Management Services

Figure 4.5 illustrates a model of a proxy integration of different Management Services, where the proxy 'translates' the requests of the different Management Services. The proxy is a piece of software that sits in the intersection between two different environments with different object models. The proxy behaves like a mediation device to translate object definitions and functions from one environment to another [Bleimann1996]. One way of interfacing CMIP and SNMP based management environments is to build a gateway between the CMIP and the SNMP environment were the proxy performs the protocol translation between these two different technologies.

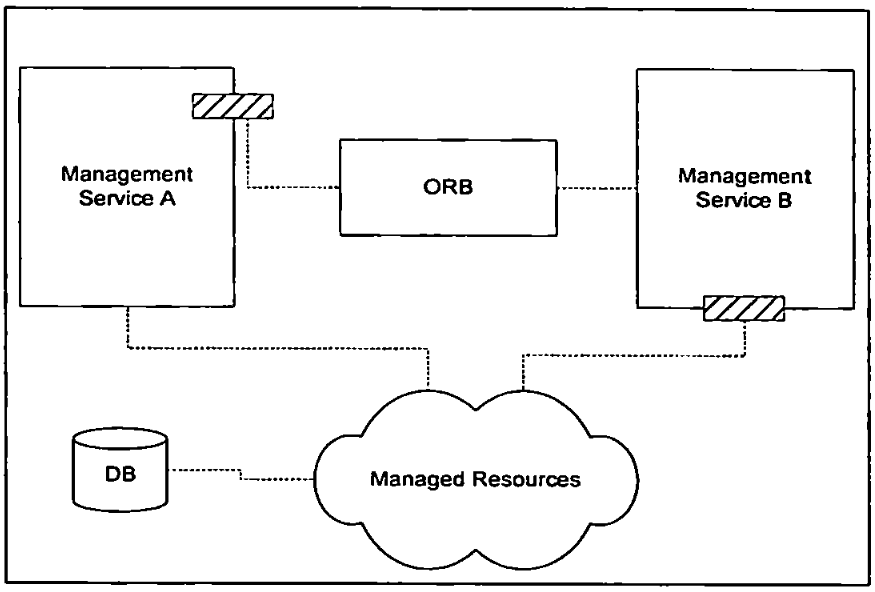


Figure 4.6: Integration via CORBA Services

Figure 4.6 illustrates a conceptual model for the integration of Management Services via middleware services, where the Object Request Broker (ORB) is acting as a broker between the various parts of the system in a Common Object Request Broker Architecture (CORBA). Using this approach, a interface based on CORBA standards could be used for the exchange of data between Management Services or parts of the Management System [Keller1999].

It can be concluded, that it is possible to achieve basic communication and a limited level of integration between different Management Systems. However, the communication and integration is limited to the proprietary interfaces developed, in this case using the Management API's of the used IBM Tivoli, Cabletron Spectrum and HP OpenView releases [Bleimann1996]. Furthermore, the adopted approaches are far from generic and need to be redeveloped for other management tools and other versions of Management API's.

4.4 Extending existing Services and Systems

Research and standardisation activities are currently underway to develop and implement standards for Integrated Management Services for heterogeneous networks as it is becoming more and more crucial to monitor and control the IT infrastructure and to enable reliable management of enterprise critical applications [Patel2002] [TMF-NMDOM1999]. As with any large undertaking of such complexity and broad scope, it is important to bring these worlds together to enable successful deployment of integrated Management Services. This research has gathered and examined INSM and networking requirements from a variety of academic research as well as from practical industry projects. Due to the complexity and broad scope of areas and functions that are involved with an INSM implementation, it is logical to begin with a model of the Management System when developing standards.

User requirements specify management functionalities that are crucial to enable the monitoring and management of an IT infrastructure [Ray1998] [TMF-NMDOM1999]. The architectural requirements specify network management functions that are essential building elements for an INSM implementation from an architectural point of view. They include the model and the components that are required by all the INSM services that the system must provide. The system has to meet user and architectural requirements to provide a future-proof management solution.

Many users require network-wide distributed management control for large networks. Such a Management System may contain many distributed network managers rather than a single centralised one. An INSM implementation needs to be capable of performing the required Management Services on any component of a multi-technology and multi-vendor network. INSM users need the flexibility to customise and adjust the INSM system to meet their own requirements. INSM users expect the system to maximise the availability and performance of the IT resources, whether those resources are communication links, processing nodes, software or data. The INSM system should be robust enough to continue network operation in spite of most common hardware and/or software failures.

The framework for INSM needs to describe the functional requirements and the essential components. The framework needs to define the organisation of the network Management System, including the relationships, interactions and interdependencies among its components. The framework describes the structure and architecture of the system so that it can provide a formal basis for developing the elements of the network Management System in a systematic manner.

4.5 Development Context and Methodology

The notion of *right technology for right problem domain* reflects the dilemma of the industry towards new IT technologies. This notion asserts in the case of the telecommunications and networking industry that it is the business problems and challenges caused by deregulation and competition that drive the progression of the management solution development process. Technologies are important but are only the means to achieve the business objectives [Mayerl2001]. This paradigm shift is essential to reflect the challenges of network operators. The management solution development needs to be oriented towards end-user solutions and hence needs to focus on end-to-end management. The aim is to develop integrated solutions and to avoid technology conflicts (i.e. practitioners need to be more concerned with integrating different existing technologies rather than picking the best technology) [Chen2000] [Patel2002].

From the analysis of concepts and methodologies applied to Integrated Network and System Management, the author concludes that the area of development frameworks and methodologies for the given problem domain has not been extensively investigated and the level of experimental rigour adopted is low. Most research consists of assertions made by their authors that were based on their own, specific experiences of using a technique. Only a few reported case studies conducted on the application of a technique in a wider context or provided subsequent lessons learnt from such projects. The author found little assessment of the application of methodological techniques to INSM development. Of the development approaches presented in the related research (see Chapter 3), most were based on paper studies that only extended as far as a system design specification. The minimum, implicit quality check that results from observing if the design led to a working implementation within reasonable costs was therefore largely absent.

Based on the qualitative material available it can be observed, that specialised development approaches seem to be driven by their use in standards bodies rather than clear results concerning their utility in industrial applications (e.g. the use of business process modelling promoted by the TeleManagement Forum [Lewis1999]). However, as these bodies represent major current sources regarding the provision of integrated management solutions, these approaches must be accommodated in addressing the needs of an INSM model. Drawing upon the qualitative material available it can be observed that INSM development techniques borrow heavily from software engineering techniques popular at the time [FlowThru2000] [Keller1998] [Mayerl2001] [Chen2000].

4.6 Criteria for a Futureproof Management Solution

The accelerating pace of development in the areas of communication and distributed systems has resulted in countless management standards and approaches. Combined with the complexity of management models (e.g. OSI or IETF management models), the interest in new object-oriented frameworks and architectures has emerged as enabling technology to meet the requirements [Lewis2000] [Patel2002]. The proposed management approach is based on a Componentware architecture to meet the requirements of an integrated Management Architecture [Knahl1999] [Knahl2000a].

Most management applications are difficult to use, configure and to extend and are very resource intensive [Jander1999] [Doman2001]. One of the reasons for this is that the developers of existing network management applications have emphasised communication aspects more than application efficiency. This is because most of these developers come from the 'protocol school' (i.e. telecommunications rather than system development background) and typically did not consider such concepts as distributed systems or Componentware.

The explosion of Internet accessible resources, the relative low cost of network equipment and the consequent proliferation of network devices have contributed to make the situation even more difficult. Users demand fast and reliable networks, networks are getting bigger and even more heterogeneous, mobile computing is replacing conventional site based IT infrastructures. Therefore, network management applications have to scale up in order to manage more and more devices and services, and hence to become more efficient and less resource intensive. Moreover, management applications have to leave their familiar environment composed of powerful servers and expert users and become applications that provide the integrated provisioning of Management Services whilst meeting the architectural and operational requirements.

The following sections provide a discussion of the generic principles and a development approach for future Management Systems is proposed. This is followed by an analysis of Architectural and operational requirements for a new Management System.

4.6.1 Generic Principles

There are several open and proprietary software architectures that are applicable to the problem of Integrated Network and System Management development for an open IT environment [Lewis2000] [Patel2002]. It is difficult to prove whether any of these possess "correct" functional principles and services. However the fact that no architecture dominates the market to the exclusion of others suggests that no one of them is superior, resulting in a non-standardised approach to management provision [Patel2002]. To meet architectural and operational requirements for the proposed Management System, the concepts of the main relevant software architectures need to be taken into account, as appropriate, for the proposal of a new solution.

The software architecture of a Management System should provide a stable basis for the system. The software architecture must address the generic principles of the provision of Management Services. As technology evolves and components are added, interchanged and upgraded, the underlying architecture remains stable. The architecture lifecycle is equivalent to the system lifecycle and transcends the lifecycles of individual component subsystems.

4.6.1.1 Stability and Cost Control

Two of the most essential characteristics of a good architecture are long-term stability and cost minimisation [Mowbray1995]. A stable design implies that the architect has successfully supported the required functionality without having exaggerated the implementation specifics. The long-term stability of an Integrated Management System is mainly influenced by the flexibility of the network model and its ability to incorporate enhancements and extensions of the managed equipment and services. Regarding the architecture of an open system, it is necessary to become independent of the underlying operating system and the embedded products used for realisation. While the former is essential to let the Management System run in heterogeneous environments, the latter is important to minimise the risk of a long term technology and product lock-in. An example is the current situation in the software middleware market. At the moment, a system developer can choose between several architectures, namely CORBA from OMG, DCOM from Microsoft and Enterprise Java Beans from Sun (see also section 6.4.2). It is unpredictable which of these technologies will dominate over the next years. Nevertheless, an open Management System will be required to cooperate with other systems using one or the other technology. Therefore, it is wise to define an architecture being independent of the middleware with an abstraction layer wrapping product and technology specific details.

Network and System Management Software is generally regarded as being expensive due to its complexity and the nature of the developed software [Jander1999] [Patel2002]. Quoting Mowbray, minimising cost is the other benefit good software architectures [Mowbray1995]. According to Horowitz, about 70% of the cost of a software system is incurred for operations and maintenance after the system is operational, while savings due to a well structured architecture typically exceed more than 50% [Horowitz1993]. The use of a software architecture minimises cost by providing clean, well-documented interfaces between system components and by isolating components from unnecessary interdependencies. Each time the system needs to be extended or maintained, the architecture is a living design that limits the effects of changes across component boundaries.

4.6.1.2 Scalability and Distribution

Scalability describes the systems immanent ability to grow or shrink with the actual demands [Hawkins1992]. In the case of a Management System this means primarily the ability to keep up with a growing IT infrastructure that has to be managed. Traditional software architectures for Management Systems have not been scalable in any way. These systems have explicitly been defined to run in a specific environment with a fixed number of computing devices and a fixed process structure. Therefore, systems being based on such an architecture cannot grow with the actual demands. One example is a network operator starting with a small network of around 100 Managed Objects in total. Over the years, the network is extended to hundreds or even thousands of Managed Objects over several locations (with their own management environments). While the Management System was initially centralised running on a single server, the distributed management must now extend to several servers over different locations to provide the required Management Services. However, this transition is not a trivial one because it must be performed incrementally at runtime due to the high availability demands. Hence, the overall architecture of the system must enable distributed deployment with the ability to relocate parts of the Management Services at runtime. The distributed nature of Management Systems is caused by the physical separation of users and management Service providers as well as by the distributed infrastructure itself in which hardware and software entities are spread throughout the network.

Management Systems that consist of a number of separate components that run in parallel and may be spread over a number of geographically separate locations face additional challenges [Chen2000] [Goldszmit1993]. Distributed systems are generally more complex than centralised systems and it is hardly possible to administrate and configure them if the overall complexity is not hidden by any means and therefore the overall architecture must be transparent to the user. Because transparency effectively shields the user of an architectural service (e.g. interprocess communication) and from the service details (e.g. selection of an optimal correlation for the handling of errors), systems applying the concept of transparency will lead to lower development and maintenance costs as well as an increased ease of use.

4.6.1.3 Availability and Versioning

Management Systems have to run continuously. Any downtime, caused by temporary faults or maintenance operations like software updates or modifications of the computing equipment must be addressed by appropriate mechanisms to ensure the required high availability of the system. This becomes obvious if one considers the consequences of a downtime, e.g. critical network state changes are not recognised and thus cannot be handled in terms of restoration and trouble shooting. Due to the high active rate within a network, a temporary outage of the Management System raises the need for synchronisation between the physical network and its representation in the managed model. Typically, an availability of 99.5 % (equals an average downtime of about 4 hours per year) or higher is asked for by customers of networking equipment, which is a hard requirement for a distributed system. Therefore, mechanisms are needed to avoid system downtimes by incremental runtime reconfiguration and a high degree of fault tolerance.

Availability is also affected through the dynamics of required system upgrades and modifications that occur over the time. While future Management Systems may have a substantial number of interaction points internally and with other Management Systems and Applications, it is essential to support incremental software updates and data migration. Examples are 3rd party systems using a specific interface of the Management System that has to be changed can be updated at the same time. Even if only an internal Management System component needs to be modified (e.g. a new element manager to support new Managed Objects), simultaneous updates of all affected client and server instances would effectively lead to severe downtimes. One solution to minimise the downtime would be to update one instance after the other. However, this leads to situations where old and the new components work in parallel. The architecture requirement for such an approach is to support multiple versions at least on the interface level in parallel (versioning of interfaces). If not only the interface but also the data model needs to be changed from one version to another, it is necessary to migrate the existing management data during this transition. Doing this gradually requires versioning of data as well, which is by far more complicated to achieve than the case previously mentioned.

4.6.1.4 Security

Sufficient support for security is essential for any distributed system offering Management Services via open interfaces in a network infrastructure. This situation is obviously given, regarding the trend of integrated communication services (e.g. via the Internet) or outsourcing of Management Services (e.g. SLAs). Both types will cause new security risks for network providers as they have to open their Management Systems for external communication with 3rd party Management Systems.

The requirement for security management is not only a matter of securely linking different systems together, but also to support operator profiles with different access rights on Management Services. Confidentiality is required to guarantee that information is not made available or disclosed to unauthorised individuals, entities or processes. Authentication and authorisation must provide mechanisms to guarantee that the source of data received is as claimed and the granting of right, which includes the granting of access based on access rights. Data Integrity is required guaranteeing that data has not been altered or destroyed in an unauthorised manner.

4.6.1.5 Simplicity

Simplicity is one of the most neglected criteria of good architectures [Koch1996]. The architecture should facilitate development, deployment and change. This can be reached better if it is based on a simple and flexible architecture. One way to achieve flexibility is abstraction, i.e. it is often possible to treat procedures that seem to be different upon first inspection in a uniform manner by abstracting common properties and behaviour. A concrete realisation might then add the necessary specifics, but the architecture remains robust to changes. Following this guideline, a robust architectural basis is also the key for reuse.

4.6.2 Management Development

Traditional management applications are built around a block that performs all necessary functions. This paradigm is reaching its limits in terms of code reusability, extensibility, configuration and especially concerning the speed and size of the development process. One reason is that the conventional development cycle specifies that the application must contain the entire functionality even if much of the functionality is only required for very particular tasks. This paradigm also requires that every extension of the initial design be integrated by an application developer familiar with the system (typically the original developer). But the user, not the developer, is often the one who best knows the requirements. The natural consequence of this would be to provide an interface that allows the user to add new functionality to the monolithic block and that defines a migration path towards compound applications.

To overcome the limitations of existing systems, a generic meta-model should be used that provides guidelines and ensures consistency of the systems generated. The core structural concept of the proposed management approach is the usage of generic Building Blocks to build Management Systems. A Building Block in the context of this thesis is a unit of deployment that provides interfaces to allow interaction in a distributed environment. In the INSMware Framework, Building Blocks can be regarded as units of deployment and management, the Building Block Interfaces as the units of interoperability and security, and the distribution and binding of Building Blocks via the Building Block Interaction Broker the units of distribution. This concept has been informed by Software Engineering principles initially advocated by Telcordia Technolgies [Telcordia1992a] [Telcordia1992b] [Telcordia1993a] and also considered by the TeleManagement Forum [TMF-TIM2000]. In the early 90s, research was concluded covering concepts of object Packaging, the design of well-formed interfaces and three tier distributed computing [Telcordia1992a] [Telcordia1992b]. The aim of the research was to specify a set of software development requirements and principles to assure the correct and robust behaviour of deployable Building Blocks in a distributed environment [Telcordia1993a].

The popularity of Object-Oriented (OO) development has increased considerably in the past few years and has been taken up by the INSM development community [Knahl1999] [Lewis2000] [Souza1999]. OO development offers many advantages over traditional development, such as allowing programmers to define objects that can be easily extended and composed in order to build a software application. Although powerful, OO development lacked a standard framework through which software objects created from different suppliers can interact with one another. The major result of this trend has been applications based on objects that cannot interact across application boundaries and the requirement to provide an interaction mechanism to extend the scope of OO has been identified [Hofmann2000] [Phippen2001]. For this reason, many frameworks and architectures have been developed to address this problem. Unfortunately applications based on such frameworks often had a monolithic structure mostly because object-oriented techniques had been misused (e.g. inheritance through introducing cross dependencies among classes). The obvious consequence had been that objects were so tightly coupled that even the simplest application had to link the entire system.

Software Components are a potential solution to these architectural problems [Hofmann2000] [Phippen2001]. As the name suggests, a Software Component is based on the notion of a component, which is a reusable piece of software that can be "plugged" with other components (e.g. a component from a 3rd party provider) with relatively little effort.

Software Components provide a different approach to the development process. Like a child does with Lego™, one can build a compound application by using several simple blocks – called Software Components – rather than a monolithic entity. Once the application has been built, it works like a monolithic one but it has the advantages of being distributable, scalable and that it can easily be modified and extended and improved by adding pieces or replacing Software Components.

4.6.3 Requirements for new Management Systems

In the course of the problem analysis, a number of requirements that a Management System should meet have been identified. To further structure the combination of the derived key aspects into a common requirement catalogue, the distilled requirements are further subdivided into two categories, namely *Architectural* and *Operational* requirements (see Table 4.1).

Archtitectural	Operational
1. System Distribution	1. System Evolution
2. System Extensibility	2. System use and development
3. Compact Architecture	3. Support of distributed environment
4. Modular Architecture	4. Efficient resource utilisation
5. Based on open standards	5. System portability and generality
6. Based on generic development methodology	6. Support of Internet standards
7. Support of existing development languages and technologies	7. Support of management standards
8. Integration with 3rd Party Building Blocks	8. System scalability and performance
9. Reuse of existing Building Blocks	9. Support of specific platform features
10. Security support	10. System administration

Table 4.1: Requirements for new Management Systems

Architectural requirements refer to overall characteristics and criteria that the Management Architecture should fulfil and are of primary concern for this research. Operational requirements refer to an actual implementation and operation of the Management Architecture. The validation of the proposed new system concerning the architectural and operational requirements can be found in Chapter 8.6.

4.6.3.1 Architectural Requirements

The following list identifies and outlines the architectural requirements.

1. System Distribution

   The usage of a distributed environment and distribution of the system (e.g. in terms of mobility and replication) is offering enhanced functionality and enables the realisation of a distributed (as opposed to monolithic) Management Architecture [Dornan2001] [Goldszmit1993] [Knahl1999]. The Management Architecture has to enable the distribution of Management Services to reflect the requirements of a given infrastructure [Lewis2002]. The extensive overhead resulting from this requirement results from the challenges associated with the distributed system technologies of the existing and future heterogeneous infrastructures and the wide variety of the mobile systems and services. Therefore the architecture must neither impose additional constraints on the underlying distributed systems technology nor rely on specific system environment (e.g. with respect to the location of certain services and resources). This further facilitates future extension of the architecture.

2. System Extensibility

The architecture must support extensibility to reflect the dynamic nature of Management Service provisioning. New standards and technologies are introduced frequently and must be supported by existing Management Systems as well as by new Management Systems [King2000] [Patel2002]. Hence, it is required to provide Management Systems that are able to meet existing requirements and are able to accommodate future extensions.

3. Compact Architecture

It is mandatory that the architecture is kept slim and efficient and that the architecture facilitates the complete lifecycle (i.e. development, deployment and change) to minimise complexity [Koch1996]. Furthermore, various heterogeneous technologies and protocols must be integrated and the resulting development and support effort should be kept low [King2001] [Lewis2002].

4. Modular Architecture

The Management Architecture should be based on a modular approach and result in distributable units of Building Blocks for the provision of Management Services. The concept of breaking up functionality into smaller parts facilitates the development and deployment process and helps to overcome many problems that affect existing implementations [Emmerich2000] [TMF-OM2000] [Phippen2001]. This is especially true in areas such as Network Management where monolithic systems are often difficult to extend and configure [Hofmann2000] [Knahl1999] [Phippen2001]. In addition Building Blocks have other potential advantages in terms of reuse (e.g. reuse of existing software code) and application design, which are mandatory to create a new generation of Management Systems.

5. Based on open standards

The architectural concepts and principles must enable the adaptation and integration of open standards (as opposed to reinvent new technologies) [Goldszmit1993] [Lewis2002] [Patel2002].

Although open systems must be based on open standards, proprietary protocols are often used to implement secondary functionality instead of using existing standard protocols (e.g. only a few large Web Servers or SQL databases can be managed using a standard management protocol such as SNMP). Hence, these protocols must be supported to integrate systems based on these proprietary protocols. This requirement prevents such systems from being managed like other Managed Objects and adds additional costs. For this reason, an architecture that is based on standard protocols must also allow the integration of proprietary protocols. However, the architecture should not be based on proprietary solutions.

6. Based on generic development methodology

The generic concepts of Building Blocks and Distributed Systems Architectures must be supported by a generic development methodology. The development methodology for a generic Management Architecture should describe the different phases and activities required to develop a system model based on Building Blocks and to assemble generic Management Systems.

7. Support of existing development languages and technologies

The motivation behind a generic Management Architecture is to be independent from specific programming technologies and languages. Hence the implementation of a Management System should not depend on a specific programming language or technology. In particular this must be applied to the proposed INSMware Framework and the INSMware Architectural components, which should be implemented on different platforms using different languages without relying on specific features.

8. Integration with 3rd Party Building Blocks

This issue is relevant from both a commercial and a technical point of view [Knahl2000] [Lewis2000] [TMF-OM2000]. The integration of the Management Architecture with other commercial applications (and potentially other existing Building Blocks) facilitates the integration of various management implementations. As another of the requirements is the adoption of open standards, the main issues involved are related to interoperability (i.e. to ensure the implementation of open standards to support the integration with 3rd Party Building Blocks).

9. Reuse of existing Building Blocks

Reuse of design principles and existing code is becoming increasingly important in the software industry [Emerich2000] [Hofmann2000] [Phippen2001]. Software Components are a way of achieving code reuse, whereas a good architecture facilitates design reuse. Reuse is an important issue because it saves development time and costs. Moreover, reusing well designed and tested implementations significantly reduces testing time, hence minimising the overall development process.

In the area of Integrated Network and System Management, the reuse of existing solutions is very important as it is not feasible to implement a new Management System from scratch considering the aspects of development time and costs. In addition it is quite often required to integrate legacy binary code in order to integrate the system with existing systems.

10. Security support

Security in the context of this research must consider the identification and implementation of functions that enable the distributed environment to be utilised in a safe way. Possible intruders must be prevented from interfering with the Management System. Furthermore security mechanisms should prevent unauthorised users from access to critical network resources (e.g. access to critical routers).

4.6.3.2 Operational Requirements

The following list identifies and outlines the operational requirements.

1. System Evolution

System Evolution in the context of this research refers to the ability to change (e.g. to extent or to modify) the behaviour of a Management System. System Evolution is a key requirement as Management Systems have to be adapted to the changing requirements and to the different environments in which they run [Knahl1999] [Lewis2002]. For instance, a Management Service that monitors TCP/IP networks will have to be significantly modified and extended when the network migrates to ATM. This is because ATM networks support new characteristics such as Quality of Service and capacity reservation, which typically were not present in TCP/IP networks.

2. System use and development

Complex Management System implementations as well as distributed systems are often regarded as complex systems that can only be operated by highly skilled specialists [Dornan2001] [Jander1999] [Lewis2002]. Furthermore, open standards may be complex. Ease of development concerns the overall development environment (e.g. programming languages and APIs which are used for the implementation of the Management System). In fact many problems are derived from the loose integration of the various object models and APIs. Ease of use concerns issues such as installation, customisation and operation of a Management System.

3. Support of distributed environment

It is required to support distributed systems technologies for the management of distributed, heterogeneous IT environments. This requirement ranges from the distribution of Management Services between remote locations to the exploitation of distributed environment services (e.g. increasing application performance and robustness).

The principal goal of the overall management solution is to provide a distributed Management System that allows the distribution of small, lightweight Management Services across the overall managed system as close to the managed resources as possible. Executing Management Services on or near managed resources reduces the amount of traffic due to management operations because management data is consolidated and refined at the source. Local execution of management tasks can be more efficient, making a richer set of local functions, data and resources available to the tasks. Partitioning of management functions into smaller components opens the way to potentially more flexible and effective deployment strategies for management software.

4. Efficient resource utilisation

As distributed systems and networks are complex structures, a significant amount of resources is usually required in order to manage these structures. For instance, an industry standard Network Management Platform typically requires at least processing power in the range of a medium-sized Unix server system in order to install and run the daemons and applications that make up the management environment [Dornan2001] [Jander1999]. The complexity of management applications and architectural implications has led to the creation of complex and "resource-hungry" applications. However, an implementation of a Management Systems should not demand a great amount of resources and should be able to run in environments of limited computing power (e.g. small to medium-sized environments where a Unix Workstation is not available). This has become even more important since the advent of mobile computing, as many users have replaced their desktop computers with mobile ones with limited processing resources that should be used efficiently.

5. System portability and generality

System portability and generality are important requirements, as the same Management System may have to run in different environments using various operating systems. This may impose constraints on the architecture that must be generic enough to be adaptable to different platforms but still able to meet the various requirements.

6. Support of Internet standards

Internet protocols and services are increasingly becoming an advocated way to access information (and probably applications in the future) [Box2000]. Therefore it is necessary to consider and incorporate these protocols and services in the architecture and to implement them when it is appropriate.

7. Support of management standards

When a standard protocol is accepted in research and industry, it has to be implemented in the management environment [King2000]. In general, the architecture should support a given standard although some functionality may not yet be implemented. This is significant for networking and management standards where vendors have to specify in a 'Protocol Implementation Conformance Statement' which standards have been implemented and the extent that they comply with the standard specification.

8. System scalability and performance

The INSM system must be scalable to adapt to a given distributed system. In the area of Network Management, network devices and services are constantly changing thus making it necessary to provide Management Systems that can be scaled according to current requirements.

Furthermore, Management Services must provide sufficient functionality and performance in order to be used for everyday and mission critical tasks. This is necessary in order to validate the architecture and to show that it can be used for "real" Management Service deployment.

9. Support of specific platform features

The support and exploitation of specific management and distributed systems features allows Management Systems to perform and behave as required and to be adaptable to specific system environments and requirements. However, as the main objective of this research is to be general, being both general and able to exploit platform specific features seems somehow contradictory. Although this requirement may be relevant for commercial products or to implement specific proprietary services (e.g. a vendor specific management protocol), the overall architecture must be based on open standards providing standard interfaces. Modifications to these interfaces or adapters may be incorporated to provide proprietary management solutions.

10. System administration

An implementation of an architecture that is difficult to install, operate and customise will ultimately lack acceptance with the administrators and users of the Management System (i.e. such an implementation may only be used by experts in a specialised domain) [Dornan2001]. It is mandatory to provide facilities that make installation, operation and customisation easy and enable the usage of the architecture not only for experts but also for users with limited experience with a particular management implementation [Dornan2001] [Jander1999].

4.7 Conclusion

This chapter has outlined the requirements and architectural principles to build and provide a comprehensive Management System. After identifying these requirements through the analysis of management development and provision, it was possible to start formulating, developing and refining the ideas for a flexible, composite and comprehensive integrated Management System capable of meeting both the immediate requirements and accommodating those which will arise as technologies develop. The next chapter moves on to discuss the ideas defined to facilitate the development of a suitable management framework based on a novel conceptual management approach.

Chapter 5

A New Management Approach

"We can think of an architecture of a system as the common vision that all the workers (i.e. developers and other stakeholders) must agree on or at least accept." [Jacobson1999]

"Design is ... a contingent process, subject to changes brought about by conditions that come to the surface after the big decisions have been made." [Ferguson1992]

In the previous chapters of the thesis, the state of the art in networking and management technologies and its implications upon new management requirements have been discussed. It has been argued that existing standards and technologies do not meet current and future challenges and the requirements upon a new Integrated Management System have been identified. This chapter gives an overview of the conceptual design of a novel Integrated Network and System Management Framework, INSMware, which is capable of meeting the requirements of Management Service provision.

The following sections identify and discuss the major components and interfaces of the INSMware Framework. The structural composition, interfaces and capabilities of the conceptual design are considered. The design and deployment environment to provide the means to develop a system without compromising its composite and comprehensive nature is also commented upon.

5.1 INSMware Challenges

From the analysis, a collection of both existing and envisaged management requirements was derived, thus identifying challenges that are important in driving the work be addressed by an innovative management approach. The following list summarises the most important challenges and groups related aspects together:

- Integration of Management Services;
- Low cost of Management Service deployment (e.g. automation of Management Services, sharing of Management Services, comprehensive Management Service development, distribution of Management Services);
- Easy access to Management Service information (e.g. usage statistics and network performance information, fault monitoring, Accounting and Billing);
- User control of Management Service (e.g. Management Service customisation);
- Support for a number of interfaces (e.g. availability of inter-domain Management Service information, use of different resources);
- Integration of legacy systems (e.g. reusing existing functions or integrating existing systems).

It is envisaged that other aspects and functions will become more important in the future:

- Integration and support of new services (e.g. integration of new multimedia services);
- Integration of various stakeholders (e.g. access to multiple service providers, intra-domain Management Service composition and deployment);

- Integration of operational environments (e.g. integration of Management Services into Workflow Systems);
- Extended co-ordination between Management Services and co-operation between Management Actors (e.g. Management Service composition over administrative domains, co-operation between multiple Management Actors);
- Extended user control over services (e.g. different levels of Management Actor access to Management Services).

5.2 Modelling of INSMware Building Blocks

The following concepts are used as a basis for creating an abstract model that enables a description of the architectural components of the INSMware Framework. This basis does not imply any dependencies on existing management frameworks, distribution concepts or any specific implementation. The main idea is to subdivide an overall system into smaller Building Blocks. These Building Blocks can represent a particular problem domain or provide a particular service.

For the communication between the Building Blocks, Interaction Calls and Interaction Events containing Identifier(s), Command(s), Data and Authentication between a client Building Block and a server Building Block must be provided. In the areas of Network and System Management, some Management Services require higher security functionality than others. Hence, security mechanisms must be provided (i.e. there needs to be control of access to Management Services and Managed Objects). An Access Relation is the collection of the Access Rights (i.e. a client is having access rights such as 'Read Data' on a server). To reflect different communication patterns between Building Blocks, Interaction Calls and Interaction Events must exist. A synchronous interaction between a requesting Building Block Instance and a providing Building Block Instance is defined as an Interaction Call. An example for an Interaction Call is a SNMP Get request. A requesting instance is asking for management related data from a SNMP Management Agent (i.e. providing instance). On the other hand, An Interaction Event is defined as an asynchronous notification between an initiating Building Block Instance and a receiving instance. An example for a Building Block Event is an SNMP Trap, where an SNMP Agent is issuing an asynchronous Notification to an SNMP Manager (i.e. receiving Building Block instance).

However, to further generalise interaction patterns between Building Blocks, an interaction and distribution model is introduced. The following aspects must be considered:

- Architectural Components supporting the interaction between functional units that can be distributed over different distributed entities or even locations are required.
- Regarding interoperability, no further implications may exist between the Building Blocks in a system. Hence, an Interface Component has to be introduced, which hides different design and implementation approaches and which hides the heterogeneity of different IT architectures.
- The access rights and procedures belonging to Interactions have to be incorporated into the interaction and distribution model through a realisation of an access control scheme. Two different realisation methods are being considered:

  (i) Internal Control Instance

  Every Interaction contains its own Access Control Instance. This enables a direct control, which typically comes with memory and processing overheads for the Building Block.

  (ii) External Control Instance

  An External Control Instance performs the control of the access rights independently from the individual Interaction. This might be less secure than an internal security scheme (e.g. if the external security component can be bypassed).

These aspects suggest the requirement for generic Interfaces that must be present in a Building Block. A Building Block Interface is linked to a Building Block to enable the exchange of data and functions with other Building Blocks via the Building Block Interaction Protocol. Hereby, the Building Block Interface has several duties:

- Hiding of specific implementation methods (i.e. processing of Building Block Interactions in messages of the Building Block Interaction Protocol);
- Concealing of heterogeneous data formats (i.e. the transformation between internal data types and the data format types of the Building Block Interaction Protocol);
- Security services.

Hence, the main role of the Building Block Interface is the adaptation of information flows. This incorporates the transformation of the local syntax into a common transfer syntax (and vice versa). The identified requirement for a Building Block Interface reflects the realisation approach of an 'External Control Instance', whereas the required support of heterogeneous interaction patterns reflects the realisation approach 'Internal Control Instance'. Hence, both architectural approaches might be considered for an implementation of an interaction and distribution model.

The composition of the INSMware Framework is based on the concept of Building Blocks. The utilisation of the INSMware Framework aims to result in an application architecture built using deployable units of software which possess similar forms and characteristics, as stated in the requirements. The following constituents lead to the INSMware Building Block Interaction Model as illustrated in Figure 5.1.

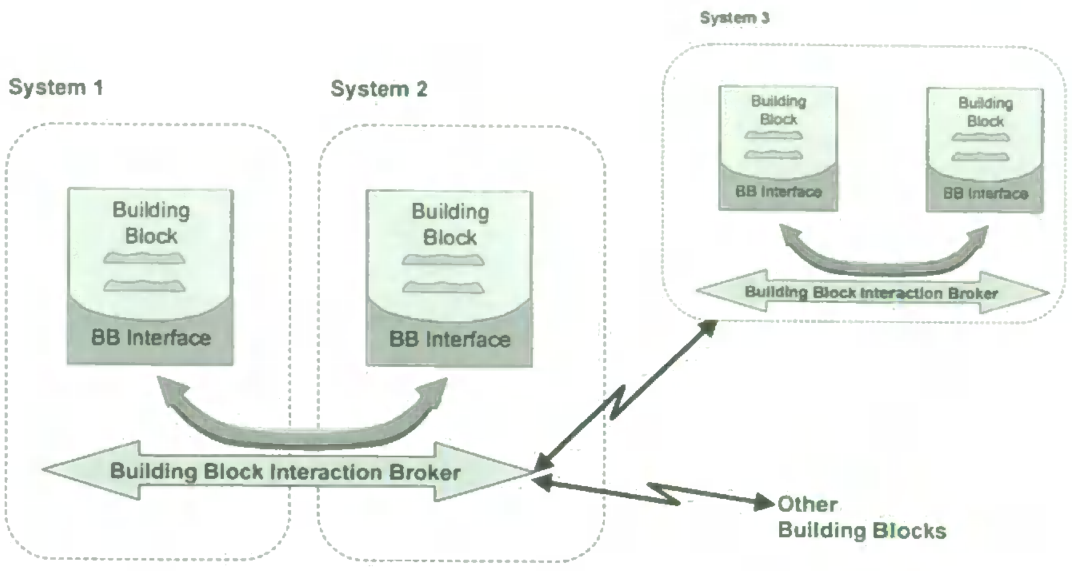


Figure 5.1: INSMware Building Block Interaction Model

A Building Block provides a set of interfaces to other Building Blocks that can be located on the same system or on different systems (providing Interaction Call and Interaction Event functions) to make it accessible (see Figure 5.1). These can be regarded as contracts as they specify the interaction between different Building Blocks. Hence, a Building Block contract must have attributes and operations that relate only to a service provided by the Building Block or to the management of the Building Block's services. These are referred to as service contracts and management contracts. Certain system management interfaces useful to system administrators should be common to every Building Block. If a Building Block provides multiple services, they should be accessed via different interfaces.

The communication between the Building Blocks is transmitted via the Building Block Interaction Protocol (BBIP). The BBIP is concerned with:

- Information aspects (how Data Formats are applied for transmission);
- Communication aspects (what message types are being used).

With the definitions so far, interactions between known communication endpoints would be possible. However, due to requirements concerning flexibility and distribution, a Management Service should be freed from the search and lookup of the Building Blocks from the various locations. This duty is delegated to the Building Block Interaction Broker (BBIB). The BBIB acts as a communication broker between different Building Blocks and performs the localisation and activation of interactions and the security services that are required with these interactions.

Hence, a BBIB can be regarded as a bus in which Building Blocks can commonly 'plug' into. Contracts must be known by type and a system administrator must be able to register the contract types of a Building Block with the BBIB when the Building Block is installed in the network. Hence, the BBIB must provide a Contract and Registration Service that enables the registration of contracts (e.g. when a Building Block is installed on a system).

When an instance of a Building Block begins to run on a system, its contracts become real instances of registered interface types. The Building Block, as part of its initialisation activities, must be able to announce its contract instances to the network via the BBIB. This announcement consists of informing the BBIB of the type and the attribute values of each contract instance. If the Building Block dynamically creates new contract instances at a later time, they must be similarly announced and registered. The BBIB must provide to a Building Block a service that enables it to publish the details of its contract instances (type and attribute values) to all other Building Blocks in the network. A Building Block must be able to find a registered instance of a contract type that it wants to use via the BBIB. Thus, the BBIB must provide to a Building Block a service to locate and select a registered contract of a particular type with particular values (or ranges of values) of attributes.

The BBIB must provide distribution transparencies to the Building Blocks (e.g. for a mobile Building Block that has no fixed location or that resides in a system that has no fixed location). These transparencies are BBIB facilities to hide the details and complexities of distributed computing from the Building Blocks. At a minimum, the BBIB must provide access transparency (i.e. the hiding of differences in data representations and invocation mechanisms) and location transparency (i.e. the hiding of the fact that two Building Blocks are co-located or in remote address spaces). Further transparencies may be addressed by BBIB implementations [ITUX901-1995]:

- Concurrency Transparency: The masking of concurrent use of the Building Block by multiple clients while insuring consistent use;
- Failure Transparency: The automatic restoration of a failed Building Block, possibly in a new location, while hiding the effects of the failure from the clients;
- Replication Transparency: The allowing of multiple instances of a Building Block for better availability and performance while hiding the effects of replication from the clients;
- Migration Transparency: The ability to change the location of a Mobile Building Block while participating in a (client or server) interaction and hiding the effects of the location change from the client or the server.

Furthermore, the BBIB must provide support for authentication and access control of human users and Building Blocks that can be distributed across a network.

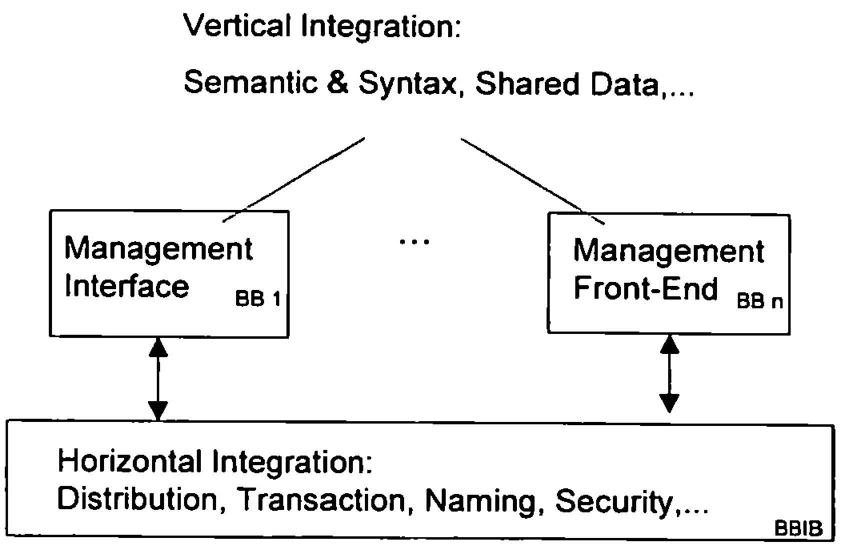

Vertical Integration:
Semantic & Syntax, Shared Data,...
Management Interface BB 1
...
Management Front-End BB n
Horizontal Integration:
Distribution, Transaction, Naming, Security,...
BBIB

Figure 5.2: Areas of BB and BBIB Integration

When building an INSMware Management System by assembling Building Blocks, the various integration and distribution aspects need to be addressed and agreed upon by Building Block, Building Block Interaction Protocol and Building Block Interaction Broker providers so that different Building Blocks are interoperable. In an open Building Block market, this can potentially lead to the reduction of the development burden for common tasks. Figure 5.2 illustrates the 2 main areas of integration, namely Horizontal Integration and Vertical Integration that should be addressed by standardisation. In addition to the underlying services being shared (addressed by Horizontal Standards), Building Blocks must also inter-operate within a service domain in which Building Blocks will inter-operate (addressed by Vertical Standards).

5.3 INSMware Top-Level View

A top-level view of the INSMware Management Framework is presented in Figure 5.3. It consists of a number of domains [Hawkins1992] that are illustrated as separate packages [Souza1999]. The

Management Service Domain groups elements that fulfil the functional requirements of the system. The ITU-T concepts of Fault, Configuration, Accounting, Performance and Security (FCAPS) were used as an initial basis for the breakdown of functionality to be modelled for the provision of Integrated Management Services. The Management Actor Domain provides Management Users with support and information regarding the actors involved in the management business process. The Service Support Domain groups elements that model the resources required by the Management Services. The Management Engine Domain provides the core Management related functions required by generic Management Services.

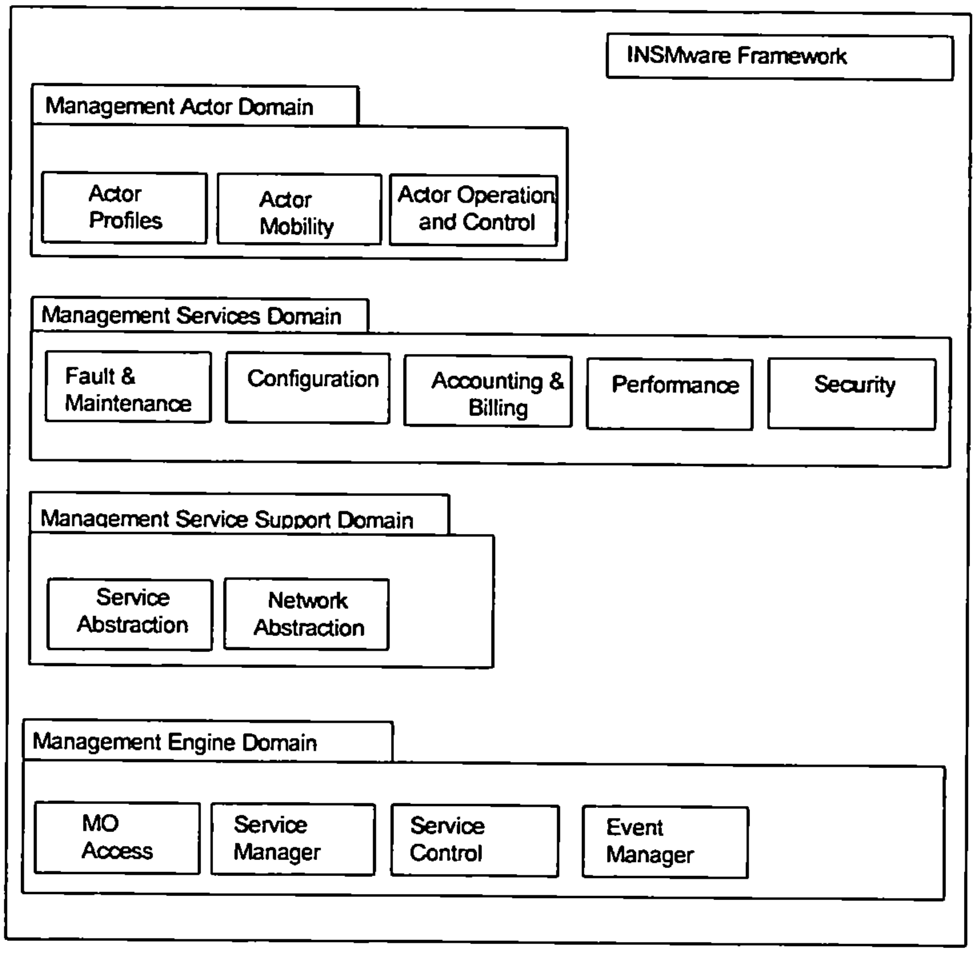


Figure 5.3: The domains of the INSMware Framework

Each of these domains contains services and functionality required for the provision of Integrated Management Services which together make up the overall management system.

The Management Actor Domain groups services and functions required by a Management Actor (i.e. a user accessing Management Services). A Management Actor is furthermore described through its role that determines the basic characteristics. It is envisaged that a Building Block is provided for a particular user providing the required functionality:

- Actor Profiles: Provision of Management Profiles for Management Actors (e.g. Management Service Administrator or Management Service User) utilising a specific profile for a given Management Actor. Furthermore, the exchange of information between different Management Actors must be enabled. This information will be used by other domains to enable the customised provision of maintenance, accounting, performance and security functions.

- Actor Operation and Control: Provide access to the service information and profile information, and to perform controlled management activities.

- Actor Mobility: Provide mobility facilities to the management actor, specifically personal mobility capabilities such as remote access or access through mobile equipment (e.g. mobile phones or other Personal Data Assistants).

The Management Services Domain provides the core Management Services that are required. This set of core Management Services can be further refined to meet specific requirements The FCAPS management areas were used as a starting point for the breakdown of functionality required:

- Fault & Maintenance: This service is responsible for detection and resolution of service related problems (e.g. network resource failures). The responsibilities can be further refined to include maintenance capabilities.

- Configuration Management: This service is responsible for configuration of service entities at the request of a Management Service or a Management Actor, including network resources and service resources.

- Accounting and Billing: This service is responsible for billing of service users and payment of service suppliers.

- Performance Management: This service is responsible for the monitoring of service quality.

- Security Management: This service responsible for ensuring security in the management of the services. Areas such as authenticity and integrity will be considered.

The Service Support Domain provides generic functions regarding Network and Service Abstraction:

- Service Abstraction: A layer grouping and modelling service resources that are required for a particular service. In the case of a Network Management Service, this layer provides generic Management Services (e.g. Router Monitoring) and can usually be found on top of the Network Abstraction layer.

- Network Abstraction: A layer grouping entities that represent the network resources involved in providing the services to be managed. The details of the network abstraction depend on the nature of the service being managed. However, in general, the Network Abstraction Layer provides generic information (e.g. HW / SW description) and models the current state (e.g. availability and activity) of the service resources.

The Management Engine Domain contains generic Management Service access, operation and distribution capabilities:

- Managed Object Access: Provide access and communication to Managed Objects. Typically, a number of different Managed Objects (e.g. using different management protocols)exist that must be supported.
- Service Manager: Provision and operation of Management Services.
- Service Control: Configuration, monitoring and scheduling of Management Services to be used by the Service Manager.
- Event Manager: Collection, correlation and distribution of events (e.g. report events such a faults and status messages received from Managed Objects).

A standard methodology should be used when describing the relevant functions and services within each of these domains. The Requirements have to be outlined in order to provide a context on the domain's intended functionality. Additionally, the internal working of the relevant domain has to be described in the Analysis section. This description gives a detailed account of how the specific domain achieves its functionality as detailed in the requirement section. Furthermore, the Architectural Dependencies with other functions and services need to be identified. This gives a description of how a domain's functionality fits in with the overall INSM system.

The INSMware Management Framework provides the means to group Building Blocks at a conceptual level, to create an Integrated Management System. It is generic enough to be applied across all possible existing and future systems and flexible enough to integrate existing solutions as well as offering itself as a possible solution to be amalgamated into future architectures. The core benefit of using the framework is that it provides a generic solution for implementing a management system specifically for Integrated Network and System Management Services based on generic modelling concepts. These are supplemented, where it is felt that this is required, with guidelines and additional design methods. The framework intends to provide a clear and concise template of the functionality necessary to implement an Integrated Management System.

5.4 INSMware Framework Reference Model

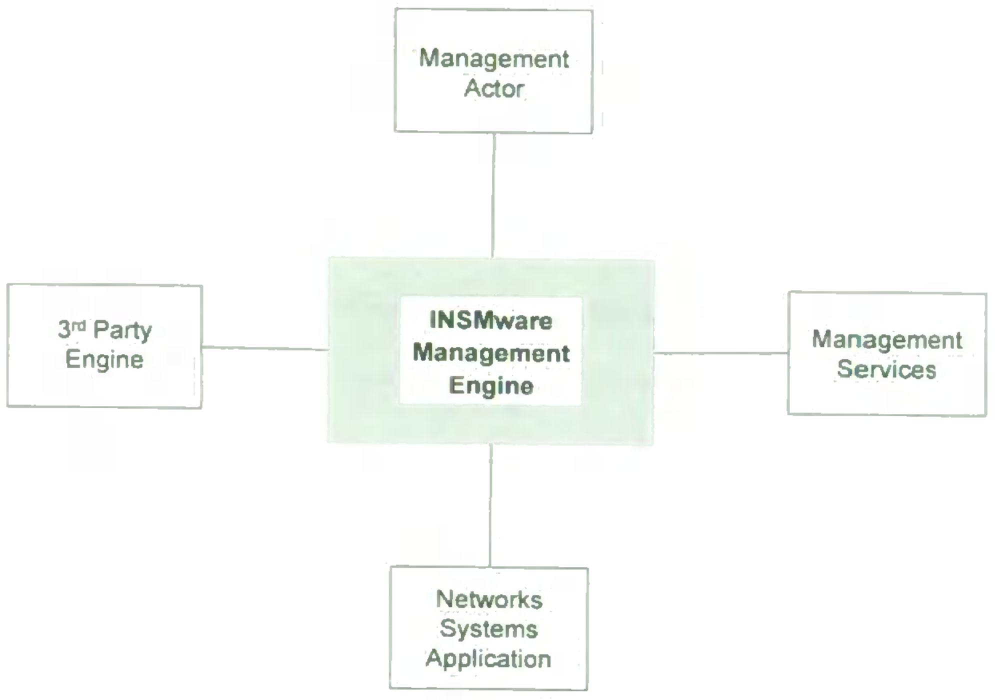


Figure 5.4: INSMware Framework Reference Model

Figure 5.4 illustrates the INSMware Framework Reference Model. The INSMware Management Engine is the central instance of the framework, as all other domains access its services. The 3rd Party Engine Domain consists of other systems that should be integrated with the INSMware Management Engine to provide Integrated Management Services (e.g. other Management Engines). Furthermore, the INSMware Engine must provide access to the Managed Objects (Networks, Systems and Applications).

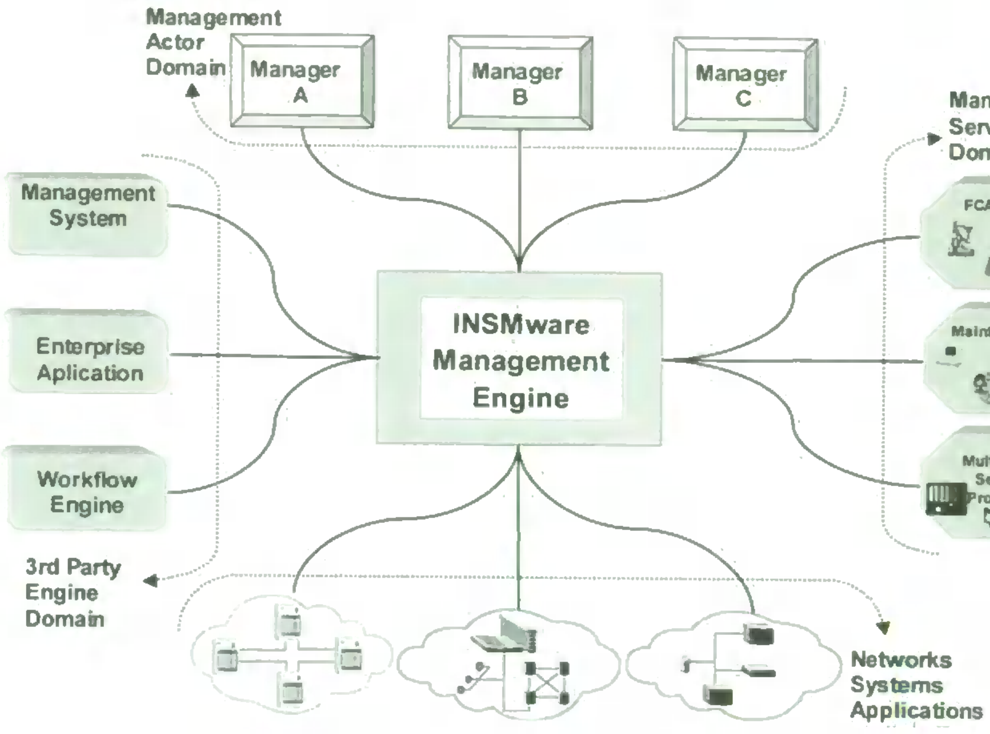


Figure 5.5: INSMware Reference Model including Domains

An extended illustration of the INSMware Management Framework is presented in Figure 5.5 It contains the following domains that form separate spheres of control:

- INSMware Management Engine: It offers a set of generic management related functions and acts as a middle layer between the Management Actors and the management environment. It provides the generic Management Services and relates management requests into a set of operations that can be applied to the Managed Objects.

- Management Actor Domain: It offers the functionality for Management Actors to access Management Services and to control the overall system.

- Management Services Domain: The Management Services Domain provides all necessary information about the Management Services to be processed by the Management Engine. This includes information about the generic network and service models and references to

Management Service specific Building Blocks that may be included in the provision of a Management Service.

- 3rd Party Engine Domain: Coexistence and integration with other Management Engines and IT systems (e.g. an Enterprise application such as SAP) through common interfaces.

- Managed Objects (Networks, Systems, Applications): Heterogeneous objects that support a number of different management protocols and mechanisms must be integrated to enable seamless end-to-end management.

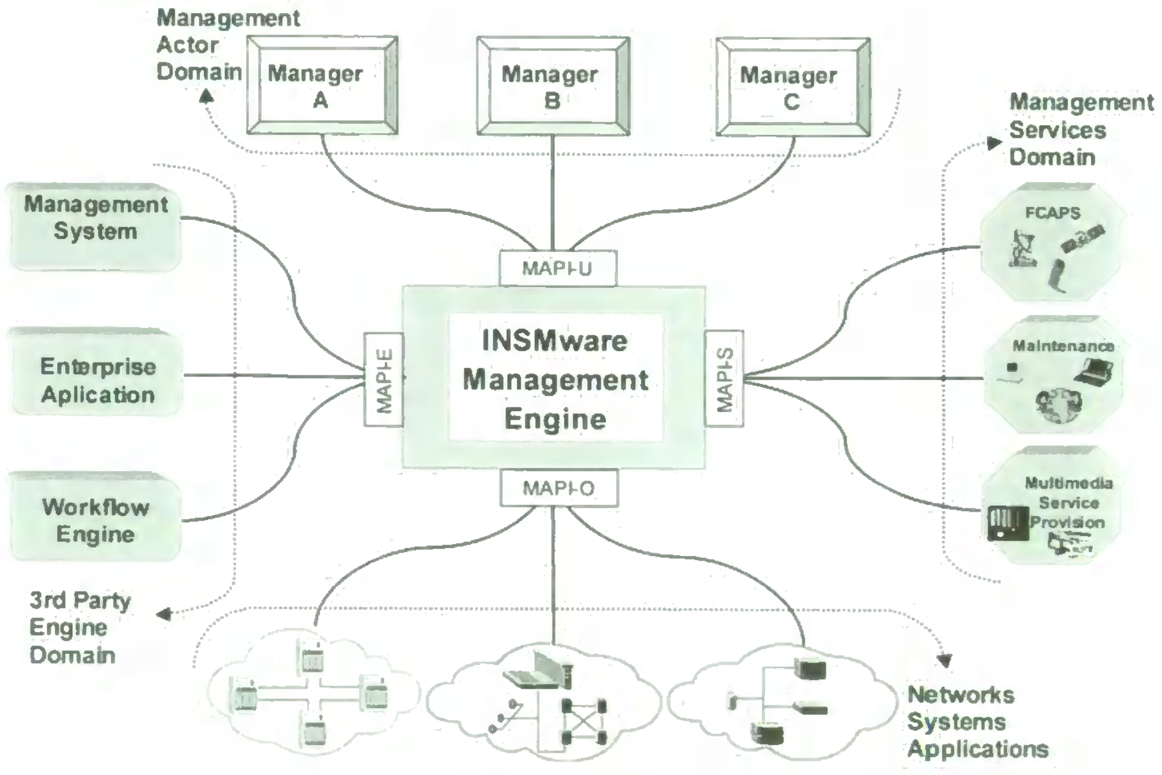


Figure 5.6: INSMware Reference Model incl. Interfaces

Although the INSMware Reference Model shows 4 separate interfaces, a number of functions within each of these interfaces are common (e.g. Management Service status calls may be issued from a Management Actor or another Management Engine). Therefore, the interface around the Management Engine is referred to as MAPI (Management Application Programming Interface). MAPI is used to access and control the communication between the Management Engine and the different domains. A subset of interactions will be in common to two or more domains. Hence, MAPI can be considered as a general interface description. MAPI-Engine (MAPI-E), MAPI-User (MAPI-U), MAPI-Services (MAPI-S) and MAPI-Objects (MAPI-O) are domain specific extensions (see Figure 5.6).

Scalability inevitably requires that the INSMware Framework and its corresponding Building Blocks are themselves distributed. Therefore, the various parts of the framework are separated into several domains that correspond to functional areas. An implementation of the INSMware Framework may consist of a number of Building Blocks covering all the outlined domains or a specific subset to meet a subset of requirements.

At the same time, the distribution of the overall framework has to be transparent for the users and services. Distribution in this context does not mean the use of a distributed database or the front-end interfaces but a distribution of the entire structure of the management. Mechanisms as defined in the Building Block Interaction Broker are required for the transparent transfer of messages between the distributed entities of the core infrastructure. In the range of functions the management framework must be suitable for the management of the entire IT services of today's and future infrastructure.

5.4.1 Management Engine

The Management Engine is the central instance that defines, creates and manages the execution of Management Services. It can be regarded as an assembly of core parts providing a runtime environment for the provisioning of Management Services. The Management Engine must be able to interpret the Management Service information running on one or more Management Engines, interact with Management Actors and invoke required Building Blocks. Hence, the Management

Engine is the central management control instance providing the INSM provision, INSM support and INSM data.

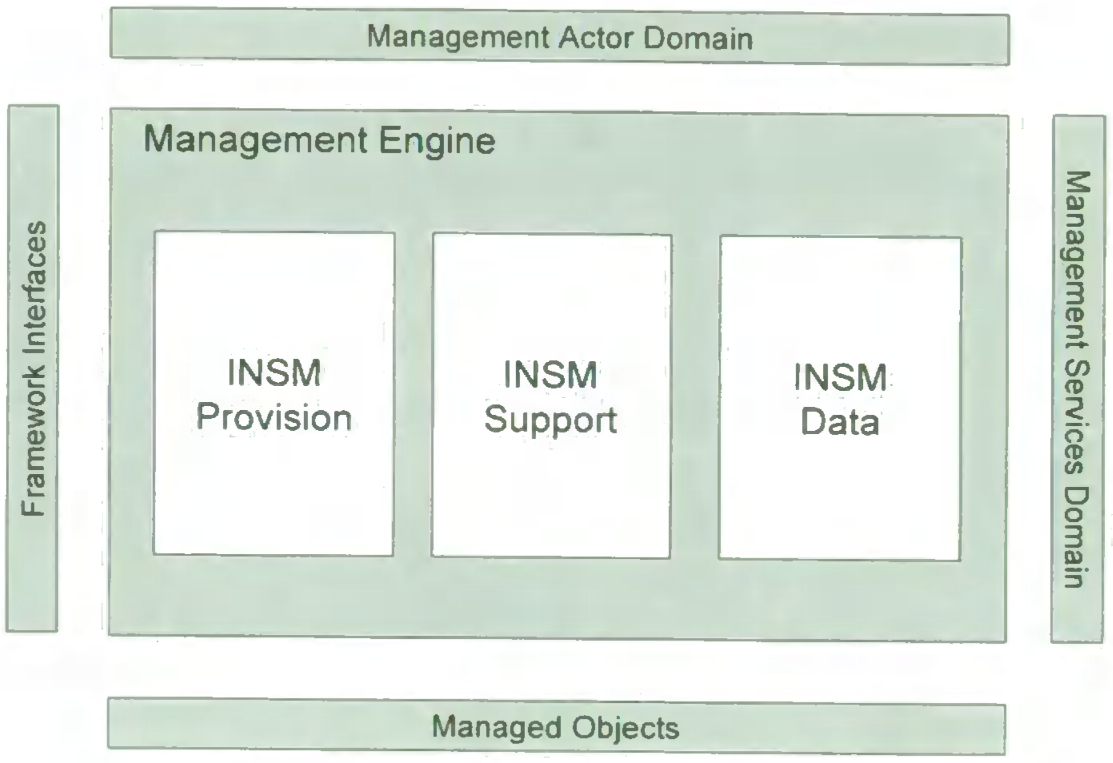


Figure 5.7: Management Engine#1

The Management Engine controls the flow of Management Services through the system (i.e. sequences of management activities that form a Management Service) by applying the Management Service rules to determine the scheduling of required activities and to invoke the required Building Blocks. The main responsibilities of the Management Engine are:

- Creation, deletion and management of Management Service over the whole lifecycle (i.e. from instantiation to completion);

- Interaction with associated domains (e.g. other Management Engines or Management Actors) which request or initiate Management Services;
- Interaction with Managed Objects;
- Monitoring and control of the Management Services.

The following are the key services provided by the Management Engine itself. The ultimate goal is to provide these services without having to adapt and modify the overall framework itself for the different identified management scenarios. Key services of the Management Engine to provide the basis core functions:

- Receipt and processing of management data. Typically, an existing enterprise data warehouse will be used to store data (e.g. an Oracle database). A special case would be that no integrated database exists, then a dedicated management database will be used instead.
- Discovery and Mapping of Managed Objects.
- Integration of management data from different sources to enable integrated processing. This requires that different data formats are mapped to a common format.
- Filtering and processing of notifications to eliminate obsolete, duplicate and unimportant ones (i.e. grouping, correlation and automatic forwarding of notifications).
- Assignment of priorities to notifications on the basis of diverse criteria such as type of notification, source or time of day according to the business processes in the organisation.
- Integration of existing and future technologies (e.g. management protocols).
- Distribution and control of Management Services.

- Adaptation of different management scenarios (e.g. a Network Integrator such as PanDacom as a provider of Remote Management Services for its customers, or a network manager in an enterprise for integrated network and system management).

The provision of Management Services by the Management Engine can be considered as a State Transition Machine where individual Management Service Instances change states (e.g. in response to external events such as completion of a Management Service) in a way that processes under the control of an Operating System change state [Stallings2001].

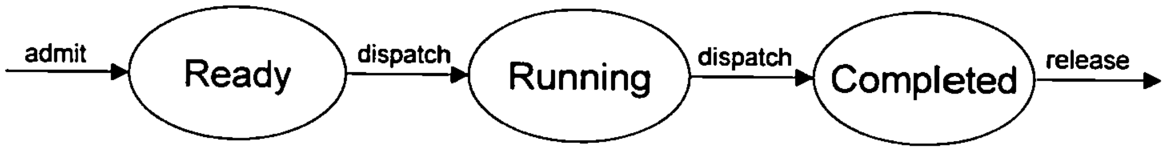


Figure 5.8: Three-State Management Service Model

Figure 5.8 illustrates a simple illustration of the basic states and transitions of a Management Service. The three states of a Management Service according to Figure 5.8 are:

- Ready: A Management Service that is prepared to be executed;
- Running: A Management Service is being executed;
- Completed: A Management Service has been released from execution (e.g. finished its execution).

However, a particular Management Engine implementation may support additional state types or incorporate different state transitions that take place in case of a MAPI command or certain Management Service condition (see also Operating Systems state models [Stallings2001] [Silberschatz2002]).

The Management Engine defines a number of Management Services and supports generic Management Service related support activities. Figure 5.9 illustrates the generic service processes provided by the Management Engine.

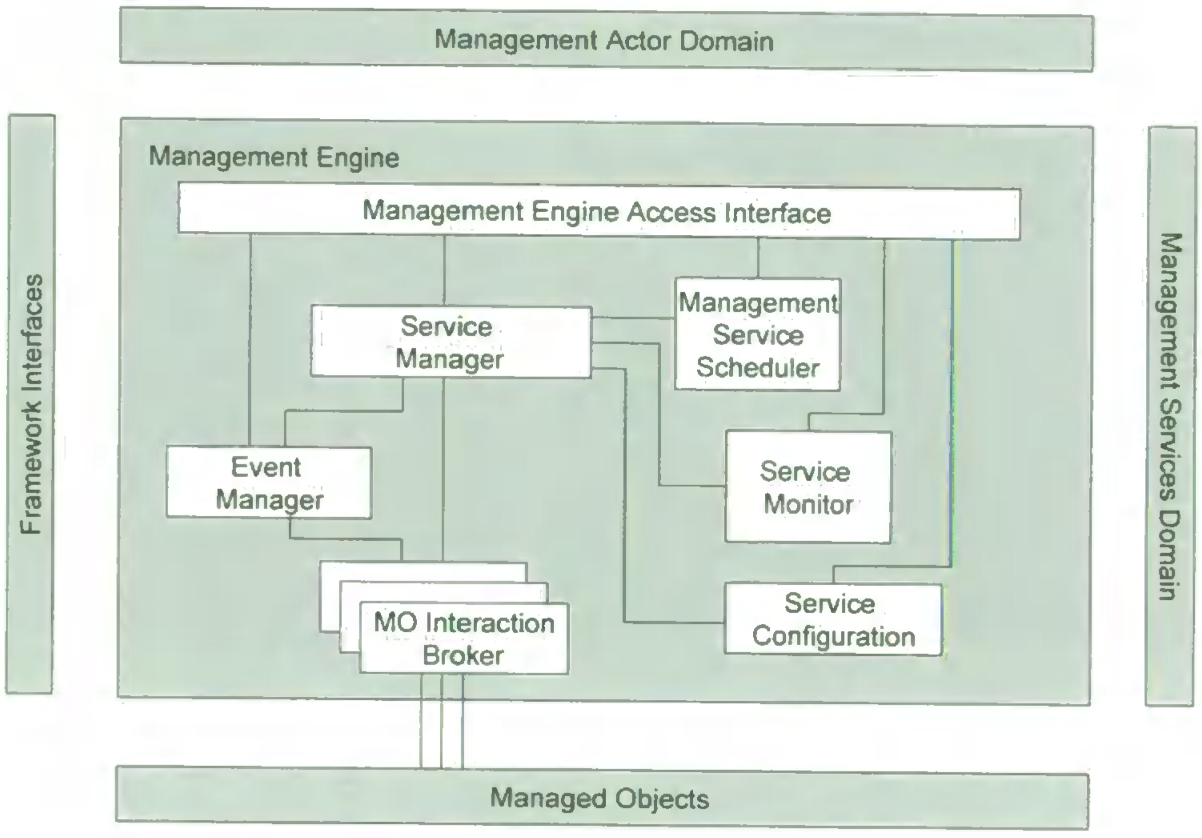


Figure 5.9: Management Engine#2

The Service Manager supports the creation and provision of a Management Service. It provides the Management Engine Access Interface with a set of interfaces for requesting services. The actual services are based on the functions and requests provided by the Management Service Domain. The operations on this interface include:

- Management Service request and configuration;

- Management Service status enquiry;
- Management Service deletion.

The Service Manager interacts with the other processes to achieve the required connectivity to supply the service specified. The Service Manager performs the following tasks:

- Determine the feasibility of a Management Service;
- Configuration of Management Service;
- Activation of Management Service;
- Monitoring of Management Services;
- Modification or deletion of services.

The Service Manager models the current state of the service resources in terms of their availability, activity, etc. The Service Manager performs these actions by invoking the service configuration process. The Service configuration process is capable of configuring the Managed Objects and acts as an integration point between service processes and network processes.

The Service Configuration Process is a process to support the service configuration. It is a key process to support the integration between abstract Management Services and Network and System Management processes. The Service Configuration Process groups elements, which represent the resources, involved in providing the Management Service. The details of the Service Configuration Process depend on the nature of the Management Service. The Management Service Scheduler and Service Monitor further support the Service Manager in the tasks of Management Service administration (e.g. integration of established thresholds in the form of SLAs or scheduling mechanisms).

The Managed Object Interaction Broker supports the communication with the various, heterogeneous Managed Objects and must integrate different management protocols and technologies:

- Access to networks;
- Universal service access;
- Integration of legacy systems;
- Independent evolution of services and network infrastructure.

To provide Integrated Management Services, it is required to achieve integration between different management technologies and domains. A network environment may use different management technologies such as CMIP and SNMP in different subsystems. Integration is thus required in order to support the communication between these subsystems. Furthermore, it might be advisable to place the Managed Object Interaction Broker close to the Managed Objects (e.g. in the physical segment where the Managed Objects can be found) to minimise management related overhead. Thus, it might be required to operate a number of Managed Object Interaction Brokers simultaneously.

The Event Manager, that is either invoked by a request in the Service Manger or by events from the Managed Object Interaction Broker, deals with event notification and information forwarding of Management Service related problems. The process is entered when events are reported to the management system (e.g. interruption or low quality in a communication service). The process is responsible for ensuring the notification of events.

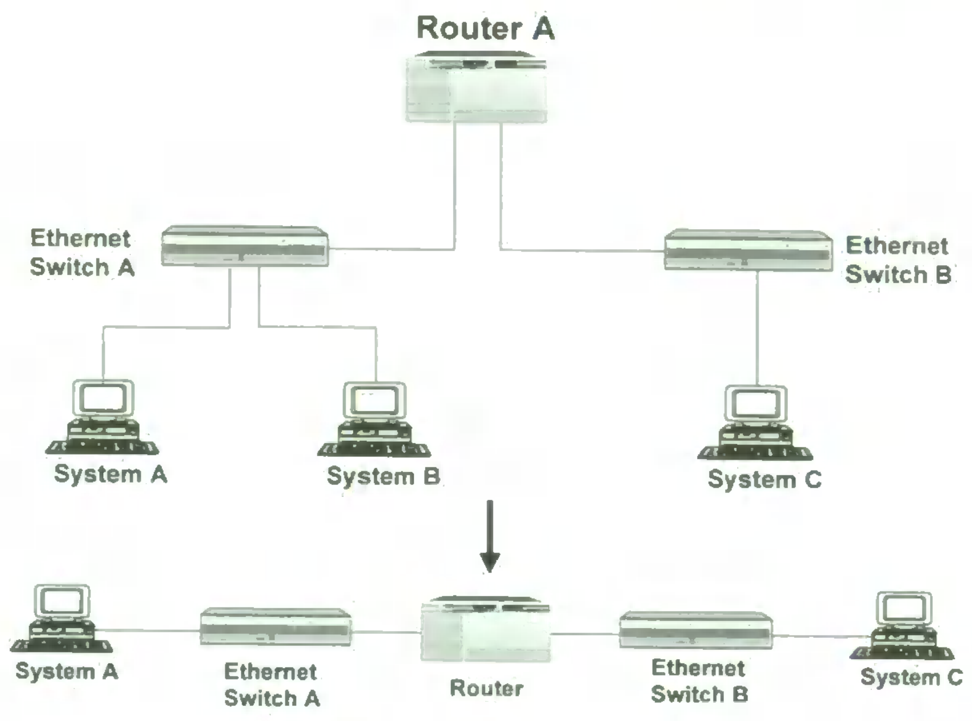


Figure 5.10: Communication Scenario A

In the example outlined in Figure 5.10, the Communication between all systems is based on Ethernet (OSI Layer 2). However, depending on the actual communication service and pattern (i.e. systems involved in communication, type of communication, higher-layer protocols), different systems might be involved. A communication between System A and System C involves Ethernet Switch A, Router A and Ethernet Switch B. Hence, to manage the End-To-End communication between System A and System B the Management System must be aware of the actual communication topology and must be able to map the different heterogeneous devices onto that topology. Hence, the Management Service 'Availability-of-A-to-B Connectivity' relies on data from a number of heterogeneous objects. Data in these objects (e.g. network element specific data provided by a SNMP MIB) might be of limited use for the network-administrator unless it is presented in an abstracted and for the network-administrator understandable fashion.

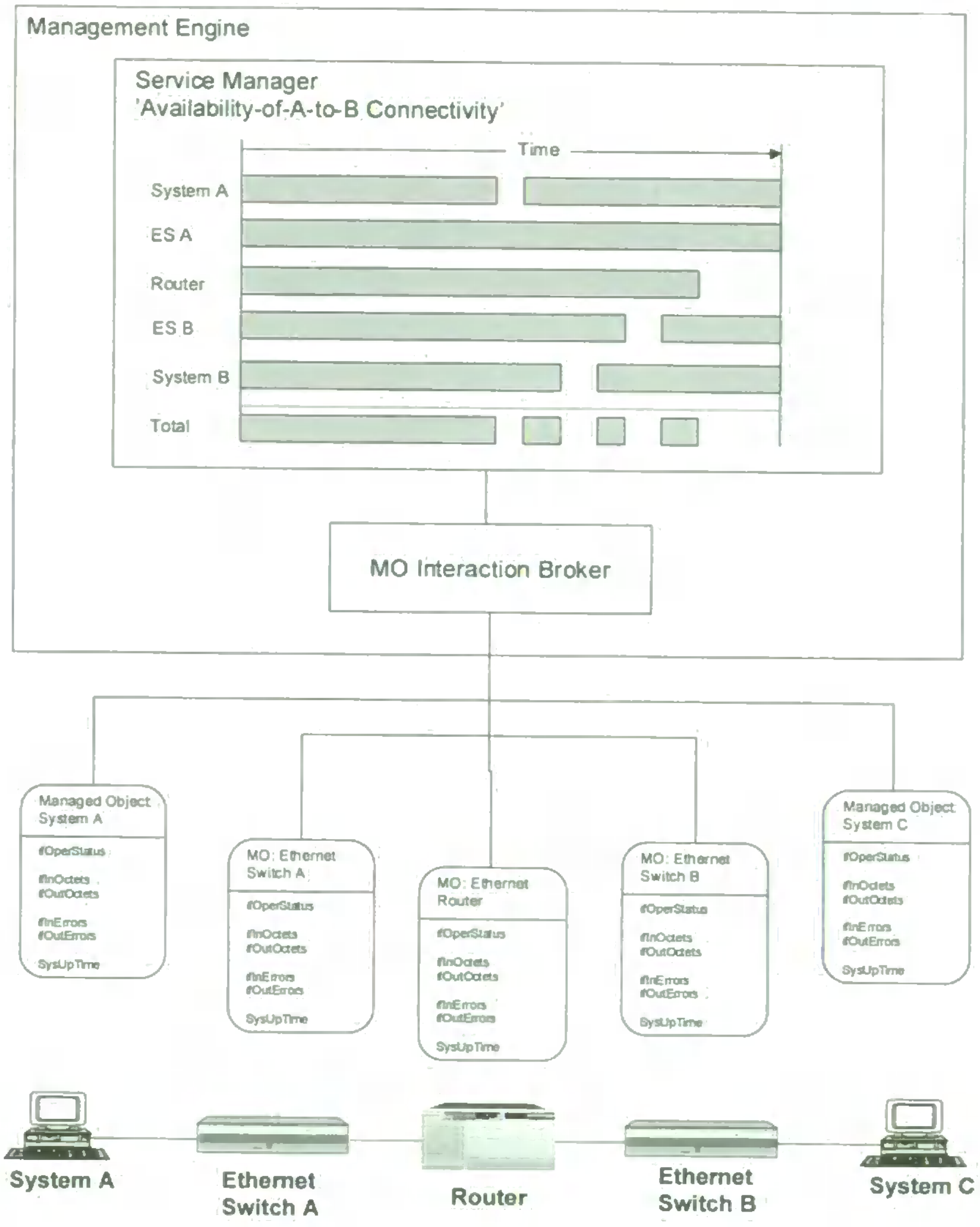


Figure 5.11: Integration of Managed Objects

Regarding the Management Service 'Availability-of-A-to-B Connectivity', the 'ifOperStatus', Interface statistic variables such as 'ifInOctets' and general variables such as 'SystemUpTime' can be analysed to establish an indication for the availability of the network resources in the case of SNMP management. Figure 5.11 illustrates the resources involved in the management process. The combination of the 'SysUpTime' can be used to establish the availability of a communication link and the other variables can be used to further consider the performance and operational status of the link. In this case, the Service Manager is required to retrieve the required data and to apply the data to a given Management Service.

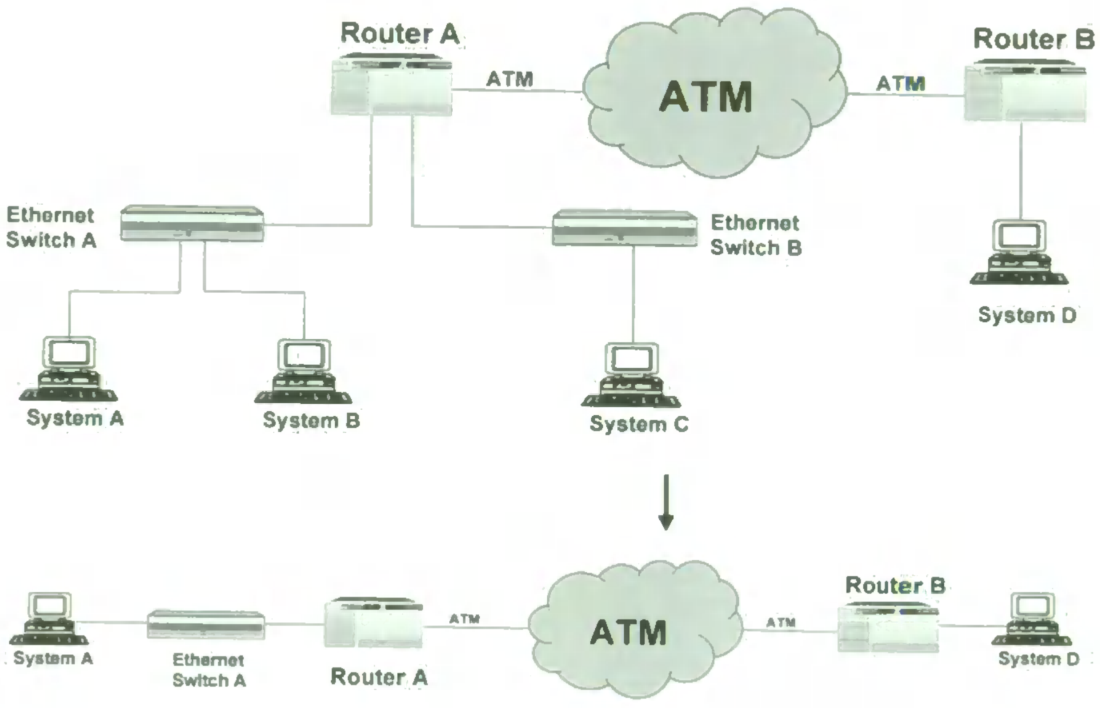


Figure 5.12: Communication Scenario B

Figure 5.12 illustrates an extension of the previous scenario. In this scenario, the Management Service requires the underlying network management systems to provide an integrated end-to-end view of the networks function. The ATM network could be an enterprise backbone or a Public Carrier network. To establish the availability of a communication link, data from the ATM network needs to be incorporated in the Management Service. In a typical network deployment, these technologies have their own management systems with different interface specifications. This could be using the same SNMP parameters (in case the ATM internetworking devices provide this information), by presenting a standard ATM M4 Public Network View [McDysan1999] for its connections or by incorporating data provided by the ATM service provider.

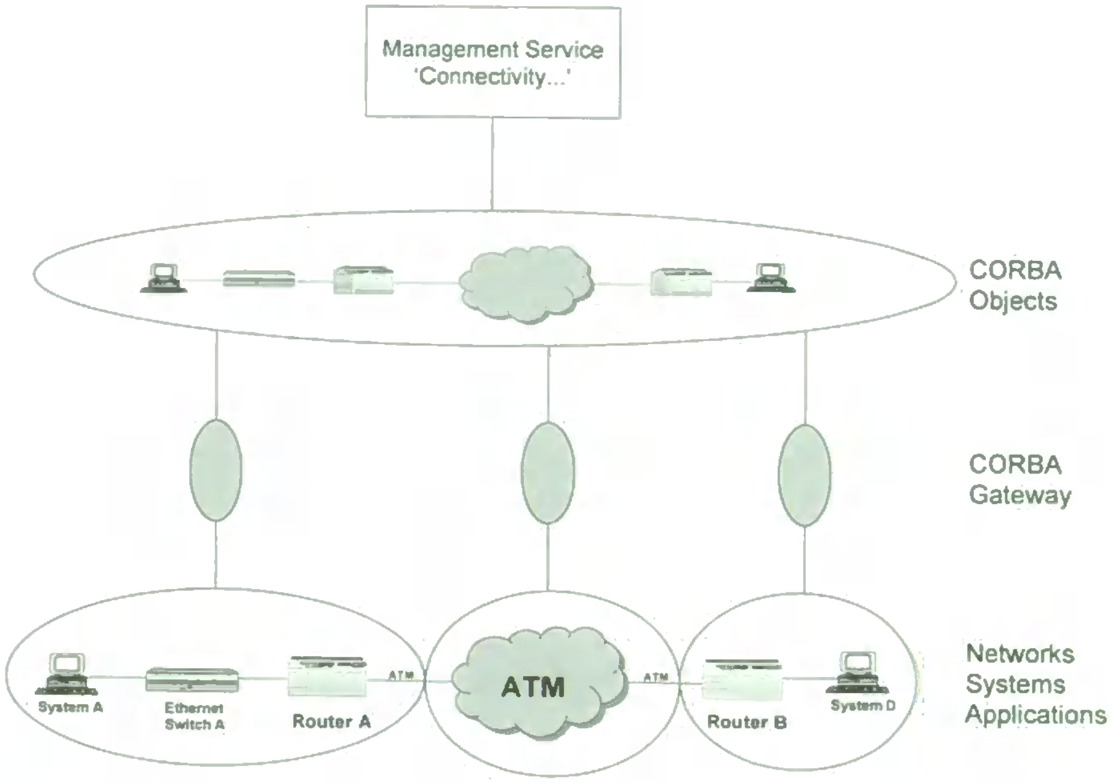


Figure 5.13: CORBA for Integrated Management Services

A unified end-to-end network interface may also be defined by using CORBA IDL, and the management system of this interface can be developed by using CORBA Components [Keller1999]. A number of gateways can be produced to adapt the unified interface to the individual technology domain providing a common CORBA interface to access network-functions in a generic and end-to-end manner. CORBA provides facilities for the definition and implementation of Software Components and it also provides an environment for the invocation of Software Components. The CORBA IDL allows shared operations to be specified with generic interfaces. This enables service objects to be rapidly developed and deployed. These features make

CORBA technology an interesting candidate as an environment to implement Integrated Management Services. In particular, this environment can be based on the distributed management as presented in Figure 5.1. A Building Block approach can be used to add more functionality to form an Integrated Management System.

Within this section, the Management Engine has been discussed as a single logical entity, although it may consist of a number of Building Blocks and these different Building Blocks may be distributed.

5.4.2 Management Actor Domain

The Management Actor domain provides functions and services regarding the actors involved in the provision of Management Services. It provides the means by which humans can make use of the functionality of other Building Blocks of the framework to meet their management requirements [Telcordia1993b]. The framework must provide user management with several user classes, such as Management-User or Management-Administrator, and extensive functions for access control and view creation. Management Actor Building Blocks will therefore be specialised for an understanding of the user's management requirements and tasks and intends to hide the underlying complexity from the user.

Typical Management Actors are:

- Network oriented staff in Networking department of a company to provide Network Management;
- End-System oriented staff in an IT department of a company to provide End-User support ;
- System Integrator or Network Integrator to provide Management Services to customers (e.g. a Network Integrator such as PanDacom to provide remote management services to customers from an in-house Network Operation Centre);

- Service Provider to provide service level management and meet Service Level Agreements for customers.

The support of these different Management Actors requires the implementation of front-end Building Blocks to access the Management Engine (i.e. Management Service) functions. Such front-end applications can be realised or provided by management application providers or management systems integrators. Integration between management and other desktop applications (e.g. e-mail or spreadsheets) is also a common target. This allows e a single end-user interface and set of function, regardless of the number and types of management Building Blocks in an installation. Therefore, the Interface between the Management Actor Domain and the Management Engine (MAPI-U) must provide the following:

- Management Data types and structures (e.g. Management Service identifiers, Managed Object identifiers and addresses);
- Management Service control functions (e.g. terminate a Management Service);
- Management Service enquiry functions (e.g. retrieve status of a specific Managed Object).

The MAPI-U and its parameters must be mapped onto several underlying operating platforms (e.g. different Operating Systems or communication protocols) to ensure the integration of different systems.

The Management Actor BB provides functionality that supports interaction with a human (see Figure 5.14). Structuring of information to users of a Management System is encapsulated within the Management Actor BB and elements of that function must not take place in any other Building Block. The most fundamental aspect of interaction with a human is the Front-End Interface (typically provided in the form of Software) and the attached Front-End infrastructure (typically provided in the form of Hardware and Software, e.g. X-Windows terminals, Web interfaces or PCs running Windows).

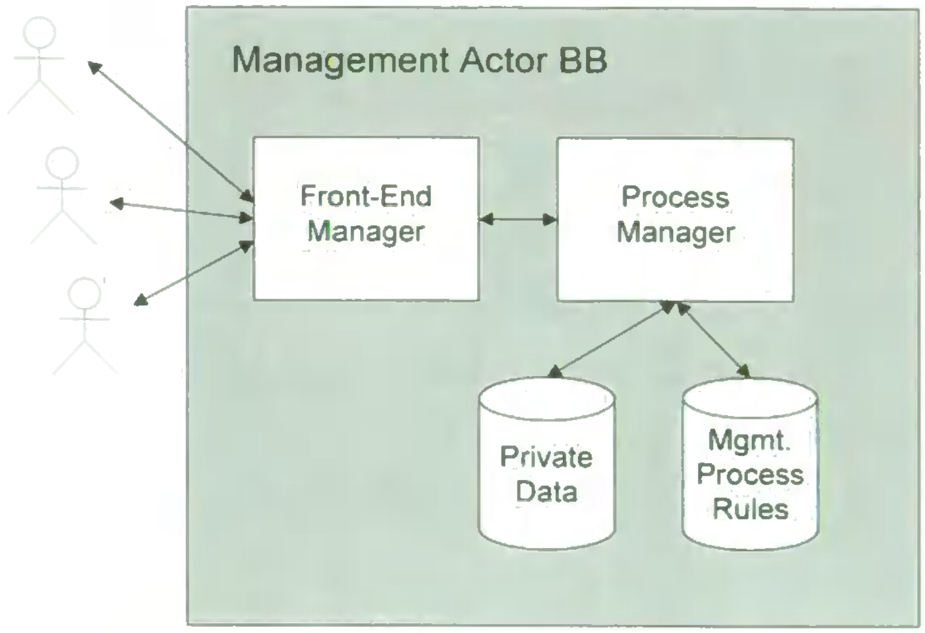


Figure 5.14: Management Actor Building Block

To meet the requirements of the Management Actor, a Management Actor Building Block must encapsulate an understanding of the Management Actor goals and tasks and translate user tasks into a set of activities. It therefore contains Management Actor specific *Management Process Rules* that must supported. These Management Process Rules contain a model of the management procedures (i.e. the operations required to perform a Management Service) for the user to accomplish its context specific tasks. It is the responsibility of the Management Actor Building Block to provide information in a form appropriate for a specific user (e.g. taking the user's profile and role into account). *Private Data* contains the operational context of user-specific semantics for the operations and may provide individual user customisation of the end-user presentation. The *Process Manager* accomplishes the tasks that model the specific Management Actor and its Management Services. The *Front-End Manager* provides a display interface to interact with humans. Since user interactions are encapsulated within the Management Actor Building Block, security functions related to human access arise (in addition to generic BB related security requirements). Furthermore, security functions must provide authentication and secure communication with other Building Blocks. It can be envisaged to provide a single Building Block for a given user. To support Management Actor Mobility, it would be provided as a Mobile Building Block.

5.4.3 3rd Party Engine Domain

Provision of Management Services will coexist and must be integrated with other existing and future business processes. Therefore, a management system needs the means to integrate and adapt existing and new, management and non directly-management related functions. Examples include:

- Integration of a Management Engine with another Management Engine;
- Integration of Management Engine with other business applications (e.g. SAP R/3);
- Integration of Management Engine with legacy Management Systems.

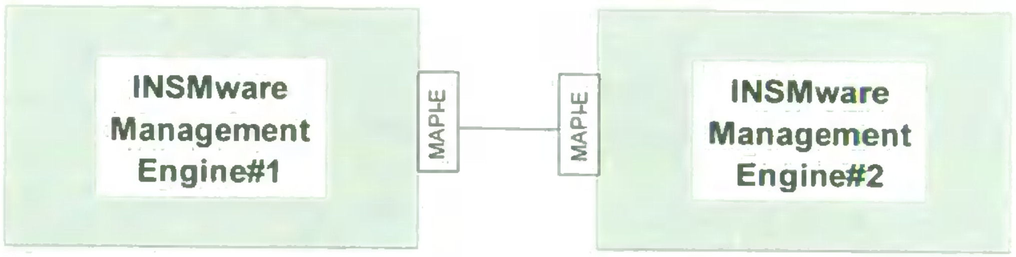


Figure 5.15: MAPI-E Interface

The MAPI-E interface provides the mechanisms that are required to enable one Management Engine to make requests to another system to effect the selection, activation and control of Management Services by that other engine or vice versa (see Figure 5.15). The requesting Management Engine is also able to request or pass management data and to receive back management data and the results of the enactment of Management Services. As far as possible, this has to be done in a way that is transparent to the involved systems and users. The levels of interoperability between Management Engines are distinguished by the architectural and consequent operational characteristics of implementations of Management Engines. The depth of integration is distinguished by the extent to which Management Service Definitions and the various types of Management Service control functions and data can be exchanged between the different systems. To achieve full interoperability, all MAPI Data Formats and Function Calls must be standardised and implemented (i.e. all related Building Block communication including the transmission of management related data and Management Service definitions). To achieve this level of interoperability, different Building Block provider may be required to support a number of different mechanisms through which such interoperation can be effected.

Many of the MAPI functions that can be found on the other interfaces can also be applied to the MAPI-E interface (e.g. Management Service control and enquiry functions, import and export of Management Service Definitions). Full interoperability would assume that in addition to the preceding requirements, all corresponding products present the user with a standard user interface. However, for commercial and practical reasons, this level may never actually be attained.

Where the involved systems can process the same Management Service definition (e.g. generated from the same Management Engine processes), a single view of the Management Services can be achieved. This potentially enables Management Engines to transfer execution of activities or sub-processes to heterogeneous workflow engines within the context of a common Management Service. Using this approach, several systems can be fully integrated and operate on a peer-level. Where the sharing of Management Services is not applicable, the alternative approach of exporting details of a Management Service may be feasible. In this case, the interface provides the means of requesting data regarding a particular Management Service, thus enabling a Management Engine to obtain data relevant to support the provision of Integrated Management Services. Where data interchange by either of the outlined approaches is infeasible, interoperability is constrained to a gateway approach, in which service names and attributes can be mapped between the two environments via gateway. In a simple case, two Management Engines use different service definitions and object models and any communication between the two is handled within the gateway.

5.4.4 Management Services Domain

The envisaged Management Service Domain of the framework consists of the services identified for the Management Domain and the Service Support Domain as illustrated in Figure 5.16. The Network Abstraction layer models the service resources whereas the Service Abstraction layer is responsible for the modelling of the overall Management Service. The specification of Management Services can further be subdivided into several categories, depending on the type and scope of a Management Service (e.g. ITU FCAPS management, QoS or Multimedia).

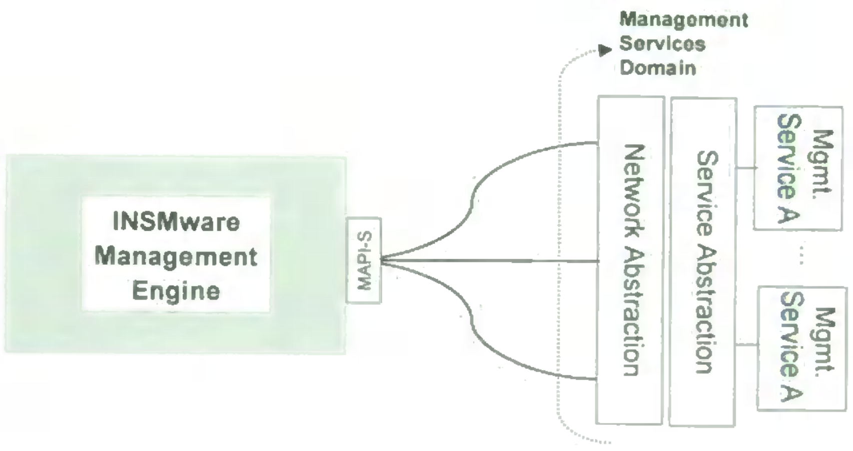


Figure 5.16: Management Services Domain

The Network Abstraction layer groups and abstracts Managed Objects that represent the objects involved in providing the services to be managed. In general, Network Abstraction models the current state of the objects in terms of their connectivity, availability, etc. The Service Abstraction layer groups and abstracts management related service elements with respect to the Management Services (e.g. grouping of Router Management Services).

The interface between the Management Service Domain providing Management Service modelling and definition services and the Management Engine is termed MAPI-S. The nature of this interface is to provide a format and a set of functions for the exchange of Management Service related information. The interface may support the exchange the support of a complete Management Service definition (e.g. for the provisioning of a new Management Service) or a subset (e.g. to modify a particular Management Service). It defines a point of separation between the modelling and runtime environment of Management Services offering the potential to export common Management Service definitions to different Management Engines. Therefore, a common model to define object attributes and their relationships and a common set of function calls to access Management Service definitions are required to allow to exchange Management Service information.

The Service Abstraction Layer and Network Abstraction Layer are responsible for the bridging of differences between service management environment and network management environment and act like a vertical mediation between these two environments. This layer contains the integration between the different object and the technology integration strategies. Different object models represent Service Management and Network Management. The Service Management object model represents more dynamic and complex services whereas the network object model represents relatively stable and less complex but finer grain network objects.

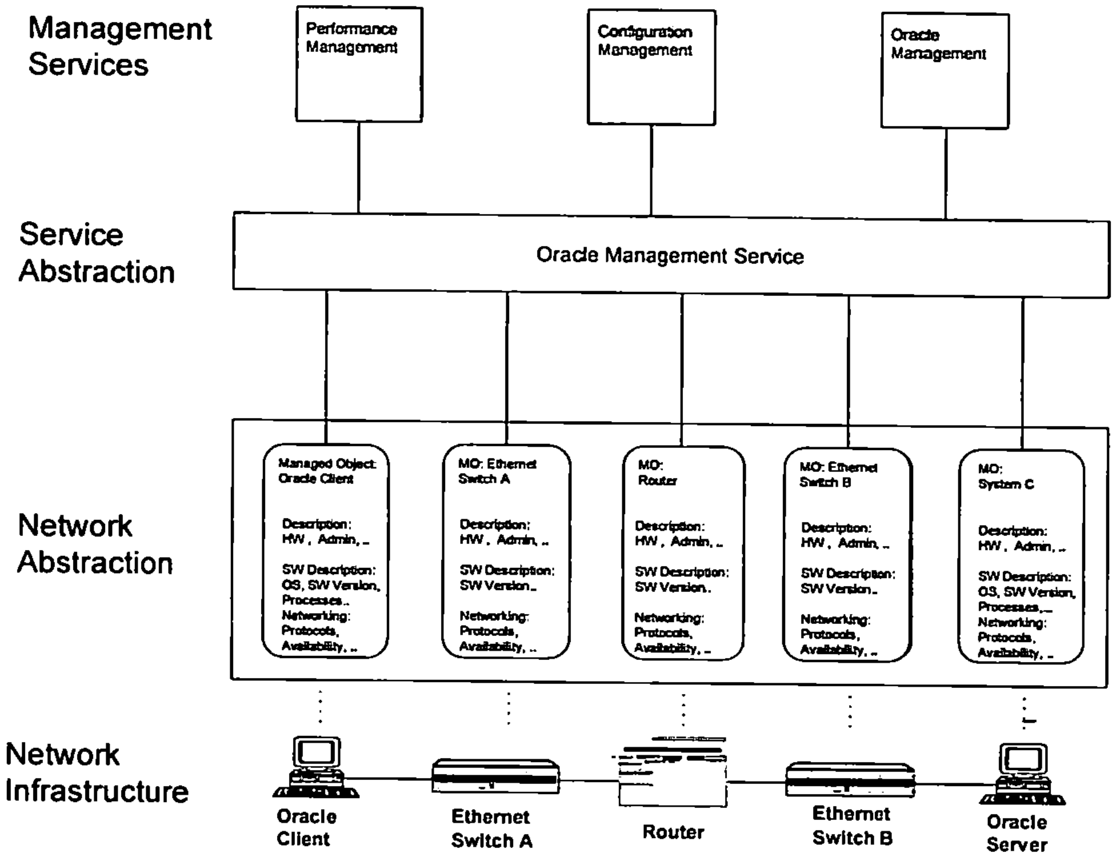

Management Services
Performance Management
Configuration Management
Oracle Management
Service Abstraction
Oracle Management Service
Network Abstraction
Managed Object: Oracle Client
Description: HW, Admin, ..
SW Description: OS, SW Version, Processes..
Networking: Protocols, Availability, ..
MO: Ethernet Switch A
Description: HW, Admin, ..
SW Description: SW Version..
Networking: Protocols, Availability, ..
MO: Router
Description: HW, Admin, ..
SW Description: SW Version.
Networking: Protocols, Availability, ..
MO: Ethernet Switch B
Description: HW, Admin, ..
SW Description: SW Version..
Networking: Protocols, Availability, ..
MO: System C
Description: HW, Admin, ..
SW Description: OS, SW Version, Processes,..
Networking: Protocols, Availability, ..
Network Infrastructure
Oracle Client
Ethernet Switch A
Router
Ethernet Switch B
Oracle Server

Figure 5.17: Oracle Management Service

Figure 5.17 models an example 'Oracle Service' that controls the access and availability of an Oracle database. The overall service consists of a number of Management Services (e.g. Performance Management to control connectivity and performance parameters or Configuration Management to modify the behaviour a given environment). On the Network Abstraction Layer, the entities and attributes required for the service are being modelled.

5.4.5 Managed Objects (Networks, Systems, Applications)

A general model for the integration of Managed Objects is illustrated in Figure 5.18. It consists of distributed Managed Objects that incorporate Management Agents. The Management Agents provide access to management related data from the processes and operating platform that are being monitored.

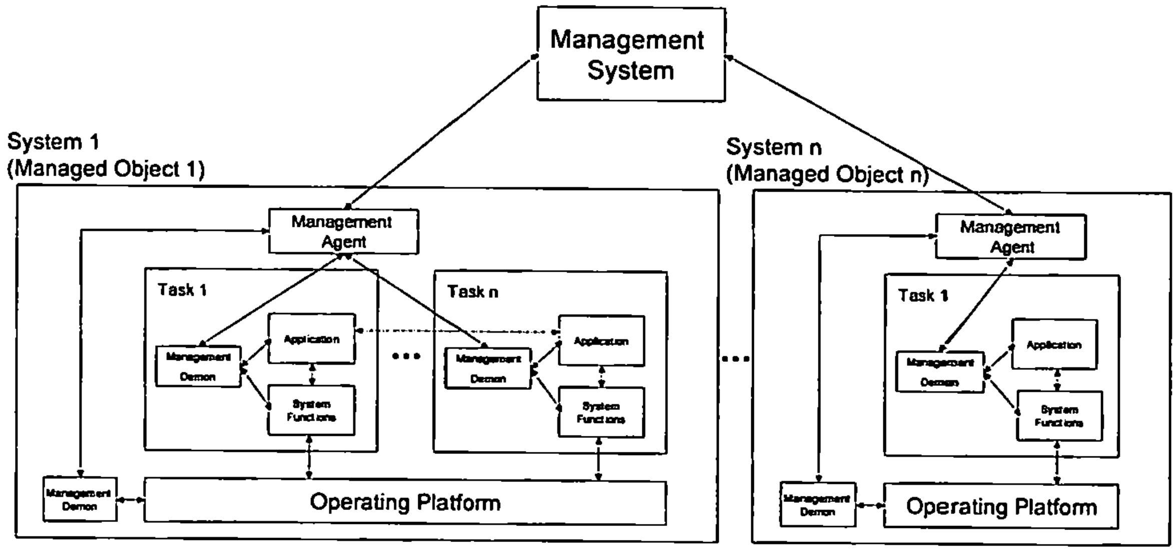


Figure 5.18: Managed Objects

In the generic model provided, the information gathered by a Management Demon that is concerned with a task on a system (e.g. a running application and the related operating system processes) is forwarded to the Management Agent. This Management Agent typically runs outside the other processes. However, in most cases the same Management Agent is responsible for the provision of Management Tasks for a number of different processes (e.g. System 1 in Figure 5.18). The Management Agent then forwards the information to the Management System.

The communication system must support the various management protocols and management technologies used in Managed Objects to enable Integrated Management Services and to access the Management Agents (see Figure 5.18). In addition to an implementation of SNMP and CMIP, an integration of Remote Procedure Calls or other protocols has to be provided to enable the communication with the Managed Objects and to provide Management Services for the infrastructure.

The interface between the Management Engine and the Managed Objects plays an important role in the entire INSMware Framework. This interface is responsible for bridging the differences between the Management Service environment and the Managed Objects. It therefore acts as a mediation layer between these two environments.

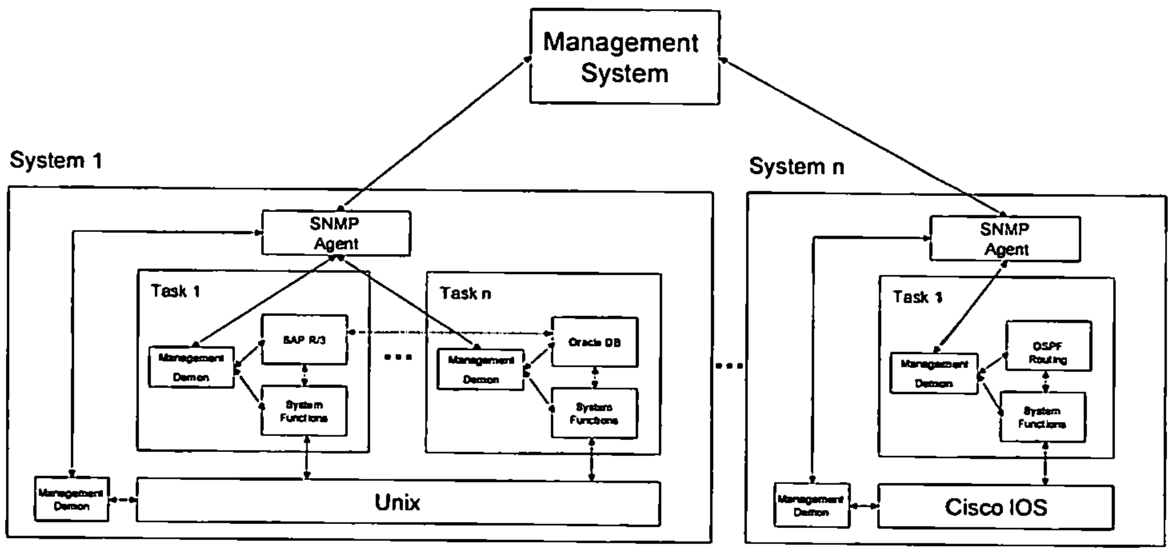


Figure 5.19: Heterogeneous Managed Objects

Figure 5.19 illustrates the integration of heterogeneous Managed Objects into the Management System. The actual Managed Objects operate using different Operating Systems (in the illustrated example Unix and Cisco IOS) and run various processes. However, both systems feed the Management related data into an SNMP Management Agent that can be accessed by the Management System.

5.4.6 Conclusions and further recommendations

The proposed ISNMware Framework can be used as a conceptual model to build Integrated Management Systems. The INSMware Framework provides a template of the functionality, components and interfaces necessary to implement an Integrated Management System. It is intended that the resulting management systems be reusable and extensible. Furthermore, due to the Building Block definition, implementations of packages of the system can be easily interchanged. This approach leaves the way open for later replacements of legacy systems that have been incorporated into the design of a system. However, the range of possible interactions and state transitions is very extensive and lies outside the scope of this project.

5.5 Business Process and Legacy System Integration

The planning, running, maintenance and configuration of an IT infrastructure and services is embedded in the Business Processes of an organisation. The better the management tools integrate with Business Processes the more useful they are from the viewpoint of the IT service provider and hence, the higher the acceptance [Mayerl2001]. Therefore the overall aim is to integrate the Management Services with the Business Processes through the Management Framework as illustrated in Figure 5.20. The co-ordination of these activities requires support for defining and implementing task dependencies, business rules and data across the various domains of a given organisation.

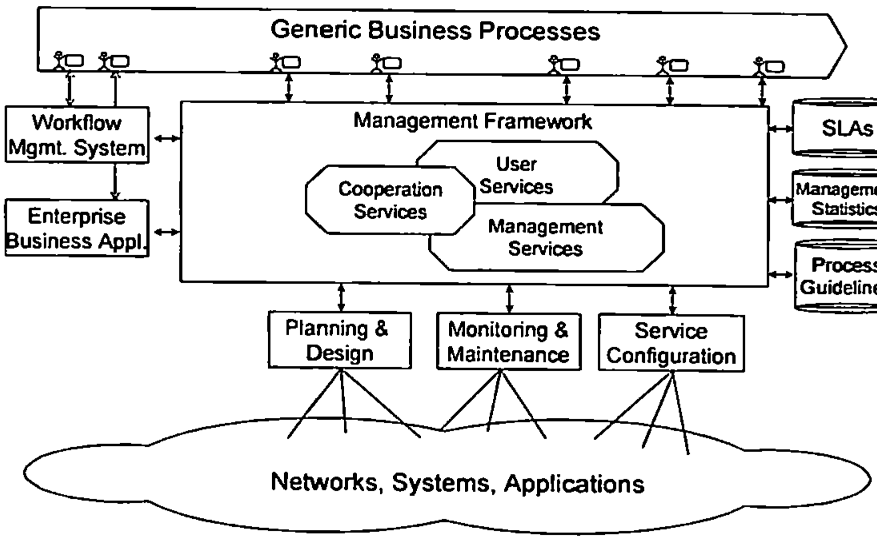


Figure 5.20: Process integration

The Management Services provide the fundamental mechanisms for the management of the integrated IT services. The Management Services provide the required management functions and support protocols such as SNMP or CMIP to access the elements of the IT infrastructure. The User Services support the interaction with Management Actors. These services provide a role oriented view on the operational process and support the Management Actor to perform the required Management Tasks. The Cooperation Services provide the cooperation between the different systems involved in the running of the business processes. The Management Framework interfaces must enable the integration, definition and customisation of Management Services utilising the Process Guidelines Repository, Service Level Agreements and Management Statistics that provides existing guidelines. An implementation of the proposed Management Framework has to support the lifecycles of a network (i.e. initial planning and design, implementation, operation, extension and modification of infrastructure and services).

The proposed Management Framework provides guidance in the complex process of developing Integrated Management Services. The Management Framework addresses the issues and problems involved and provides guidelines to users as well as informative reference material for state-of-the art technologies expected to be used when applied (see also Chapter 5.6). However, every developer and provider of Management Services must be aware of the problem of integration. The requirement for integration of legacy systems might be further exacerbated by the heterogeneity of current infrastructures and the requirement of End-to-End service provision. Because legacy systems integration has such an impact on the architecture of new systems, it should always be treated at the architecture level not left as a deployment detail. Besides, if the integration of applications sharing the same environment is itself a rather broad and challenging issue in the field of Management Services, the integration of legacy systems will be an even more complex task.

5.6 Implications of proposed Management Approach

The following sections discuss major implications regarding the conceptual INSMware Framework. The research has been undertaken at a time of many advances in software technology and the implementation of Building Blocks and other INSMware concepts may be addressed by a wide range of tools and technologies. A number of key developments have a significant impact upon this research project and must be further assessed. Their presentation also provides a reference guide for the area of software technology.

5.6.1 Deployment of Building Blocks

This section identifies and illustrates the major implications for the deployment of the INSMware Framework and its Building Blocks to be taken into account in order to implement a real life management architecture.

The main requirement on Building Blocks is the ability to interact in a loosely coupled manner and to interact via standard interfaces to achieve interoperability. This implies that an implementation of the INSMware system should support different technology bindings for interface implementations via its Building Block Interaction Protocol (e.g. Internet Inter ORB Protocol for CORBA environments, Simple Object Access Protocol for 'standard' Internet access [Box2000] [SOAP1.2-2001] or SNMP for access to Managed Objects). To provide interoperability across technologies, adapters will be required to achieve interoperability between Building Blocks with different contract technologies.

The INSMware Framework further stresses *separations of concerns* as a design foundation for interoperability. Separation of concerns is a fundamental principle of Software Architectures [Orfali1996] [Souza1999]. By identifying the key concerns of an application and functionally separating them to be handled individually, it is widely agreed that many positive benefits related to design, development, and maintenance of software are achieved. However, the envisaged research goes beyond those benefits and advocates industry agreements about the nature of the separations, so that interoperability may become possible. Without clear, widely-accepted definitions of non-overlapping roles for Building Blocks, it is not likely there will ever be a substantial level of multi-technology and multi-vendor "Plug-and-Play' interoperability for the provisioning of Integrated Network and System Management Services.

Building Blocks from different sources should be integrable. Standardisation groups and industry forums have to define and test principles and interfaces for Building Blocks. Ideally, a Management System designer should be able to make a purchasing decision for a Building Block from among a number of suppliers. Thus, a Building Block from one supplier must be able to inter-operate with one from another supplier through its interfaces. A developer must be able to add functionality to a distributed computing network by designing a Building Block that leverages the capabilities of existing Building Blocks from any source.

A supplier of Building Blocks should be able to support the different types of Building Block Interaction Broker that are used. The supplier's Building Block should be able to access the enterprise data of the customer through widely accepted interfaces, alleviating the need to modify the software for each customer. A supplier of Building Blocks should be able to leverage the capabilities of Building Blocks that the customer has purchased from other suppliers, invoking their contracts as needed.

5.6.2 INSMware Design

Before building any structure or system a comprehensive set of plans, for that structure and its components, should first be drawn up. This set of plans illustrates the requirements to successfully complete the structure and the dependencies between the components of it. This is a familiar concept to almost everyone, its application being most apparent in disciplines such as civil engineering or electrical engineering. However, it is equally valid in other engineering faculties. The Software Architecture envisaged by the INSMware Framework provides a functional decomposition in which components and sub-systems are defined and their interaction to fulfil the goals of the system are characterised. Defining and using an established Distributed Systems Architecture in the development of large system carries with it a number of benefits [Jacobson97] [Phippen2001] [Weber1998]. Furthermore, Distributed Systems Architecture aim to facilitate system designers and developers in comprehending large systems by expressing the system's high-level design in a clear manner.

The distributed technologies that are being used to support Management Services are also evolving and enabling interoperability between distributed applications. This middleware technology is of vital importance for the performance, availability and reliability of Management Services and the research has based its work on the assumption that such middleware technologies are available. Technologies for the realisation of Componentware based systems (e.g. OMG's CORBA, Sun's Enterprise Java Beans or Microsoft's DCOM) support connectivity between components in a distributed environment running on multiple platforms. Such distributed processing technologies provide a unified modelling paradigm for developing applications and potentially enable management applications providing different Management Services to be integrated into the general purpose distributed environment and to take advantage of general purpose support services. By assuming the existence of such middleware support, designers of Integrated Management Systems can concentrate on the functionality of the Management Services they are developing rather than become involved with details of the underlying communication and interoperability mechanisms.

Reuse in the context of the INSMware Framework is envisaged in two ways. First, a software architecture itself can be reused. Second, Software Architectures can support and facilitate the reuse of common components if the roles and responsibilities of the components are clearly defined and documented. Architectural descriptions separate the concerns of the functionality of components from the ways in which components interact. This allows the components to evolve without unduly impacting their interaction with other components.

The INSMware Framework defines and describes, at the conceptual level, generic concepts regarding Building Blocks of an Integrated Management System along with dependencies, interactions and communications between these Building Blocks. After the Analysis and Modelling phase of the INSMware methodology that will be presented in chapter 6, the respective Building Block Model should be mapped to an Software Components.

5.6.3 Generic Principles

The INSMware Framework has two main roles in the design and implementation of the Integrated Management System. Firstly, at the initial design phase, a system planner can immediately see from the Framework, those Building Blocks that he may be able to reuse from any existing infrastructure and those that must be implemented from scratch. Secondly, the INSMware Framework allows for the meaningful gathering of modelled system Building Blocks and shows how these are to interact and the dependencies between them. This proves invaluable as the designer follows the Methodology to create the required management system.

The implemented INSMware Architecture provides a research prototype based on Software Components that provide 'real-life' implementations of the INSMware Framework Building Blocks. It can be considered as a set of base functionality like generic component communication that enable an Integrated Management System to run in a flexibly configurable, heterogeneous, distributed processing environment. INSMware is intended as a layered structure of distributed components, where a higher layer makes use of the functionality provided by a lower layer. As such, INSMware at its lowest layer consists of a number of Software Components that may have to run on different systems in order to ensure portability and to support heterogeneous environments. This allows, for example to operate a component like a simple Front-End Component on a low cost PC while a components with higher requirements can be executed on a pool of workstations. On top of the Operating System layer, a number of embedded standard middleware products will realise a substantial degree of functionality. These products must be available on all intended operating systems. The Middleware Abstraction Layer basically integrates the products and provides an interface and generic services for the upper layers (e.g. IDL for a CORBA based implementation). The development effort for this layer depends on the usable functionality of the embedded products and in particular on their support for a pure Componentware based architecture. This kind of wrapping allows the provision of access to components that should not influence the architecture. This is especially useful if components are used that might possibly be replaced or reengineered in the future. The performance penalties of such a wrapping can be kept low if forms of component adaptation carrying small overhead are introduced [Hofmann2000]. The next higher layer implements the core INSMware Software Components and adds the required management functionality to the system.

5.7 Summary

This chapter specified the conceptual design for a new Integrated Network and System Management approach, INSMware. It sought to discuss the form, appearance and capabilities of the proposed management framework given by the conceptual design. Equally, the purpose, form and utilisation of the design environment has been described as a means of enabling the realisation and development of the INSMware framework within the management provision environment. The INSMware Framework is intended to last longer than applications that were developed using it. Therefore, the generic principles and domains of the framework were specified in a way allowing room for future additions. The INSMware Framework provides the context for Integrated Management Services and, in conjunction, it contains essential input to the Development Methodology and to the INSMware Architecture. These results should be regarded as complementary to the research and design work that has been part of models and implementations to date. The next chapter moves on to discuss how the conceptual design and the design environment can be realised to provide a system which benefits the provision of Management Services. In Chapter 7, a system based on the identified concepts is proposed and designed that is then prototypically implemented and evaluated.

Chapter 6

Realisation of the Management Approach

"Forget programming and objects as you know them. The future is giant Lego blocks, the kind a toddler can handle" [Sarna1994]

This chapter provides an overview regarding the realisation of the INSMware conceptual principles and management provision scenarios. The various aspects and benefits given by the INSMware approach are commented on with respect to the provision of Management Services and the characteristics of the envisaged Distributed Management Architecture

The impact and leverage of distributed systems technology is prevailing not only for design and implementation of user applications and services but indeed also for the deployment of management systems. Thus far, management systems have typically been of two categories. Either specialised along one dimension (e.g. vertically targeting one or a few management aspects such as configuration or performance management) or horizontally dedicated to management of a specific layer (e.g. management of network elements). Hence they have resembled monolithic systems based to a large extent on proprietary solutions. The presented research uses contemporary concepts to leverage a modular approach to design management systems based on the Building Block principles, thus facilitating openness and extensibility on one hand and adaptability (e.g. distribution of Management Services) on the other.

Distributed object-oriented systems represent the logical development of the object model supporting the distribution of Building Blocks to physically distinct locations [Orfali1996] [TMF-

OM2000] [Zitterbart1997]. Distributed System Architectures, also referred to as middleware architectures, such as CORBA or DCOM form the basis for the development of distributed object-oriented systems. Providing a mediation layer for distributed object communication can be enhanced by the provision of a framework to support reuse. In this context, the Building Blocks for the INSMware Architecture will be assembled using *Software Components.* They represent physically immutable pieces of software that can be assembled with other components to form new, more complex components or even entire applications. The latter is referred to as *Componentware* and forms an architectural basis for the development and operation of component-based systems.

6.1 INSMware Management Provision

The previous chapter defined the conceptual design of the INSMware Framework and a design environment of both its initial realisation (independent of implementation approach) and its continuing evolution. However, it did not give any feel for the development, form or use of the system from a user perspective, and thus, did not demonstrate the true benefits available to management provision from the proposed approach.

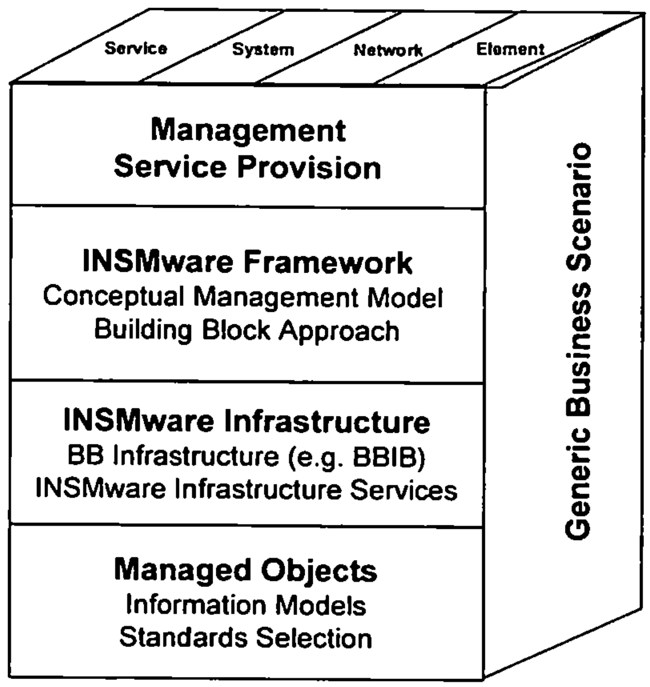


Figure 6.1: ISNMware Layers

Figure 6.1 illustrates the layers addressed by INSMware (i.e. collections of processing functions that together comprise a set of rules and standards for successful Management Service provision). The INSMware Framework offers a conceptual model for an application architecture providing the means to group related functions into Building Blocks that use the underlying infrastructure in a regulated way. On the highest layer of the model are the Management Services that support generic business processes. Below this layer are the layers addressing the INSMware Framework and INSMware Infrastructure principles that are utilised to implement the Management Services. The problem is how to provide deployable units of Management Building Blocks that can inter-operate with other Building Blocks, perhaps developed by different Building Block providers independently from one another at different times in different organisations.

The INSMware Framework concept of Building Blocks deployment on an enterprise level facilitates distribution, software reuse, security services and other functions. Distribution requires Building Blocks to communicate with each other over electronic networks (such as local area networks or the Internet). Modelling aspects and the functions of a distributed, Building Block based Integrated Network and System Management framework have been identified and will be further expanded upon to prove the relevance to build a novel management framework using Software Components.

The presented approach aims to provide a conceptual framework for the provisioning of Integrated Network and System Management Services. It can be seen as a part of an enormous effort in research and industry towards the integrated management of all resources in a distributed system. The frequently cited OSI Management Framework that categorises the management tasks into the five classes Fault, Configuration, Accounting, Performance and Security Management only gives a functional decomposition of this multidimensional problem. Another relevant problem dimension are the resources (i.e. managed objects) within a distributed system upon which the management functions are applied. In the past, mainly the logical communication resources (e.g. communication protocol entities) and the physical network components (e.g. repeaters, bridges, routers) have been addressed by the standardisation bodies and industrial products. Consequently, the term network management is commonly used to cover these aspects. In the meantime, hardware and software resources of the end-systems (e.g. disk space, CPU time of applications) are included in the management view expanding the network management to systems management.

Beside the integration of established management procedures and managed objects, the proposed framework requires functionality to be instrumented in order to enable the monitoring of distributed applications - one step in the direction of application management. However, purely isolated component-oriented management is not what the end-user requires. Integration of management solutions with the aim of a single architecture that may cope with all type of resources is demanded. For instance, by means of commercially available network management applications network operators get a management view upon the network resources and this helps to answer questions about the connectivity and availability status of network devices and end systems or the load on the network segments. Additional integration is necessary in order to close the gap between the applications and the network to establish integrated Management Services (e.g. which application components are running on which end systems, what is the communication behaviour of the applications or how much load is produced by the individual applications). To gain this knowledge is currently very tedious, as the mapping relations between applications and network resources is often changing, possibly without the knowledge of the network operator. The situation will become even worse when object migration technologies or automatic fault recovery procedures will be widely used. What is required is a common and integrated view upon all resources of a distributed system on the network, (operating) system, middleware and application levels.

6.2 INSMware Development Methodology

This section introduces the INSMware Development Methodology for an Integrated Network and Systems Management Architecture that provides development principles and suggests a number of steps and tasks that should be taken into account during the various phases. The methodology provides guidelines to structure the development process of the new management approach in the analysis, design and implementation phases.

The proposed Methodology consists of building an initial model of a system and then adding further implementation and process details to it during the design of a system. The proposed methodology emphasises the iterative nature of the development process and the pre-eminent place of the Building Block definition within it. Representing Building Block at an analysis level as well as at a design level enables Building Block reuse to become much more central to the analysis process and to become directly relevant to the business process reengineering activity. Given a Building Block Interaction Broker that a set of Building Blocks can "plug into," the underlying goal is to agree upon a set of principles which will create a sense of uniformity among Building

Blocks on as many issues as is useful and practical for a given environment. Building Blocks should be deployed in the same manner in any given environment.

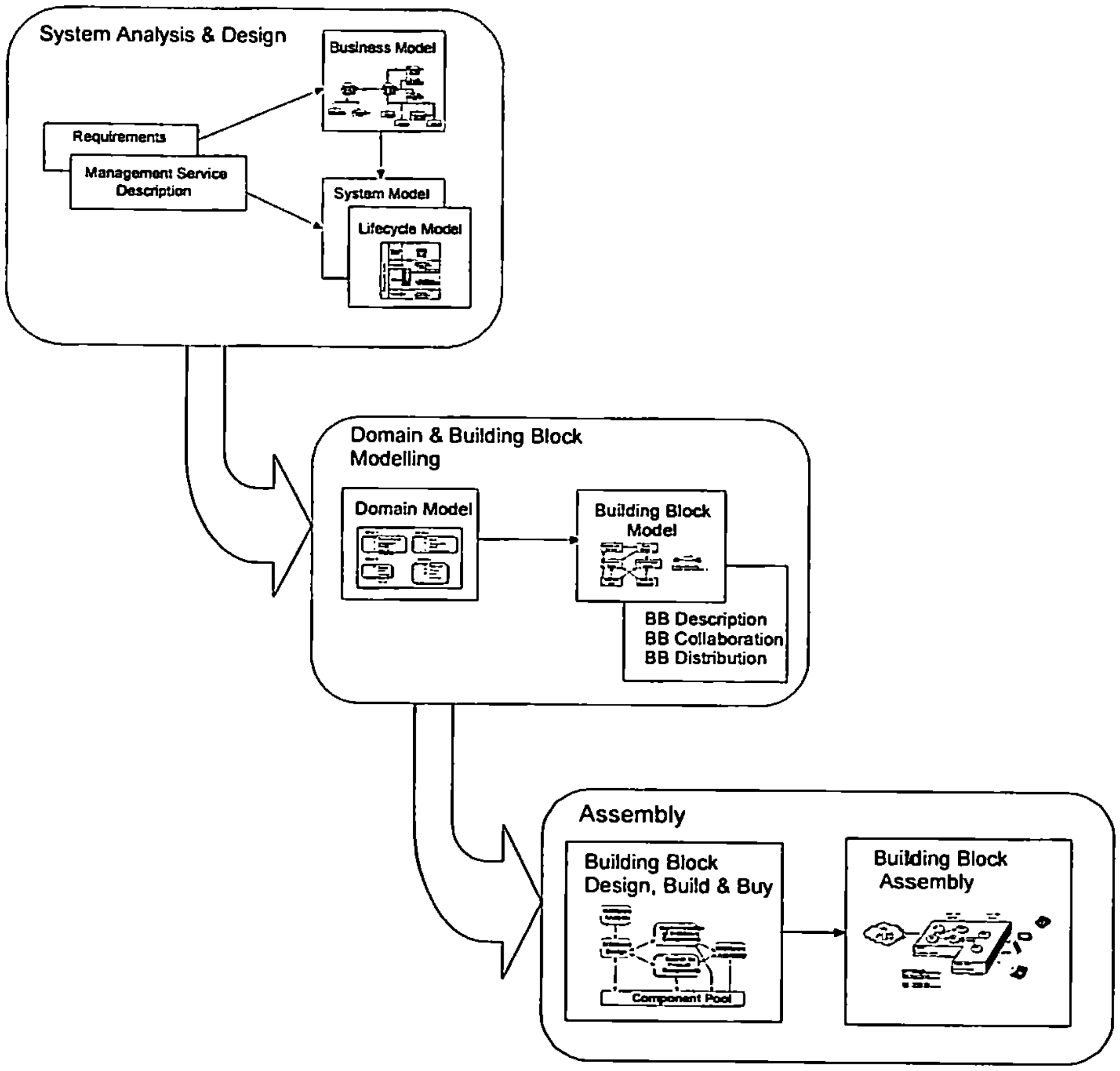


Figure 6.2: Overview of INSMware Methodology

The main activities of the INSMware Development Methodology are the capturing of requirements, specification of Management Services, system modelling and system assembly (as illustrated in Figure 6.2). It consists of three phases:

- System Analysis and Description;
- Domain and Building Block Modelling;
- Assembly of Management Architecture.

The phases are further subdivided into separate activities. The main activities of the design and implementation process are the capturing of requirements, specification of Management Services, system modelling and system assembly as illustrated in Figure 6.2. Each activity may be influenced by, or take as input the results of another activity. The major relationships between the different activities are represented by arrows. However, the Methodology process is iterative (all activities may be repeated several times or the entire methodology may be repeated as a whole) and some phases can be carried out in parallel.

The INSMware Methodology has been assessed after the proposal of the INSMware Management Framework. It provides an approach to encompass aspects regarding Management Services and the related organisational environment. The results of the implementation activities should be kept aligned with further development in the fields of development technologies and follow the telecommunication standards and developments with a focus on existing and new networking services.

A basic assumption of the proposed management framework paradigm is that it is layered on top of existing systems and services in order to provide core integration services. The user plugs in some combination of management Building Blocks (i.e. the most appropriate subset) that applies to the specific mix of required Management Services. The proposed framework with open interfaces and a shared user interface, allows the customer to build an integrated yet customisable management environment.

Given the complexity of modern networks and services, it is necessary to adopt a structured approach to their design, construction and management [Lewis2000] [Schmale1999]. A key part of this is the development of well-founded, structured methods that provide a reproducible and reliable solution to the overall problem. The creation of methods, algorithms and techniques is usually not the concern of the user, more a specialist activity carried out for system development.

6.2.1 System Analysis and Design

The main aim of the analysis and requirements phase is to identify the requirements and the overall context of the Management Architecture. In this phase the following activities should be carried out:

- Description of Management Services;
- Identification of the different stakeholder and Management Actors;
- Identification of system lifecycle;
- Description of the Architecture.

The description of the required Management Service defines the basic management issues to be solved and identifies the nature of the Management Services. The aim of the Management Service Description is to give a concise overview of the problems and the core concepts. The initial Management Service Description should contain enough detail to enable the extraction of the then following activities. Business requirements and general management requirements (such as the ones included in the Framework) may already be included in the Management Service Description.

The INSMware Business Model identifies the different stakeholders and Management Actors and specifies their roles and their relationships within the overall Management System. The Management Service Description is the main input for the Business Model. Furthermore, the Business Model can be used to determine the interactions associated with the different Management Actors. The Business Model provides an organisational context for the Management Services described in the Management Service Description.

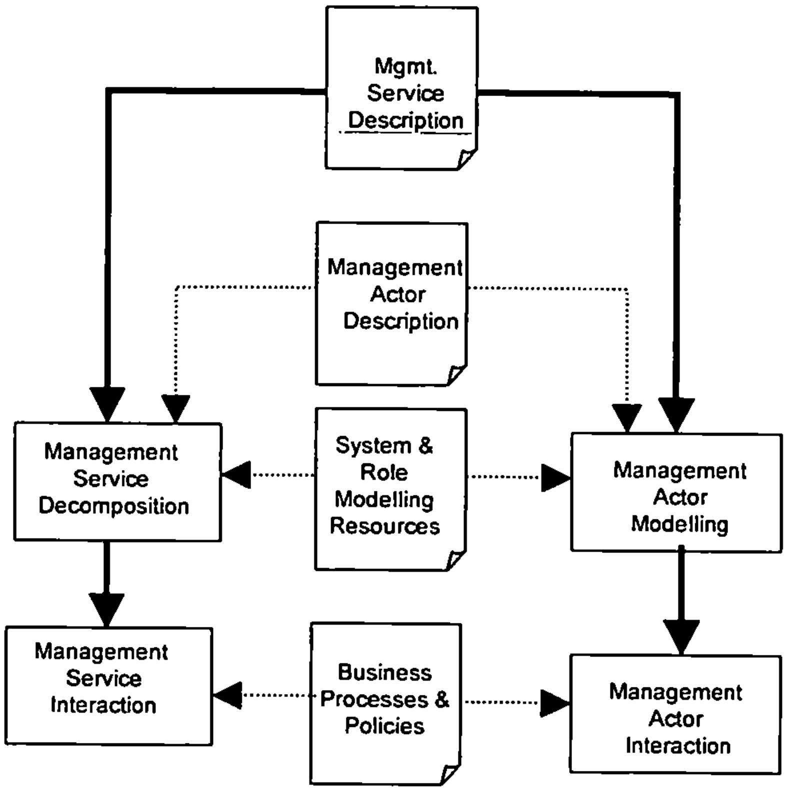

Mgmt. Service Description
Management Actor Description
Management Service Decomposition
System & Role Modelling Resources
Management Actor Modelling
Management Service Interaction
Business Processes & Policies
Management Actor Interaction

Figure 6.3: INSMware Business Modelling

For the modelling of Management Actors, the Management Service Description can be used as a main input. Additional resources such as existing descriptions of Business Processes and Policies or Management Actor descriptions in conjunction with investigations regarding the interactions between the relevant actors and other existing resources can be used to obtain a precise model. These Management Actors and their roles form a starting point for Management Actor modelling using the concepts described in the INSMware Framework. In addition, the interactions between the Management Actors in the Business Model must be described. An example flow of the Management Actor Modelling activities is shown on the right hand side in Figure 6.3. The Business Model can further be used for the refinement of the envisaged Management Services (e.g. establish organisational boundaries or context of Management Services).

A lifecycle defines the series of developments that a system undergoes in the course of its progress from an initial planning stage to a final termination state. The envisaged INSMware Lifecycle Model must identify the set of functional stages (e.g. Management Service Utilisation stage) and areas (e.g. Management Service Provider perspective). It is further used to identify the relations between Management Actors and Management Services with respect to the different stages and areas of Management Service provisioning.

From the Management Service Description and the Business Model, the views of the System Model can be derived. These views serve as descriptions of the different ways of using the system from the viewpoint of the Management Actors and are specifying the different roles. From the System Model and Service Description, instances of the System Models can be derived that identify sequences of tasks. The System Model provides a description of the Management System in terms of the Management Actors and their roles, the interactions between them and the Management System.

Typically a number of iterations between the activities will occur both at the initial modelling phase but also further on in the design process.

6.2.2 Domain and Building Block Modelling

The main aim of the Domain and Building Block Modelling phase is to identify a distinct set of domains and Building Blocks that contribute to the same goals (with an emphasis on aspects such as coupling and messaging between domains and encapsulation of functionality). The two key activities in this phase are Domain Analysis and Building Block Modelling.

Domain Analysis for the provision of an Integrated Management System is the process by which information used in developing system components is identified, captured, and organised with the purpose of making them reusable when creating a new systems. Domain Analysis can be regarded as the activity that will identify objects and operations of a class of similar systems in a particular problem domain [Jacobson1999].

The Domain Model provides a representation of the domain problems that need to be addressed. Through this model the information needed to develop a specific service in a specific domain can be obtained. The Domain Model can be something that already exists, in its whole or just in part, available to be reused. Otherwise, after analysis has been completed, potentially reusable Domain Classes can be identified and then the next time that a Management Service that requires a particular same domain is being developed, those reusable Domain Classes may be taken into account.

Domains will represent important parts of the overall organisation and infrastructure containing the different resources that provide a common set of functions for the provision of a Management Service. Domain Analysis allows the integration of the reuse process in the software production life cycle and facilitates a systematic, formal and effective practice of software reuse. It is through Domain Analysis that domain knowledge can be transformed into generic specifications, designs and architectures for reuse in developing new systems within the domain. The identification of Domain Classes to be reused and can benefit from the usage of existing management models (e.g. TMN or TeleManagement Forum) [TMF-OM2000] [ITU1992].

The Building Block Model describes the assembly of Building Blocks for the provision of Management Services within a Domain Model's scope. The Building Block Model may be built taking into account the Domain Classes that could be reused (e.g. as a result of Domain Modelling). Starting from the requirements capture in the System Model analysis, other models and refinements of existing ones are added to the model, refining associations between them.

The following activities can be applied for the Building Block modelling phase to support the design process:

- Building Block Model, which identifies Building Blocks grouping objects and domains of the system for the provision of Management Services oriented on the Domain Model;
- Building Block description to further specify the functions of the identified Building Blocks;
- Identification of static and dynamic collaboration description illustrating the connectivity and interaction among Building Blocks;
- Description of Building Block distribution scenarios illustrating distribution aspects.

These specifications can be used to identify required interactions, methods and attributes of the design to further refine the Building Block Model. The System Model is further refined and all the activities are repeated until an entire architectural model for the provision of Management Services has been obtained.

6.2.3 Assembly of Management Architecture

This phase is related to a specific application domain of Integrated Network and System Management following the principles of the INSMware conceptual model. The main concepts to take into account are utilisation of existing components and the adoption of an appropriate packaging notation. Once the Building Block Model has been built, the following tasks can be undertaken:

- Identify Building Blocks that can be reused. Identified Building Blocks that are part of the Architecture, may appear in other architectures (e.g. IETF or TMN based Management Architectures or other application domains) and can be reused.
- Develop Building Blocks.
- Group and assemble Building Blocks according to the specified management scenario.

- Integrate Legacy System with the INSMware system.

The author of this research suggests the usage of the Unified Modelling Language (UML) to describe and support the implementation of INSMware Building Blocks [Jacobsen1999] [Souza1999]. The INSMware Framework provides the main concepts and domains to be considered in developing an Integrated Management System and informs the phases of the methodology. To assemble the operational system, the Building Block Model should be mapped onto a 'real-life' environment.

6.2.4 Conclusions and further recommendations

This section presented a development methodology that can be used in conjunction with the INSMware Framework for the specification and development of Integrated Management Systems. The results from the methodology provide an initial framework for the design of an Integrated Management Architecture. Hence the INSMware Management Architecture will evolve as the result from the design cycle. It will be evaluated and relevant additions can be subsequently incorporated into the Architecture, thus providing openness towards future requirements.

The development methodology is suggested as an area where a common approach can be of great benefit to developers of open Management Systems. Adoption of a common modelling notation will ease the communications between different stakeholders in the development of open Management Systems and the components they reuse [Lewis2000] [Schmale1999]. However, to further align the INSMware development process with related development principles, general principles of System Modelling approaches and notations such as the UML could be further adopted for the modelling phases of the proposed methodology [Lewis2000]. The Unified Modelling Language integrates the concepts of Booch [Booch1996], Rumbaugh [Rumbaugh1991] and Jacobson [Jacobson1995] into a common and widely used modelling language [Jacobson1999]. The use of a structured and well documented development approach that has found widespread use and acceptance can be advocated as it offers a coherent approach for system development [Lewis2000] [Phippen2001] [Souza1999].

6.3 INSMware Characteristics

The INSMware Framework is intended to facilitate distributed computing with potentially large numbers of Building Blocks interacting concurrently as part of a potentially wide-area distributed computing network to form many concurrent, overlapping logical systems. As the complexity of the network increases, both the planning issues and the variety of possible abnormal scenarios grow excessively. Automation of tasks in normal and in failure recovery scenarios is needed to reduce the burden on human administrators and to maximise reliability.

Building Blocks of a distributed management system are logically separated from one another and can be physically separated from one another depending on the requirements of a given Management Service. They can be placed, for example, on the same system, on different systems in the same network or in different networks (different locations). The execution of Management Tasks and other processing operations can operate where it is required to (e.g. reflecting a particular infrastructure or organisational requirements).

A distributed Management Systems can consist of a great number of Building Blocks. Transactions that are managed during the execution of a Management Service (such as polling of a large number of managed objects) have a beginning and an ending state, and various intermediate states. So a polling procedure could have the local state "management data received" in local segment (e.g. a segment in Darmstadt), whereas it has the state "processing polling procedure" in a remote segment (e.g. a segment in Plymouth).

Aside from the addition or removal of Building Blocks, Managed Objects as well as Management Actors in a INSMware system can also change their physical location (e.g. when a department of an organisation moves or when a Managed Object is mobile). Furthermore Building Blocks may migrate to other locations in a distributed system because of performance or security issues. However, a user should be unaware of the fact that the location of a Building Block he is using has changed. Hence, a runtime re-configuration of a system (i.e. dynamic re-configuration) is necessary if the system is not to be shut down with each addition or removal of a Building Block (i.e. "Plug-and-Play" for dynamic re-configuration). It can be anticipated that the system architecture will change continuously because of he addition and removal of Building Blocks and the evolution of distribution and interaction technologies.

INSMware Building Blocks may be used simultaneously with other Building Blocks. This parallel processing of tasks in a distributed system can increase performance significantly over non-distributed applications. However, when distributing tasks over different software Building Blocks, the communication overhead has to be considered. As communication over long distances can cause delays in processing, an asynchronous execution of different tasks by quasi-parallel processing can save time. However, an asynchronous execution of different actions is only possible up to a fixed level because the results of a set of elemental actions are required in order to perform higher-level actions. Furthermore, a plan of execution sequences has to be drawn up to minimise the total task processing time.

The INSMware Framework makes no requirements regarding the internal programming methods used by the designers and developers of Building Blocks. However, it is intended to act as an enabler for Componentware based development, in which pre-tested components are composed into applications by means of open interfaces to enable their interactions.

At the inter-Building Block level, the clients are intended to find their needed services dynamically by means of requesting services via the INSMware Building Block Interaction Broker (i.e. further decreasing the level of coupling between Building Blocks). One purpose of this concept is to establish a clear boundary between the arena of 3rd Party development and the arena of End-User development and usage. A Building Block is an encapsulated entity (e.g. a deployable unit of computer software) that cannot be directly modified by the End-User (e.g. customer using a Building Block from a 3rd Party Provider or Management Component Provider). It can only be adopted by means of flexibility features built in by the provider of a Building Block. On the other hand, a logical system of Building Blocks is extensible by the customer. Although a single vendor may provide all of the Building Blocks for a particular system, the customer or another vendor can supplement the given system by adding a new Building Block, or replacing an existing one, with one of their own design. This would be difficult if all the Client-Server interactions of the logical system were predefined.

Transparency refers to the hiding of implementation details of separate Building Blocks from the system. It should make no difference to the user whether he accesses the services of local or remote Building Blocks and the mechanisms that are necessary for accessing remote Building Blocks should be transparent to the user. The location of Building Blocks of a distributed system and whether a Building Block has been replicated should be transparent to other Building Blocks and is usually of no interest to their users. The way of processing a specific task (e.g. sequential or parallel) and whether one or more Building Blocks are responsible for processing a task should be transparent to the user.

Individual INSMware Building Blocks should be customisable. Hence, Building Block providers must provide the features to enable the level of required adaptability. Adaptability must cover behaviour customisation of Building Blocks and must enable the integration of additional Building Blocks.

6.4 INSMware Development Principles

"The ideal of component-based development is that application development becomes an assembly process, built on substantial reuse of standard components. In theory, more than 95% of an application can be based on reused software." [Kiely1998].

In the INSMware management approach, every interaction takes place between a client and a server. Any single INSMware Building Block interacts with any other in one of these two roles. Because a Building Block exists to provide a service (server role) which usually then requires it to act as a client with underlying servers, nearly all Building Blocks are acting in both roles concurrently with many parties.

Since the early inception of distributed management, a number of related technologies tried to provide a uniform and ubiquitous platform for building distributed management solutions [King2000] [Patel2002]. The most relevant of those technologies include OMG's CORBA [OMG1998] and Microsoft's DCOM [Eddon1998]. Since the requirements associated with INSMware represent a large scale distributed system it is valid to consider its mapping onto these distribution services.

A key motivation for incorporating these distribution technologies together with the Componentware approach in INSMware is to enable and facilitate the implementation and integration of Building Blocks. INSMware is conceived as a distributed management framework that consists of Building Blocks that use the distribution services of a generic Building Block Interaction Broker. Distributed Systems Architecture, such as CORBA, provide such a framework and it is likely that it will be used in the future for supporting distributed management applications and advanced telecommunications services (e.g. those conforming to the emerging TINA infrastructure) [King2000] [Lewis 2000] [Patel2002].

Software Components based on these distribution technologies represents a relevant approach in software development [Hofmann2000] [Phippen2001]. Numerous sources extol the virtues and discuss the implications of component based development, and propose it as enabling technology and prerequisite to enable software engineering to became a true industrial process (e.g. comparable to the production of cars) [Butler1998] [Forrester1996] [Kiely1998] [Phippen2000] [Zedan2001].

The concept behind component-based development is straightforward – a Software Component represents an encapsulated piece of functionality that is reused at a *binary* level. This means that the reuse of each Software Component is implementation independent – one of the primary differences between component-orientation and other software reuse techniques. The evolution of software systems to component-orientation moves software development from continuous development of new applications through writing of new code towards the assembly of systems from reusable components. This, in turn, should result in an increase in development productivity, as a greatly reduced amount of the system has to be written with original code.

In order to understand the proposed perspective it is important to be aware of the fundamental principles upon which the proposed management solution is based. These reflect current trends and technologies and are considered as essential for future developments. The convergence between Network, System and Service management, in terms of common platforms and techniques, is an essential principle adopted in the proposed INSMware Framework. Even though the managed entities obviously present distinct intrinsic characteristics and properties, the representation of the entities serves as a common ground for management. Management Services can be outsourced as well as being federated. The Management Architecture must be technology independent. At a time of rapid change in networking technologies, IT services need to run over a variety of different network types. It is therefore a fundamental principle of the research that the Architecture must be applicable to any network technology and IT environment where integrated Management Services are required.

6.4.1 Component and Componentware Paradigm

"The best way to attack the essence of building software is not to build it at all. Package software is only one way of doing this. Program reuse is another. Indeed, the promise of easy reuse of classes, with easy customisation via inheritance, is one of the strongest attractions of object-oriented techniques" [Brooks1995]

"It's very risky to expect that emerging industry standards, like CORBA, DCOM or Java RMI middleware, will automatically eliminate the complexity of developing ... software" [Schmidt1999]

Software reuse has long been held up by IT analysts as a way to increase the productivity of software development [Booch1997]. Prior to component-orientation, other techniques (modular programming, object-orientation) have been put forward as the way to achieve large-scale software reuse, but have arguably failed to provide the anticipated gains. Component technologies are the latest candidate to facilitate large-scale software reuse and transform software development into an industrial process [Szyperski1998]. Within the INSMware development process, component technologies were used effectively to incorporate functionality between components within systems and between systems. The technologies provide the potential for straightforward software reuse, but they do not automatically make software reuse easy [Hofmann2000] [Phippen2001].

It seems certain that component technologies have a future in software development. The market forces and the requirements from the industry behind component technologies are so strong that "progress" will be hard to resist [Kiley1998] [Butler1998]. Supporters of Component technology claim that the use of this approach will result in a simpler, more productive development process. However, Hofmann, Schmidt, Phippen and others suggest a more cautious stance and stress that software development is not just a collection of development technologies [Hofmann2000] [Phippen2001] [Schmidt1999]. However, nearly every entity in software development can be called a component in the sense of being a constituent or a part of a more complex entity. Such an understanding would also cover units such as classes.

A definition for Software Components can further be based on a hardware-software analogy that also emphasises the criterion of immutability. Taking the use of system components in electrical engineering as an example, it can be seen that electronic components are immutable (e.g. a Central Processing Unit). It is not possible to change a Central Processing Unit's internal logic or to modify its internal structure. However, they can be integrated with other electronic components. This is also referred to as *black-box reuse* [Gamma1995]. The advantages of black-box Software Component reuse are:

i. complexity of development is reduced since the functionality encapsulated inside a component is entirely unknown to its user;

ii. development time is reduced as recurring development tasks are implemented by self-contained components;

iii. development costs are reduced since components can be reused;

iv. product quality is improved since components are pre-fabricated, pre-tested units.

The following definition incorporates the requirement of immutability and also establishes the requirements of integration and configuration for Software Components and will be used throughout this thesis:

Definition 6.1: Software Component

A Software Component is an immutable package of software with one or more well-defined interfaces that is integrable and configurable.

The existence of well-defined interfaces and the generic capabilities regarding configuration and integration enable a Software Component to co-operate with other Software Components. This also applies to the deployment of Building Blocks in the context of this research. Configuration concerns the setting of parameters to change a Software Component's behaviour while integration focuses on component interoperability. Immutability is an outer criterion describing the physical shape of a Software Component. Hence, Software Components provide an enabling technology for the implementation of Building Blocks.

Immutability as a key criterion for Software Components can be achieved by physical encapsulation. Entities in computing which are immutable are binary files, such as executable files or dynamic link libraries. This suggests the view of Software Components as a packaging technology (i.e. containers for source code) [Hofmann1998].

Even though single Software Component can be regarded as an isolated entity (e.g. in terms of development or deployment), *"it must be adequately prepared to socialise with a population (or populations) of components in the context of an enforced etiquette"* [Lycett2001]. The term Componentware is widely perceived as a generic term for software being built by using Software Components. Therefore, definitions of the term Componentware only show rudimentary differences that are mainly caused by dissimilar interpretations of the term Software Component. Eddon defines Componentware as *"software composed of Building Blocks"* [Eddon1998]. This definition obviously focuses on a software hardware analogy. The 1995 Ovum report presents a similar, but more precise definition:

> *"Software designed to enable application elements to work together that were constructed independently by different developers using different languages, tools, and computing environments." [Ovum1995]*

Though distributed architectures such as CORBA or DCOM provide features for the realisation of Componentware, distribution is not inherent in the Componentware model. Hence, Componentware may also be used in a homogeneous, non-distributed system environment. An example for such an application is Microsoft's ActiveX technology [Denning1995] that can be regarded as a desktop precursor of Componentware.

The following definition will be used to describe Componentware throughout this thesis:

Definition 6.2: Componentware

Componentware is a constitution of co-operating Software Components.

As with INSMware Building Blocks, Software Components will collaborate with other Software Components for the provisioning of services. To provide a Componentware based implementation, an additional Componentware Infrastructure is required offering distribution services to the Software Components. Hence, Componentware provides the means of implementing the conceptual INSMware Framework.

6.4.2 Distributed Systems Architectures

The basic principle of Distributed Systems Architectures is to enable system functionality and data storage to be broken up into components that may be located in any part of a computer network [Orfali1996] [Zitterbart1997]. Sharing the processing load across a number of computing resources has many advantages, including the potential for improved performance, availability and reliability. The distributed systems architectures [Weber1998] are currently being developed and exist on the market. However, the most relevant architectures are the Object Management Group's (OMG's) Common Object Request Broker Architecture (CORBA), Microsoft's Distributed Component Object Model (DCOM) and Enterprise JAVA Beans from SUN Microsystems [Chen2000] [Phippen2001] [Pryce2000]. Several definitions can be found in literature that describe distributed systems:

"A distributed system consists of autonomous subsystems co-operating in a coordinated manner to fulfil a common task" [Geihs1995]

"A combination of heterogeneous hardware, software and network components, in which the Software Components reside and execute on two or more of the hardware components, communicating and interacting using a network" [Freestone1996]

The complexity of the management processes does not decrease in distributed systems, instead it can well increase due to heterogeneity of co-operating systems, new possible fault scenarios regarding communication, consistency, security problems and others [Geihs1995] [Dornan2001]. However, Freestone suggests that the benefits from system distribution can be massive [Freestone1996]. The key benefit of distributed processing comes from the modularity of distributed components, which is natural for integrated network and system Management Services.

There also exist specific approaches and extensions of distributed technology tailored towards the provision of Management Services (e.g. to be applied to TMN based management). TINA-C's Distributed Processing Environment (DPE) is specially created for the operation and management of telecommunications networks [TINA-SA1997]. In ITU-T, the work in this area is focusing to the application of principles of the Open Distributed Processing to the OSI reference model to create the Open Distributed Management Architecture (ODMA) [ODMA1997]. In the past decade, CORBA has emerged as the most commonly used and propagated Distributed Systems Architectures in different industry domains. This also applies to the telecommunications and networking industry. Although TINA-C network and service architectures was tailored towards communication service provisioning based on the DPE, most of its global demonstration showcases and R&D efforts were developed by using CORBA as Distributed Systems Architecture [Trigila2002] [Vandermeulen2002]. TeleManagement Forum's Technology Integration Map also advocates CORBA as the base technology to integrate multiple TMN management functions and applications [TMF1999] [TMF-OM2000]. However, it is not the sole technology and other approaches such as DCOM are also subject to research and development in this area [Adamopoulos2002].

For the implementation of INSMware Building Blocks using Software Components, Distributed Systems Architectures must provide the following:

- General purpose distributed computing environment and distribution services;

- High-Level interface description methods to specify access to the functions of INSMware Building Blocks;

- Integration platform for heterogeneous systems and services that are not necessarily developed for a particular Distributed Systems Architecture.

Software Components must be able to work together and interact over heterogeneous environments. This implies that an implementation of the INSMware system should support different technology bindings such as Internet Inter ORB Protocol (IIOP) for CORBA environments and should consider protocols such as Simple Object Access Protocol (SOAP) for the provision of Web-Services based on 'standard' Internet access methods [Box2000] [SOAP1.2-2001]. It is interesting to note that SOAP cannot directly be compared with CORBA. SOAP provides a protocol for the access of distributed objects for TCP/IP based environments and corresponds to IIOP for CORBA environments. Thus, it seems more appropriate to compare IIOP and SOAP (i.e. IIOP is only a part of CORBA).

Item	**Web Services**	**CORBA**
Protocol	SOAP, HTTP, XML	IIOP, GIOP
Location Identifiers	URLs	IORs, URLs
Interface Specification	Web Service Definition Language (WSDL)	Interface Definition Language(IDL)
Naming, directory	Universal Description, Discovery and Integration (UDDI)	Naming Service, Interface Repository, Trader Service

Table 6.1: Web Services and CORBA

Table 6.1 illustrates the corresponding parts of Web Services and CORBA. This technology mapping illustrates the different aspects involved in each approach that affect the implementation of a system and the provided functions. Currently, a lot of interest and emphasis (i.e. some analysts refer to hype when they analyse the current market situation) is going towards SOAP and Web Services rather than more established technologies such as CORBA. It also has the support of many key players on the software market (e.g. Microsoft's ".net") [Platt2002]. However, Web Services are basically a reinvention of CORBA based services and is currently lacking a full set of standards (i.e. the first SOAP specification appeared in 2000 whereas CORBA 1.0 was specified in 1992) [Cerami2002].

The aim of this research is not to promote any underlying technology environment. However, it is important to state that they form the basis of the practical work of the research. The practical part of the INSMware project focuses on the specification of higher-layer Building Blocks that are using the services of an underlying middleware technology. The most advanced and influential middleware technologies are CORBA, DCOM and Enterprise Java Beans. CORBA and DCOM are briefly described in the following sections.

6.4.2.1 CORBA

Since 1989 the Object Management Group (OMG) has been working to create standards for object-based component software within the framework of its Object Management Architecture. The key component is the Common Object Request Broker Architecture (CORBA) [OMG1998]. CORBA defines interoperability between objects in heterogeneous systems. The OMG specifies a common architecture, including a common Object Requester Broker (ORB), allowing interoperability between vendor middleware across dissimilar platforms.

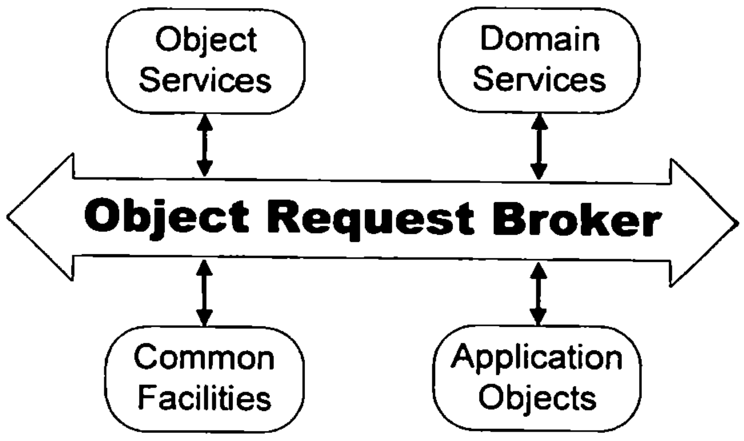


Figure 6.4: The OMG Object Management Architecture

The OMG Object Management Architecture (OMA) is the architectural basis of OMG's standardisation efforts since it defines both a communication infrastructure for distributed objects and a variety of services of different granularities necessary for application development [OMG1997]. Figure 6.4 shows the structure of the OMA with its backbone, the Object Request Broker (ORB) [OMG1998]. The CORBA specification defines the ORB and its associated structures. The ORB mediates requests between clients and object implementations (servers) and enables Software Components to communicate and interact regardless of their underlying operating systems, the programming languages that were used to build them, and their physical location. Thus it supports transparency of access, location, migration, resources, and connections. Because of its similarity to a hardware-bus (e.g. as used in PC technology), an ORB is sometimes referred to as an object bus [Hofmann2000]. It also provides a high-level Interface Definition Language (IDL) to specify the interfaces between Software Components.

6.4.2.2 COM/DCOM

As it does in other areas, Microsoft proceeds to define and propagate its own set of standards in the area of Componentware. Microsoft's initial Object Linking and Embedding (OLE) technology came into being in 1990 to provide a cut-and-paste capability for Windows. When OLE became OLE2 for more general communication between Windows applications, its new communication model was given a separate name, COM, which was said to stand for either Component Object Model or Common Object Model. Components in C++, Visual Basic and other languages that drove applications via OLE automation came to be called OLE Controls. At this stage COM began to be promoted as a general-purpose infrastructure for building component-based software applications. Distributed COM (DCOM) and COM+ are extensions of COM that support distributed object invocations and system independent services to simplify the development and deployment [Eddon1998].

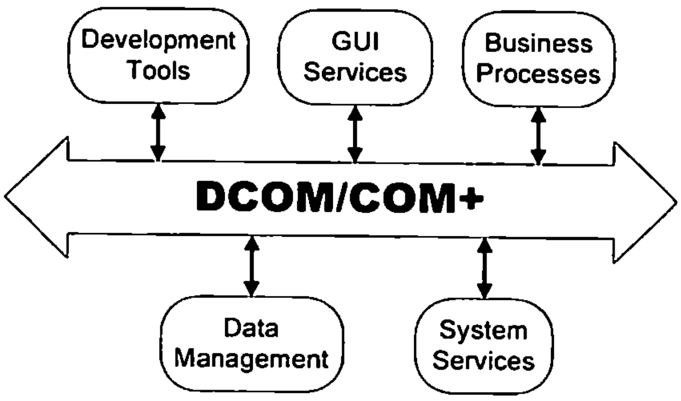


Figure 6.5: The Microsoft Distributed Architecture

DCOM/COM+ form the backbone of the Microsoft Distributed interNet Application Architecture (DNA) and allow distributed objects to communicate by using their facilities [Microsoft1997] [Souza1999]. Figure 6.5 shows the structure of the DNA. Development Tools include tools for HTML and script authoring, for the development of distributed objects, for rapid application development (RAD) and for team development while GUI Services provide mechanisms for GUI representation and user navigation. Business Processes include back-end technologies such as Web server technology, transaction management, message queuing, and scripting. Data Management and System Services represent a group of lower-level services. The former supports different mechanisms of data storage, ranging from flat files to object-oriented data storage and the latter includes services such as security and the management of distributed objects.

6.4.3 Using Components to Enable Integrated Management Services

It has been shown that existing frameworks and implementations do not meet the identified requirements due to architectural limitations. The implementation technologies used require the development of new management applications that operate beside existing applications if new networking technologies and products have to be incorporated into existing management systems. The usage of Software Components in an Integrated Management System that interact via middleware technology such as CORBA or DCOM will allow the reuse of existing components and the integration of new components into the management system.

The system has to provide distributed management capabilities to meet the organisational and functional requirements. Management Services, such as Network and System Management, have to be integrated into one Management Architecture that provides interfaces for the required Management Services, data storage and a common Graphical User Interface (GUI). The system has to be able to integrate new services and new technologies in a practical and economic manner.

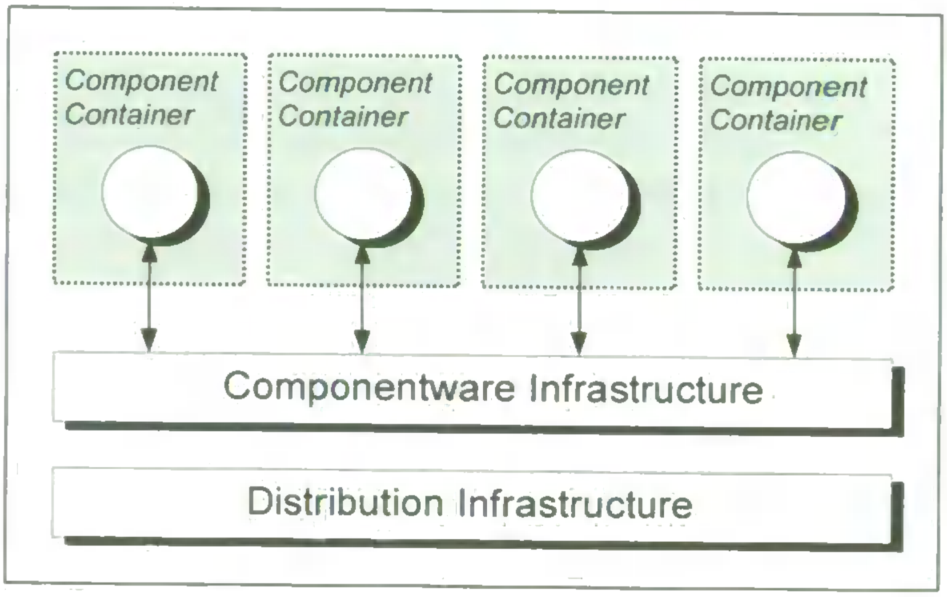


Figure 6.6: Componentware Architecture

The suggested system is based on the Componentware Architecture as illustrated in Figure 6.6. Software Components have to be combined in a Componentware architecture that consists of several elements providing different services for the realisation of component based application building. Each Software Component is embedded in a Component Container and uses services of the Component-Infrastructure. The Component Infrastructure for its part uses services of the Distribution Infrastructure. The Component-Infrastructure offers mechanisms for component management or component communication. The Distribution Infrastructure incorporates the communication and distribution features of different middleware technologies (e.g. CORBA or DCOM).

Using a component based approach for the implementation of a novel management system offers several advantages. Componentware provides open, flexible and modifiable application structures that can be adapted to the identified requirements. It allows the reuse of existing components that offer Management Services and reduces redundant application functionality through integration of different management disciplines into one management system. The usage of middleware technologies allows a highly flexible and distributed management solution to be built.

As technology and requirements evolve, Building Blocks in the form of components are added, interchanged and upgraded while the architecture remains stable. The architecture life cycle is equivalent to the system life cycle and transcends the life cycles of individual component subsystems.

6.4.4 INSMware Development Context

One of the overall interoperability goals is to use Software Components as Building Blocks and their design principles. The existence of reusable Building Blocks can potentially lead to a market for Building Blocks. These Building Blocks can be regarded as *Common Building Blocks*. The concept of *Common Building Block* finds its usefulness because of possible Building Block reuse across multiple organisations sharing some similar or identical elements of business processes (i.e. management related processes in the context of this research).

These Common Building Blocks will exist in two categories. Some Building Blocks, such as those that provide management data warehousing activities, are useful across different environments. These are referred to as "Horizontal Domain" Common Building Blocks. Other Building Blocks (e.g. a Building Block that performs SNMP management in a Router) are useful broadly within a particular type of Management Service or within a single application domain (e.g. for Network and System Management). These are referred to as "Vertical Domain" Common Building Blocks. Beyond these types of Building Blocks, there may also be *"Private Building Blocks"* which are proprietary Building Blocks most often specific to a single enterprise.

Regarding Integrated Network and System Management, common Building Blocks will emerge to fulfil requirements based on analysis of the requirements and business processes. Within a given logical system, a service provider should be able to purchase the Building Blocks from different vendors on a mix-and-match basis, purchase some portion of the Building Blocks and develop the others, or develop the entire logical system in-house. High levels of interoperability and flexibility are the keys to making such an approach work. However, an organisation requires guidance in what functionality is provided by Componentware technologies and which technologies will best suit their purposes. This requirement is well served by industrial IT research companies (for example Forrester or Ovum) and domain specific industry and research consortiums (e.g. TM Forum) that all produce relevant literature (e.g. [Butler1998], [Brown1998]).

However, choice of product and platform, while important, is only one of a number of questions an organisation should address when contemplating the adoption of Componentware techniques [Phippen2001].

Others have expressed this conclusion more explicit:

"So, the answer to the question 'why are components important?' is simple: money. The promise of saving it in your own developments, and making it if you're a small outfit aiming to make a splash selling pre-fabricated Software Components. But remember that tools aren't magic wands. Reuse isn't a place, but more a state of mind." [PCWeek1997]

In a market of Software Components, different stakeholders will have different roles and requirements [Lewis2000]. It is necessary to identify and highlight the relationship of stakeholder involved in the development and operation stages of a Management System to put the INSMware approach into a market context [Lewis2000] [TMF-OM2000]. The INSMware Development Context Model identifies the different actors and their roles within the INSMware development process. It explicitly identifies a Management Component Provider to acknowledge the Componentware based development approach.

The following roles have a significant stake in the INSMware development and are tailored to aid in the identification or requirements for the development of an open INSMware Architecture (see Figure 6.7):

- INSM Service Provider: A role that provides Management Services to a Management Service Customer.
- 3rd Party INSM Service Provider: A role that the INSM Service Provider must collaborate with in order to provide the required service to the customer (e.g. provision of additional management data).
- Management Service User: A role that uses Management Services provided by an INSM Service Provider.
- INSMware Developer: A role that undertakes the development of the INSMware System required by a service provider to manage its services.

- Management Component Provider: A role that develops management components (e.g. to be sold on the market or distributed free of charge).

- Standard Provider: A Role that develops standards relevant for the provisioning of Management Services and system development (e.g. regarding Network and System Management protocols or relevant development platforms).

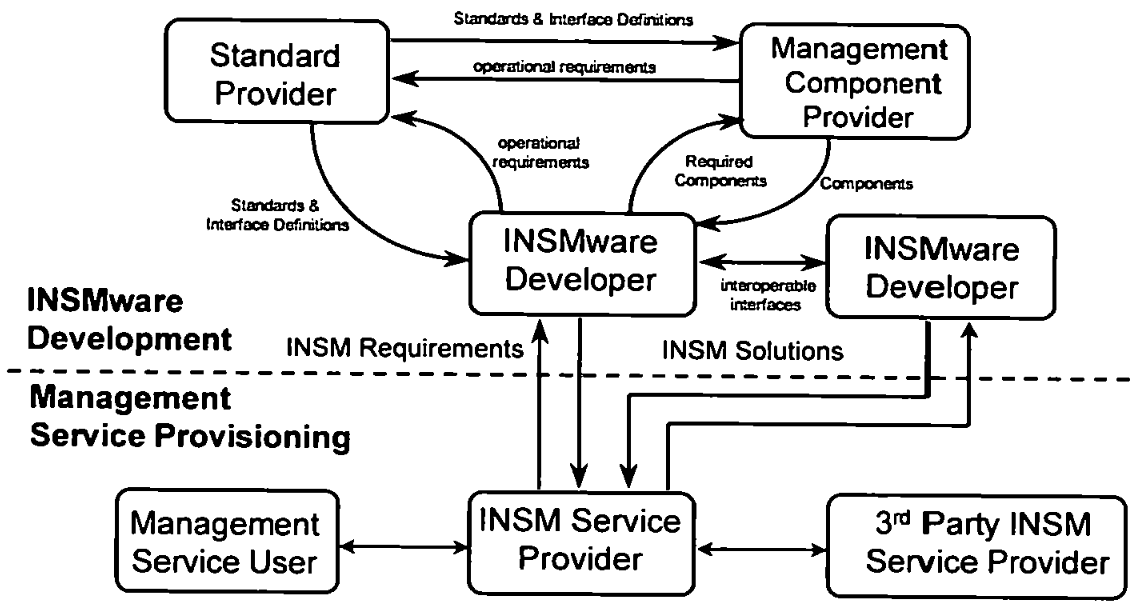


Figure 6.7: INSMware Development Context Model

Figure 6.7 identifies the relations in the overall development context. However, further connections that are not identified in the model might exist (e.g. a Management Service User or INSM Service Provider may make direct requests to a Standard Provider or a Management Component Provider).

One of the main aims of the INSMware Developer is to increase productivity and profitability (e.g. reduce development and maintenance costs) in order to secure continued development contracts from an INSM Service Provider. With respect to the INSMware Developer a number of limitations and considerations must be taken into account (e.g. limited start-up resources for a new venture, limited development capabilities and market requirements that may result in limited capability of initial release). However, it is required to provide full interoperability to gain market acceptance (e.g. ensure that small INSM Service Provider systems can interact with large Service Provider systems).

The INSMware Developer develops the management system for the INSM Service Provider. An INSM Service Provider may select different INSMware Developers for the provisioning of Management Services. The model separates these roles in the anticipation that the development, integration and deployment will be increasingly independent.

The INSM Service Provider role is concerned with the provisioning of Management Services. Hence, its main concerns are the provision of Management Services as well as the planning and design of new Management Services. A small INSM Service Provider may only require a limited number of Management Services and small number of users that need to be supported. Therefore, a small INSM Service Provider typically only requires a limited set of Management Services (e.g. only one Management Actor and no requirement for distributed services). On the other hand, a large INSM Service Provider may want to have a system offering a 'complete' set of Management Services.

The INSM Service Provider is the main source of functional requirements for the INSMware system that is developed by the INSMware Developer. These requirements may include adherence to specific open standards as part of an overall business strategy. However, these requirements are also shaped by the relationships the INSM Service Provider has with other INSM Service Providers. Relationships with these stakeholders may also include agreements to adhere to open interface standards for operational interaction at the Management Service level. The current lack of inter-domain Network and System Management standards and the lack of integrated products that support such interactions implies that implementation of interoperable interfaces between INSMware Developers must be addressed. This relationship introduces the requirement for the INSMware Developer to interact with other developers to provide interoperable Management Services to the INSM Service Provider.

Software reuse is widely seen as a key approach in meeting the challenges of developing software within cost and time constraints [Phippen2001]. One of the most promising techniques is the reuse of commercial off the shelf Software Components developed by a third party, here modelled as Management Component Provider. With Software Component reuse, the costs of developing and maintaining the INSMware Software Components falls on the Management Component Provider, who recoups it by selling the component to as many customers as possible, the customer in this model being the INSMware Developer. In addition, the INSMware Developer can also practise software reuse internally by reusing solutions to frequently occurring problems, or by implementing standard management solutions popular with Management Service Providers.

The Standard Provider will develop standards to establish new technologies and as a response to requests from the other roles. The Service User may ask for open service management interfaces that enable it to move between INSM Service Providers easily, thus encouraging competition between them. INSM Service Providers may ask for management interface standards to ease the implementation of third party relationships or to meet regulatory interoperability requirements. The INSMware Developer is motivated to support these standards, as they will promote the reuse of software that implements them between different management development projects. The Management Component Provider will also have management related standards since they reduce the risk of investing in the development of management Software Components due to improved reselling options. Both the INSMware Developer and the Management Component Provider will also have an interest in using standards that promote software portability and integration (e.g. DCOM or CORBA).

This model therefore emphasises the need for open standards to underpin a competitive market both in the delivery of Management Components, Integrated Management Systems and the delivery of Management Services. An approach to the design of an Integrated Management Framework that takes into account by the requirements of standard providers as well as system developers rather than being mainly driven by architectural and technological considerations of a particular (and specific) environment and will potentially result in more powerful and integrable management solutions.

6.5 INSMware Distribution of Management Services

The heterogeneity and sheer complexity of management provision requires the distribution of INSMware Building Blocks. Furthermore, a management system based on INSMware principles can be split into a number of Building Blocks that can run on different systems and transparently connect to another system's Building Blocks on other systems. The Componentware paradigm provides a basis for the implementation of INSMware Building Blocks. The functionality that a Software Component exposes via its interfaces can be reused by another Software Component. As a result, the entire network appears to be one large computer with enormous processing power and capacity. Thus it can be concluded that general purpose functionality, such as network monitoring, event correlation have to be implemented by a component which can then be used by the different Management Services.

6.5.1 System Performance

As technical environments affect the performance of a distributed system, dynamic models have to be developed that consider performance issues. The system-technical parameters that influence system performance for management related processes are:

(1) Implementation and algorithmic aspects;

(2) Communication capacity (e.g. network bandwidth);

(3) Number of clients per server component;

(4) Component location.

Parameter 1 affects both distributed systems and local, monolithic applications. Implementation or algorithmic performance problems can be tackled by code optimisation and smart algorithms [Sedgewick1992]. Parameters 2 and 3 are particularly important for distributed systems because the communication of Building Blocks making up an INSMware implementation is not done through local mechanisms, but via remote communication channels. As many distributed objects can share one single network, performance can only be determined by testing. If, for example, Building Blocks of a distributed system are communicating via the Internet, time-specific performance issues have to be considered. At first sight, parameters 2 and 4 seem identical as the network bandwidth used to access an INSMware Building Block is directly related to its location. But, for example, *mobile Building Blocks* could also be part of a distributed system.

Irmscher discussed the problems of mobile clients in distributed systems [Irmscher1997]. In addition to the CORBA trading concept, Irmscher introduces the concept of *mediators*. A *mediator* mediates between a mobile and static server objects. This includes the mediation of data between mobile and fixed regions, handling of transmission errors between client and server objects, continuance of execution after intentional communication interruptions, and caching of server data. This concept could be adapted to Building Block mediation.

6.5.2 INSMware Component Distribution

Beside the organisational and technical requirements for the distribution of management tasks, using distributed software technology enables the provision of a new management methodology rather than one monolithic management system for the distribution provides further advantages.

A large amount of components and information required to provide Management Services is distributed over different systems and cannot be managed by one single, central administrator. Furthermore, a single administrator will not have the knowledge about the structure and the relevance of the overall Management Services. So when the components of an infrastructure are distributed, management tasks for them should be distributed (i.e. decentralised) too. Hence, flexible and autonomous management is required. One scenario could be that a central system administrator has a general, global view of a system, and can view and manage system parameters whereas a local administrator could only view and manage parameters within his fixed and limited component environment. Furthermore, the local user of a Software Component can usually perform local management tasks easier and more efficiently than a central administrator. For example, a local administrator can easily reboot a computer or change the toner cartridge of a printer. On the other hand, a central administrator will monitor and configure the important communication links of a network. The separate systems can be used for different tasks involved in a Management Service (e.g. event correlation and data storage).

The realisation of distributed software systems is made possible by the increase of low-cost bandwidth on Wide Area Networks, for example the Internet, a new generation of network-enabled desktop operating systems and middleware technologies such as CORBA and DCOM that enable the distribution of Software Components. The possibility to access remote resources (e.g. via the Internet) enables software developers to distribute their Software Components, even over the boundaries of a corporate network (see Figure 6.8).

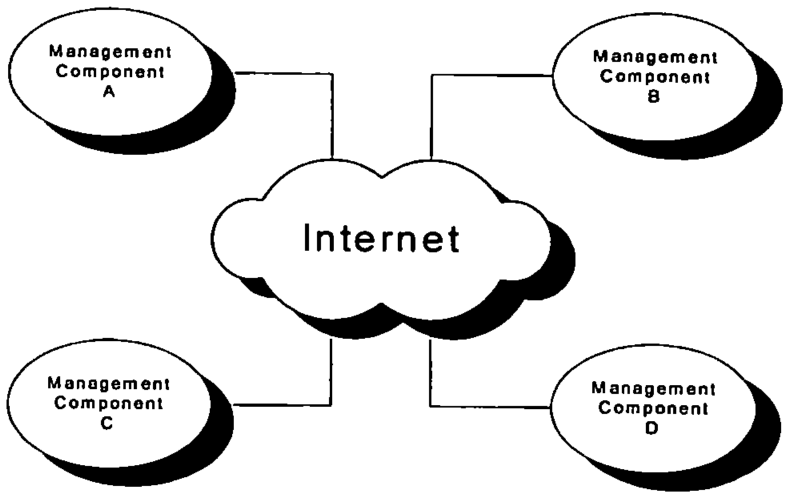


Figure 6.8: Distribution of Management Components over the Internet

The underlying middleware technologies (e.g. CORBA or DCOM) allow the distribution of the Software Components in the Network. One example could be the installation of a Management Interface Component at a remote site. There, the managed objects will be monitored locally by the Management Interface. Another local Filter Component could then analyse the management data and decide whether or whether not to forward the data to the corresponding Software Components in the Management System.

The connection between different components of a distributed management system is software-technically (logically) realised through middleware that implements the Building Block Interaction Broker and hardware-technically (physically) realised via communication lines (i.e. a communication network). Therefore, the *logical* and *physical* connections between distributed components must be differentiated. A *distributed component transaction* is a transaction in which more than one Software Component participates. The *logical connection* is directed to represent responsibilities and communication characteristics of components. The *physical connection* is undirected and merely represents the physical distribution of components. A Physical *Connection*

Graph can be used to show the connection between distributed management components. A Software Component is represented by a node and a connection is represented by an edge.

Figure 6.9 illustrates five management components that are distributed over two systems (System 1: $\{C_1, C_2\}$; System 2: $\{C_3, C_4, C_5\}$). There is one physical communication path (e. g, an Ethernet connection). Two *marshalling nodes*, C_{M1} and C_{M2}, have to be introduced to the physical connection graph that provide the communication and distribution services (e.g. using the TCP/IP protocol stack) [Emmerich2000]. Additional marshalling nodes have to be added to the physical connection if more than one communication path exists.

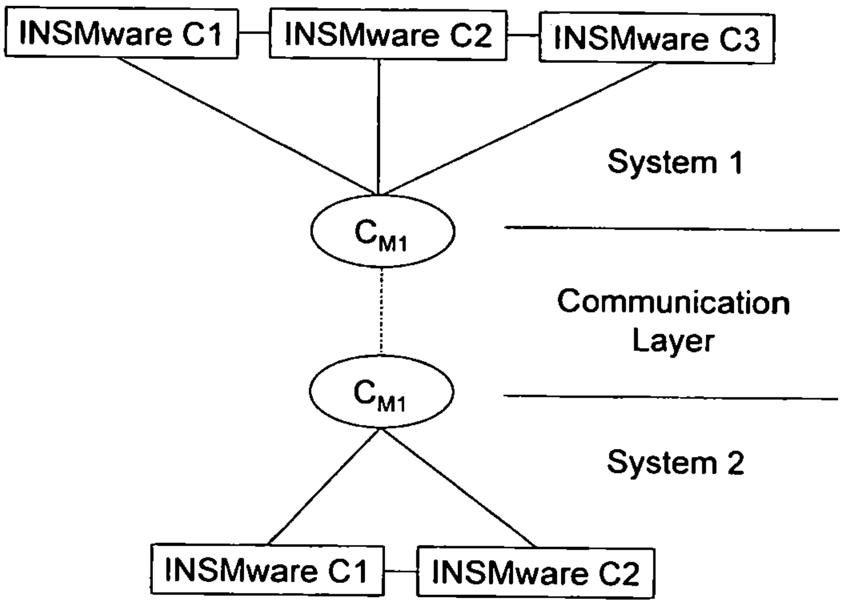


Figure 6.9: Physical Connection Graph

When, for example, the edge $\{C_{M1}, C_{M2}\}$ is disconnected, the components located on one machine can only communicate with local components, not with components located on the other (remote) machine. The edge $\{C_{M1}, C_{M2}\}$ represents the physical connection between machine 1 and 2 (e.g. an electronic network) and is a connection-critical edge.

Reactions on upcoming technical system requirements can be done by re-configuring the system (e.g. by configuring a component or replacing a component by another). Also, the connection between components or the location of components can be changed in order to configure the system. By having the capability to re-configure a system by configuring or changing its components, the necessity of building a new system because of changing requirements is decreased (i.e. facilitating configuration in conformance with system requirements).

The risk of a system overload or a failure of single Software Components can be minimised by a successive system extension. By systematic replication of Software Components, the likelihood of failure is diminished (i.e. risk minimisation).

Only few Software Components may be affected by a system failure or system crash. The other Software Components can still work autonomously. When using local data caching or data replication, a system can work with local copies of normally distributed data (i.e. autonomous operation).

6.5.3 INSMware Operation Principles

The proposed Framework provides a model where Software Components can be deployed dynamically for distributed management. INSMware can consist of a number of distributed Management Entities (each Management Entity provides a single or a set of management components that communicate using the middleware infrastructure). These management entities form a network of peers over which hierarchies or cooperating sets of management components can be run.

Nodes on which Management Components should be deployed must be instrumented with the appropriate execution environment [Feridun1999]. Management Systems consist of one or a set of management components that carry out management tasks of different levels of complexity within the overall INSMware model. The services required for the distribution and communication (i.e.

Installation and Distribution Services) as illustrated in as shown in Figure 6.10 are either provided by the middleware infrastructure or must be incorporated into the INSMware implementation.

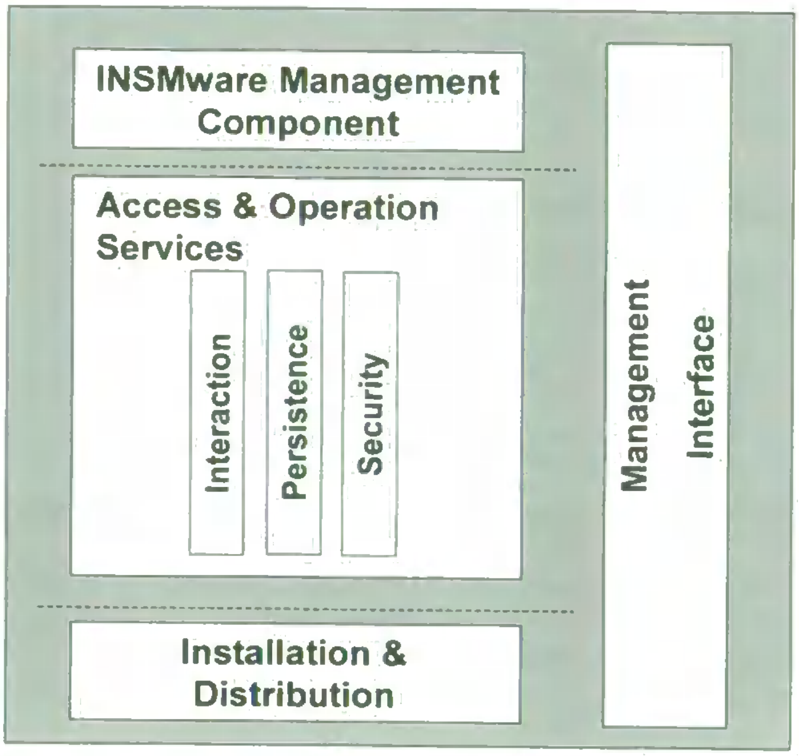

INSMware Management Component
Access & Operation Services
Interaction
Persistence
Security
Installation & Distribution
Management Interface

Figure 6.10: Distributed Management Entity Architecture

The purpose of the Installation and Distribution services is to facilitate and expand the operation of a system (e.g. perform installation of a downloaded Software Component) and to reduce the requirements on the Management Entity (e.g. in terms of CPU performance and memory requirements).

The Access and Operation Services provide the essential services for the Management Components. The Interaction Service provides communication between the management components. This communication can be either local (i.e. Management Components are installed on the same Management Entity) or remote (i.e. Management Components are installed on different Management Entities). The Persistence service allows Management Components to be stored in the persistent storage and the stored components to be reloaded back into memory. This service can be used after system crashes, where persistent parts of the Management Component can be reloaded into memory after reboot or to save system resources (i.e. components that are not used are being moved into persistent storage and reloaded as necessary). Management Components may keep their internal states including the results of the completed management operations. The aim of Security Services is to provide secure communication and transport between Management Components and between Management Components and Managed Objects.

The Management Interface enables access to services and components available on the Management Entity. Furthermore, it provides access to tools provided by the local operating system that can be used for management (e.g. a tool such as *ping* can be used to test availability of particular systems in an IP network). It may also initiate dynamic installation & removal, configuration and registry of Management Components.

6.6 Adaptation of INSMware Components

The underlying principle of INSMware system construction is that to assemble or extend Management Services, the designer and provider of a Management System

(a) Determines Building Block composition;

(b) Connects interface of one Building Block to provided interface of other Building Blocks to plug them together;

(c) Provides connectivity between the Building Blocks.

Using Software Components for the implementation of Building Blocks, Software Components might need to be *adapted* to one another. If the Software Components available in-house or from commercial sources do not provide interoperable interfaces or do not meet the exact requirements of a particular INSMware implementation, the possibility of adaptation enables the integration of these Software Components [Knahl2002]. Software Components are defined by their provided services only. This forces a client to use existing interfaces to connect to required services, and therefore increases Software Components' dependence on their context (i.e. reducing the scope for their reuse). The concept of component adaptation allows the existing component to be wrapped with a layer providing adaptation services (i.e. to be wrapped with a Component Adapter) [Hofmann2000] [Troya2001]. For instance a management component may expect SNMP management data in the form 'MIB Instance' in one field with 'MIB Value' in another. If an existing management component implements 'MIB Instance, MIB Value' then it is conceptually straightforward to produce a Component Adapter that makes the translation and embeds the management component within it.

6.6.1 Component Adapter

A Component Adapter is a Software Component that represents a specific view of a Software Component to other Software Components. A Component Adapter acts as a surrogate and maps requests of client components to appropriate implementations provided by server components. Thus Software Components are accessed via surrogates (i.e. computational entities representing the actual Software Components). The surrogates can be substituted by Component Adapters that delegate requests to the original surrogates or to surrogates of other Software Components. For instance, the connection of two components A and B ($A \leftrightarrow B$) is changed to $A \leftrightarrow B' \leftrightarrow B$, where B' represents a Component Adapter for B.

Figure 6.11 illustrates the structure of a generic Component Adapter [Hofmann2000]. Client Software Components access a Component Adapter's incoming interface that represents a view to one or more server components. Each interface member (IM) of the Component Adapter's incoming interfaces is mapped to one or more implementations provided by server Software

Components. This mapping is installed by configuration and includes parameter and return type conversion if necessary. Without these conversion facilities, only implementations of server components could be used whose interfaces are subtypes of the client component's outgoing interfaces.

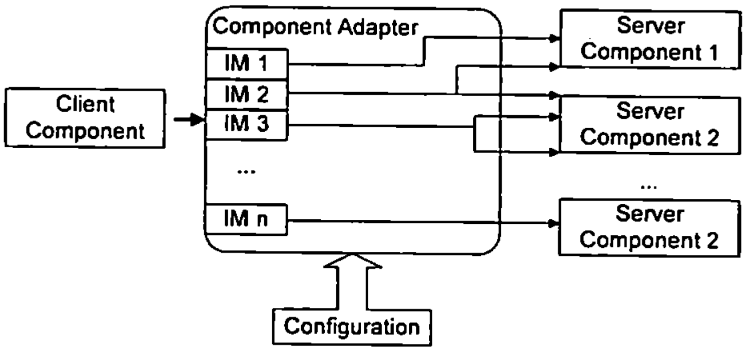


Figure 6.11: Structure of a Component Adapter

The Component Adapter implements all the necessary adaptation behaviour; no modifications are required to the Software Components concerned. The concept of the Component Adapter approach is architecture-neutral (i.e. it can be used with any current or future underlying Componentware architecture) [Hofmann2000].

For example, a pair of Component Adapters may be developed that implement security functionality for an INSMware implementation, e.g. using data encryption and data decryption. A Component Adapter that is local to a client component encrypts data and sends it to a Component Adapter that is local to a server component. The server-side Component Adapter decrypts the data and forwards it to the server component. In this scenario, Component Adapter management has to ensure that both Component Adapters are installed locally to their respective associated components. If, for example, the client-side Component Adapter is located on a host different from that of the client component, data transmission may be insecure.

6.6.2 Management of SW Components

The requirements for System Management, which includes the management of applications, have been identified as one aspect for Integrated Network and System Management. Hence, the integration of Componentware based applications and services is adding further complexity and new requirements for the provision of Management Services. A number of requirements for applications management have been identified and different approaches regarding Application Management have been proposed [Debussmann2002] [Hauck2001] [Ressmann2001b] [Kreger2001]. Component monitoring provides data about the run-time state of a set of Software Components. CORBA Assistant [Fraunhofer1996] is an example tool used to monitor CORBA components. Data such as server host utilisation and memory usage of Software Components can be monitored. However, a constraint is that Software Components have to be especially designed and implemented for the use with this tool. To illustrate a generic approach for the Management of Software Components, a concept based on Component Adapters can be considered (see also [Knahl2002]).

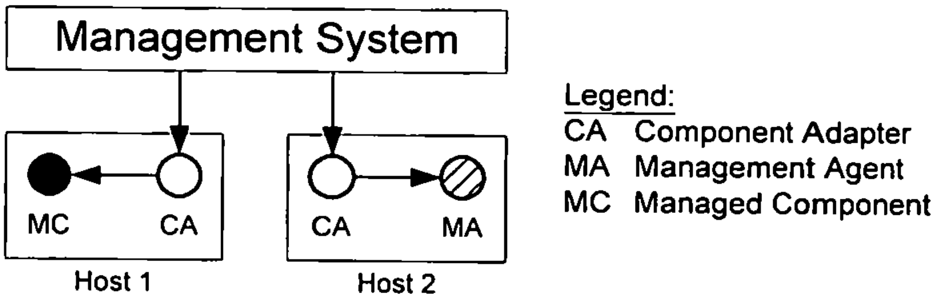


Figure 6.12: Component Adapters & Component Monitoring

Figure 6.1 shows two types of Software Component monitoring. The idea is to use Component Adapters to manage Software Components with or without individual monitoring facilities. In Host

1, a Component Adapter directly monitors a managed Software Component (MC). This approach can be applied if a MC supports monitoring functionality or if the Component Adapter gathers information about a component by making calls to its operational interface. In Host 2, the Component Adapter monitors a management agent (MA) that gathers information about a Software Component's environment, for example, the number of running processes or Software Component instances on a particular host or the operational status of hardware components (e.g., on/off/stand-by). Using the Component Adapter approach, monitoring Software Components could be integrated with new or existing management systems [Knahl2002]. If, for example, Component Adapters implement an SNMP interface, they can be integrated with any SNMP-compliant management system.

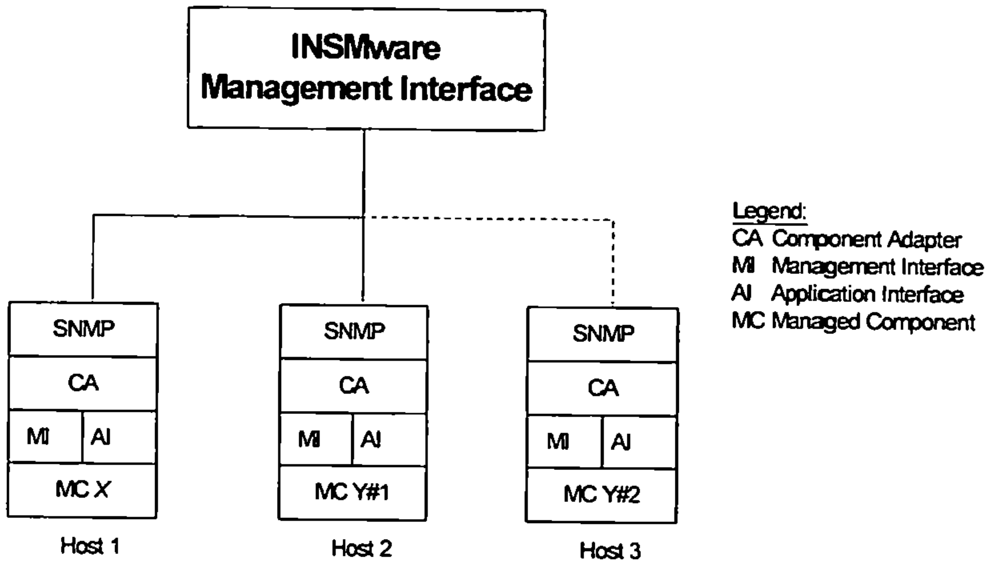


Figure 6.13: Management of distributed SW Components

The Management Interface of the Software Component provides management relevant data and can be wrapped via SNMP (see Figure 6.13). The management relevant data can be transformed into SNMP and forwarded to the INSMware Management Interface (i.e. the Software Component that provides access to Managed Objects). If a Software Component is exceeding the capacity of the system where it is installed, a new instance of the component might be started on another system with the corresponding parameters. The management is then "wandering" with the component to enable management of this component.

The management extension inside the application emits creation events at start up and deletion events when the application shuts down or an object instance is destroyed. These events are received and processed by the Management Interface. To ensure interoperability, the communication of the Management Interface of the Software Component to the Management Interface of the Management Framework can be wrapped into SNMP to enable the seamless management of legacy SNMP systems such as routers and workstations and distributed applications. Hence, no adoption of the Management Interface for SNMP would be restate quired to integrate the management of Software Components.

6.7 Conclusions

This chapter has outlined the realisation of the new Management Approach. It can be seen that the INSMware concept achieves it aims, in that it enables the provision of comprehensive, composite and distributed management provision, freeing IT service provision from the limitations imposed by the use of current management systems. As such, it offers a way to co-ordinate and advance the provision of Management Services now and in the future.

The development goal for INSMware is to define a model addressing the needs for Integrated Management Services. One of the major characteristics of the implemented INSMware Architecture is that it is a solution that concentrates on implementing a management system specifically for the provision of Network Management services. It draws on experience from this research and from a number of sources, merging existing knowledge. These sources are supplemented, where it is felt that this is required, with guidelines and additional design methods. The next chapters move on to discuss the design and prototypical implementation of the INSMware Architecture that have been developed to express and to evaluate the key features of the conceptual model defined.

Chapter 7

INSMware Prototype Design

The previous two chapters have discussed the conceptual aspects for an Integrated Network and System Management framework. To aid the evaluation of the design and principles formulated, it was felt that there was a need to apply the INSMware conceptual principles to establish its feasibility and success. Thus it was decided that a prototype system should be constructed as a basis for demonstration. This would simulate an evolutionary implementation of the INSMware framework given by the conceptual design, showing the operations of a distributed, Building Block based management system proposed in the framework. Before embarking on the development of the demonstration system, careful consideration was given as to what should be shown. Hence, a number of the design's key features felt to capture the essence of the proposed management framework where defined.

The INSMware Architecture is designed for the management of communication networks and services. It provides a template of the functionality necessary to implement an Integrated Management System. This architecture is reusable and extensible. Furthermore, due to the definition of the Building Blocks in terms of their interface, implementations of Building Blocks can be easily interchanged. This approach leaves the way open for later replacement of legacy systems. The following sections describe the design and principal mechanisms of the INSMware Architecture.

7.1 Realising INSMware Architecture

To enable the capabilities of the proposed INSMware framework with respect to the provision of Integrated Management Services, an example in form of an architecture (and prototypical implementation) of the INSMware Framework principles and of its utilisation is described. The proposed solution is based upon the following, fundamental principles:

- Conceptual design functionality: The proposed INSMware Architecture supports the generic capabilities given by the conceptual design;
- Scope of design environment: The proposed INSMware Architecture includes a number of relevant Management Services within which the INSMware framework is based;
- Evolutionary implementation: Existing systems are maintained where appropriate and integrated into the INSMware operations.

The above basis was further determined for the prototypical implementation as it represents the type of management implementation which could realistically be achieved at this point in time.

INSMware	
Architecture Domain	Integrated Management of Open Distributed Systems, Networks and Services.
Requirements	1. System Distribution 2. System Extensibility 3. Compact Architecture 4. Modular Architecture 5. Based on open standards 6. Based on generic development methodology 7. Support of existing development languages and technologies 8. Integration with 3rd Party Building Blocks 9. Reuse of existing Building Blocks 10. Security support
Architectural Characteristics	1. External: Distributed Management Services 2. Internal: Based on Building Blocks
Genericity	INSMware supports the protocols and services required in the context of integrated management.
Scalability	1. Addition of Building Blocks 2. Distribution of Building Blocks
Reuse	At Building Block level.
Interoperability	1. Application level 2. Building Block level
Building Block Interaction	Message and event passing. All the interactions between Building Blocks are mediated by the Building Block Interaction Broker (i.e. no direct Building Block interaction). INSMware Administrator configures other INSMware Building Blocks.

Table 7.1: INSMware Architecture

The relevant architectural aspects of INSMware are summarised in Table 7.1. The proposed INSMware Architecture is intended for the development of open, distributed and scalable Management Services. It needs to support different Management Standards and Protocols and also changes continuously [Patel2002]. The architectural goals are derived from the requirement catalogue defined as part of this research (see Chapter 4.6.3). Externally, INSMware is a distributed architecture that is utilised in the context of a distributed environment and based on the Client-Server model. Internally, INSMware is divided in co-operative Building Blocks that are disposed in a meshed architecture. The resulting connection graph is a mesh where components co-operate by using services from other Building Blocks and providing services to other Building Blocks. Genericity and scalability are important issues since the Management Protocols and Services change continuously in the lifecycle of a distributed system. INSMware facilitates reuse at Building Block level (i.e a Building Block can be reused in a different context).

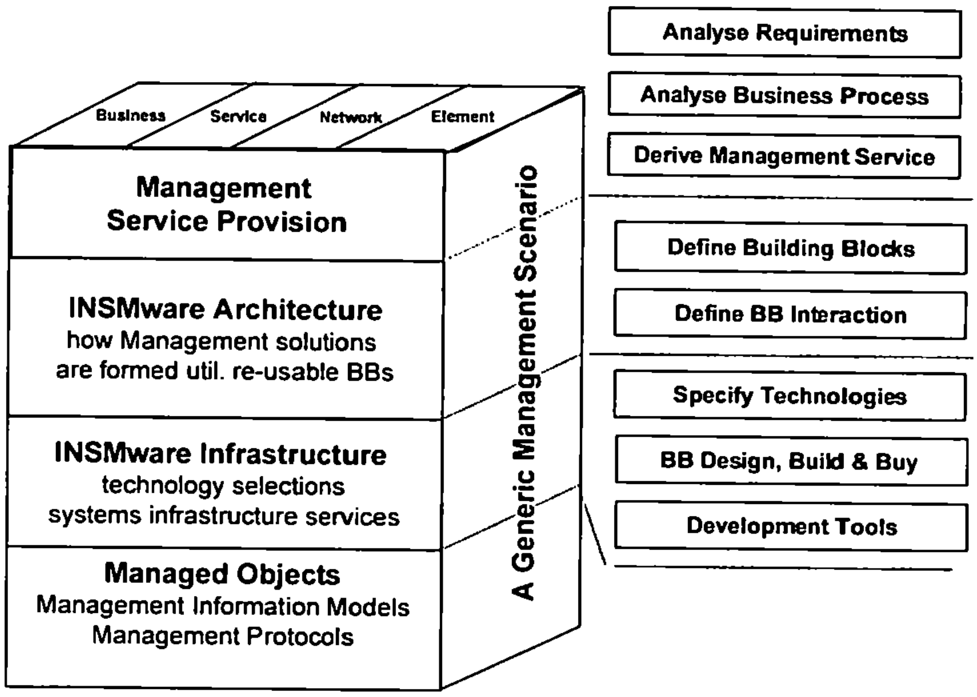


Figure 7.1: INSMware Layers & Development Aspects

Figure 7.1 illustrates the different INSMware layers and development aspects that must be addressed. One of the overall INSMware Framework goals is to use Building Blocks and their design principles to establish a market for Common Building Blocks to be used for different Integrated Network and System Management (INSM) Implementations. Some INSMware Building Blocks, such as those to provide user notifications and data storage, are useful broadly across all INSM environments (i.e. Horizontal Domain Common Building Blocks). Other INSMware Building Blocks, such as those to provide Configuration Management of a proprietary Managed Object, are useful within a single (or limited number) of management environments (i.e. Vertical Domain Common Building Blocks).

7.2 INSMware System Specification

This section outlines the system specification leading to INSMware Architecture. It starts with the Business Model identifying the relevant stakeholders and provides a specification of the INSMware Lifecycle. Then, the domains and the INSMware Architecture are discussed.

To develop a Management System based on INSMware concepts, it is required to perform system analysis and design to assemble a set of appropriate Building Blocks. However, the implications of system analysis and design will change with the use of Building Blocks when compared to traditional software development. Using Building Blocks, a system designer does not need to know the internal concepts of a Building Block (i.e. how things are done internally) but merely needs to know what is provided by a Building Block. In an ideal case, a system designer only needs to know the particular application domain for which a system is required. However, in most cases the highest level of knowledge and expertise of an application domain can be found by system-users (e.g. a network administrator using a management application) rather than by system-developers (e.g. a developer of a management application). Users that also have high understanding of data-processing concepts are called Power-Users [Orfalli1996]. Power Users are usually not experienced in modelling and software development. However, they can intuitively formulate requirements and perform system analysis and design.

7.2.1 System Analysis and Description

The following sections briefly identify the requirements and the overall context of the Management System.

The aim of the system is the provisioning of generic Management Services. In heterogeneous networks, different communication protocols, technologies and services exist and are commonly used to provide communication services. In the deployment of heterogeneous networks, a number of states and events can occur. No operator can steadily control all the resources, processes and events, but several events can occur which require attention from a Network or System Administrator. A system could, for example, have a malfunctioning interface that disrupts a number of services.

A software system must be developed to support the management of these systems. The system should notify users of specific states of systems (e.g. "high-error rate on router interface") using computer (e.g. Graphical User Interface on a PC) and non-computer (e.g. voice message) based means of communication. Additionally, users of the system should be able to obtain information about current states and administer the system. It is further required to enable system administrators to monitor and control the management system itself.

The generic business model is comprised of Management Actors (i.e. an acting person or organisation that provides or consumes Management Services), the various roles these Management Actors play and the conceptual relationships between these roles in a specific context. The INSMware Architecture is based on a business model that reflects the development context outlined in Chapter 6 (see also operational domain of the INSMware development context model) and a generic service market consisting of Management Service Customer and Management Service Providers [Redmond2000] as basis for its administrative domains. The generic INSMware business model is shown in Figure 7.2.

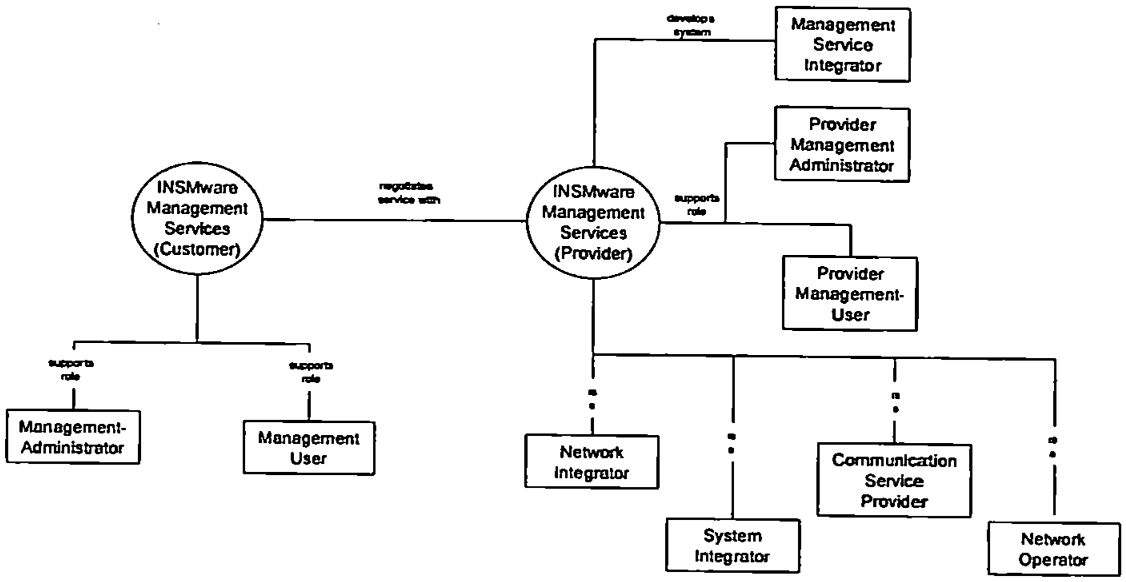


Figure 7.2: INSMware generic Business Model

Two primary stakeholders can be found in this model: the *Management Service Customer* and the *Management Service Provider*, each supporting different roles.

The Management Service Provider can be further specialised (e.g. a Network Operator who provides basic network connectivity or a Communication Service Provider who provides a value added service over basic network connectivity such as Virtual Private Networks). A Management Service Integrator develops and supports the system for the Management Service Provider. Furthermore, the roles of Management User and Management Administrator can be found in a typical Management Service Provider.

On the customer side, the roles of End-User or Customer-Administrator (who administrates the customer premises) can be identified. The provider supports the roles of Provider Management User and Provider Management Administrator (i.e. supervising the negotiated Management Services).

For the remaining discussion, two general roles are defined: INSMware Administrator (i.e. a Management Actor that controls and administers Management Services) and INSMware User (i.e. a Management Actor that uses Management Services).

The INSMware Lifecycle Model provides the set of management functional areas envisaged to address the entire lifecycle of Management Services (see Figure 7.3). The following sections introduce the INSMware service lifecycle model that has been inspired by the TINA services lifecycle model [TINA-SA1997].

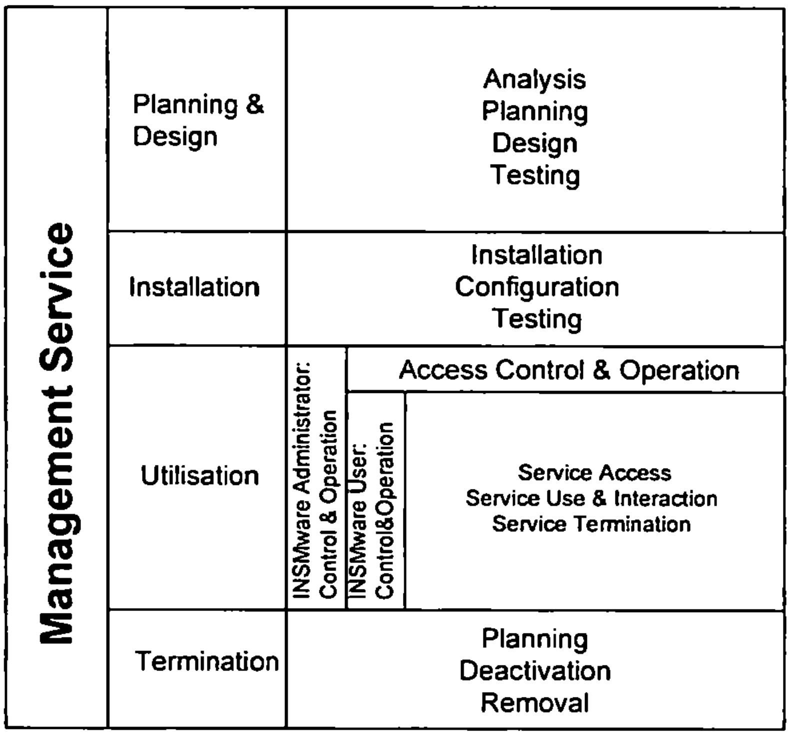


Figure 7.3: INSMware Management Service Lifecycle Model

The INSMware Management Service Lifecycle Model defines the various functional areas and facilitates the identification of the dependencies between actors. Several management areas from different Management Actors' perspectives must be considered for the lifecycle phases of

Management Services to be offered to customers. The different contents of the lifecycle phases are further refined in the proposed model (e.g. Installation, Configuration, Testing for the lifecycle phase Installation). The different lifecycle phases, namely Planning & Design, Installation, Utilisation and Termination, can be viewed from the perspective of the INSMware Administrator providing the service and INSMware User using the service. This enables specific management functionality to be associated with a particular actor's management system and ensures that the management functions available in the INSMware Administrator and INSMware User are complementary and consistent. The Management Service areas that must be addressed are:

- INSMware Administrator: Provision of Management Services from the Management Service Provider perspective and INSMware User support;
- INSMware User: Management Service usage capabilities.

INSMware Administrator Management Services affect monitoring, control and maintenance of Management Services (i.e. the installation, removal and creation of a Management Service). The aim of the Planning & Design phase is to analyse, plan, design and test the feasibility and workability of a Management Service. The aim of the Installation phase is to introduce a Management Service. Required functions are related to the installation of Management Services and include composition, installation and distribution. The aim of the Utilisation phase is the provision of Management Services to the INSMware Users throughout the lifecycle of the Management Service (e.g. support INSMware User in the utilisation and configuration of a Management Service until the Management Service is withdrawn). The aim of the Termination phase is to remove a Management Service Management Service from the environment in which the service was running.

INSMware User Management Services provide certain sets of Management Tasks provided by the INSMware Administrator (i.e. usage of Management Services). The INSMware User is not involved in the Planning & Design and Installation phase. However, it must be ensured that the Management Services are running and interact properly with the overall system. Further activities in the Utilisation phase include Management Service configuration and maintenance functions. Typical examples of this are changes to the service characteristics made by the INSMware Users (e.g. modifications regarding monitored parameters). When a Management Service is terminated, the INSMware Administrator must ensure that the Management Service is correctly removed and that all information connected with the service is saved or deleted.

The different Management Actors have only a specific view on the set of management information. However, in order to provide co-operative Management Services there is a strong dependency of management information related to the different functional areas associated to Management Services that is maintained for the different Management Actors. This requires a common modelling of required management information objects.

In the Management Service Installation Phase, the Management Administrator composes a Management Service. Management Service related data is stored in the Management Service repository. Once the Management Service is composed, it is available to Management Users through the Management Service repository. In the utilisation phase a user finds out about the Management Services offered through a request to find an appropriate Management Service available in the Management Service Repository. The Management User might further customise the Management Service, in which case configuration information is stored in a Management User Profile Repository.

The Management Service usage request activates Management Service Installation that creates an instance of the Management Service to be used. The service operates on data in the Management Service Profiles and Management User Profiles Repositories whilst it is being used. The Management User carries out the Management Services (covering all required management functions) and may need to interact with the Management Service Administrator regarding the operation and use of a Management Service.

7.2.2 Domain Specification

A number of different approaches exist such as performing an object-oriented design or brainstorming for the identification of external objects and domains. Typical management users (e.g. a network administrator) and Power-Users may only be able to do brainstorming while professional developers are able to practise both approaches. Indeed, the best way is to practise both to evaluate the results of one approach against the results of the other.

During analysis, a model of the "real World" is created. *Objects* that need to be taken into account exist in the "real world" (e.g. Routers and Switches) and in a system-internal model. Therefore, the notion of *External Objects* can be used for real world objects that can be groped to Building Blocks and the notion of *Internal Objects* can used for internal objects [Meyer1990]. The analysis phase has to consider the grouping of external objects to INSMware domains.

The following external objects have been identified during analysis:

- Network Element (e.g. Router, ATM-Switch, Server, End-System);
- Management Interface (i.e. interface that allows access to management data);
- Network Element Component (component of a Network Element, such as Network Interface);
- Event (state transition that triggers one or more Management Task, e.g. Reset of Network Connection);
- Management Event (triggers the recording of specific states of network elements);
- Message Event (e.g. to trigger the notification of users);
- Management Service (i.e. a meaningful set of Management Tasks, created to provide a solution to a management related problem such as Router monitoring);

- Data (e. g. related to a Management Service and stored in a database);
- Voice Message (i.e. natural language message);
- Text document or spreadsheet (e.g. for visualisation of data);
- Person (system users);
- Phone/Pager call (notification of users).
- Fax message (notification of users).

The next step is to break up the system into a small number of constituents that share properties or functionality [Rumbaugh1993] [Jacobson1999]. These constituents are referred to as domains. A domain can be identified by the services it exposes to other domains. Hence, each domain can be modelled by one or more Building Blocks. Possible generic domains are data storage or messaging. The domains and functionality of the INSMware Architecture have been derived from the identified requirements for an Integrated Network and System Management Architecture. Furthermore, brainstorming in conjunction with Management System Users design has been undertaken to identify the domains of the INSMware Architecture. The identified requirements have been used to identify domains. The following generic domains were identified:

(1) Connection of management system to managed elements

The network elements have to be connected to the information system (software) so that data can be received from the network elements. Also, the system should be able to control/set the network elements if necessary.

(2) Mapping of network element states to management system states (event notification mechanism)

Specific network element states should be mapped to system states. This must be done by filtering and analysing data received from the network. The system states cause one or more actions to be performed (for example, notify user A by E-Mail, user B by Voice-Mail).

(3) Management Services and Data storage

The system should enable the creation and modification of Management Services. Data about network element states, users of the system, and configuration data should be stored in a database.

(4) Distribution of management information (Communication)

The system should communicate with users to notify them of specific system states.

The following sub-domains of domain 4 were identified:

- Sending of voice mails;
- Sending of SMS Messages;
- Sending of faxes;
- Sending of e-mails.

(5) Visual representation of system information to the INSMware User

The user should be able to view current system states and a history of past system states on a Graphical User Interface.

(6) Administration of the management system by the INSMware Administrator

A system administrator should be able to configure the system by using a Graphical User Interface.

The domains (1) to (4) can be assigned to the objects we identified during analysis (see Figure 7.4). The domains (5) and (6) are extensions of the analysis model and can therefore not be assigned to objects identified during analysis.

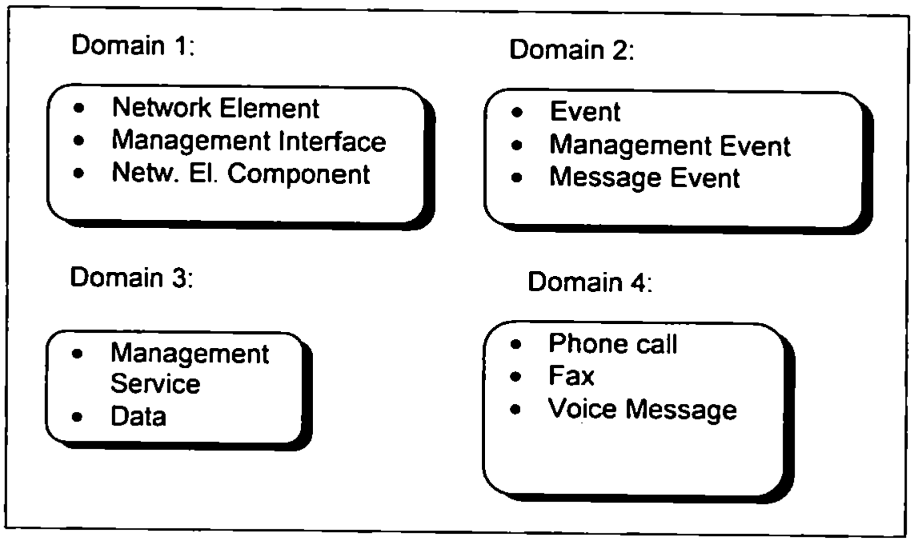


Figure 7.4: Assignment of Objects to Domains

7.2.3 Building Block Model

The research has identified that the INSMware Management Architecture has to enable distributed Management Services and must provide open interfaces that are based on industry approved standards. It has also been proved that an integration of Management Services for End-Systems and networking technologies such as ATM into Network and System Management Architectures is required to enable integrated Management Services.

At the highest level of abstraction, the INSMware system consists of six Building Blocks. This subdivision reflects the subdivision of the INSMware Application Domains that was made in order to break down the required functionality into self-contained pieces that could be implemented as Building Blocks.

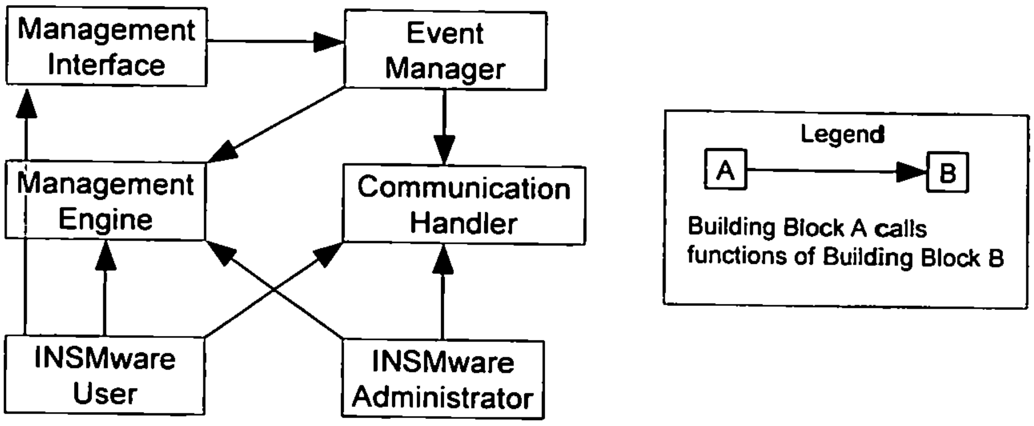


Figure 7.5: INSMware Building Blocks

Figure 7.5 illustrates the different Building Blocks of the proposed management architecture. The Management Interface implements the connection between the management system and the Managed Objects. The Event Manager deals with events and notifications received by the Management Interface and assigns them to a set of actions (connection from the Management Interface to the Event Manager). The Management Engine provides the means for the creation and maintenance of Management Services and provides a common data repository. The Communication Handler is used to distribute and forward system state information. The INSMware User and INSMware Administrator integrate the management users into the system. These Building Blocks access the Management Engine Building Block and the Communication Handler to supply the users with the required information. Users that are only allowed to monitor do not have direct access to the Management Interface, whereas users with further permissions and access rights can access the Managed Objects via the Management Interface (e.g. to perform a Management Task at a Managed Object).

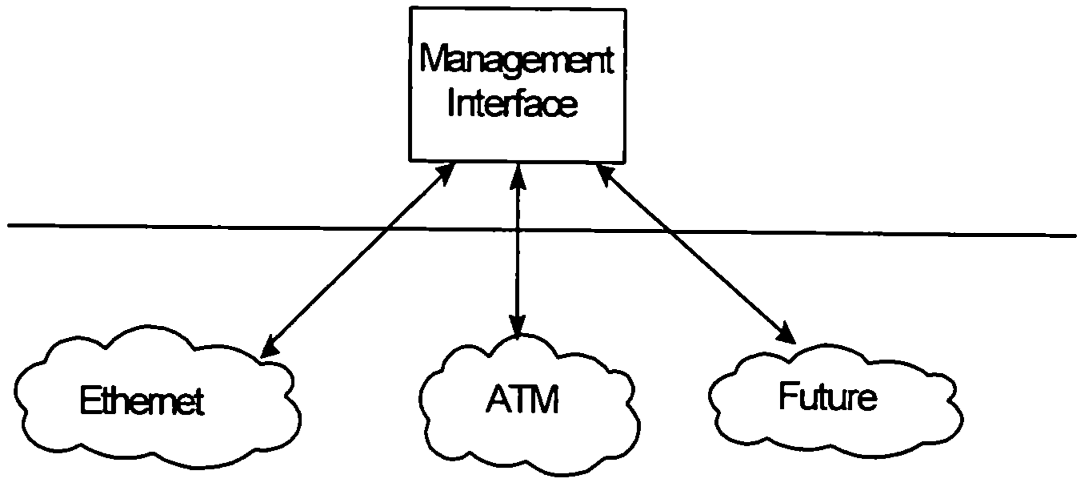


Figure 7.6: Network Management Interface

The suggested Management Architecture must enable the integration of existing Management Services as well as the integration of future Management Services that may be required to integrate as yet unknown technologies. Furthermore, the suggested Management Architecture must ensure that multiple software developers are able to integrate additional Management Services to the Architecture using its core services.

A prototypical implementation must demonstrate the architectural principles and illustrate management capabilities to provide end-to-end Management Services. Heterogeneous Managed Objects must be incorporated via a common Management Interface into the system to demonstrate the integrated management architecture (see Figure 7.6). A prototypical implementation of the outlined management framework and the described management methodology based on the Componentware paradigm to implement the various Building Blocks is presented and evaluated (see Chapter 8). Middleware technology such as CORBA or DCOM and object-oriented programming languages such as C++ are used for the implementation of the Management Services. This Componentware architecture enables the abstraction and reuse of implemented services, and, therefore eases the integration of new services and technologies.

7.3 INSMware Building Blocks

The following sections are not intended to provide a complete design documentation, but give an overview of design decisions that were made in order to provide Building Blocks that can be implemented using Componentware-oriented software development. Designing a Building Block in the context of this research first of all means designing it not for a specific project or application, but for multi-purpose use. The INSMware Management Architecture is designed to support the application domain of Network Management and System Management. The different access technologies for the Managed Objects such as SNMP should be easy to integrate. However, it will be shown that generality cannot be realised in all cases because of very specialised domains that have to be implemented.

7.3.1 Management Interface Building Block

The Management Interface Building Block implements the specialised domain of communication with the Managed Objects. Therefore, it must be adapted for different Management Services and can be regarded as a Vertical Domain Common Building Block. For example, a Management Service that monitors Software Components through a proprietary Software Component's application interface requires other access mechanism for monitoring than Network-Elements running CMIP or SNMP Agents. Because of its purpose, the Management Interface can be compared with a device driver (e.g. a printer driver). Like a device driver, the Management Interface connects a network element to a management system. Each type of Management Interface component should be developed for a specific problem domain. Different applications of a Management Interface inside a domain are possible by re-configuring the component. Hence, the Management Interface must obtain the relevant management parameter (e.g. variables to be monitored and thresholds) from the Management Engine Building Block that contains Management Service specifications.

The key to the flexibility of management modelling can be derived form the Flatland principle [Abbott1992]. In the story of Flatland the various shapes which lived on the two-dimensional surface had no knowledge of the three-dimensional world, although three-dimensional beings could see the two dimensional surface and the internal structure of the objects there. In a similar way the high-level management view allows visibility from its privileged vantage point and access to the objects of the element-level models that have no knowledge of the higher-level Management Services. The high-level management does not prevent proprietary features from being added to the generic model because these can be included as additional objects contained in the objects of a Building Block. This way of extending the high-level management view has the advantage of avoiding the need to use proprietary subclasses, so the core model remains generic and unchanged.

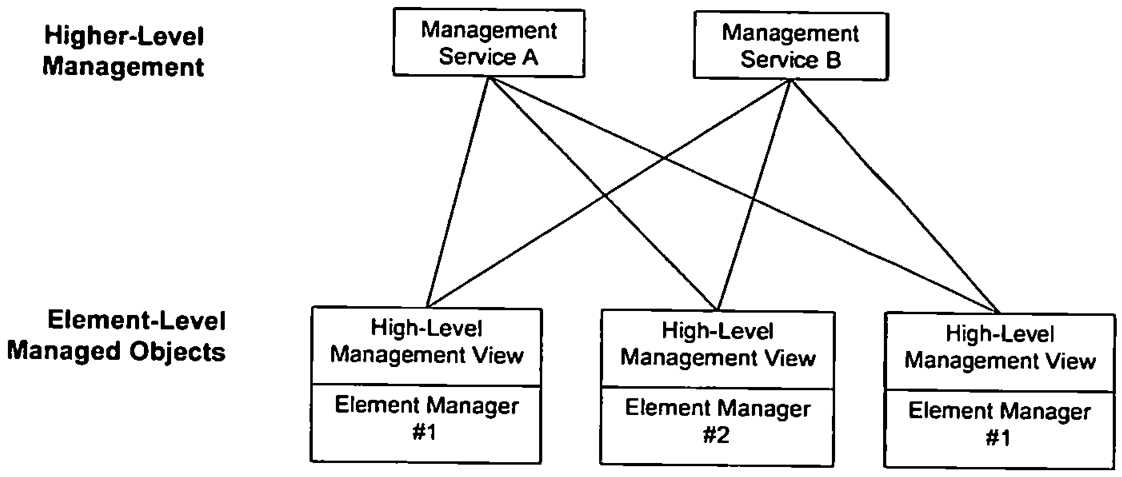


Figure 7.7: Management of different technologies

The use of the technology independent high-level management view gives an operator the flexibility to deploy different technologies within the network but to manage them in a uniform way (see Figure 7.7). Element managers can be given the task of mapping between a technology-dependent element view thus avoiding the need for higher-level management tools to take account of the details of a particular technology. This also enables to extend the high-level management so that additional features can be added without having to upgrade the entire existing system.

The INSMware Architecture must enable the integration of the various Network Management Standards and Protocols. SNMP is the de-facto industry standard Network Management Protocol and therefore is the first choice for being implemented in the Management Interface Component. Furthermore, the Management Interface must be open to integrate other standard management protocols (e.g. OSI CMIP) as well as proprietary or future implementations. In the practical part of the research programme, SNMP is integrated into the Management Interface Component through SNMP++ libraries from Hewlett Packard (see Chapter 8). In order to meet the challenge of management, systems are required that allow technologies and concepts such as Asynchronous Transfer Mode (ATM), Virtual-LANs (VLANs) and End-To-End Service Management to become integrated part of management architectures.

7.3.2 Event Manager Building Block

The Event Manager Building Block is responsible for the management of events that may occur in the system. The Event Manager Building Block is a multi-purpose Building Block. It can be used for any application that uses event mechanisms and can be regarded as a Horizontal Domain Common Building Block.

The actions performed by the Event Manager Building Block when a specific event happens should be transparent to the users of the system. So the Event Manager encapsulates the main parts of the application logic of an event-driven system. The assignment of events to a set of actions to be performed can be done by decision tables. The assignment of events to individual actions to be performed should be done by configuring the component.

Hence, the Event Manager Building Block must provide a generic method via its interface:

DoEvent (in EventID, in TimeString, EventDesciption, EventData).

The first parameter *EventID* is the identifier of an event. The parameter *TimeString* is a string that indicates when the events occurred (including the date). The parameter *Event Description* is a string that provides a description of the event and the parameter *EventData* provides additional data that belongs to the event.

7.3.3 Management Engine Building Block

The Management Engine Building Block is dedicated to information management and serves to separate the concerns of information management from those of Management Tasks and user interaction. This Building Block provides Management Services (i.e. creates, updates and deletes Management Services) and Management Service related data. Thus, it must provide access to data (e.g. predefined queries of management data) and maintain the integrity of the management related data. Hence, it can be regarded as a Vertical Domain Common Building Block.

The Management Engine Building Block is responsible for maintaining and providing access to all management information as an interconnected whole. It is not merely a data access layer but provides update operations that will preserve the integrity of the management information. The Management Engine Building Block is a major candidate of reuse, as the provision of management related data is the reusable basis for the provision of management procedures. It is not specific to a particular Management Service but is responsible to support any authorised client. Therefore, it defines the complete set of management related data. All data is either available for sharing with other Building Blocks, is private to a subset of Building Blocks or is private to one Building Block. All shared data is located, logically, in the Management Engine Building Block and therefore subject to its requirements for managing the integrity of the data and the complex issues that arise from sharing. On the other hand, private data can exist in any Building Block and is owned by the Building Block that contains it and is never shared with another Building Block, except as data belonging to interactions (e.g. return parameters).

The Management Engine Building Block provides generic access to a database that is installed on a computer. It must be accessible by local as well as by remote Building Blocks. A change to the internal scheme of a database must not require any modifications of the database component (to develop a new version).

7.3.4 The Communication Handler Building Block

The Communication Handler Building Block is used by other Building Blocks to notify users via standard communication services (e.g. fax, e-mail or voice-mail) of specific system states and provides users with an access point to interact with the system using standard communication services. It aims to provide generic messaging services and can be regarded as a Horizontal Domain Common Building Block. Several communication patterns are envisaged [Hock2000]:

- Asynchronous transmission of system states (e.g. system informs network administrator of network problems);
- User initiated transmission of system states (e.g. network administrator accesses system to follow-up network problem);
- Dialogue based transmission of system states (e.g. voice dialogue between system and user).

7.3.5 The INSMware User and Administrator Building Block

The fundamental purpose for the Human-Interaction Building Blocks of the INSMware system is to provide the means by which humans can make use of the functionality of the overall system (i.e. the other Building Blocks) to meet the management requirements. Typically, a Human-Interaction Building Block is specialised for an understanding of specific user's management requirements and procedures (and can therefore be regarded as a Vertical Domain Common Building Block). Furthermore, it hides the underlying complexity of the accomplished management procedures from the user.

To meet the complex needs of the human user, INSMware User Building Block must encapsulate an understanding of the user's goals and tasks. It must contain a model of the user's management procedures that have to be supported, the operations required to perform these management procedures and the user's workplace semantics for these management procedures.

The INSMware User Building Block supports the translation of the user's management procedures and requirements into a set of activities and a set of invocations and interactions with other Building Blocks. The INSMware User Building Block accesses the Database Building Block and the Communication Handler Building Block. Because this Building Block provides access to the overall INSMware system, it must be able to provide interaction with security management Building Blocks. The INSMware User Building Block may contain private redundant data (e.g. for specific user setting of the graphical display of management related data), but they must neither host general management related data nor do they contain shared redundant data.

The user interactions must be translated into executable management procedures that are:

- Easy to learn;
- Unlikely to produce errors;
- Efficient in executing Management Procedures.

The INSMware User Building Block enables the separation of human interaction activities from all other activities and supports human interaction with the overall management system. Information exchange might be active (e.g. displaying a menu for the user to make a choice) or passive (e.g. when a report is generated automatically). This might be customised for the individual user. Separation brings the following benefits:

- The INSMware user is protected in some measure from the failure of other Building Blocks;

- The user is less affected by changes in other Building Blocks and the overall distribution infrastructure;
- Changes in User-Interface technology (i.e. User Interface Hardware and User Interface Software) and adoption do not effect the other Building Blocks;
- Deployment flexibility for new User Interface technology is increased;
- It is easier to modify the User's task without affecting the other Building Blocks:

In addition to the services provided by the INSMware User Building Block, the INSMware Administrator Building Block is further used to administer the INSMware system. It is used to check the status and operation of the INSMware system itself and to create, modify and remove Management Services and management users and can be regarded as a Vertical Domain Common Building Block.

7.4 Design of Building Block Interaction

The set of Building Blocks defined by the INSMware Architecture forms the basis for the provision of Management Services (i.e. no single Building Block can provide a Management Service by itself). The set of Building Blocks is connected together by the Building Block Interaction Broker (BBIB). In order to provide vertical integration, the Building Blocks have well defined roles and interfaces. In order to provide horizontal integration, the Building Blocks must be used in conjunction with a standards based BBIB.

7.4.1 Static Collaboration Description

The following section shows the static aspects of a collaboration description. The main aspect is to define which Building Blocks must have access to the interface of another Building Block so that

7.4.1 Static Collaboration Description

The following section shows the static aspects of a collaboration description. The main aspect is to define which Building Blocks must have access to the interface of another Building Block so that they are able to exchange messages. Figure 7.8 shows the *logical connection graph* of the INSMware architecture. The edge number does not indicate an execution path.

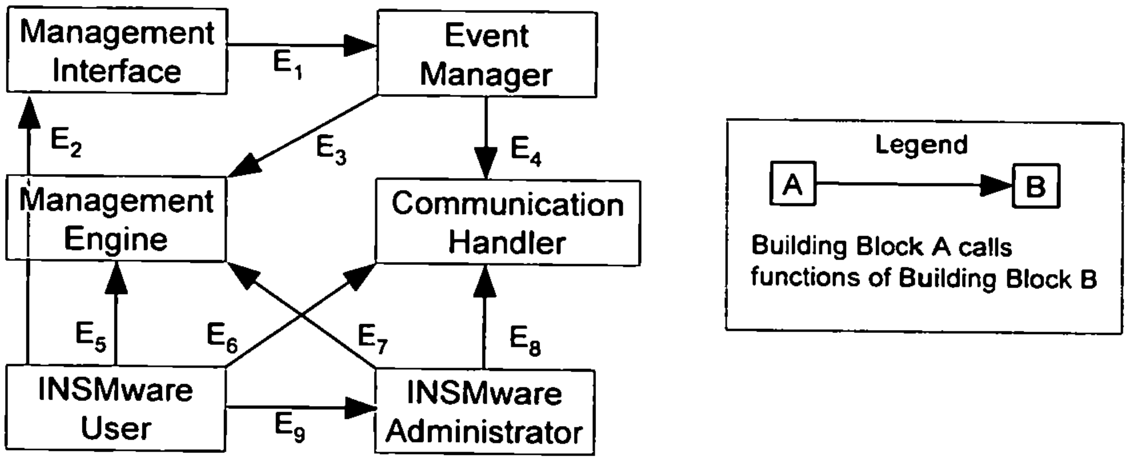


Figure 7.8: Logical Connection Graph of INSMware

Table 7.2 describes the actions performed when a communication over a specific edge takes place.

Edge	Description
E_1	Management Interface notifies Event Manager about a specific system state (e.g. a Network Element failure).
E_2	INSMware User instructs Management Interface to perform a set operation on the managed-object.
E_3	Event Manager writes data about events and protocol data to a database using the Management Engine.
E_4	Event Manager instructs the Communication Handler to notify one or more users via fax, e-mail, or voice mail.
E_5	INSMware User accesses the Management Engine to read data about the current system state.
E_6	INSMware User instructs the Communication Handler to notify one or more users via fax, e-mail, or voice mail.
E_7	INSMware Administrator configures the system by accessing the system's Management Engine.
E_8	INSMware Administrator configures Communication Handler.
E_9	INSMware User sends a message to the INSMware Administrator.

Table 7.2: Edge Description of the Logical Connection Graph

7.4.2 Dynamic Collaboration Description

To design the dynamic aspects of Building Block collaboration, *Dynamic Building Block Collaborations* have to be identified. Unlike the static collaboration description, they include execution orders. The following Building Block transactions (scenarios) can be detected for the INSMware Architecture (path indications are related to the paths shown in Figure 7.8 and Table 7.2):

1. The Management Interface Building Block reads data from the Network Element, analyses it, and transfers it to the Event Manager Building Block. The Event Manager writes the data to a database using the Management Engine Building Block (Path: $\{E_1, E_3\}$).

2. The Management Interface Building Block reads data from the Network Element, the Event Manager Building Block detects that an event occurred, and instructs the Communication Handler Building Block to notify one or more users. The information about the event is stored by the Event Manager Building Block using the Management Engine Building Block (Path: $\{E_1, E_4\}$ and $\{E_3\}$).

3. As the user wants to gather information, the INSMware User Building Block reads data from the Management Engine Building Block and represents it graphically to the user (Path: $\{E_5\}$).

4. As the user informs another user (e.g. send a message requesting a Management Service to the Administrator) the INSMware User Building Block sends a message to another user (Path: $\{E_9\}$) or instructs the Communication Handler Building Block to notify one or more users (e.g. send a voice message (Path: $\{E_6\}$).

5. As the administrator configures the system, the INSMware Administrator Building Block configures (sets parameters of) the Management Engine and/or Communication Handler Building Block (Path: $\{E_7\}$ and/or $\{E_8\}$).

6. The user instructs the Management Interface Building Block to perform a set operation on the managed-object (Path: $\{E_2\}$).

7.4.3 Building Block Distribution

To design the distribution aspects of Building Block collaboration, *Distributed Building Block Scenarios* have to be identified. As each Building Block of the INSMware Architecture can use the services of the Building Block Interaction Broker, theses can be freely distributed over an electronic network (e.g. over an Enterprise Network or the Internet).

Figure 7.9 shows a distribution scenario where the Front-End Building Blocks (i.e. the INSMware User or the INSMware Administrator Building Block) are separated from the other Building Blocks and operation takes place on 2 separate systems (i.e. Back-End Building Blocks are separated from Front-End Building Blocks).

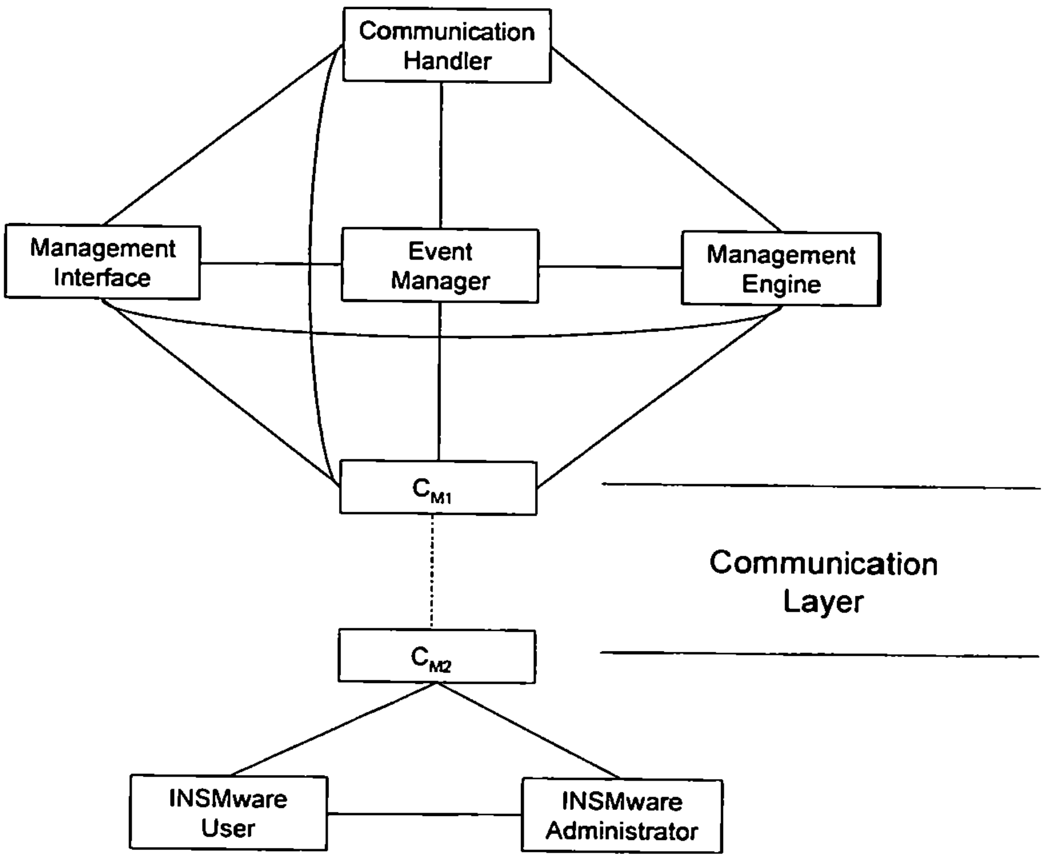


Figure 7.9: Distribution Scenario I

The following basic Building Block interactions can be identified :

1. The Management Interface reads data from a Managed Object, analyses it, and transfers data to the Event Manager. The Event Manager forwards data to the Management Engine (Path: {MI-EM} and {EM-ME}).

2. An event occurs causing the notification of one or more INSMware users and the data related to the event is processed by the Management Engine (Path: {MI-EM, EM-CH} and {EM-ME}).

3. A user requests a current system state (Path: {U-C_{M2}, C_{M2}- C_{M1}, C_{M1}-ME}).

4. A user sends a message to another user (via Communication Handler) or to the administrator (Path: {U-C_{M2}, C_{M2}- C_{M1}, C_{M1}-CH} or {U-A}).

5. Administrator configures system (Path: {A-C_{M2}, C_{M2}- C_{M1}, C_{M1}-ME} and/or {A-C_{M2}, C_{M2}-C_{M1}, C_{M1}-CH}).

6. A user modifies a Managed Object (Path: {U-C_{M2}, C_{M2}- C_{M1}, C_{M1}-MI}).

Examining the sub-graphs of Building Blocks participating in the transactions with respect to the reliability, it can be seen that the cut-vertex for the transactions 3, 4, 5, and 6 equals 1 (i.e. are not reliable if a communication link fails). In a corresponding INSMware system, the edge {C_{M1}, C_{M2}} represents the physical connection between 2 systems and is a connection-critical edge.

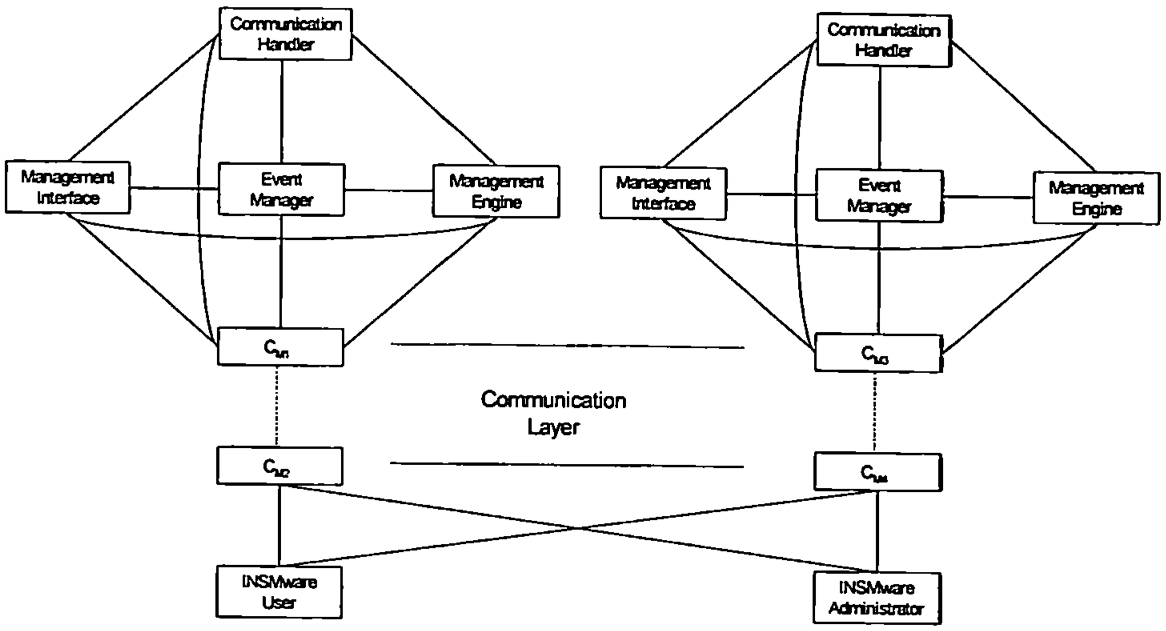


Figure 7.10: Distribution Scenario II

Figure 7.10 shows a possible solution to this problem. The Back-End Building Blocks are replicated. In this distribution scenario, the cut vertex for each distributed Building Block transaction is greater than one. However, this solution adds further complexity to the system due to the existence of *replicated data* (e.g. data contained in the separate Management Engines must be synchronised).

Figure 7.11 shows a distribution scenario where the Back-End Building Blocks are distributed to separate systems. This distribution scenario enables load-balancing among separate systems.

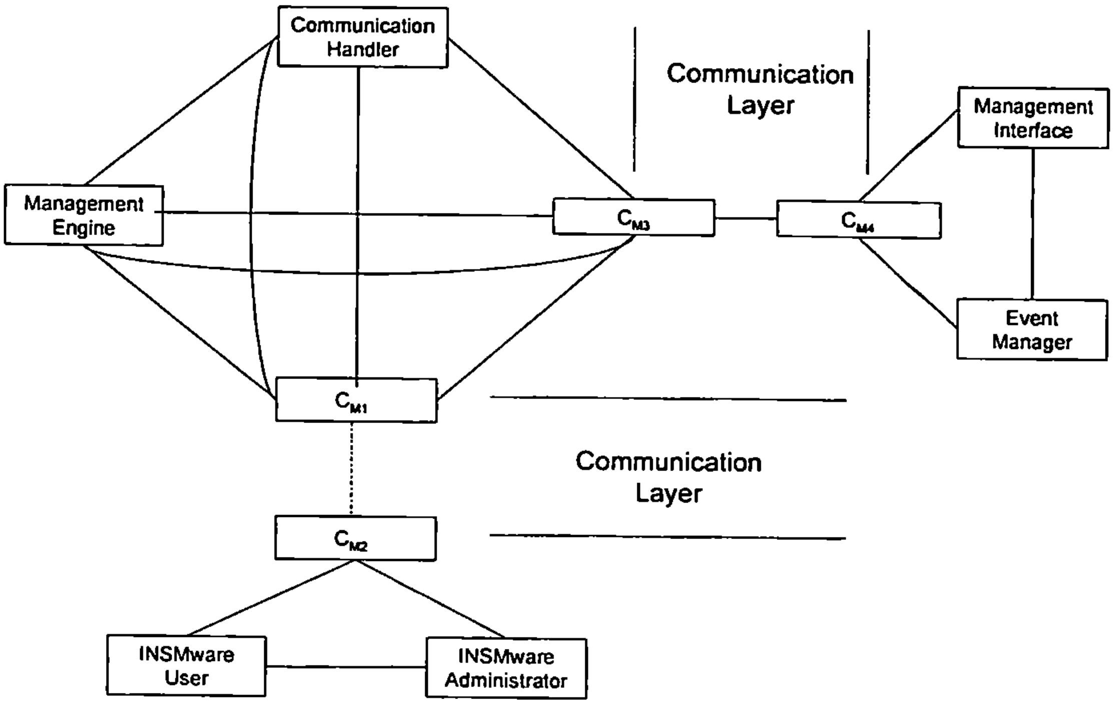


Figure 7.11: Distribution Scenario III

When applying the distribution scenario illustrated in Figure 7.11, the Building Block transactions can have the following paths:

1. The Management Interface reads data from the Managed Object and transfers it to the Event Manager. The Event Manager writes data to the Management Engine (Path: {MI-EM} and {EM-CM_4, CM_4-CM_3, CM_3-ME}).

2. An event that causes the notification of one or more users occurs and data is processed in the Management Engine (Path: {MI-EM, EM-CM_4, CM_4-CM_3, CM_3-CH} and {MI-CM_4, CM_4-CM_3, CM_3-ME}).

3. A user informs requests current system state (Path: {U-C_{M2}, C_{M2}- C_{M1}, C_{M1}-ME}).

4. A user sends a message to another user (via Communication Handler) or to the administrator (Path: {U-C_{M2}, C_{M2}- C_{M1}, C_{M1}-CH} or {U-A}).

5. The administrator configures the system (Path: {A-C_{M2}, C_{M2}- C_{M1}, C_{M1}-ME} and/or {A-C_{M2}, C_{M2}- C_{M1}, C_{M1}-CH}).

6. A user modifies a Managed Object (Path: {U-C_{M2}, C_{M2}- C_{M1}, C_{M1}-C_{M3}, C_{M3}-C_{M4}, C_{M1}- MI).

In this distribution scenario, none of the distributed Building Block transaction is *reliable (i.e. n*one of the sub-graphs representing the components participating in a specific transaction has a cut vertex greater than one).

As can be seen from the described distribution scenarios, there are a number of solutions for the distribution of Building Blocks, with different advantages and disadvantages. The best practical solution is to identify and remove potential performance bottlenecks and to identify and secure distributed Building Block transactions that are most important for a system to work correctly. Fundamental Building Block interactions are transaction 1 (i.e. the system should be able to read system data at any time and process data in Management Engine) and 2 (i.e. it is critical that the system informs users about important events).

7.5 Conclusions

This chapter has sought to outline the design and purpose of the INSMware Architecture. It discussed the key features and characteristics and how it can be applied for the provision of Integrated Network Management Services and the management of Software Components. The next chapter discusses further development issues of the system and presents the actual prototypical implementation and evaluation of INSMware.

Chapter 8

INSMware Prototypical Implementation & Evaluation

The following sections describe the prototypical implementation of the INSMware Management Architecture. It exploits Componentware technologies for the realisation of the conceptual principles. The prototype provides a demonstration platform of an open and scalable system. It utilises the Simple Network Management Protocol (SNMP) for the integration of Managed Objects.

8.1 From Open Distributed System to Network Management System

The management of networks and open distributed systems is a very broad area. It includes very different systems ranging from connectionless transmission of data to distributed multimedia services. Among those services are Network Management Services, which are used to manage communication networks ranging from LANs to WANs. As integrated management is a very broad area that can be outlined but not implemented properly within the scope of a PhD project, the developed prototype will focus on a subset of Management Services, namely on Network Monitoring based on SNMP. It offers a practical implementation of management of SNMP capable network elements and services. This area has been selected because:

- It is an area of primary relevance for "real-life" implementations;

- Existing Management Systems are far from being perfect, and problems and open issues present in this area have stimulated most of this research work;

- Recent technological innovations in computer science may be profitably applied to this field, hence there is a need to define a way to build applications that solve common problems and that exploit these technologies.

In the past, Network Management was relevant driven by manufacturers of communication equipment that needed to provide management applications for their specific products. Today, Network Management is perceived as an integrated set of tools able to manage efficiently a network made up of products from different vendors. The predominant standard protocol for Network Management is SNMP [King2000] that is implemented in the prototype. Having established the design for an INSMware system, it was also necessary to implement and evaluate aspects of management provision in a more practical context. However, rather than present a detailed treatment of a specific management Building Block, it was decided to present a general overview and implementation of an INSMware implementation which provides general management functionality based on the most relevant network management protocol. The research interest in network management is based on:

- The many yet unsolved problems which range from the integration of heterogeneous systems to scalability issues present in large networks and the associated management environments;

- Dissatisfaction with the usual way to approach the field, which neglects aspects such as usage of a structured design methodology and ease of use in favour to cost and performance;

- The absence of an architecture that focuses on the seamless integration of various protocols and services into a homogeneous and user-friendly environment.

Network Management has not only significantly stimulated the author's interest but it also has been test case of the conceptual principles and Management Architecture presented in this thesis.

INSMware	
Application Domain	Management of SNMP Networks
Operational Requirements	1. System Evolution 2. System use and development 3. Support of distributed environment 4. Efficient resource utilisation 5. System portability and generality 6. Support of Internet standards 7. Support of management standards 8. System scalability and performance 9. Support of specific platform features 10. System administration
Implementation	Front – End: Visual Basic Back – End: C++ Database : MS Access
Generality	Componentware based implementation. Support of CORBA and DCOM
Scalability	Distribution of components
Reuse	At Software Component level
Interoperability	1. Application level 2. Software Component level

Table 8.1: INSMware Prototype Principles and Requirements

Table 8.1 summarises the main aspects of the INSMware prototype. The prototype is intended to support the Network Management based on the SNMP protocol. The outlined goals for the prototype are derived from the requirement catalogue defined as part of this research (see Chapter 4.6.3 for details) and will be used to evaluate the system (see Chapter 8.6).

8.2 Conceiving INSMware Prototype

The aims of INSMware are to hide the complexity of the heterogeneous network environment and underlying technologies and to provide a universal framework for the various Management Services. In addition, INSMware is designed to facilitate the addition and integration of new networking technologies and Management Services by employing Software Component reuse.

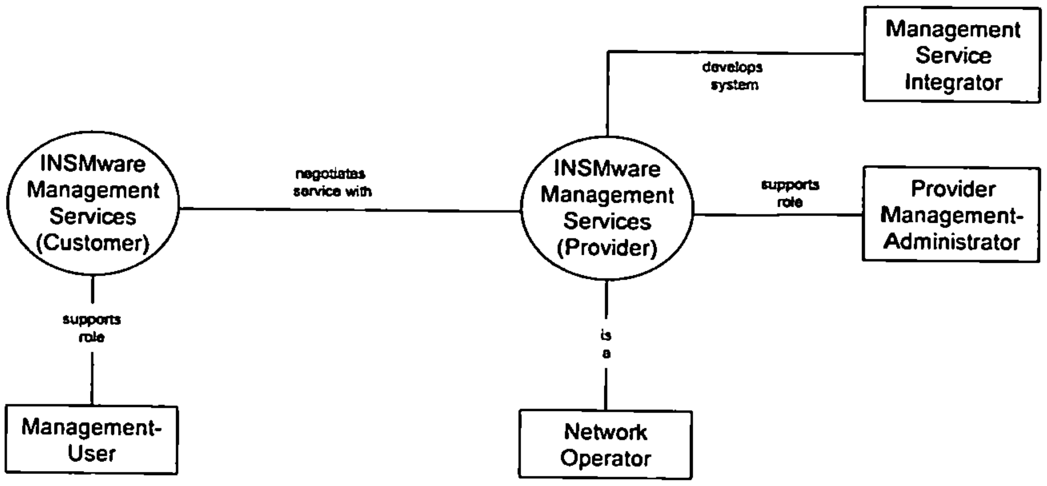

Management Service Integrator
develops system
INSMware Management Services (Customer)
negotiates service with
INSMware Management Services (Provider)
supports role
Provider Management-Administrator
supports role
is a
Management-User
Network Operator

Figure 8.1: INSMware Business Model

Figure 8.1 illustrates an instantiation of the generic business model (see Figure 7.2) that has been chosen the basis for the INSMware trial. In this model, a Network Operator acts in the role of Management Service Provider. The Management Service Provider develops the system in its role as Management Service Integrator and acts as a Management Administrator (i.e. supervising and administering the system). AT the Management Service Customer site, Management Users use the system (i.e. Network and System administrators).

Componentware based systems allow the partitioning of applications into logical and physical self-contained entities. A Software Component is an immutable piece of software with one or more well-defined interfaces that is configurable and integrable. Configurability refers to the ability to set parameters affecting the properties of a component without requiring its modification. The integration of Software Component means the connection of incoming and outgoing interfaces (i.e. interfaces being used in the client role and in the server role of a client/server component communication), while immutability requires a Software Component to be physically immutable. The most important criterion of the Software Component definition is immutability (Black-Box Reuse) since it allows dissociation from Object-Oriented concepts such as Classes and Design Patterns (White-Box Reuse).

During the course of this project, it became clear that the development of Componentware software differs in some aspects from non-Componentware development [Muench1997] [Souza1999]. Table 8.2 illustrates the main development phases and the main differences of conventional Object-Oriented and Componentware development utilised for INSMware for the main phases of software development.

System Analysis Phase	
Requirements analysis	No differences
Requirements definition	No differences
Requirements specification / modelling	If necessary, describe and model logical components and their message flow on a higher level than normal objects
System Design Phase	
Transfer of the analysis model into an architectural model	Additional steps: - Perform search for components - Add additional components from other sources into the design model - Add reusable components into the design model
Specification of the development environment	Special issues exist regarding distribution aspects: - Is required middleware technology (e.g. an Object Request Broker) available for the hardware platform and programming language? - Is OLE / DCOM available for the platform?
User interface design	No differences
Implementation and Test Phase	
Programming	Use of special tools like ORB/OLE - IDL compilers and special implementation steps
Defining test conditions	Define special test conditions for the communication between distributed Software Components running on different systems

Table 8.2:Conventional Object-Oriented versus Componentware development

During analysis, a model of the real world was established identifying the external objects. An appropriate packaging of external objects and domains to Software Component can then be done in the design phase. The design phase further deals with the selection of a development and runtime platform search for useful existing Software Component. The design of INSMware Software Component differs from the design of "conventional" OO software as only the interfaces are visible to other Software Components.

Figure 8.2 shows an overall view of the INSMware development process. After the INSMware domains and Building Blocks have been specified, Software Components implementing them have been searched for (see also [Muench1997] or [Ressmann2001a] for a discussion regarding the search of Software Components or Software Classes concepts and approaches). A number of components developed in the Distributed Application Systems Research Group at the University of Applied Sciences Darmstadt could be integrated into the INSMware prototype, namely the Communication Component, Event-Controller and Database Component [VAS2000]. The remaining Software Components have been developed for the INSMware prototype as no matching Software Components were found.

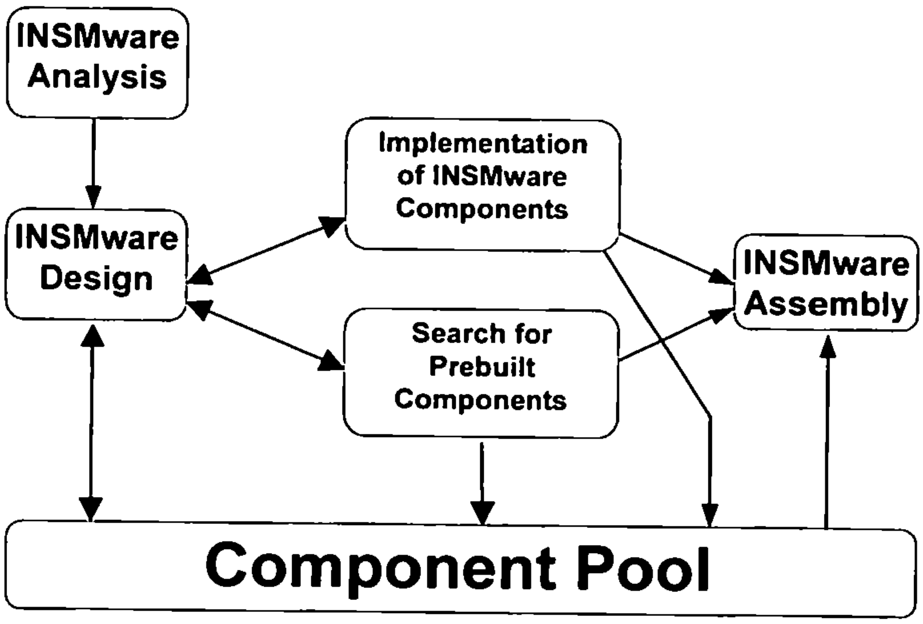

INSMware Analysis
INSMware Design
Implementation of INSMware Components
Search for Prebuilt Components
INSMware Assembly
Component Pool

Figure 8.2: INSMware development process

When Software Components are used to build a system, existing components can have impacts on the design process (as the usage of existing Software Component may influence the design of Software Component to be developed). Also, method signatures, parameter names and types may have to be adopted for existing Software Components in order to work together properly. Furthermore, the INSMware Software Components need to be assembled (i.e. embedded into an operational infrastructure). In the case of CORBA or DCOM based distributed systems, an implementation specific Interface Definition Language Compiler is used to create Component Containers at compile time that can be utilised by a particular middleware infrastructure.

The functionality of INSMware system developed consists of the processing, filtering, and analysis of management relevant data, the presentation in a graphical user interface and user notifications at the occurrence of predefined states that represent important network states. The system allows that one or more users may be notified over varying communication channels.

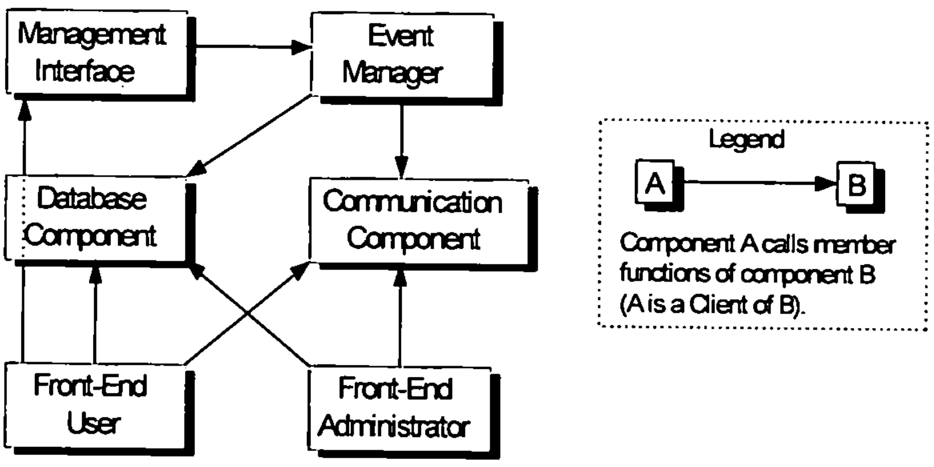


Figure 8.3: INSMware components and their connectivity

The design of the individual INSMware components is based on a domain specification that subdivides the entire application domain into sub domains as illustrated in Figure 8.3. First, the data processing system requires a connection to a data source (i.e. a Managed Object). This is realised by the Management Interface Component that exists, similar to device drivers of an operating system, in several different forms and must configurable, as required, for different data sources. The Management Interface Component interprets the received data, filters and analyses it, and notifies the Event Manager Component when particular pre-defined exception states occur. Data storage is accomplished by a call to the Database Component and user notification is effected over the Communication Component. It must be emphasised that all information about the users that need to be notified (e.g. access to user, user's role regarding the monitored processes) is stored in the system. The Communication Component itself consists of a set of several components that again implement sub domains (e.g. sending of faxes, voice mails, e-mails).

Figure 8.4 presents the operational Architecture of the INSMware system that will be described in greater detail in the following sections. It is subdivided into a Back-End System (invisible to INSMware user) and a Front-End System (visible to INSMware User and INSMware Administrator). It should further be noted that the connection between the Front-End System components and the MI component in the Back-End system has been omitted in Figure 8.4 to facilitate the readability of the illustration (i.e. illustrating a monitoring approach).

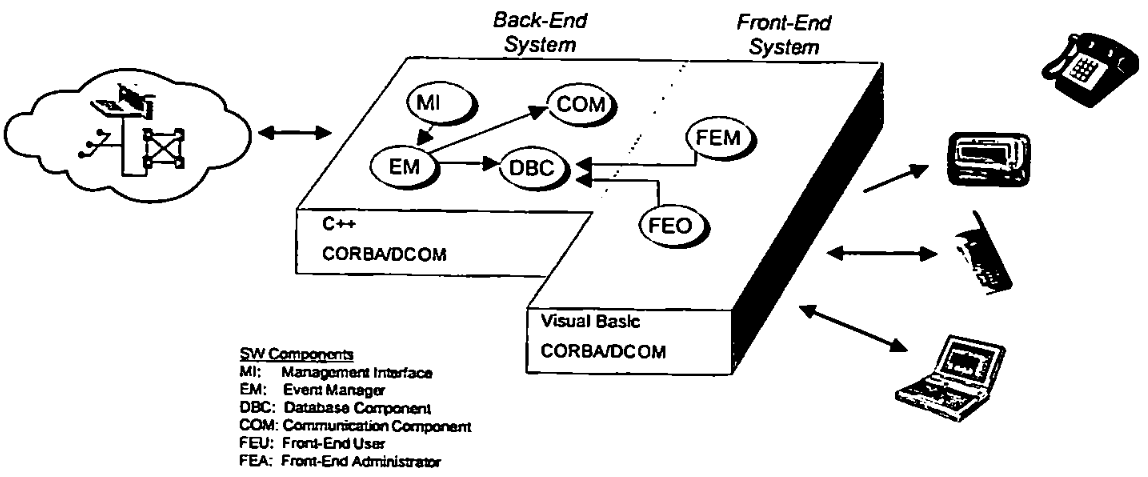


Figure 8.4: Operational Architecture of the INSMware System

The user can visualise system states by using the Front-End user component and can maintain the system by using the Front-End administrator component. INSMware was originally conceived to monitor the various network elements and to provide Management Services in heterogeneous environments (e.g. heterogeneous TCP/IP and ATM network environments). With INSMware, a timely user intervention in the running of the network processes is made possible when required (e.g. user notification when a network critical condition occurs).

To test the workability of the Componentware based approach and to examine aspects of reuse at component level, INSMware was applied to the application domain of monitoring SNMP based managed objects. Two of the components, namely the Management Interface component and the Front-End user component, have to be modified to integrate new Management Services into the management framework. The database structure has to be adapted to the management relevant information, while the remaining four components can be reused with no modifications. Adaptation is required in the Management Interface component since the data acquisition mechanism differs between the different physical and logical managed objects with different management protocols. The application domain implemented by the Management Interface component is actually very limited and a universal Management Interface component was proposed which can be adapted to different systems by configuration [VAS2000]. The graphical data representation supported by the Front-End component to visualise the management information has to be adapted to the respective application domain and remain user-friendly and simple.

8.3 INSMware Back-End System

The following sections describe the Software Components of the components of the INSMware Back-End System, namely the Database Component, Management Interface Component, Event Manager and Communication Component.

8.3.1 Management Interface Principles

One advantage of Componentware based software is the relatively easy reuse of individual parts of the system. It became clear after the first stages of the project that major parts of INSMware will be reusable with no or relatively little changes to the components within the application domain of Integrated Network and System Management. However, modifications to the system to integrate new services and to integrate existing components are required on the interface to the user (Front-End component for user interaction) and the interface to the managed network (Management Interface component for the communication with the managed objects).

8.3.1.1 Management Interface Layers

To improve the reusability of the Management Interface component it was further abstracted. Therefore, the functionality of the Management Interface component was further subdivided into several smaller components as illustrated in Figure 8.5.

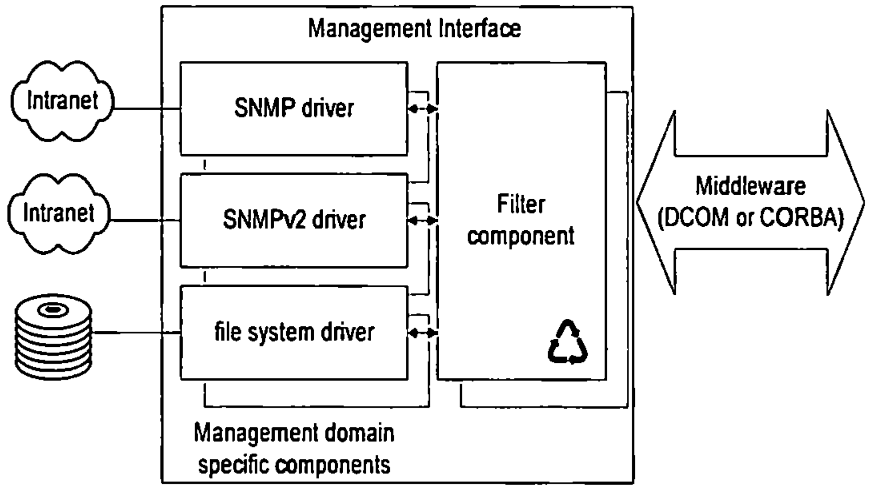


Figure 8.5: Architecture of Management Interface

The general functions for filtering of management information offer a high potential for reuse. This part analyses the information from the managed objects and decides whether a critical value has been exceeded or whether a critical event has occurred. This evaluation can proceed without further knowledge of the underlying technology (e.g. SNMPv1 or SNMPv2, Hub or Router) of the managed objects because solely the protocol independent data stream (from the managed objects) has to be analysed. Therefore (because of this high potential for reuse), the filtering was encapsulated and implemented into an individual component within the Management Interface. Hence the problem occurs of how to bring the data from the various kinds of managed objects into a syntax that can be understood and processed from the filtering component. The initialisation and de-initialisation of a connection with a managed object is individual for every kind of managed object with different management functionality (management protocols), e.g. the initialisation of a connection and the access to a SNMP managed object very much from the access to an ODBC database or a file system. The potential for reuse in this domain is relatively small and the component cannot integrate new access technologies through component configuration but through re-implementation and integration of the new access technologies. One exception are data sources that are similar concerning their access (e.g. different ODBC databases where minor adjustments are required). To be able to integrate different access technologies within the management domain a concept was developed that is similar to the driver concept of operating systems. For each access technology (e.g. SNMP), a specific component is developed which implements a specific defined interface and which can be used from the general Filter Component. This specific component is responsible for the communication with the managed object and transforms/translates the received data into a format that can be used by the Filter Component.

The introduction of an additional abstraction layer within the Management Interface leads to components with a very small level of granularity. No middleware for distribution was introduced but the concept of Dynamic Link Libraries (DLL) was used to minimise the thereby generated disadvantages (e.g. loss of performance and stability). Due to the fact that it is an In-Process-Server, a complex binding process is not required and the communication can be performed very fast and without changing tasks.

8.3.1.2 Integration of SNMP Managed Objects

SNMP is the de-facto industry standard Network Management Protocol [King2000]. Therefore, it was decided for the INSMware prototype to support SNMP to integrate SNMP managed objects. In our case, implementing Network and System Management Procedures using Object-Oriented (OO) Software technologies with the programming language C++, the integration of SNMP Management functionality into the Management Interface is required.

Object-Oriented tools provide many of the benefits required to meet this challenge. Up to now, network management development has been a cumbersome task requiring the expertise of many resources. Integrating management functionality into distributed, component based frameworks offer the object advantage to network management development and in doing so allows the provision of powerful Management Systems. There are a number of Management Protocol APIs available on the market (e.g. SNMP API from Hewlett Packard [Mellquist1997]) that can be used for the creation of network management applications. The majority of these APIs provide extensive libraries of functions that require the programmer to be familiar with the inner workings of SNMP and resource management. Most of these APIs are platform specific resulting in SNMP code specific to an operating system platform and thus not portable. However, application development using object oriented programming languages (e.g. C++) has entered the main stream; and with it, a rich set of reusable class libraries are now readily available. The research aims to facilitate standard, reusable class libraries for Integrated Network and System Management, to provide openness and flexibility that would otherwise be difficult to achieve.

SNMP++ is a set of C++ classes that provide SNMP services to a network management application developer [Mellquist1997] [Katz1999]. SNMP++ is public domain software and is developed and provided by Hewlett Packard. SNMP++ provides C++ classes to implement SNMPv1 and SNMPv2 management operations and is not an additional layer or wrapper over existing SNMP architectures. SNMP++ is not meant to replace other existing SNMP APIs such as WinSNMP or HP OpenView, it rather aims to provide development capabilities that would otherwise be difficult to manage and implement. In order to meet the challenges of integrated SNMP management, the framework supports the various versions and implementations of the SNMP standard [Mellquist1997] [Katz1999].

8.3.2 Management Interface Component

This section briefly describes the implementation of the layered Management Interface component and provides the class diagrams and an Object Interaction Diagram of the Management Interface Component (see also [VAS2000]).

8.3.2.1 Design of Management Interface Component

The following Class-Diagram illustrates the generic part of the Management Interface and serves as a reference model. The generic Management Interface can be applied upon the various management scenarios, e.g. management of SNMP components or management of File Systems.

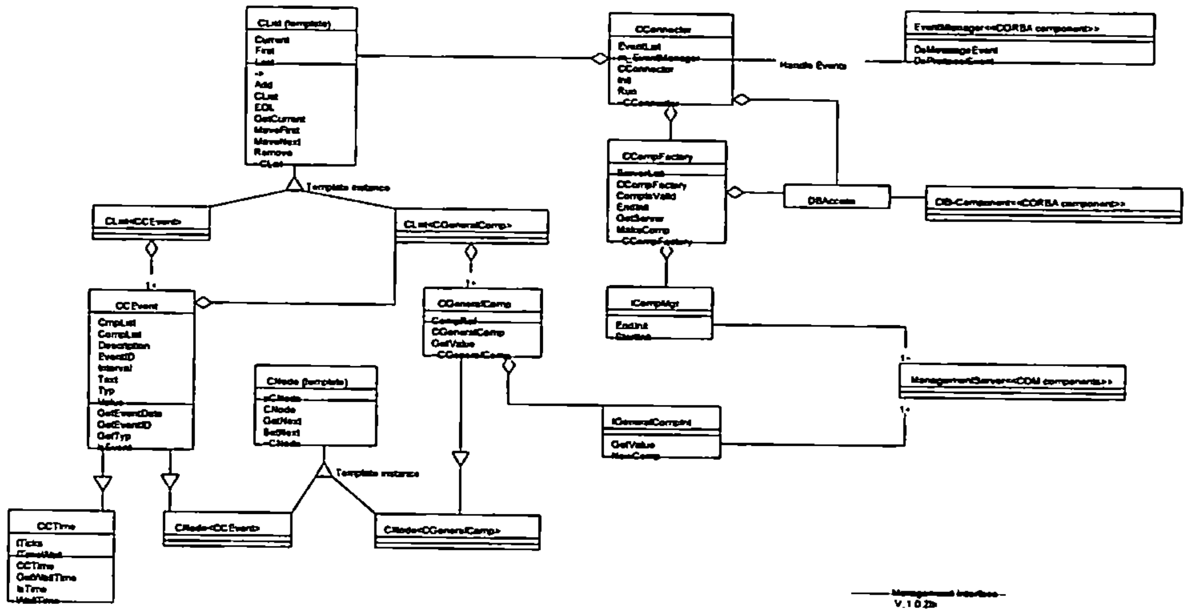


Figure 8.6: Class Diagram of Management Interface

8.3.2.2 Design of Server Component

The following Class-Diagram illustrates the generic reference model of the Server Component. The generic Server Component can be applied upon the various management scenarios (e.g. SNMP).

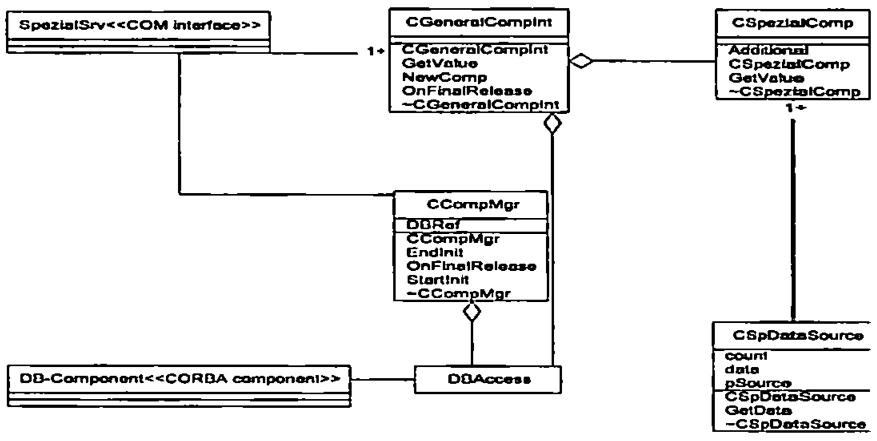


Figure 8.7: Reference Model for the implementation of Server components

8.3.2.3 Object Interaction

Figure 8.8 illustrates the object interaction of the "MakeComp" method of the class "CcompFactory" through an Object Interaction Diagram (OID).

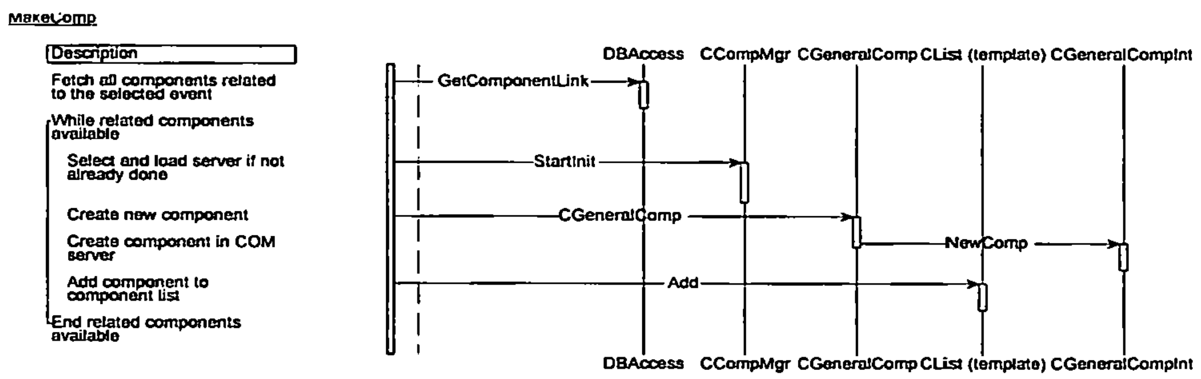

MakeComp
Description
Fetch all components related to the selected event
While related components available
Select and load server if not already done
Create new component
Create component in COM server
Add component to component list
End related components available
DBAccess
CCompMgr
CGeneralComp
CList (template)
CGeneralCompInt
GetComponentLink
StartInit
CGeneralComp
NewComp
Add

Figure 8.8: Object Interaction Diagram "MakeComp" Method

First, the data required to create a new Managed Object (i.e. component) is read from the database. Then, the Server Component of the Management Interface is initiated and a new component is created.

8.3.2.4 Design of SNMP Server

The following class diagram illustrates the implemented Server Component for the management of SNMP managed objects.

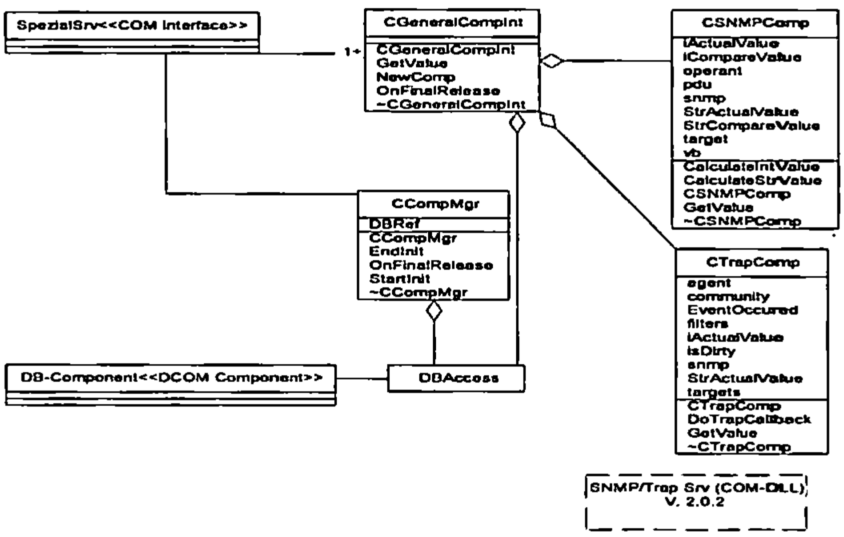


Figure 8.9: Class Diagram of SNMP Server

The analysis and design stage of the SNMP-Interface was based on the layered model of the driver-components of the Management Interface. Dynamic access mechanisms have been implemented to read the configuration from an ODBC database. The prototypical implementation of the Management Interface component is written in C++. The SNMP functionality has been implemented using two different SNMP frameworks based on C++. An early version of the SNMP interface used the Microsoft Foundation Classes (MFC), which only implements SNMPv1. To extend the capabilities of the Management Interface, the SNMP++ framework from Hewlett Packard was adopted which offers support for SNMPv1 and SNMPv2c [Mellquist1997].

Functions of the implementation include:

- Monitoring and collection of SNMP Traps. Every SNMP-based system in a network can send an event/notification to INSMware, e.g. if a reset or cold-start of a system occurs (passive monitoring);
- Active monitoring of SNMP managed objects using SNMP Get / GetNext. This enables to actively monitor and control changes within the SNMP Agents.

Another advantage of the design is the focus on integration of new Management Services and managed objects is the easy adaptation of the INSMware management framework for new developments, e.g. for new versions of SNMP. SNMP standards, such as SNMPv3, start to occur and can be implemented into the SNMP++ framework [Katz1999] and, therefore, with minor modifications into the INSMware framework. Figure 8.10 illustrates the layers of the Management Interface (see also Figure 8.5) and the implementation of the different SNMP frameworks within it.

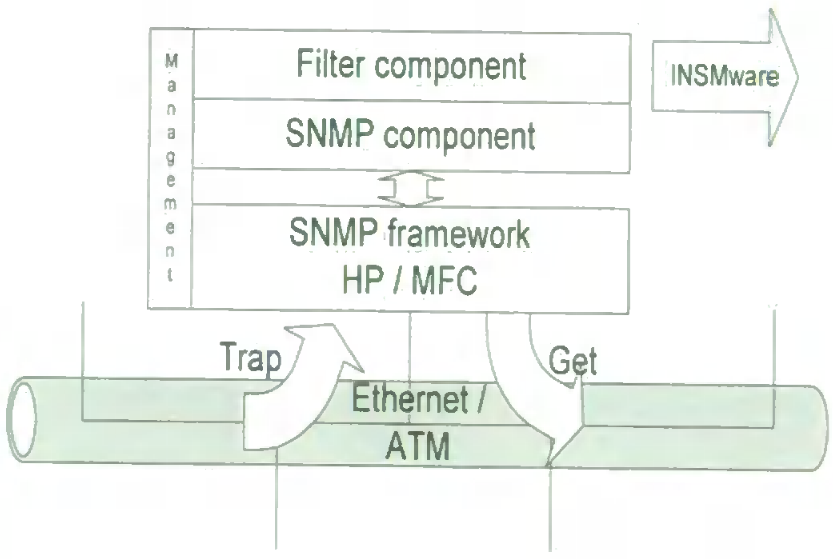


Figure 8.10: Management Interface layers

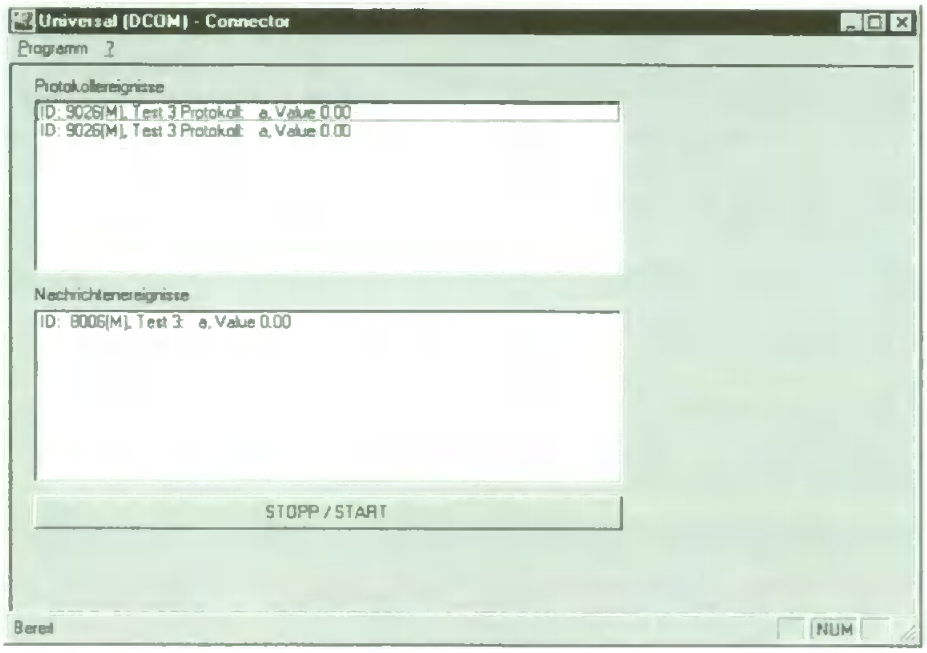
The INSMware management prototype offers Management Services for the collection and monitoring of SNMP-Traps and offers services for SNMP Get and GetNext operations. This enables INSMware to act as a SNMP Manager. Traps, which are sent from SNMP managed objects such as routers or even from software in the network, are collected and further processed for the Filter Component. The SNMP component then forwards the data to the Filter Layer for further analysis. The Filter Component then analyses the data using the configuration parameters from the INSMware Framework. The filtering uses the source address as well as the Object ID (OID) of the trap (e.g. to discover a cold-start of a managed object).

Contrary to the passive management based on traps, where the managed object itself is initiating and sending management data, the Management Interface also enables active monitoring based on SNMP Get and GetNext operations. Here, the Management Interface reads management relevant data from the Management Information Base (MIB) of the managed objects to monitor and discover status changes or network conditions. The SNMP++ C++ libraries from Hewlett Packard support the different data types (Integer, Unsigned Integer32, Octet, OID, IP-Address, Counter32, Gauge32 and Timeticks) which are defined for SNMP. This MIB management data is compared against predefined values using different operands (e.g. '<', '>' or '=') to discover status or administrational changes such as change of System Administrator or critical network conditions, such as high collisions on an Ethernet or the failure of a WAN connection. When the Management Interface discovers that a critical event occurred (e.g. due to an incoming Trap or a MIB value) it stores the event and arranges the notification of the related INSMware user on it's GUI or forwards the notification to a manager via an external communication (e.g. telephone or E-Mail). The messaging functions are provided by the Event Manager and the Communication Component.

The specification and manipulation for the configuration of the SNMP managed objects (e.g. IP-Address, SNMP Version and Community) and of and the events for the related Get/GetNext and Trap events (e.g. OID, Value, Operand) is implemented via the INSMware Front-End Component and saved in the Database Component. The type of notification can be dependent on different parameters (e.g. dependent of day or time) and can be configured for each individual event that occurs. Furthermore, it is possible to set the intervals for the polling and controlling of events according to the requirements and to individually add/delete intervals according to specific management requirements.

8.3.2.5 Operation of the Management Interface Component

To start the monitoring of Managed Objects, the Management Interface component needs to be launched (i.e. start 'Connector.exe' [VAS2000]). A Control log window that provides a visualisation of the events that occur in the Management Interface Component is launched automatically (see Figure 8.11).



```
ID: 9026(M), Test 3 Protokoll:  a, Value 0.00
ID: 9026(M), Test 3 Protokoll:  a, Value 0.00
```

```
ID:  8006(M), Test 3:  a, Value 0.00
```

Figure 8.11: Management Interface Control Log

It is advisable to test the behaviour of the Management Interface to confirm to operational status of the Management Interface. Therefore, a program "snmptraps.exe" has been developed using the SNMP++ library that can be used to test whether the Management Interface receives SNMP Traps to ensure that the Management Interface Component is working correctly [VAS2000]. The program creates SNMP Traps that can be sent to an IP Address for testing purposes. For example "snmptraps 127.0.0.1" sends a ColdStart Trap to Adress 127.0.0.1 (i.e. the local TCP/IP stack). This Trap is received by Management Interface Component and displayed in the event log window if the Management Interface is working correctly.

8.3.3 Event Manager

The Event Manager Component is the "glue" of the INSMware system. It gets input from the Management Interface, processes the transmitted data, and passes results to the other components. It is written in C++ and provides two interface methods: the first handles protocol events and the second handles message events (see also [VAS2000]).

Using ISNMware in a CORBA environment, the Object Request Broker (ORB) is used to enable access to distributed Software Components at runtime. The Event Manager Component provides functions to other components and registers as Server an interface in the Implementation Repository of the ORB. Every Software Component that wants to use the functionality of the Server makes a request to the ORB that is required to locate the Server Component and provide access to that component. Interfaces are declared using the CORBA Interface Definition Language (IDL) and are compiled using an IDL compiler. The following code extract illustrates the Event Manager Interface:

```
enum EM_SUCESS_CODE { EM_OK, EM_FAILURE};

interface EventManager
{
        oneway void DoMessageEvent(in long EventID, in string
        TimeString, in string ID, in string EventString);

        oneway void DoProtocolEvent(in long EventID, in string
        TimeString, in string value, in string ID, in string
        EventString);
};
```

These interfaces provide the means to access the Software Component and provide the separation between the actual component implementation based on a particular Hardware, Operating System and programming language and its generic interface.

Under DCOM, the interfaces were implemented using the Microsoft Interface Definition Language (MIDL). The Universally Unique Identifiers are unique identifiers generated for every object by an UUID-Generator. IUnknown provides the methods to connect to the EventManager interface (QueryInterface, AddRef, Release). The following code extract illustrates the Event Manager Interface:

```
// EventManager.idl : IDL source for EventManager

[
        object,
        uuid(7301322E-9832-11D1-81E6-00403399D908),
        dual,
        helpstring("IEvent Interface"),
        pointer_default(unique)
]

interface IEvent : IDispatch
{
        [id(1), helpstring("method DoMessageEvent")] HRESULT
        DoMessageEvent(long EventID, BSTR TimeStr, BSTR ID, BSTR
        EventString);
        [id(2), helpstring("method DoProtocolEvent")] HRESULT
        DoProtocolEvent([in] long EventID, [in] BSTR TimeString, [in]
        BSTR value, [in] BSTR ID, [in] BSTR EventString);
```

The Event Manager is a multipurpose Software Component responsible for the processing of events that occur in the system (i.e. it can be used to process events from different types of sources) [Hock2000] [VAS2000]. The actions performed by the Event Manager Building Block when a specific event happens are transparent to the users of the system. The assignment of events to a set of actions to be performed is done through the usage of decision tables [VAS2000].

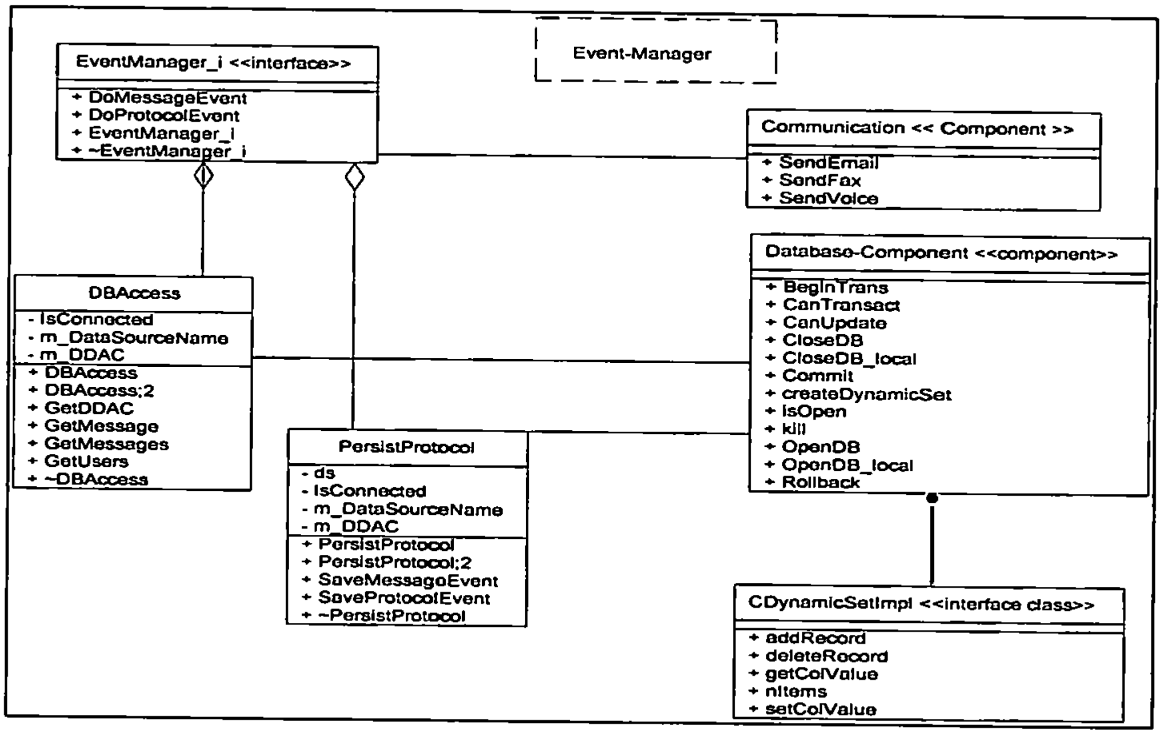


Figure 8.12: Class Diagram of Event Manager

The structure of the Event Manager is illustrated in Figure 8.12. If a protocol event is to be handled, the Event Manager Component invokes the Database Component, opens the table with the resulting protocol, adds a new record and saves the remaining information such as the time when an event happens and its state or value. If a message event happens, all applicable users are determined by reading the relevant communication settings in the "CommSetting"-table of the Database Component. The day and time when these users can be reached is compared with the actual event time. If these coincide, the communication media and the related text or voice mail files are determined. Then, the message transmission is done via the Communication Component and the results of this transmission are stored in the protocol table of the database.

8.3.4 Communication Component

The Communication Component provides user notification to users via standard communication services (e.g. fax, e-mail or voice-mail) of specific system states [VAS2000]. Under Microsoft Windows, different application programming interfaces (APIs) exist that allow access to hardware at a higher level. The APIs that can be used for communication purposes are the Messaging-API (M-API) and the Telephony-API (T-API).

The Communication Component manages all Message Event transmissions through external communication channels (e.g. telephone). To do this, it uses two MS-windows extensions: the Message-API and the Telephony-API. Thus, whenever possible, other development tools (here M-API) have been used. Another advantage of M-API is that it is available as an OLE-control, while T-API is only available as a library. To use T-API in C++, an encapsulation in the form of a dynamic link library (MS-Windows DLL) was chosen. This following communication scenario was designed to transmit voice message files :

1. After dialling and establishing a connection, a voice mail file to identify and authenticate the users is played periodically;

2. The user must identify and authenticate himself with a code by touch-tones to be recognised by the system;

3. After authentication, the system plays the real message file with the event message.

The M-API is a set of functions for creating, manipulating, transferring, and storing fax and e-mail messages. M-API allows the integration of applications with Microsoft Exchange. Two controls, the M-API session control and the M-API messages control, are provided with Microsoft Visual Basic and Microsoft Visual C++ providing access to the M-API. To use the M-API, a M-API session has to be established first. This functionality is encapsulated by the M-API session control. When a session is established, the M-API messages control can be used to send, receive, and manipulate messages.

The T-API is based on Microsoft's extended unimodem architecture. It allows the access of modems to transfer files or voice data. The T-API is encapsulated by a Software Component with a small surface area to hide the complexity of the T-API (see also [Hock2000]). The T-API component was implemented as a Dynamic Link Libraries (DLLs) that can be used in many development environments under Microsoft Windows. T-API is a set of functions for all kinds of telephony: automatic dialling, data-transmissions, voice-mails, answering machines, etc. T-API contains the additional functionality to send faxes and e-mails like Message-API (M-API).

8.3.5 Database Component

The Database Component provides the services for storage and access of management related data [VAS2000]. All access to the management related data is handled through the Database Component. It is a general purpose Software Component as it can dynamically access columns in tables within databases that offer an Open Database Connectivity (ODBC) driver. It is written in C++ and uses the encapsulation of ODBC via the Microsoft Foundation Classes (MFC). The physical database is implemented using MS-Access as this product was available for this project. Furthermore, it is relatively easy to administer and use and can be installed on standard desktop PCs that were available for the project.

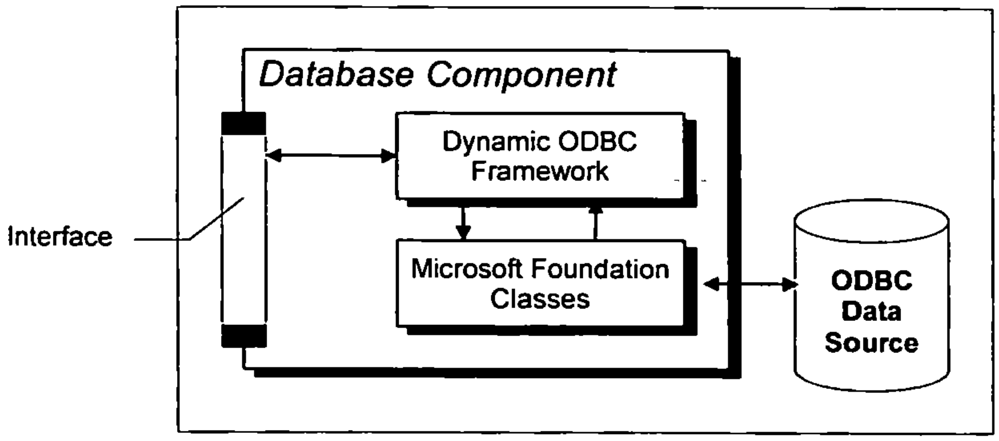


Figure 8.13: Structure of the Database Component

The Database Component provides access to any ODBC database that is installed on a computer. It is accessible by local as well as by remote clients. A change to the schema of a database will not require any modifications of the Database Component (i.e. it is not required to develop a new version of the Database Component). Figure 8.13 shows the structure of the Database Component. The functionality of the Database Component is exposed via the interface of the component [VAS2000].

For the INSMware prototype, a generic database structure that can be supported on any SQL database was designed as illustrated in the next tables. For the actual implementation of the database system, the MS Access database product has been used. Reasons for this implementation choice were availability of the product and previous experiences and knowledge gained with this database system.

Network Structure

Key	NodeID	Net ID	Caption	Base Key	URLHelp
1	1	0	University Region Darmstadt	0	network2.htm
2	2	0	University of Appl. Sciences Da., Branch	1	network.htm
3	2	0	Darmstadt University	1	network.htm
4	2	0	German Research Network	1	network.htm
5	2	0	University of Applied Sciences Darmstadt	1	network.htm
6	3	0	IGD/VAS Laboratory	5	network.htm
7	5	0	Router 141.100.225.254	6	network.htm
8	6	0	141.100.225.9	7	hp-ux.htm
9	7	0	141.100.225.11	7	
10	8	0	141.100.225.13	7	
11	3	0	FBI	5	network.htm

Table 8.3: Network Structure

The Network Structure table provides the basic data related to the existing network infrastructure. This data is used to build the network view.

- Key: Key for every element in the network.
- NodeID: Connection to table SNMP_ID, where the SNMP specific management data is stored.
- NetID: Network ID (i.e. identifier of individual networks, e.g. when different user work with different managed networks).
- Caption: Description of the network Element.
- BaseKey: Specifies the next higher element of the network infrastructure (or tree vie).
- URLHelp: Specifies the URL address for this node.

SNMP Nodes

NodeID	IPAddress	Read Community	Group	Type	Status	Write Community	Management Protocol
5	141.100.225.254	public	2	1	4	public	0
6	141.100.225.9	public	4	2	6	public	1
7	141.100.225.11	public	4	1	5	public	2
8	141.100.225.13	Public	4	2	6	public	3

Table 8.4: SNMP Nodes

The SNMP Nodes table provides the SNMP related management data. These parameters are required to access the SNMP nodes to retrieve and manipulate SNMP management parameters. Note that Table 8.4 presents the key information found in this table (i.e. parameters TimeOut, Retries, NetProtocol, Aliasname, MgmtProtOther are missing).

- NodeID: Key for NodeID table.
- IPAdress: Specifies the IP Adress of the Network Element.
- ReadCommunity: Specifies the Read Community of the Network Element for the retrieval of management data.
- WriteCommunity: Specifies the Write Community of the Network Element for the manipulation of management data.
- Group: Specifies the group of the Network Element (1: Network, 2: Network Device, 3: Building, 4: System, 5: Software).
- Type: Specifies the type of network element, dependent on its group (Network: 1=Root, 2=Subnet; Network Device: 1=Router, 2=Switch; System: 1=PC, 2=Unix WS, 3=Printer).
- Status: Status of the network device.

- ManagementProtocol: What management protocol is being used, i.e. which version of SNMP is being used.
- TimeOut Amount of time until an SNMP operation is timed out.
- Retries: No. of retries when SNMP operation failed.
- Netprotocol: Specifies what transport protocol is being used (0: IP, 1: IPX).
- Aliasname: Aliasname of the network element.
- MgmtProtOther: Specifies wheter and what other Management Protocol than the standard SNMP is being used.

SNMP Variables

CompID	Key	OID	Name	Operant	Value
9035	8	1.3.6.1.2.1.1.4.0	sysContact	=	Test
9036	8	1.3.6.1.2.1.1.1.0	sysDescr	!=	Ttt
9038	9	1.3.6.1.2.1.1.4.0	sysContact	!=	=x

Table 8.5: SNMP Variables

The SNMP Variables table specifies the SNMP variables that are being monitored by INSMware and its related operant and value.

- CompID: A unique identification number is created for every variable and its monitoring process. If a variable is being monitored several times (i.e. several monitoring processes for the same variable), a unique identification number is created for every monitoring process.
- Key: Key of the network element from table Network Structure.
- OID: Here, the OID of the variable is specified.

- Name: Name of the variable (e.g. Sys Contact).
- Operant: Operant of the monitoring process.
- Value: Value against the actual data is being compared using the operant.

SNMP Components

CompID	Address	Communit	OID	Version	Retries	Timeout	Operant	Value
9002	127.0.0.1	public	1.3.6.1.2.1.1.4.0	1	1	1000	=	Simon
9003	127.0.0.1	public	1.3.6.1.2.1.1.7.0	1	1	1000	=	76
9004	127.0.0.1	public	1.3.6.1.2.1.1.3.0	1	1	1000	=	13
9005	127.0.0.1	public	1.3.6.1.2.1.4.20.1.1.12	1	1	1000	=	127.0.0.1
9035	141.100.225.9	public	1.3.6.1.2.1.1.4.0	1	1	1000	=	test
9036	141.100.225.9	public	1.3.6.1.2.1.1.1.0	1	1	1000	!=	ttt
9038	141.100.225.11	public	1.3.6.1.2.1.1.4.0	1	1	1000	!=	=x

Table 8.6: SNMP Components

This table is being used by the Management Interface Component to gather the required management processes and to provide these parameters for the Management Interface component.

- CompID: Here, the CompID from SNMP_Variables is being gathered. It is checked, whether an entry with the same OID and IP address already exists to guarantee, that no variable is being incorporated twice.
- Address, Community, OID, Version, Retries, Timeout, Operant, Value : Required configuration parameters for the Management Interface component. Gathered from the SNMP Variables and SNMP Nodes tables.

Event Table

EventID	Description	Type	Interval
9001	SNMP Test SysContact	M	5000
9026	Test 3 Protokoll	P	5000
9100	Trap lokal	M	5000
9101	Trap entfernt	M	5000
9999	Dummy	M	5000

Table 8.7: Event Table

This table provides the system settings for every INSMware event category, specifying the type of INSMware event and its interval.

- EventID: Here, an unique ID for every Event is created. If a SNMP variable is being monitored, the CompID from SNMPComp is being used. It is checked, whether an entry for the same management process (e.g. several users monitor the same variable) already exist to guarantee, that no variable is being incorporated twice.

- Description: Description of the event. In the case of a Message-Event, the description generated in the Front-End (Dialogue 'Watch Variable') is being used. In the case of a Protocol-Event, the description from the dialogue 'Add Variable' is being used.

- Type: Type of the event (M: Message-Event, P: Protocol-Event).

- Interval: Interval in milliseconds in which the values are being checked.

Event Component Link

ID	EventID	CompID	CompTyp
70	9001	9001	Z
71	9100	9100	T
72	9101	9101	T
75	8003	9020	M

Table 8.8: Event Component Link

This table links the EventID and the CompID and specifies the operational setting for the INSMware events. CompTyp specifies the operational setting of the event (Z: ‚Sleeping', is not monitored; M: Management Information based on OIDs, T: Trap Information based on SNMP Traps).

Message Event

ID	Message	WAV_Message	FAX_Message
9001	SNMP Ereignis	Goat2.wav	fax.txt
9100	SNMP Trap Ereignis 127.0.0.1	Goat2.wav	fax.txt.
9101	SNMP Trap Ereignis 141	Goat2.wav	fax.txt
9999	Geschäftsbericht 1997	Goat2.wav	fax.txt

Table 8.9: Message Event

This table specifies the message to be forwarded in the case of an event.

- ID: Here, the EventID from Event is being incorporated. Every Message-Event is linked to an individual message.
- Message: Here, the message to be forwarded is being specified.
- WAV_Message: Specifies a corresponding 'WAV' file related to the specified event.

- FAX_Message: Specifies a corresponding 'TXT' file related to the specified event to be sent as a fax message.

Communication Settings

ID	User_ID	Message_ID	Time_ from	Time_ until	Day_ from	Day_ until	Comm_ Type	Comm_ Number
21	1	1022	00:00	00:00	0	6	Phone	#1
24	1	1040	00:00	00:00	0	6	Pager	QUIX/30
25	1	1060	00:00	00:00	0	6	Phone	#1
27	1	1061	00:00	00:00	0	6	Phone	#1
39	1	8010	00:00	00:00	1	7	Phone	#1
40	1	8011	00:00	00:00	1	7	Phone	
41	1	8012	00:00	00:00	1	7	SMS	

Table 8.10: Comm Settings

This table specifies the Communication Settings of the INSMware system. In this table, the notification for every Message-Event is being specified. An event can initiate a number of Message Events. In this case, multiple entries can be found in this table for this event.

- ID: For every type of notification, a new entry is being incorporated in this table with an unique ID.

- User_ID, Message_ID: The User_ID and the Message_ID (from the Event table) is being specified.

- Time_from and Time_until, Day_from and Day_until: Specifies the period of time (from – until) and the period of days for which the management is being active.

- Comm_Type and Comm_Number: Type of notification (Phone, SMS, Fax, Pager, E-Mail) and the corresponding number (or address).

8.4 Front End System

The following sections discuss the INSMware Front-End System.

8.4.1 Implementation of Front-End Components

The second component of INSMware that has to be adapted for new services in the domain of Integrated Network and System Management is the Front-End Component. The Front-End user component is implemented using Microsoft Visual Basic. As can be observed from the INSMware component model, the Front-End user component accesses the Database Component and the Management Interface. The Database Component supplies the Front-End User component with the data needed to display the current system state and a state history while the management interface component is only used to set parameters (e.g. to reset a switch port) of the network element if needed.

Two different design approaches were considered for the implementation: A pure graphical approach that uses overview maps to represent the network structure as known from commercial Network and System Management platforms such as HP OpenView and a more Microsoft Windows Explorer like view showing the network structure as tree. The first approach visualises the network structure very clearly, particularly for experienced users of management platforms such as HP OpenView. Several network management tools (e.g. HP OpenView) use such a map-based interface. Therefore, they are typically well known for experienced users of management tools, but they are also very space consuming and become unclear for really large networks, e.g. if an internetworking map consists of 30 routers and 50 networks.

The tree view is less space consuming and for this reason more suitable for large networks. A problem of this approach is that the network structure is not always hierarchical (e.g. cross links) and therefore the tree representation might not represent the logical and physical network connections as clearly as the established network maps.

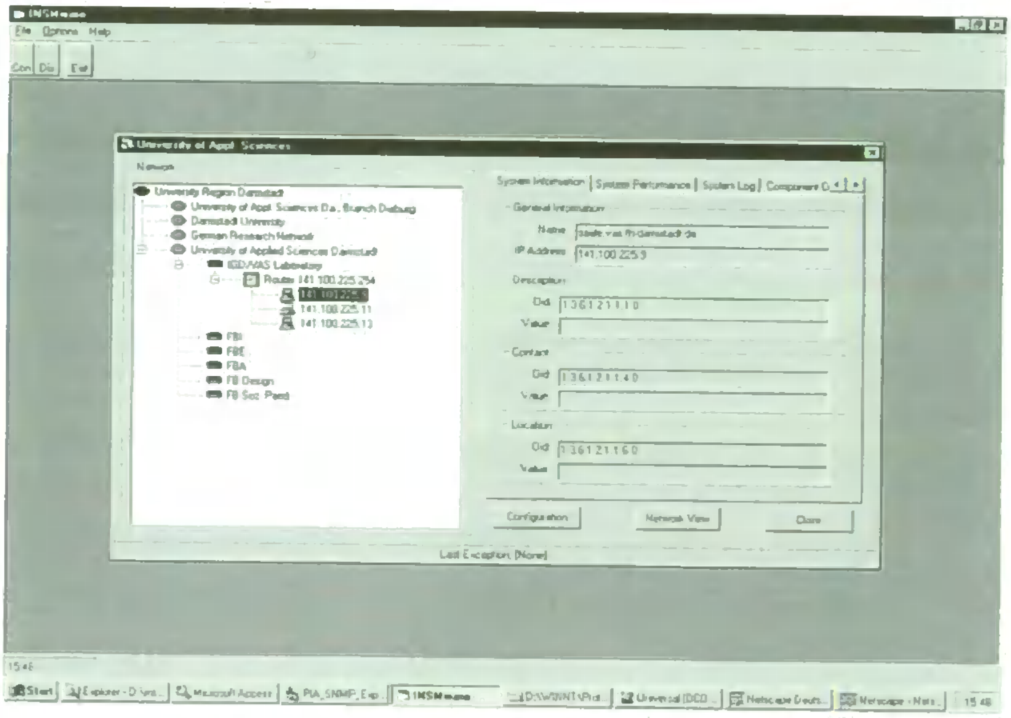
The first prototype of the INSMware Front-End User component is realised as a Visual Basic program, but it has been envisaged to produce a web based front end in order to allow management and configuration from everywhere without installing client software.

8.4.2 INSMware User Interface

This section illustrates the prototypical implementation of the INSMware User Interface.

8.4.2.1 Main User interface

The main user interface of INSMware is shown in Figure 8.14. The network structure is represented by a tree at the left side. At the right side, four tabs show information about the node which is activated at the moment. If desired the user can demand an overview map by pressing the "Network View" button which then shows a network map for the actual network or domain.



Name: [illegible]
IP Address: 141.100.225.9
Description Oid: 1.3.6.1.2.1.1.1.0
Contact Oid: 1.3.6.1.2.1.1.4.0
Location Oid: 1.3.6.1.2.1.1.6.0

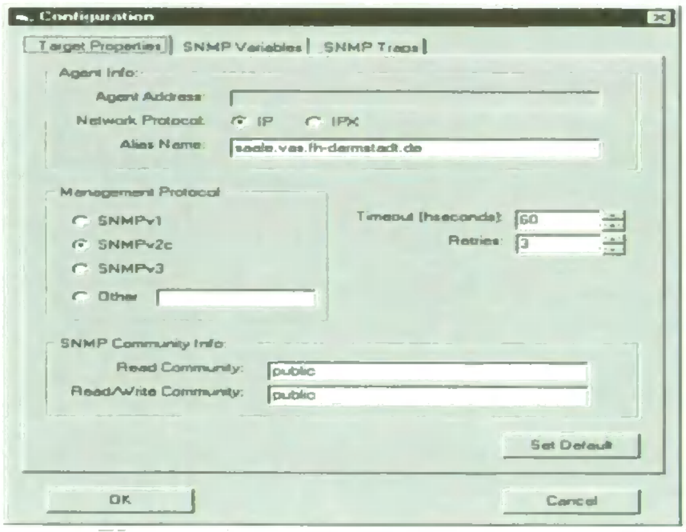
Figure 8.14: Main User Interface

The view of the network is configurable for every single user. Every user can name the nodes in his preferred way and the part of the network, which is shown by the tree, is determinable by the user. In order to enable such flexibility, all the data that is presented by the Front-End is stored in the system database. This concerns the entire network structure represented by the tree as well as the user settings for the configuration.

The four tabs at the right hand side of the GUI provide quick access to the most important information of every network node. The "Information" tab displays the basic information like node name, description, location and contact person at a first sight. The second tab visualises the performance of the network node. The network traffic, utilisation and error rates of the whole network or every single port in case of a router or a switch are presented. The "Log" tab displays an event log for every node. Restarts, port failures and breakdowns of the system will be registered with date and time. The 'Component configuration' tab enables the user to reconfigure the managed object, e.g. to reset a network interface. All the configuration data, including the settings for the notification events, are stored in the Database Component.

8.4.2.2 Configuration

The Front-End supports graphical operation of the management configuration as shown in the next illustrations. This includes the basic configuration of the managed elements and the specification of specific management parameters and INSMware messaging services. In order to make changes to the configuration of the Management Services, the user can start the configuration from the main user interface ("Configuration" button in Figure 8.14). This will display a configuration dialogue with three tabs



Configuration

Target Properties | SNMP Variables | SNMP Traps

Agent Info:

Agent Address:

Network Protocol: [✓ IP] IPX

Alias Name: saale.vas.fh-darmstadt.de

Management Protocol

SNMPv1

[✓ SNMPv2c]

SNMPv3

Other

Timeout (hseconds): 60

Retries: 3

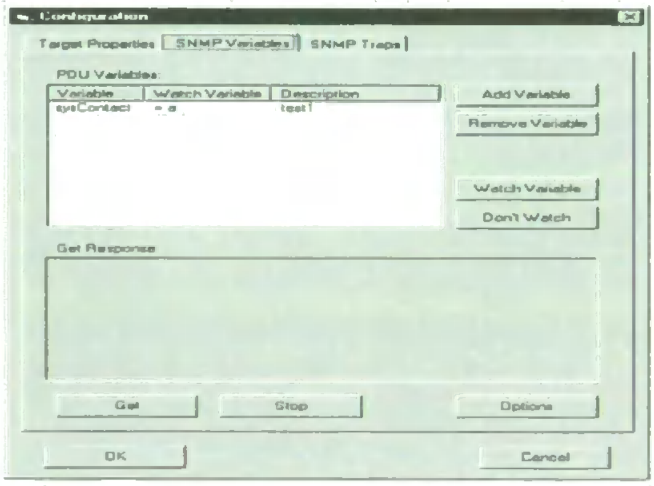
SNMP Community Info:

Read Community: public

Read/Write Community: public

Figure 8.15: Target Properties (Configuration of Management)

Figure 8.15 illustrates the management of the target properties. This includes the setting of the agent address and the specification of the management protocol.



Configuration

Target Properties | SNMP Variables | SNMP Traps

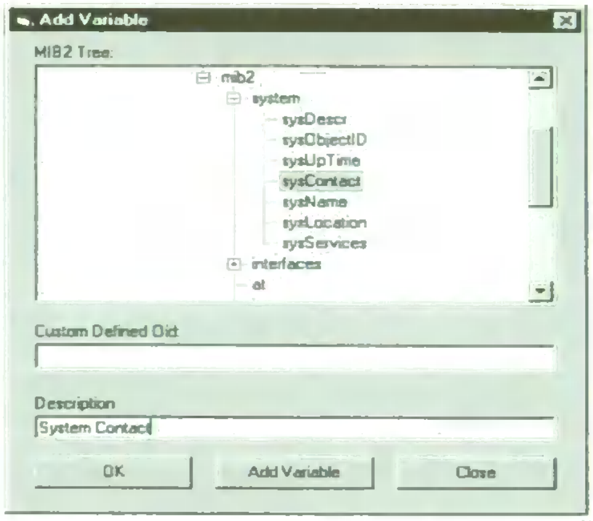
PDU Variables:

Variable	Watch Variable	Description
sysContact	= a	test1

Get Response

Figure 8.16: SNMP Variables (Configuration of Management)

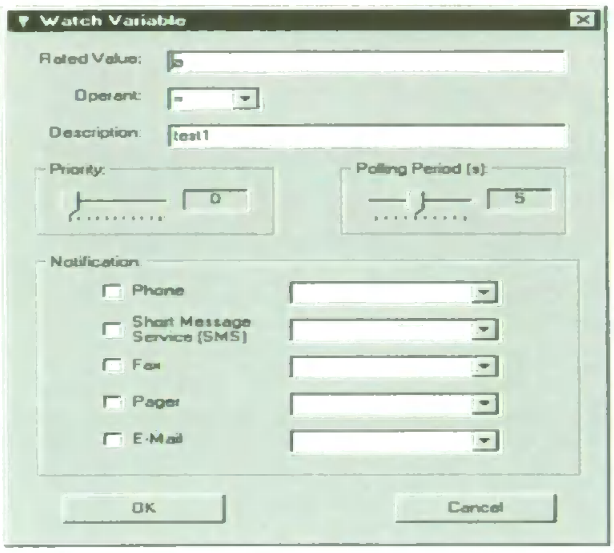
The next tab leads to the configuration of the SNMP variables (see Figure 8.16). A list of specified SNMP variables presents the actual managed SNMP variables. Functionality include adding and removing of variables to be monitored and the specific INSMware configuration of the Management Services.



Description: System Contact

Figure 8.17: MIB Tree Representation (Configuration of Management)

Figure 8.17 illustrates the implemented MIB tree to add a variable. The user can either select a variable using the MIB tree or using a 'Custom Defined OID'. The description is then used to label the variable with a user-friendly name (the default names tend to be cryptic).


Watch Variable

Rated Value: 0

Operant: =

Description: test1

Priority: 0

Polling Period (s): 5

Notification

☐ Phone

☐ Short Message Service (SMS)

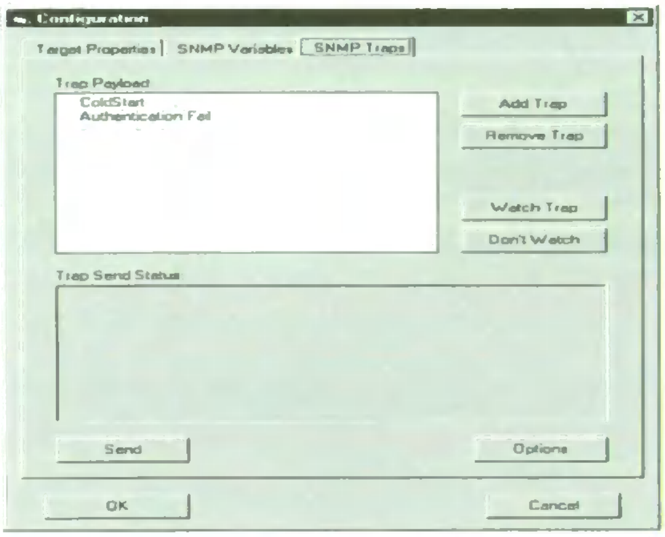
☐ Fax

☐ Pager

☐ E-Mail

Figure 8.18: Thresholds & Notification Services

Figure 8.18 illustrates the INSMware setting of thresholds and notification services. The user configures the monitoring of a SNMP variable and selects the appropriate notification medium from amongst the different communication channels that are provided by the system.


Configuration

Target Properties | SNMP Variables | SNMP Traps

Trap Payload

ColdStart
Authentication Fail

Trap Send Status

Figure 8.19: SNMP Traps (Configuration of Management)

Figure 8.19 illustrates the INSMware setting of SNMP Traps.

8.5 Distribution and utilisation of INSMware components

INSMware allows the distribution of Management Components throughout the network utilising the distribution and communication services provided by the underlying middleware infrastructure. The communication between the components is transparent to the user. A sensible distribution of Management Components will take organisational and geographical considerations as well as traffic patterns into account.

During the course of the project, INSMware was tested and demonstrated in industry and research environments. The tests aimed to demonstrate and evaluate the different services available with INSMware (e.g. communication services) and to demonstrate the distribution of the INSMware infrastructure itself. The reactions from the users made it clear, that usability is a key criteria regarding the acceptance of a new management tool. The Front-End Component, that was designed in conjunction with networking professionals in industry (e.g. design sessions with PanDacom network support engineers) received positive critique during the presentations. The deployment of external communication services (e.g. notification via telephone) was seen as a relevant feature. Furthermore, the distribution of the infrastructure itself was regarded as a key feature (e.g. a Network Integrator such as PanDacom could install Management Interface at a remote customer site to offer remote Management Services to customers). However, it was clearly felt that the prototype needed further extension with respect to functionality and usability.

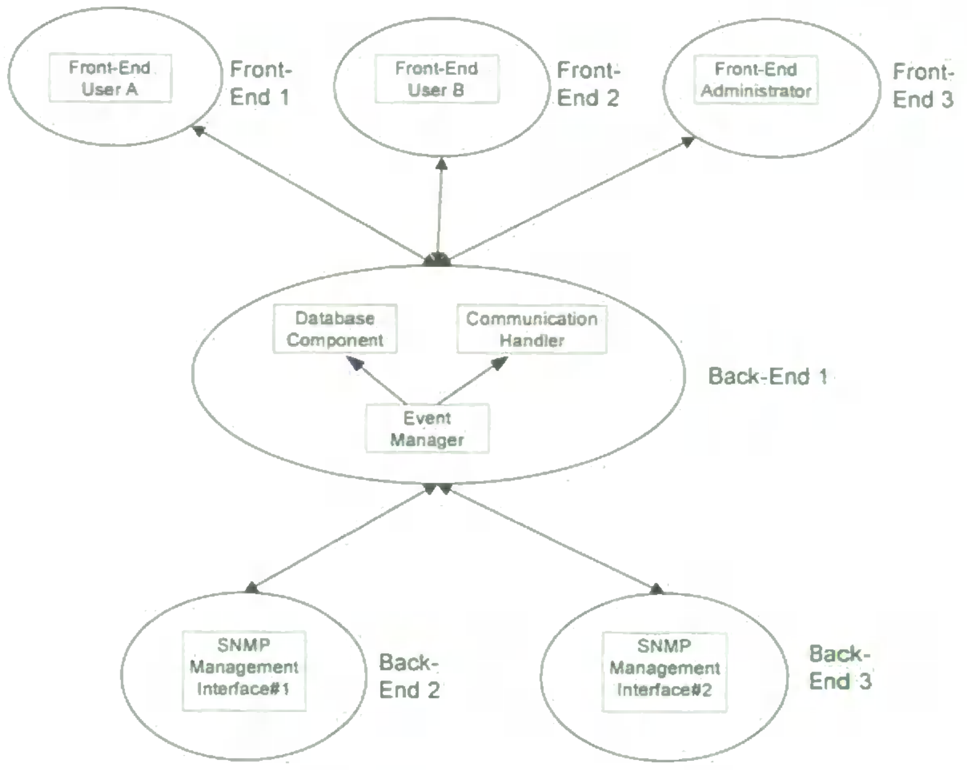


Figure 8.20:INSMware Distribution

Figure 8.20 illustrates the distribution of INSMware components over 6 systems. Prime components for distribution are the Management Interface and Front-End components. The distribution of Front-End components allows different users individual access to the system without the requirement for the full component set. In our example, 2 systems acted as Front-End Users and 1 system acted as Front-End Administrator. The central Back-End system provided the Database Component, Event Manager and Communication Handler. Furthermore, 2 Back-End Systems running the Management Interface Component provided distributed data collection and analysis (i.e. the distribution of Management Interfaces enables the gathering of management related data were it occurs thus reducing management related overhead).

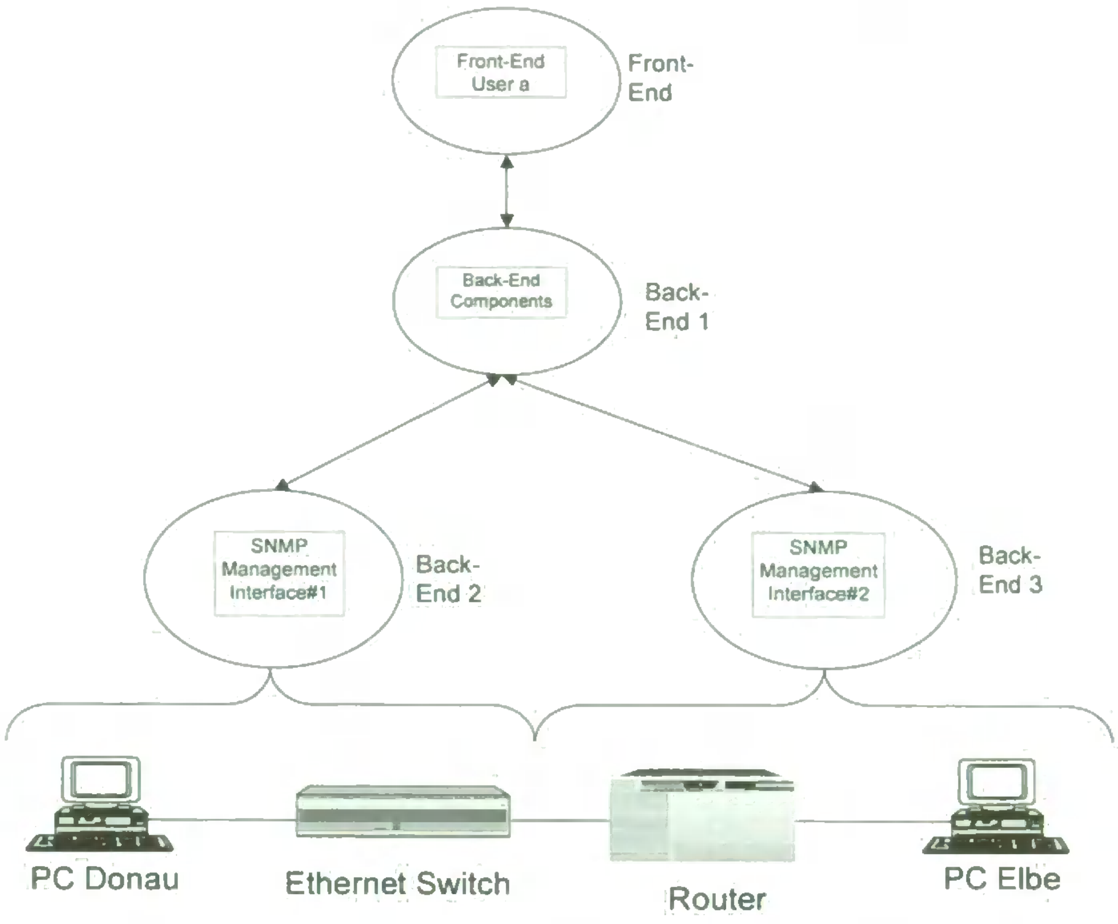


Figure 8.21: INSMware Monitoring of SNMP System States

INSMware can be utilised to gather SNMP related data for the provisioning of integrated Management Services. Figure 8.21 illustrates the usage of a distributed INSMware environment to monitor the availability of an End-to-End connection. Two Management interfaces are being used to gather data from SNMP Managed Objects (for example the SNMP MIB II parameter 'SystemUpTime' can be used to gather general information about the time a system has been running). The data is received and analysed by the local Management Interfaces and then processed by the Back-End Components.

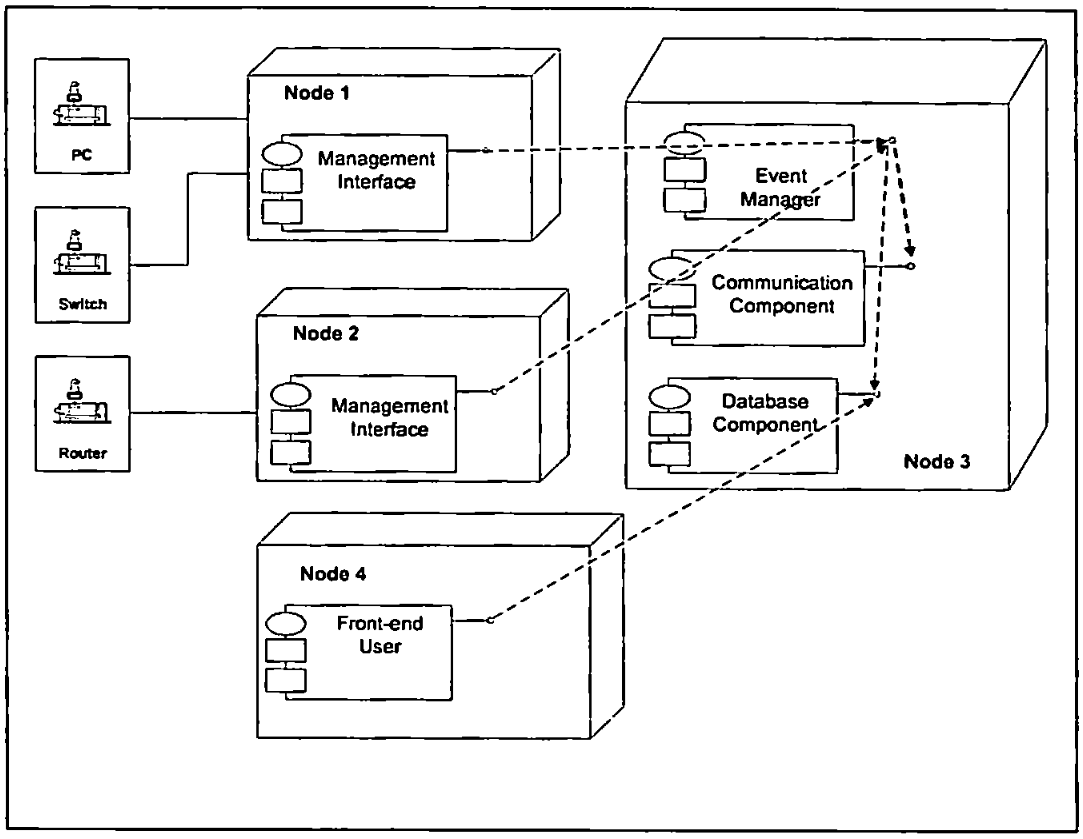


Figure 8.22: INSMware Distribution

Using the INSMware components several distribution scenarios can be realised. In the smallest configuration, all components are installed on one system. In this scenario, remote users would connect to this system to receive all information. In a fully distributed configuration, all components would be installed on different computers. However, in practical terms it is better to leave the Database Component, the Communication Component and the Event Manager on one system. Using this distribution, a good performance can be achieved as this distribution situation avoids heavy traffic through the network connecting the different computers. Furthermore the Management Interface may require a lot of hardware performance and could affect the performance of the overall system. Figure 8.22 illustrates this physical distribution.

8.6 Prototype Validation

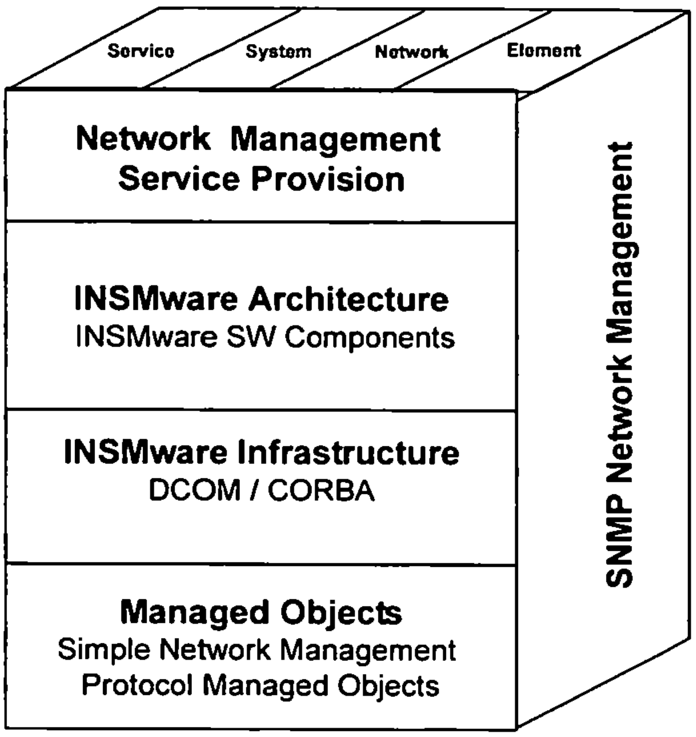


Figure 8.23: Layers of the INSMware Implementation

Figure 8.23 illustrates the different layers of the INSMware prototypical implementation in the context of the overall research programme to outline the scope of the prototype. The prototype can be utilised for the provision of Network Management Services in the context of SNMP Managed Objects. It is composed of the INSMware Software Components running on top of industry standard distribution architectures.

8.6.1 Architecture Evaluation

In order to validate the proposed architecture, it is necessary to return to the 'Architectural Requirements' (see Table 7.1) and to evaluate the INSMware Management Architecture with respect to the outlined requirements.

1. System Distribution

   The INSMware Architecture is comprised of a set of Building Blocks that can be distributed over the network. It is envisaged that the location of a Building Block can be assigned to meet the requirements of a specific Management Service (e.g. Management Interface can be located close or in the corresponding Managed Objects to minimise management related overhead). This design principle is fundamental to the INSMware approach. Applying a Componentware approach based on current industry good practise to implement INSMware Building Blocks, the distribution services provided by distributed systems architectures such as CORBA and DCOM can be used.

2. System Extensibility

   INSMware extensibility has been achieved by introducing Building Blocks for specific purposes. For example the Management Interface Building Block implements a specialised domain (e.g. a specific management protocol) can be replaced by another Management Interface. Furthermore, Building Blocks can be split into several layers that provide domain specific Management Services and generic services.

   These mechanisms allow extensibility in addition to facilitating architecture tailoring, which can be done by adding and distributing further Building Blocks.

3. Compact Architecture

   Given the broad scope of this thesis, which encompasses two distinct areas such as generic Management Architectures and provision of Management Services, it is worthwhile to summarise the results that are relevant for this research and can be applied to other projects. Architectures, as applications, are never complete. In order to support evolution, an architecture for an Integrated Network and System Management System must clearly separate mandatory (i.e. the set of services that are necessary to every INSMware Architecture) and optional components (i.e. the set of specific and optional services). It is not always possible to know in advance the specific implementation requirements or future requirements and herewith the complete set of services a Management System must support.

The INSMware Architecture is implemented using Software Components that have the potential to offer the implemented functions to other applications and to share generic services. This allows incremental application development and avoids existing services and functionality to be re-implemented every time a new application needs to be developed. The proposed INSMware Architecture is rather slim (i.e. 6 Building Blocks can be found in the Architecture) and the different identified domains were assigned to specific Building Blocks.

4. Modular Architecture

The INSMware Architecture is based on a Building Block paradigm. The required functionality was split into a number of Building Blocks in the design stage resulting in a modular architecture. The actual implementation of the Management System is based on distributable units of Building Blocks in the form of Software Components that were assembled to provide Management Services.

5. Based on open standards

The INSMware Architecture is based on generic concepts and was applied to existing standards and technologies. INSMware is fully documented and is not a black-box which uses proprietary technologies. INSMware does not define new management protocols nor does it specify the set of protocols that is suitable for a certain activity. The proposed prototype incorporates the standardised SNMP protocol and is based on industry-standard middleware.

6. Based on generic development methodology

   It follows a generic development methodology that was proposed for the design and implementation of INSMware based systems and it is not dependent on a particular implementation technology (e.g. programming language or distribution middleware). However, the author suggests for future research and development in this area the use of a standard design approach and notation for all stages (e.g. adoption of Catalysis approach based on UML principles as it offers a general modelling notation for the different stages of the development process [Souza1999]).

7. Support of existing development languages and technologies

   In addition to the points mentioned in point 6 above, the fact that the INSMware prototype has been implemented on two different middleware platforms (DCOM & CORBA) demonstrates that the architecture is independent of specific technologies. Moreover, Building Blocks do not rely on specific feature of a language such as C++ and have been implemented using different programming language (i.e. Visual Basic and C++).

8. Integration with 3rd Party Building Blocks

   Further work should focus on the integration with 3rd party products (e.g. commercial Network and System Management Platforms). This issue has not been dropped because none of the available solutions were deemed suitable for use in the context of this project. However, the integration of the INSMware Architecture with commercial Network and System Management Platforms presents certain problems. This is because the INSMware Architecture attempts to solve problems typical of conventional management applications, hence the integration of INSMware into commercial platforms should concentrate on the exchange of Management Service related data. Further work should concentrate on the integration of 3rd Party Building Blocks into the proposed architecture.

9. Reuse of existing Building Blocks

Reuse has been facilitated through the adoption of the Componentware paradigm that enabled the reuse of pre-fabricated Software Components that were developed as research prototypes in the Distributed Application Systems Research Group at the University of Applied Sciences Darmstadt.

INSMware Software Components can provide multiple services or can be designed to facilitate a particular service. Since services can be accessed in a distributed manner, it is not necessary to have the Software Component locally but it is possible to issue remote interactions. This way to reuse services and Software Components is preferable when different geographical locations and systems are involved, because it allows services to be reused without system modification (e.g. without adapting code). Follow-on research should further built upon the pool of reusable INSMware Software Components.

10. Security support

Security is a rather wide and complex field, encompassing several aspects ranging from peer authentication to encrypted communications. Security demands in the context of INSMware can be partitioned into the area of authorisation and access control, and the area regarding security of data transmission. It should be possible to grant access permission for the services of a system and to get access statistics for the different services. Future work on the INSMware system should therefore consider the authorisation of users (e.g. control access to Front-End components).

Furthermore, the data transmission between components of a distributed system has to be encrypted if necessary. Security support varies depending on the specific environments, the required Management Services and the management protocols being used. The integration of security mechanisms that can be included to secure INSMware's system interactions should be further considered in future research.

8.6.2 Prototype Evaluation

In order to validate the implemented prototype, it is necessary to return to the 'Prototype Goals' (see Table 8.1) and to evaluate the prototype using the outlined requirements

1. System Evolution

   INSMware can be adapted to changes and evolutionary changes through the adoption of Componentware principles. Applications that must modify their behaviour due to changing requirements can do so by replacing components with new component versions. Additional Software Components can be linked with the core system and can migrate to other networked machines thus allowing the system topology to evolve and be adapted to new management requirements and network topologies. For example INSMware extensibility has been further facilitated by splitting the Management Interface component into several layers (i.e. management-domain specific component and generic Filter Component). The services provided by the Filter Component within the Management Interface component are general. To integrate Managed Objects that require different management connectivity (e.g. another Management Protocol), new "drivers" can be added. This mechanism allows extensibility in addition to facilitating application tailoring, which can be done by adding and distributing further components.

2. System use and development

Developers demand concepts and architectures that make their tasks easier. In the field of open and distributed systems, it is further influenced by the way installation and configuration are achieved. In the current version of the prototype, the installation and configuration of the individual components must be performed manually (i.e. no "ApplicationWizard" or other automatic installation tool exists).

Users are accustomed to application ease of use. Ease of use is determined by different factors. As INSMware is based on the idea that resources should notify their status to the operator and not the other way round, INSMware monitoring parameters of one system can be reused to incorporate new systems. However, the location of managed objects themselves is not supported by the prototype and can be achieved using systems such as HP OpenView Network Node Manager that provide Auto Topology functions. As far as distribution and maintenance of the INSMware components are concerned, they are facilitated by the Componentware paradigm.

The ease of development is facilitated by the fact that the INSMware Software Component interfaces are rather simple, thus limiting the amount of information that has to cross them. This potentially improves interoperability and development of new Software Components. The simpler design and programming is, the fewer errors are made leading to an increased productivity of the developer.

3. Support of distributed environment

INSMware incorporates facilities for Software Component communication and distribution using the Componentware paradigm. It has been tested for DCOM and CORBA based distributed environments.

However, the proposed approach could further be applied to other distributed systems architectures such as Enterprise Java Beans. This would further require the adaptation of the interfaces and the object activation, interaction and release on source code level.

4. Efficient resource utilisation

Delegating computational intensive processes to appropriate systems (e.g. high performance servers) enables load distribution. To load and execute components on demand allows applications to efficiently use system resources. This contributes to the general enhancement of overall system performance and can potentially optimise the use of system resources and avoids using unnecessary resources in an INSMware system.

INSMware has been used in conjunction with standard PCs and showed reasonable performance when managing small networks (e.g. a standard PC with 128 MB RAM and a 400 MHz Pentium-II Processor running all INSMware Components under DCOM was used to monitor a 5 SNMP variables on 1 Router, 1 Ethernet Switch and 1 PCs in the VAS Lab showed an average CPU utilisation of under 20%).

The main aim of this research was to investigate the workability of the proposed management concepts with industry standard technologies. However, in a large scale implementation with hundreds (or thousands) of Managed Objects, the performance issues would certainly become a key criteria. Further work should focus on performance analysis (i.e. analyse the performance and system utilisation of the overall system and particular INSMware components in the management of "real-life" networks) and relevant distribution overheads (i.e. analyse resulting overhead in distributed INSMware implementations).

5. System portability and generality

Identical environments and users do not exist (i.e. they are always at least slightly different). Every user always has very specific needs. Therefore, an architecture must allow applications to be built that can be customised by the end-user rather than only by developers. The conceptual separation of the Front-End System into Front-End User and Front-End Administrator enables the separation of roles (e.g. with respect to the configuration of the system). Furthermore, users can modify existing Management Services or create new Management Services using the Graphical User Interface.

For future versions, a WEB based Graphical User Interface would enable users to access the system using a standard interface without the requirement for the installation of client software.

6. Support of Internet standards

Mechanisms and protocols to distribute and access INSMware Software Components over the Internet can be applied to the proposed system (e.g. CORBA uses IIOP to provide standard Inter-ORB communication over the Internet [OMG1998]). However, due to the ever and rapidly changing nature of trends and technologies in this area, further work should consider the application of new technologies. Over the last few years, new technologies have been introduced that aim to provide solutions tailored to the requirements of Web-based distributed computing (e.g. Web Services based on SOAP that is utilising http and XML [Platt2002]). The author believes that a number of technologies and protocols that enable access to remote Software Components will coexist in the future. It is most likely that the current hype about Web Services will diminish and architectures such as CORBA and Web Services will coexist (i.e. potentially increasing heterogeneity for the foreseeable future). Web Services utilising SOAP might become a dominant force to communicate with the 'outside world' (i.e. the Internet) and CORBA will be used predominantly to communicate in the 'inside world (i.e. in Intranets); when necessary, gateway will be used to integrate the different worlds.

Further potential regarding the distribution of INSMware functionality lies in the provision of Web based Graphical User Interfaces. An INSMware GUI based on a Web Browser to access Management Data stored on a web server had been envisaged in the early stages of the project but was dropped due to time restrictions. The adoption of an XML based approach would allow the usage of the existing Internet infrastructure (e.g. usage of existing web browsers and web servers, usage of existing http security mechanisms, usage of protocols such as SOAP) and could further enhance the presentation of Management Services (e.g. XML stylesheets allow the same management information to be presented and manipulated by users in different ways). However, potential limitations and shortcoming of such an approach might result in limited market acceptance and restricted functionality (e.g. large communication overhead due to encoding, currently lack of common distributed services such as location and transaction standards, lack of industry-wide standards).

7. Support of management standards

INSMware does not impose particular limitations on the way Management Systems have to be implemented. This facilitates the implementation of management standards that can be quite dynamic and complex and potentially difficult to be fully supported. The implemented INSMware prototype supports SNMP that has been standardised by the IETF. It has been used in conjunction with DCOM and CORBA. However, the continuous evolution and modification of these distribution technologies requires an INSMware provider to also provide numerous versions of the system and to continuously update and modify the system.

Further potential lies within the adoption of a common information model for the representation of management related data (e.g. CIM) and Internet based access and presentation approaches (e.g. http and XML).

8. System scalability and performance

INSMware Software Components can be distributed to scale according to the actual environment to meet the specific requirements. However, the lack of extensive testing and performance limitations make the current prototype no a viable option for large-scale installations.

Performance is always an issue, although opinions regarding this aspect diverge. An architecture that produces elegant applications with poor performance is useless. Usually the best compromise is to balance design and performance in order to obtain a good design with good performance. On the other hand an architecture that delegates too many key decisions to the developer is not good. This is because this policy will increase the possibility that applications developed separately cannot inter-operate to share functionality. It has been shown that the proposed architecture can be applied to provide SNMP-based Management Services using different middleware technologies and in different distribution scenarios using standard PCs. Future tests should focus on performance issues (i.e. how will the system perform given an increased number of managed objects and operations to handle) and should consider overheads (i.e. how much overhead traffic is induced when INSMware is distributed).

9. Support of specific platform features

As the main objective of this prototype was to be general, no specific platform features were addressed (e.g. the standardised SNMP protocol has been implemented). Although this requirement may be relevant for commercial products, it has not been addressed by the prototype.

10. System administration

Ease of installation, operation and customisation can be achieved through the usage of tested and established industry-standard tools for the installation, operation and customisation of Building Blocks based on Software Components. However; the current version of the prototype has not automated installation facilities (i.e. the installation and configuration of the individual components must be performed manually).

To conclude the analysis and evaluation, a number benefits of a Componentware based INSMware Management System can be summarised:

- INSMware Components may be reused;
- Additional Components may ultimately be bought-in from third party developers;
- System solutions can become more modular and thereby easier to construct, maintain and operate.

8.7 Conclusions

This chapter has sought to illustrate how INSMware and the conceptual design defined, was implemented, demonstrated and evaluated during the implementation and evaluation process. It has discussed the findings of the practical research and, thus, the value of the design formulated. The next and final chapter moves on to review the overall achievements of the review programme and looks towards the future refinement and embodiment of the conceptual designs defined. It also looks at the future of communication and Management Services.

Chapter 9
Conclusion and Outlook

The current state of the art in networking technologies and various solutions to Integrated Network and System Management have been introduced and analysed as part of this research. The analysis of the current state of the art has revealed that no single architecture and technology is dominant and suited to meet all the requirements. A new management approach to alleviate the current limitations has been proposed and a prototype based on the conceptual principles has been successfully implemented. This allowed the author to conclude that the research has made a positive contribution to resolving the current difficulties faced by network and systems managers.

Yet certain topics and trends that have influenced the research area (and will do so in the future) are not fully solved. Either the underlying new technologies have not been established themselves, the relevant solutions are still at the discussion stage or the impact on new services is not predictable. Therefore, these aspects will be refined and visions regarding future developments will be discussed. The nature of these predictions forces the author, to remain subjective in the selection of topics as well as in the assessment of these different aspects.

9.1 Achievements of the Research

The main aim of the research undertaken was to provide a comprehensive concept for Integrated Network and System Management. A number of constituents contribute to meet this aim:

- An assessment of the current state of the art and isolation of requirements for a new management approach.
- Conceptual design of a Management Framework based on Building Blocks that can be used as a reference model. It adapts a high-level logical model in order to accommodate the range of requirements and environments applicable to Integrated Network and System Management.
- Specification of the INSMware Development Methodology and realisation principles to support the development process of the proposed management approach. A common set of methodological guidelines further promotes interoperability and efficient development.
- Application of the Componentware approach to implement INSMware Building Blocks to provide a prototypical implementation of an open and scalable system. It exploits existing and emerging Componentware technologies for the realisation of the conceptual principles.

The presented research provides a fundamental basis for the realisation of a new approach to Network and System Management. It has been illustrated that INSM development can benefit from a structured modelling approach and that Componentware based architectures for the provision of Management Services can offer a number of benefits. INSMware Components may be re-used and system solutions will become more modular and thereby easier to construct and maintain.

Equally, the research has given rise to a number of reviewed papers and has been presented at a number of conferences. As such, it is considered that the research has made a valuable contribution to the field of Integrated Network and System Management.

9.2 Limitations of the Research

The research outlined the principles for the implementation of Building Blocks in the form of Software Components and demonstrated its principal advantages. However, it merely illustrated the principles rather than proving a solution exclusively built using "Off-the-Shelf" Building Blocks from external sources, as a rather traditional development approach was utilised for the implementation of the prototype system (i.e. development of new Software Components and reuse of Software Components available in the research environment). Future focus of this research should further emphasise generality (i.e. provide solutions using "Off-the-Shelf" Building Blocks). A number of further limitations can be isolated:

- The Conceptual design would further benefit from additional refinement. The conceptual design adopts a high-level model to accommodate the requirements applicable to INSM and identifies the different domains and interfaces. However, to further extend the conceptual model and to provide more specific guidelines for the realisation, the required functions and interfaces should be further specified.

- The developed prototype provides rather limited functionality. In order to use the system in a productive environment, further functionality is required to facilitate the provision of Management Services and analysis of Management Service related data.

- A rather limited amount of performance testing and analysis has been undertaken with respect to the developed prototype. As the main aim of this research was to test and evaluate the architectural aspects of the proposed approach, the research focused on evaluating interoperability and connectivity on an application level.

- Because of the structure of the overall research programme, and also due to the limitations in resources and manpower, the research undertaken does not fully utilise the latest technologies and developments an the areas of INSM and Software technologies. However, references and investigations to further apply the research were made when appropriate.

9.3 Suggestions for Future Research

In further developing the research aims, a number of suggestions for future research are proposed:

- Further refinement of the conceptual design (e.g. further specification of interfaces). A prime target for future work would be the MAPI-E Interface to address interoperability issues between different systems (e.g. between Management Engines).
- Further align development methodology with established and emerging development approaches and notations.
- Adaptation to other middleware technologies (e.g. Enterprise Java Beans).
- Performance analysis of the INSMware prototype to further investigate the impacts and requirements of the proposed solution (e.g. to investigate the communication overhead that is introduced through the usage of middleware technologies).
- Extend functionality of the existing prototype for the provision of extended Management Services. The provision of a Web based GUI would further facilitate the access of remote users to the prototype.
- Further integration and testing of Management Protocols (e.g. test INSMware and SNMPv3).

9.4 Development of Integrated Network and System Management

In addition to "standard" services that must be provided by a management system, the requirement to offer customised services for users with specific requirements (e.g. specified in a SLA) will increase. This will substantially increase the importance of the area of Service Management. To cope with the challenges, Management Systems will be composed of a range of components that perform the required Management Services [Lewis2002]. Management Service operations that enable a user controlled activation of communication services are also conceivable. It is also worth considering workflow management front-ends for user's in order to channel and automate the various processes. Furthermore, changes to configurations and topologies (physically as well as logically) will continue to be made with great frequency. There will also be a change in user behaviour, particularly regarding the mobility of users. New types of users, including those who previously had little contact with computers and networked systems, will be using the communication networks and services (e.g. mobile IP) and will incorporate new management challenges. However an exploration of the Management Services and technologies that will be needed for current and future environments is still the subject of research [King2000] [Lewis2002] [Sloman2002] [Salsano2002].

System Management is mainly being resolved on an applications-specific basis. General methodologies and approaches for the management of distributed applications and services are still at the development stage (e.g. license and performance monitoring) [Hauck2001] [Debussmann2002] [Kreger2001] [Ressmann2001]. Regarding the more specific Application Management, it has to cover the entire lifecycle of a distributed application. Service Management will gain in importance compared to the management of resources and network elements. Uniform concepts for solutions are not yet available and existing approaches only touch upon certain aspects [Jander1999] [Dornan2001].

The integration of management tools and technologies is an issue that has still not been resolved satisfactorily [Chen2000] [Lewis2002]. The integration of Management Services is becoming more and more critical because of new usages and resources in networked systems and because of the diversity of management architectures. One challenge is at which level integration should take place (e.g. at the management tool level), what should be integrated (e.g. user interface, events or data) and what the integration interface should look like. Integration is an extremely important issue. Until now, only partial solutions have been available for this area where the lack of integration of Management Services for the concrete IT infrastructures resulted in higher costs and personnel requirements. It is not only a system or technology oriented problem but also an organisational problem [Mayerl2001].

These trends inevitably have an impact on the development process. Since a management instrumentation of existing resources is complicated and expensive, there will be a move to incorporate the management requirements in the development process (i.e. to take the management aspects into account at the stage of resource and service development). This has to apply particularly for application development. Developers have to become even more sensitive toward these management problems. The main reason for this is that the dynamic of some areas of existing and future IT services with great potential is being slowed down or delayed considerably due to a lack of management concepts and services. Examples include:

- The state of current network infrastructures and communication services falls far short of what is technically possible of being achieved. Some of the reasons for the relatively low deployment of state of the art enabling technologies can be seen in the management problems and the slow process of standardisation.

- The progress of Quality of Service sensitive services (e.g. Voice over IP or videoconferencing) is partly being slowed down because of the lack of end-to-end Management Services. These are required before corporations will adopt these services on a large scale (e.g. before Voice over IP will be a serious alternative to the established legacy phone services).

Rising costs in maintaining the IT infrastructures has been one of the main reasons why cost reductions anticipated with the evolving structures have not been realised [Norriss2000] [Lewis2000]. Furthermore, one outcome has been an increased heterogeneity of environments. The problems encountered in the context of management provision are promoting a concentration of technologies and manufacturers. The addition of cost considerations and the technical aspects illustrate the problem space for the provision of Integrated Management Services.

Management can either be seen as a driving force or a blocking force for the development of the services and underlying technologies. A comprehensive Management Architecture is a prerequisite for the provision of optimised services and enables the efficient deployment of these services. At the same time, it can slow down the introduction of new technologies and hence slow down the deployment of new services if management issues are not addressed carefully at the outset. Hence, it is can be regarded as essential to have an Integrated Management Architecture to achieve flexible IT services. Management issues have to be taken into account during the design of each new technology and service. The same applies to the definition of business processes in connection with the planning and operation of IT infrastructures. Any disregard of these basic rules will increase costs and will affect the viability of the provided services.

What can be anticipated is a management instrumentation of all resources, particularly the clients for applications that are seldom instrumented today. End-to-End Management, Quality Of Service Management and the Management of Service Level Agreements will require an Integrated Management Architecture based on generic concepts. The INSMware approach provides a conceptual framework on which to build management systems and offers a step towards the provision of Integrated Network and System Management.

References

[Aaron1999]	Aaron, R. ; Decina; M. ; Skillen, R. Electronic Commerce: Enablers and Applications. *IEEE Communications Magazine*. Vol. 37, No. 9, September 1999.
[Abeck1999]	S. Abeck, C. Mayerl. Modeling IT Operations to Derive Provider Accepted Management Tools. *Proceeding of IM 1999*. May 1999.
[Abbott1992]	Abbott, E. A. *Flatland – A Romance of Many Dimensions*. Dover Publications Inc. New York, 1992.
[Adamopoulos2002]	Adamapoulos, Dionisis; Pavlou, Geeorge; Papandreou. Continuous media support in the Distributed Component Object Model Computer Communications. Vol. 25, January 2002.
[Alfano1995]	Alfano, M. *A Quality of Service Management Architecture (QoSMA): a preliminary study*. Technical Report TR-95-070, International Computer Science Institute. December 1995.
[Appledorn93]	Appledorn, M. ; Kung, R. ; Saracco, R. TMN + IN = TINA. *IEEE Communications Magazine*. March 1993.
[Apptitude2000]	Apptitude. MeterFlow Network Design Decision Engine. *Apptitude Technical White Paper*. January 2000.

[ATMForum1998] ATM Forum. ATM Forum Building Awareness in Europe. *ATM Forum 53 Bytes Newsletter*. March 1998.

[ATMForumILMI96] ATM Forum. *Integrated Local Management Interface (ILMI) Specification 4.0*. af-ilmi-0065.000. 1996.

[Bai1999] Bai, Guangwei. *Load Measurments and Modeling for Distributed Multimedia Applications in High Speed Networks*. LIT Verlag, Hamburg, 1999.

[Barillaud1997] Barilaud, F. , Deri, L. , Feridun, M. Network Management Using Internet Technologies. *Proceedings of IM 97*. Chapman-Hall. May 1997.

[Bauer 1988] Helen A. Bauer, John J. Kultzer, Edward G. Sable. Designing Service Independent Capabilities for Intelligent Networks. *IEEE Communications Magazine*. December 1988.

[Berndt1999] Hendrik Berndt et al. *The TINA Book*. Prentice Hall. 1999.

[Bhagavath1997] Bhagavath, V. Technical Issues in Provisioning High-Speed Interactive Data Services Over Residential Networks. *IEEE Networks*. January/February 1997.

[Bhoj2001] Bhoj , P. , Singhal, S. , Chutani, S. SLA Management in Federated Environments. *Computer Networks*. Volume 35, No. 1. January 2001.

[Bleimann1996] Bleimann, U. *Projekt Systementwicklung: Integriertes Netzwerk und System Management*. Internal Report, University of Applied Sciences Darmstadt. March 1996.

[Booch1996]	Booch, G., Jacobson, I., Rumbaugh, J. *The unified modelling language for object-oriented development, version 0.91.* September 1996.
[Booch1997]	Booch, G. *Software as a Strategic Weapon.* Rational Software. 1997.
[Box2000]	Box, Don et. al. *Simple Object Access Protocol (SOAP) 1.1.* World Wide Web Consortium. May 2000.
[Braden1994]	Braden, R., Zhang, L. et al. *Integrated Services in the Internet Architecture: An Overview.* Request for Comments 1633. 1994
[Brandt2001]	Brandt, R. , Hörtnagl, C. , Reiser, H. Dynamically Adapteble Mobile Agents for Scaleable Software and Service Management. *Journal of Communications and Networks.* December 2001.
[Brooks1995]	Brooks, F. *No Silver Bullet Refired - In the Mythical Man Month (2nd Edition).* Addison Wesley. 1995.
[Brown1998]	Brown, J. *Software Strategies – The Component Decision.* Forrester Report. Forrester Corp. Reports. 1996.
[Brownlee1999]	B. Brownlee, C. Mills, G. Ruth. *RFC 2722: Traffic Flow Measurement Architecture.* Request For Comments 2722. October 1999.
[Butler1998]	The Butler Group. *Component-based Development. Application Delivery and Integration Using Componentised Software – Strategies and Technologies.* Butler Group Report. 1998.

[Campbell1993] Campbell, A. et al. Integrated Quality of Service for Multimedia Communications. *Proceedings of IEEE Infocom 1993.* San Francisco, CA. April 1993. Pp. 732-739.

[Cekro1997] Cekro, Z. Using the SunNet Manager Platform to monitor European ATM activities. *Proceedings of the 2nd ATM Symposium.* Brussels, November 21, 1997

[Cerami2002] Cerami, Ethan. *Web Services Essentials.* O'Reilly UK. 2002.

[Chan1998] Chan, D. L. S. , Chanson, S. T. Scalability Support for Multiparty Multimedia Communications. *Multimedia Systems.* Vol. 6, pp. 75-78, 1998.

[Chapman1994] Chapman, M. , Montesi, S. Overall Concepts and Principles of TINA. *TINA Baselines Document TB_MDC.018_1.0_94.* TINA, 1994.

[Chappel1992] Chappell, D. (1992). The OSF Distributed Management Envirment. *ConneXions.* October 1992.

[Chatt1997] Chatt, T. R. , Curry, M. , Holberg, U , Seppa, J. An Object-Oriented API for GDMO, CMIS and ASN.1. *Proceedings of IM 1997.* May 1997.

[Chen2000] Chen, Graham; Quinzheng, Kong. Integrated Management Solution Architecture. *Proceeding of IM 2000.* April 2000.

[Choi2000] Chai, M.-H. , JU, H.-T. , Cha, H.-J. , Kim, S.-H. , Hong, W.-K. An Efficient Embedded Web Server for Web-based Network Element Management. *Proceedings of IM 2000.* April 2000.

[Chorafas1997]	Chorafas, D. High *Performance Networks, Personal Communications and Mobile Communications.* MacMillan Press. 1997.
[CIMXML1999]	Distributed Management Task Force, Inc. *Specification for the Representation of CIM in XML. Specification. v2.0.* DMTF Specification. 1999..
[Clark1997]	Clark, D. , Wroclawski, D. *An approach to service allocation in the Internet.* Request For Comments. 1997.
[Clark2000]	Clark, Elizabeth. ATM: Alive and Well on the WAN. *Network Magazine.* May 2000.
[Cox1990]	Cox, B. Planning the Software Industrial Revolution. *IEEE Software.* November 1990.
[Dassow1997]	Dassow, H. Lehr, G. SNMP and TMN: Aspects of Migration and Integration. *Proceedings of ISN 1997.* Springer Verlag. May 1997.
[DATACOM1997]	Data Communications. Market Forecast 1998. *Data Communications Magazine.* December 1997
[DATACOM2001]	Data Communications. Market Forecast 2001. *Data Communications Magazine.* January 2001.
[Davison1999]	Davison, R. Turner, T. A practical Perspective on TMN Evolution. In *Proceeding of the 6th International Conference on Intelligence in Service and Networks.* Springer-Verlag. April 1999.

[Debussmann2002] Debusmann, M. ; Schmid, M. ; Kröger, R. Generic Performance Instrumentation of EJB Applications for Service Level Management. *Network Operations and Management Symposium (NOMS) 2002, Florence, Italy.* 15.-19. April 2002.

[Denning1995] Denning, Adam. *OLE controls inside out — the programmer's guide to building componentware with OLE and the Component Object Model.* Microsoft Press, Redmond, Washington, 1995.

[Deri1997] Deri, L. Ban, B. Static versus Dynamic CMIP/SNMP Network Management using CORBA. *Proceedings of ISN 97.* Springer Verlag. May 1997.

[DMTF2000] Distributed Management Task Force, Inc. *Common Information Model (CIM) Specification. v2.5.* DMTF Specification. 2000.

[Donnelly1998] Donnelly, William (Ed.). *ACTS Project AC 225: FlowThru Deliverable 3: System Specification and Models.* ACTS Project AC 225. November 1998.

[Dornan2001] Dornan, Andy. Network Management In A Crisis. *Network Magazine.* July 2001.

[EC1999] European Comission. Information Society Technologies: 2000 Workprogramme. *IST Conference.* Helsinki, November 1999.

[ECDirective1990] Commission of the European Union. Commission Directive of 28 June 1990 on competition in the markets for telecommunications services. *Journal of the European Communities.* No. L192, 24 July 1990.

[ECDirective1997] Commission of the European Union. Commission Directive 97/51/EC of 6 October 1997 for the purpose of adaptation to a competitive environment in telecommunications. on competition in the markets for telecommunications services. *Journal of the European Communities*. No. L41 12 February 1997.

[Eddon1998] Eddon, Guy; Eddon, Henry. *Inside Distributed COM.* Microsoft Programming Series, Microsoft Press. 1998.

[Emanual1994] Emanuel. Open Management – Addressing Real Business Needs". Proc. *IEEE Network Operations and Management Symposium.* February 1994

[Emmerich2000] Emmerich, Wolfgang. *Engineering Distributed Objects*. John Wiley and Sons Ltd. 2000.

[Fan1999] Fan, Z. , Lee, E. S. Pricing for Differentiated Services. *Proceeding of IDMS 1999.* Toulouse, France, October 1999.

[Feldman2001] Feldman, Steve. Reality Check: IETF meets Network Operators. *The Simple Times: The Quarterly Newsletter of SNMP Technology, Comment, and Events.* Volume 9, Number 1, December 2001.

[Ferguson1992] Ferguson, Eugene S. *Engineering the Mind's Eye.* MIT Press. 1992.

[Feridun1999] Feridun, M. ; Kastelijn, W. ; Krause, J. Distributed Management with Mobile Components. *6th IFIP/IEEE International Symposium on Integrated Management.* May 1999.

[Flegkas2002]	Flegkas, Paris; Trimintzios, Panos, Pavlou, George. A Policy-Based Quality of Service Management System for IP DiffServ Networks. *IEEE Network*. March/April 2002.
[Florissi1994]	Florissi, P. *QuAL:Quality Assurance Language. Tech. Report CUCS-007-94*. Columbia University. March 1994.
[FlowThru2000]	Lewis, Dave. *Implementing Integrated Management Business Processes using Reusable Components – A Report on FlowThru Project*. FlowThru Consortium. February 2000.
[Fluckiger1995]	Fluckiger, F. *Understanding Networked Multimedia; Applications and Technology*. Prentice Hall. 1995
[Forrester1996]	Forrester Corporation. *The Forrester Report – Internet Computing Voyage*. Forrester Corp. Reports. 1996.
[Forrester1999]	Forrester Corporation. *The Forrester Report – Internet Computing Voyage*. Forrester Corporation. 1999.
[Fraunhofer1996]	Fraunhofer Institute. Monitoring of CORBA-based Applications: The CORBA-Assistant, Release 1.1. *Fraunhofer Institut fuer Informations- und Datenverarbeitung White Paper*. 1996.
[Freestone1996]	Freestone, David, Richardson, Tony, Whittle, Ben. Distributed Processing – Managing the Future. *British Telecommunications Engineering Magazine*. Vol. 15, April 1996.
[Froitzheim1997]	Froitzheim, K. *Multimedia Kommunikation: Dienste, Protokolle und Technik für Telekommunikation und Computernetze*. D Punkt Verlag. 1997.

[Furley1997]	Furley, N. The BT Operational Support Systems Architecture. *BT Technical Journal.* Vol. 15, No 1, January 1997.
[Gamma1995]	Gamma, E. , Helm, R. , Johnson, R. , Vlissides, J. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison Wesley. Reading, MA, 1995.
[Gaspoz1995]	Gaspoz, J. P. , Gbaguidi, C. , Meinkohn, J. VPN on DCE: From Reference Configuration to Implementation. *Proceedings of ISN 95.* Springer-Verlag. October 1995. .
[Geihs1995]	Geihs, Kurt. *Client/Server Systeme – Grundlagen und Architektur.* Thomson's Aktuelle Tutorien. International Thomsosn's Publishing GmbH. 1995
[Gheti1997]	Gheti, I. G. *Networks and System Management Platforms Analysis and Evaluation.* Kluwer. 1997.
[Gibbs1994]	Gibbs W. Software's Chronic Crisis. *Scientific American.* September 1994.
[Glitho1995]	Glitho, R. H. , Hayes, S. Telecommunications Management Network: Vision vs. Reality. *IEEE Communications Magazine.* March 1995.
[Glitho1996]	Glitho, R. H. , Hayes, S. Approaches for integrating TMN in Legacy Networks: A Critical Look. *IEEE Communications Magazine.* Sept. 1996.
[Goldszmit1993]	Goldszmidt, G. On Distributed System Management. *Proceedings of the 3rd IBM/CAS Conference.* Toronto, Canada, October 1993.

[Griffin1996]	Griffen, D. (Ed.). Integrated Communications Management of Broadband Networks. *Crete University Press.* 1996.
[Gutschmidt1995]	Gutschmidt, M. Neumair, B. Integration von Netz- und Systemmanagement. *Proceedings of the 3. Fachtagung ueber Arbeitsplatzrechensysteme.* Hannover, May 1995.
[Hauck2000]	R. Hauck and I. Radisic. Monitoring Application Service Performance – Classification and Analysis of Existing Approaches. *Workshop of the OpenView Association (OVUA 2000).* Santorini, Greece. June 2000.
[Hauck2001]	R. Hauck and I. Radisic. Service Oriented Application Management – Do Current Techniques meet the Requirements? *New Developments in Distributed Applications and Interoperable Systems, Proceedings of the 3rd IFIP International Working Conference (DAIS 2001).* Kluwer Academic Publishers. September 2001.
[Hawkins1992]	Hawkins, Joyce M. (Editor). *The Oxford Reference Dictionary.* Clarendon Press, Oxford. 1992.
[Hegering1999]	Heinz-Gerd Hegering, Sebastian Abeck, Bernhard Neumair. *Integrated Management of Networked Systems.* Morgan Kaufmann Publishers. 1999.
[Hehmann91]	Hehmann, D. et al. Implementing HeiTS: Architecture and Implementation Strategy of the Heidelberg High-Speed Transport System. *Proceedings of 2nd International Workshop Network and OS Support for Digital Audio and Video.* IBM, ENC, Heidelberg, 1991.

[Hock2000]	Hock, Thomas. *Human-Computer Interaction via Telephone.* MsC Thesis, Cork Institute of Technology. 2000.
[Hofman1998]	Hofmann, H.D. , Stynes, J. Implementation Reuse and Inheritance in Distributed Component Systems. *Proceedings of Twenty-Second Annual International Computer Software and Applications Conference (COMPSAC'98), Vienna/Austria.* 1998
[Hofmann2000]	Hofmann, Holger. *Software Component Reuse by Adaptation.* PhD Thesis, Cork Institute of Technology. 2000.
[Hong1999]	Hong, J. W. K. , Kim, J. S. , Park, J. K. A CORBA-Based Quality of Service Management Framework for Distributed Multimedia Services and Applications. *IEEE Network.* March/April 1999.
[Horowitz1993]	Horowitz, B. M. *Strategic buying for the future.* Libey Publishing 1993.
[Hosoon2000]	Hosoon, K. , Forslow, J. , Park, J. Web-Based Configuration Management Architecture for Backbone Router Networks. *Proceeding of IM 2000.* April 2000.
[Irmscher1997]	Irmscher, Klaus; Richter, Klaus; Rudolf, Steffen: Temporaere Einbindung mobiler Clienten und Optimierung der Dienstauswahl in einem verteilten System. Kommunikation in Verteilten Systemen. *Proceedings of GI/ITG-Fachtagung Braunschweig, Germany.* February, 1997.

[ISO1987] ISO. *ISO 8824-1987:Information Processing Systems – Open Systems Interconnection – Specification of the Abstract Syntax Notation One (ASN.1)*. ISO Specification.1987.

[ISO1995] ISO. *Quality of Service Framework*. ISO/IEC JTC1/SC21/WG1 N9680. 1995

[ITU1984] ITU-T Recommendations. *COM XVIII-228-E*. ITU, Geneva. March 1984.

[ITU1989] ITU-T Recommendation X.700 | ISO/IEC 7498-4. *Information Processing Systems - Open Systems Interconnection - Basic Reference Model - Part 4: Management Framework*. Geneva, 1989.

[ITU1991] ITU-T Recommendation X.710. *Common management information service definition for CCITT applications*. Geneva, 1991.

[ITU1992] ITU-T Rec. M.3010. *Principles for a Telecommunications Management Network*. ITU Recommendation. Geneva, Switzerland, 1992

[ITU1992] ITU-T Recommendation X.701. *Information technology – Open Systems Interconnection – Systems management overview*. Geneva, 1992.

[ITU1992b] ITU-T Recommendation X.720 (1992) | ISO/IEC 10165-1. *Information technology – Open Systems Interconnection – Structure of management information: Management information model*. Geneva, 1992.

[ITU1992c]	ITU-T Recommendation X.721. *Information technology – Open Systems Interconnection – Structure of management information: Definition of management information.* Geneva, 1992.
[ITU1992d]	ITU-T Recommendation X.722 (1992) | ISO/IEC 10165-4. *Information technology – Open Systems Interconnection – Structure of management information: Guidelines for the definition of managed objects.* Geneva, 1992.
[ITU1993]	ITU-T Recommendations. Recommendation G.824: *Vocabulary of Terms for Broadband Aspects of ISDN.* ITU-T. Geneva, 1993.
[ITUCMIP1991]	ITU-T Recommendation X.711 | ISO/IEC 9596-1. *Information Technology - Open Systems Interconnection - Common Management Information Protocol, Part 1: Specification.* Geneva, 1991.
[ITUX901-1995]	ITU. *Information Technology – Open Distributed Processing – Reference Model Part 1: Overview.* ITU-T Recommendation X.901. 1995.
[Jacobson1997]	Jacobson I., Griss M., Jonsson P. *Software Reuse: Architecture, Process and Organization for Business Success.* ACM Press / Addison Wesley Longman 1997.
[Jacobson1999]	Jacobson, I. , Booch, G. , Rumbaugh, J. The Unified Software Development Process. Addison Wesley. 1999.
[Jander1999]	Jander, Mary. Management Framework Survey - Framework Fraud? *Data Communications International.* September 21, 1999.

[JIDM2000] Open Group. *Inter-Domain Management: Specification & Interaction Translation (The JIDM Specification Translation and JIDM Interactive Translation Technical Standards in one volume).* Open Group Technical Standard. January 2000.

[Joch2000] Joch, Alan. *Full Steam head for Navy's Network Firepower. Network Magazine.* April 2000.

[Kalakota1999] Kalakota, R. , Whinston, A. B. *Electronic Commerce.* Addison Wesley. 1999.

[Kalyanasundaram 1994] Kalyanasundaram, P., Sethi A. Interoperability Issues in Heterogeneous Network Management. *Journal of Network and Systems Management.* Plenum Publishing Corporation. June 1994.

[Karten1998] Karten, N.. *How to Establish Service Level Agreements.* Karten Associates. Randolph, MA, USA, 1998.

[Katz1999] Katz, Jochen. SNMPv3 Support for SNMP++. *The Simple Times. Volume 7, Number 1.* March 1999

[Keller1998] Keller, Alexander. Towards CORBA based Enterprise Management: Managing CORBA based Systems with SNMP Platforms. *Proceedings of the 2nd International Enterprise Distributed Object Computing Workshop (EDOC).* San Diego, USA. November 1998.

[Keller1998b] Keller, Alexander. *CORBA-basiertes Enterprise Management: Interoperabilitaet und Managementinstrumentierung verteilter kooperativer Managementsysteme in heterogener Umgebung.* PhD Thesis, Technical University of Munich. December 1998.

[Keller1999]	Keller, Alexander. Managing the Management: CORBA-based Instrumentation of Management Systems. *Proceedings of the 6th IFIP/IEEE International Symposium on Integrated Network Management.* Boston, May 1999,
[Kiely1998]	Kiely, D. *The Component Edge – An Industrywide Move to Component-Based Development Holds the Promise of Massive Productivity Gains.* Information Week April 13, 1998.
[King2000]	King, A. ; Hunt, R. Protocols and architecture for managing TCP/IP network infrastructure. *Computer Communications.* 23 (2000) 1558-1572.
[Knahl1998]	Knahl, M. ; Bleimann, U. ; Furnell, S. ; Sanders, P. Integration of ATM Management Procedures into Native Integrated Network and Systems Management Architectures. *Proceedings of the International Network Conference 1998, Plymouth.* July 1998.
[Knahl1999]	Knahl, M. ; Hofmann, H. ; Phippen, A. A Distributed Component Framework for Integrated Network and System Management. *Information Management and Computer Security, 7(5), MCB University Press, Bradford/UK.* 1999.
[Knahl2000a]	Knahl, M. H. ; Bleimann, U. ; Hofmann,, H. ; Furnell, S. An Integrated Management Architecture for Heterogeneous Networks: INSMware. *In: Jajszyczyk, Andrzej (Editor), Proceedings of the IEEE Workshop on IP-oriented Operations and Management (IPOM '2000).* September 2000.

[Knahl2000b]	Knahl, M. H. , Furnell, S.M. Management of Service Level Agreements using INSMware *Proceedings of the 2nd International Network Conference 2000, Plymouth.* July 2000.
[Knahl2002]	Knahl, M. ; Furnell, S. ; Hofmann, H. ; Bleimann, U. An Integrated Network and Systems Management Framework based on Adapted Software Components. *Proceedings of the Euromedia 2002 Conference.* Modena, Italy. April 2002.
[Koch1996]	Koch, T. , Krell, C. , Kraemer, B. Policy Definition Language for Automated Management of Distributed Systems. *Proceedings of 2nd Int. IEEE Workshop on Systems Management.* Toronto, Canada, June 1996.
[Kockelmans1995]	Maarten Kockelmans, Eric de Jong. Overview of IN and TMN harmanisation. *IEEE Communications Magazine.* March 1995.
[Köppel2002]	Andreas Köppel. Tutorium: Common Information Model – Modellierung von Applikationen mit CIM. *1 Fachgespräch Applikationsmanagement.* February / March 2002.
[Korostoff1998]	Korostoff, K. Why They Buy: A Survey of ATM Customers. *Business Communications Review.* February1998.
[Kosuga1999]	Kosuga, M. Yamazaki, T. , Ogino, N. Matsuda, J. An Agent-Based adaptive QoS Management and its Applications. *Proceeding of IDMS 1999.* Toulouse, France, October 1999.
[Kreger2001]	Kreger, H. Java Management Extensions for Application Management. *IBM Systems Journal.* Volume 40, No. 1, 2001.

[Krishnan1998] Krishnan, K. ; Fuller, W. An Overview of MIBs for ATM Network Management. *ATM Forum 53 Bytes Newsletter.* March 1998.

[Langer1998] Langer, M., Loidl, S. and Nerb, M. Customer Service Management : A More Transparent View to your subscribed services. *Proceedings of the 9th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM 98).* Newark, USA. October 1998.

[Langsfort93] Langsford, A. , Moffet, J. D. *Distributed System Management.* Addison Wesley, 1993.

[Lee1998] Lee, C. et al. (1998). The Next Generation of the Internet: Aspects of the Internet Protocol Version 6. *IEEE Network.* January/February 1998.

[Leopold1995] Leopold, H. , Campbell, A. , Hutchison, D. Distributed *Multimedia Communication System Requirements.* Research Reports ESPRIT, Project 5341.OSI95, Vol. 1. 1995.

[Levi1999] Levi, D. et al. *Coexistence between SNMP Versions.* SNMPv3 Working Group. Internet Draft, 21 May 1999.

[Lewis1999] David Lewis. *A Development Framework for Open Service Management Systems. Interoperable Communication Networks.* Baltzer Science Publications. 1999.

[Lewis2000] Lewis, Dave. *A Framework for the Development of Service Management Systems for the Open Service Market.* University College London (UCL), PhD Thesis. 2000

[Lewis2002]	Lewis, Lundy; Ray, Pradeep. On the Migration from Enterprise Management to Integrated Service Level Management. IEEE Network. January / February 2002.
[LewisL1999]	Lewis, Lundy. *Service Level Management for Enterprise Networks.* Artech House, 1999.
[Lupu1998]	Lupu, Emil. *A Role-Based Framework for Distributed Systems Management.* Imperial College of Science, Technology and Medicine, Department of Computing, London. July 1998.
[Lycett2001]	Lycett, M. Understanding 'variation' in component based development: case findings from practise. *Information and Software Technology.* Volume 43, Issue 3. 1 March 2001.
[Mac1994]	Michael Mac, Hans-Jürgen Kugler. Service Engineering versus Software Engineering – A Foundational study. *Proceedings of the International Conference on Intelligence and Networks (IS&N1994).* Springer Verlag. 1994.
[Magedanz1996]	Magedanz, T. ; Propescu, Z. Intelligent Networks. International Thompson Computer Press. 1996.
[Magretta2002]	Magretta, Joan; Stone, Nan. *What Management is, how it works and why it's everyone's business.* Free Press. May 2002.
[Manley1997]	Manley, A. et al. (1997). Evolution of TMN Network Object Models for Broadband Management. *IEEE Communications Magazine.* October 1997

[Marcus1995]	Marcus, J. S. Icaros, Alice and the OSF DME. *Proceeding of IM 95.* Chapman-Hall. 1995.
[Mayerl2001]	Mayerl, Christian. *Eine integrierte Dienstmanagement-Architektur für die qualitätsgesicherte Bereitstellung von Netz- und Systemdiensten.* Shaker Verlag. March, 2001.
[McCarthy1995]	McCarthy, K. , Pavlou, G. , Bhatti, S. Neuman do Souza, J. Exploiting the Power of OSI Management for the Control of SNMNP capable resources Using Generic Application Level Gateways. *Proceedings of IM 95.* 1995.
[McCauley1997]	McCauley, D. *The Netizen: Telco Terrorism.* Wired 5.07. June 1997.
[McDysan1999]	McDysan, David E. ; Spohn, Darren L. *ATM Theory and Application.* McGraw Hill. 1999.
[McKinney1998]	R. D. McKinney. Warren A. Montgomery, Hamza Ouibrahim. Paul Sijbem, John Stanaway Jr. Service Centric Networks. *Bell Labs Technical Journal.* July-September 1998.
[Mellquist1997]	Mellquist, Peter E. *SNMP++: An Object-Oriented Approach to Developing Network Management Applications.* Prentice Hall, 1997.
[Meyer1990]	Meyer, B.: *Object-oriented software construction.* Hanser Verlag, 1990.
[Microsoft1997]	Microsoft. *Windows DNA — Windows Distributed interNet Applications Architecture.* Microsoft Specification. 1997.

[Microsoft1998]	Microsoft. *On Complexity, the Web and the Future of Software Development.* Microsoft Developer Network. 1998.
[Mountzia1999].	Mountzia, M.-A. , Rodosek, D. Using the Concept of Intelligent Agents in Fault Management of Distributes Systems. Journal of Network and System Management. 1999.
[Mouritzen2001]	Mouritzen, Jens D. ; Lewis, Dave. The role of XML in TMN Evolution. *Proceedings of IM 2001.* May 2001.
[Mowbray1995]	Mowbray, T. J. , Zahavi, R. *The Essential CORBA Systems Integration using Distributed Objects.* John Wiley & Sons Inc. 1995.
[Mowbray1997]	Mowbray, D. J. , Malwveau, R. C. CORBA Design Patterns. John Wiley & Sons. New York. 1997.
[Muench1997]	Muench, Volker. *Tool-based individual software development in distributed computing environments.* M.Sc. Thesis, Cork Institute of Technology. 1997.
[Naur1976]	Naur, P., Randell, B., Buxton J. (eds.). *Software Engineering Concepts and Techniques – Proceedings of the NATO Conferences.* Petrocelli / Charter, New York.. 1976.
[Netman94]	RACE H408. *Accounting Management Services in TMN.* Research and Development in Advanced Communications in Europe (RACE) / NETMAN Project. 1994
[NetworkMagazine 2001]	Network Magazine. Products Of The Year. *Network Magazine.* May 2001.

[Nolle1997] Nolle, Tom. *ATM: Is There Life After The Hype?* Business Communications Review. April 1997.

[Nolle2000] Nolle, Tom. The New Order in Networking. *Network Magazine.* April 2000.

[Norris2000] Mark Norris, Steve Pretty. *Designing the Total Area Network: Intranets, VPNs and Enterprise Networks clearly Explained.* John Wiley & Sons (Wiley-BT Series). 2000.

[ODMA1997] ITU. *Information Technology – Open Distributed Management Architecture.* ITU-T X.703.. October 1997.

[OMG1995] Object Management Group. *Common Facilities Architecture, Revision 4.0.* OMG Document 98-07-10. 1995.

[OMG1997] OMG. A Discussion of the Object Management Architecture. *Object Management Group White Paper.* 1997..

[OMG1998] OMG. *The Common Object Request Broker: Architecture and Specification, Revision 2.2.* OMG Document 98-07-01, Object Management Group. 1998.

[OMG2000] Object Management Group. *Unified Modelling Language (UML), Version 1.3.* OMG Specification. March 2000.

[Oppenheimer1999] Oppenheimer, P. *Top-Down Network Design.* Cisco Press / Macmillian Technical Publishing. 1999.

[Orfali1996] Orfali, Robert; Harkey, Dan; Edwards, Jeri. *The essential distributed objects survival guide.* John Wiley & Sons, Inc., New York, Chichester, Brisbane, Toronto, Singapore, 1996.

[Ovum1995] Ovum. *The Ovum Report 1995.* Ovum Reports 1995.

[Patel2002] A. Patel. Current status and future directions of software architectures for telecommunications. *Computer Communications.* Vol. 25, January 2002.

[Pathak1994] Pathak, G.. Integrated Network and Service Management for the North Carolina Information Highway. *IEEE Network.* November/December 1994.

[Pavlou1994] Pavlou, G. , Tin, T. , Carr, A. High-Level Access APIs in the OSIMIS TMN Platform: Harnessing and Hiding. *Proceedings of ISN 1994.* Springer Verlag. 1994.

[Pavlou1995] Pavlou, G. , McCarthy, K. , Bhatti, S. , Knight, G. The OSIMIS Platform: Making OSI Management Simple. *Proceedings of IM 1995.* 1995.

[Pavlou1998] Pavlou; George. *Telecommunications Management Network: A Novel Approach Towards its Architecture and Realisation Through Object-Oriented Software Platforms.* University College London (UCL), PhD Thesis. 1998.

[PCWeek1997] PC Week Virtual Press Centre – Ovum in the Press, Components. *PC Week.* March 1997.

[Petermueller1996] Petermueller (1996). Q3 Object Models for the Management of Exchanges. *IEEE Communications Magazine*. March 1996

[Phippen2001] Phippen, Andrew David. *Component Technologies and their Impact upon Software Development.* PhD Thesis, Univerity Of Plymouth. 2001.

[Platt2002] Platt, David S. *Introducing Microsoft .NET (Second Edition).* Microsoft Press. May, 2002

[Prahalad1999] Prahalad, C. K. , Krishnan, M. S. The New Meaning Of Quality in the Information age. *Harvard Business Review.* Sept.-Oct. 1999.

[Pras1999] Pras, A. , van Beijnum, B.-J. , Sprenkels, R. *Introduction to TMN.* CTIT Technical Report 99-09. University of Twente. April 1999.

[PREPARE1996] Jane Hall (Ed.). *Management of Telecommunications Systems and Services: Modelling and Implementing TMN-Based Multi-Domain Management.* Lecture Notes in Computer Science 1116. Springer Verlag Berlin/Heidelberg. 1996.

[Pryce1995] Nathaniel G. Pryce. *Component Interaction in Distributed Systems.* PhD Thesis. Department of Computing. Imperial College of Science, Technology and Medicine. London. 2000.

[Pryce2000] Pryce, Nathaniel G. *Component Interaction in Distributed Systems.* PhD Thesis, Imperial College London. 2000.

[Rabhi2002] Rabhi, Fethi A. , Benatallah, Boualem. An Integrated Service Architecture for Managing Capital Market Systems. *IEEE Network.* Jnauray/February 2002.

[Ray1998] Ray, Pradeep. Evaluation Methodology for Network Management Systems. *Proceedings of the NOMS 1998.* New Orleans, 1998.

[Redmond2000] Redmond, Cliff. *A flexible service-level accounting architecture for telecommunications.* PhD Thesis, University of Dublin, Trinity College. October2000.

[Ressmann2001a] Ressmann, D. ; Platt, A. ; Rumsby S. Architecture to Support Performance Monitoring in Object Based Distributed Systems. *8th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS'2001).* October 2001.

[Ressmann2001b] Ressmann, D. ; Platt, A. ; Rumsby, S. Performance Monitoring of Large Global Distributed Systems. *16th Annual ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'2001).* October 2001.

[Rumbaugh1991] James Rumbaugh. *Object-oriented modelling and design.* Prentice Hall 1991.

[Rumbaugh1993] Rumbaugh, James. et al.: *Object-oriented modelling and design.* Prentice-Hall Internat., 1993.

[Sarna1994] Sarna, D. , Febish, G. The Death of Programming. *Datamation.* May 15, 1994.

[Schade1999] Schade, Andreas. *Management verteilter Anwendungen auf der Grundlage objektorientierter Interaktionsplattformen.* Shaker, Aachen. 1999.

[Scherer1988]	Scherer, J. R. , Murray, T. A. Management of the Intelligent Network. *IEEE Communications Magazine*. December 1988
[Schmale1999]	Schmale, Torsten. *Object-Oriented Network and Systems Management in a Distributed Environment*. Logos Verlag, Berlin. 1999.
[Schmidt1999]	Schmidt, D. Why Software Reuse has Failed and How to Make It Work for You. *C++ Report*. January 1999.
[Sedgewick1992]	Sedgewick, Robert: *Algorithms in C++*. Addison-Wesley. 1992
[Shenker97]]	Shenker, S. ; Partridge, C. , Guerin, R. *Specification of guaranteed Quality of service*. Request For Comments 2212. Sept. 1997.
[Shrewsbury1995]	Shrewsbury, J. K. An Introduction to TMN. *Journal of Network and System Management*. Volume 3(1), 1995.
[Sidor1998]	Sidor, D. TMN Standards : Satisfying today's Needs While Preparing For Tomorrow. *IEEE Communications Magazine*. March 1998
[Silberschatz2002]	Silberschatz, Abraham; Galvin, Peter Baer; Gagne, Greg. *Operating Systems Concepts (Sixth Edition)*. John Wiley & Sons, Inc. 2002.
[SimpleTimes2000]	Katz, Jochen. SNMPv3 Support for SNMP++. Simple Times Quarterly Newsletter. Volume 7, No. 1. 2000.
[Slasano2002]	Salsano, Stefano; Veltri, Luca. QoS Control by Means of COPS to Support SIP-Based Applications. *IEEE Network*. March/April 2002.

[Sloman1993] Sloman, M. , Magee, J. , Twidle, K. , Kramer, J. An Architecture for Managing Distributed Systems. *Proceedings of 4th IEEE Workshop on Future Trends of Distributed Computing Systems.* London, September 1993.

[Sloman1995] Sloman, M. IDSM & SysMan Common Architecture, Domain & Policy Concepts. *Proceedings of the International Workshop on Services for Managing Distributed Systems.* Karlsruhe, September 1995.

[Sloman2002] Sloman, Morris; Lupu, Emil. Security and Management Policy Specification. *IEEE Network.* March/April 2002

[SOAP1.2-2001] World Wide Web Consortium.. *Simple Object Access Protocol (SOAP) 1.2.* World Wide Web Consortium. May 2000.

[Soukouti1997] Soukouti, N. U. Hollberg. Joint Inter-Domain Management: CORBA, CMIP and SNMP. *Proceedings of the 5th International IFIP/IEEE Symposium on Integrated Management 1997 (IM97).* San Diego, USA. May 1997.

[Soukouti1997] Soukouti, N. Hollner, U. Joint Inter-Domain Management: CORBA, CMIP and SNMP. *Proceeding of IM 1997.* May 1997.

[Souza1999] Souza, Desmond Francis D' ; Wills, Alan Cameron. *Objects, Components and Frameworks with UML.* Addison Wesley Longman. 1999.

[Stallins2001] Stallings, Williams. *Operating Systems (Fourth Edition).* Prentice Hall. 2001.

[Steinmetz1993] Steinmetz, R. *Multimedia – Technologie: Einführung und Grundlagen*. Springer Verlag. 1993.

[Stone2001] Stone, G. N. et. al. Network Policy Languages. *IEEE Computer Networks*. January/February 2001.

[Strauss1995] Strauss, P. At Last: Net Managers that Distribute the Load. *Datamation*. February 15, 1995

[Stringer1996] Stringer, D. , Rutt, T. (Eds.). CORBA-based Telecommunications Management Systems. *OMG White Paper*, OMG. May 1996.

[Subranian2000] Subranian, Mani. *Network Management – Principles and Practice*. Addison-Wesley. 2000.

[Szyperski1998] Szyperski, C. *Component Software – Beyond Object-oriented Software*. Addison-Wesley. 1998.

[Telcordia1992a] Telcordia. *The Bellcore Operations Systems Computing Architecture*^{TM} *(OSCA*^{TM}*)Architecture*. Telcordia TR-STS-000915, Issue 1. October 1992.

[Telcordia1992b] Telcordia. *Distributed Transaction Processing and OSCA Building Blocks*. Telcordia SR-STS-002274, Issue 1. July 1992

[Telcordia1993a]. Telcordia. *Information Networking Architecture (INA) Cycle 1 Framework Architecture*. Telcordia SR-NWT-002282, Issue 2. April 1993.

[Telcordia1993b] Telcordia. *User Layer Building Block Configurations*. Telcordia SR-STS-002442, Issue 1. December 1993.

[Thomas2000] Thomas M. Thomas II. *Thomas' Concise Telecom & Networking Dictionary.* McGraw Hill. 2000.

[TINA1994] The TINA Consortium. *Engineering Modelling Concepts (DPE Architecture), Version 2.0.* Document no. TB_NS.005_2.0_94. 1994

[TINA1997] TINA Consortium. *TINA Glossary of Terms.* TINA Consortium. 1997.

[TINA-SA1997] Kristiansen, L. (Editor). *Service Architecture. TINA-C, Version 5.0.* TINA Specification. 1997.

[TMF1999] Tele Management Forum. *Smart TMN Technology Integration Map. GB 909, Version 1.1.* TMF. October 1999.

[TMF-NMDOM1999] TeleManagement Forum. *Network Management Detailed Operations Map.* GB908/v1. TM Forum. March 1999. .

[TMF-OM2000] TeleManagement Forum. *Telecom Operations Map.* GB910/v2.1. TM Forum. March 2000.

[TMF-TIM2000] TeleManagement Forum. *Generic Requirements For Telecommunications Management Building Blocks.* TM Forum GB909. July 2000.

[Trigila2002] Trigila, Sebastiano; Lucidi, Ferdinando; Raatikainen, Kimmo. A Service Architecture for Fixed and Mobile Convergence. *Computer Communications.* Vol. 25, January 2002.

[Troya2001]	Troya, J. M. ; Vallecillo, A. Controllers: reusable wrappers to adapt software components. *Information and Software Technology.* Volume 43 (2001). 1 March 2001
[Tsuchida1996]	Tsuchida, H. et al.. A New Service Provisioning Method in the Multimedia/B-ISDN Era. *IEEE Communications Magazine.* December 1996
[Vandermeulen2002]	Vandermeulen, Filip; Vermeulen, Brecht; Demeester, Piet; Steegmans, Frank; Vermeulen, Steven. A Generic Architecture for Management and Control of End-to-End Quality of Service over Multiple Domains. *Computer Communications.* Vol. 25, January 2002.
[VAS2000]	VAS Team. *PIA System Dokumentation.* Internal Report, Labor fuer Verteilte Anwendungssysteme, University Of Applied Sciences Darmstadt. September 2000.
[Venieris1998]	Venieris, I. S.; Hussmann, H. *Intelligent Broadband Networks.* Wiley & Sons. 1998.
[Wade1998]	Wade, Vincent (Ed.). *PROSPECT Deliverable D212b: PROSPECT Methodology for the design of Multi Domain Management Services in an Open Services Environment (Trial 2).* PROSPECT Deliverable D212b. February 1998.
[Wade2002]	Vincent Wade, David Lewis, Jacques Brook, William Donnelly. *Towards a Framework for Managing the Business-to-Business e-Commerce Chain.* Idea Group Publishing, 2002.

[Wang1998]	Wang, Z. *Towards scalable bandwidth allocation in the Internet.* Technical Report, Bell Labs, Lucent Technologies. 1998.
[Weber1998]	Weber, Michael. *Verteilte Systeme (Distributed Systems).* Spektrum Hochschultaschenbuch, Spektrum, Akademischer Verlag. 1998.
[Wegner1990]	Wegner, P. Concepts and Paradigms of Object-Oriented Programming. *OOPS Messenger.* Volume 1(1), pp. 7-87. August 1990.
[Wies95]	Wies, R. *Policies in Integrated Network and Systems Management: Methodologies for the Definition, Transformation and Application of Management Policies.* PhD Thesis, University of Munich, Germany, 1995.
[Willets1996]	Willets, K. , Adams, E. Managing a Broadband Environment. *IEEE Communications Magazine.* December 1996.
[Willets1996]	Willets, K. et al. Managing a Broadband Environment. *IEEE Communications Magazine.* December 1996.
[Wiltfang1999]	Wiltfang, H. R. *Funktionsorientiertes Management heterogener ATM Netzwerke.* INFIX. St. Augustin, Germany, 1999..
[Wired1997]	Wired. Hypelist. *Wired 5.10.* October 1997.
[Wroclawski1997a]	Wroclawski, J. *The use of RSVP with integrated services.* Request For Comments 2210. Sept. 1997.
[Wroclawski1997b]	Wroclawski, J. *Specification of the Controlled-Load Network* element service. Request For Comments 2211. Sept. 1997.

[Xiao1999]	Xiao, X.; Ni. L. M. Internet QoS: A Big Picture. IEEE Network. March 1999.
[Yamura1997]	Yamamura, T. et al. (1997). TMN-Based Customer Network Management for ATM Networks. *IEEE Communications Magazine*. October 1997
[Zedan2001]	Zedan, H. Guest Editorial. *Information and Software Technology*. Volume 43 (2001). 1 March 2001.
[Zellkowitz1998]	Zelkowitz, M. , Dolores, W. Experimental Models for Validating Technology. *IEEE Computer*. May 1998.
[Zimmermann1995]	Zimmermann, Martin. *Konstruktion und Management verteilter Anwendungen*. Deutscher Univeritaets-Verlag. 1995.
[Zinky1997]	Zinky, J. , Bakken, D. , Schantz, R. Architectural Support for Quality of Service for CORBA objects. *Theory and Practice of Object Systems*. Volume 3, Number 1, April 1997.
[Zitterbart1997]	Zitterbart, Martina (ed.). *Kommunikation in Verteilten Systemen (Communication in distributed systems)*. Informatik aktuell, Springer, Berlin, Heidelberg, New York, 1997.

Appendix: Publications

A number of papers have been published in the course of this research programme.

- Knahl, M. ; Furnell, S. ; Hofmann, H. ; Bleimann, U. An Integrated Network and Systems Management Framework based on Adapted Software Components. *Proceedings of the Euromedia 2002 Conference*. Modena, Italy. April 2002.

- Knahl, M.. , Furnell, S. Management of Service Level Agreements using INSMware *Proceedings of the 2nd International Network Conference 2000, Plymouth.* July 2000.

- Knahl, M. ; Bleimann, U. ; Hofmann,, H. ; Furnell, S. An Integrated Management Architecture for Heterogeneous Networks: INSMware. *In: Jajszyczyk, Andrzej (Editor), Proceedings of the IEEE Workshop on IP-oriented Operations and Management (IPOM '2000).* September 2000.

- Knahl, M. ; Hofmann, H.; Phippen, A. A Distributed Component Framework for Integrated Network and System Management. *Information Management and Computer Security, 7(5), MCB University Press, Bradford/UK.* 1999.

- Knahl, M. ; Bleimann, U. ; Furnell, S. ; Sanders, P. Integration of ATM Management Procedures into Native Integrated Network and Systems Management Architectures. *Proceedings of the International Network Conference 1998, Plymouth.* July 1998.

Copies of these papers are included in the pages that follow.

An Integrated Network and System Management Framework based on Adapted Software Components

Martin H. Knahl[ᶦ], Udo Bleimann[⊕], Steven M. Furnell[?], Holger D. Hofmann[°]

e-mail: martink@soc.plym.ac.uk

Keywords

Integrated Network and System Management, Reusability, Componentware, Computer Networks

Abstract.

Componentware represents an evolutionary step in software development which adds a new packaging granularity to object-oriented systems: software components. Such as components in other sciences or technical domains, software components facilitate reuse even across application domains and allow the realisation of nearly any distribution scenario. In Integrated Network and Systems Management (INSM), the subjects to management are distributed. Furthermore, the heterogeneity of managed components often requires specific adaptations to the management software. The INSMware framework approach proposed throughout this paper achieves a maximum level of manageability by combining INSM with componentware. Software components by default are immutable which would even hinder minor adaptations to the management system or management semantics. To overcome this architecture-inherent limitation of componentware, we integrate the Component Adapter approach to INSMware, which allows the integration of even semantically incompatible software components.

1 Introduction

Nowadays, distribution represents a system-inherent criterion for software and hardware systems. Hardware as well as software components are to collaborate in large-scale environments such as the Internet and thus create new challenges in managing both. We develop the concept of integrated network and system management (INSM) and propose INSMware, a framework which provides component-based INSM, managing both hardware and software components [8].

Traditional management software is monolithic: stand alone applications performing all necessary functions. This paradigm is reaching its limits in terms of code reusability, extensibility, configurability and especially concerning the speed and size development. One reason for this is that traditional development approaches require the application to contain the entire functionality even if much of it is only required for very specific tasks. This paradigm also requires extensions to the design to be carried out and integrated by an application developer familiar with the system (typically the original developer). However, it is the user, not the developer, who is often the one most familiar with the application domain and aware of required extensions. The natural consequence of this would be to provide an interface that allows the user to add new functionality to the monolithic block and that defines a migration path towards compound applications.

Software components provide a different approach to the development process. Like a child with Lego™ building blocks, one can build a compound software application by using several elemental blocks – software components – rather than a monolithic entity. Once the application has been built, it has similar functionality but has the advantages of being distributable, scalable, configurable and can be modified by adding or replacing components. Configurability in this context means to use components or component-based applications in more than one application domain by setting parameters affecting the component's behaviour. Component software aims to provide an architecture to tailor individual needs easily and at low cost. The Latin proverb 'Divide et Impera' can now be changed to 'Build and Manage'.

2 Software Components and Adaptation

The concept of software components goes back to 1969 when McIlroy envisioned an industry of reusable software components and introduced the concept of formal reuse though the *software factory* concept [9]. The idea behind the concept of software components was and is to use self-contained, pre-fabricated and pre-

[ᶦ] Network Research Group / School of Computing, University of Plymouth, Plymouth, United Kingdom

[⊕] Fachbereich Informatik, University of Applied Sciences, Darmstadt, Germany

[?] Network Research Group / Department of Communication and Electronic Engineering, University of Plymouth, United Kingdom

[°] ABB Research Centre, ABB, Heidelberg, Germany tested units in order to build more complex units or entire applications.

As there is no agreed formal definition of a software component [13], we will first present the definition that forms the basis for our work: a software component is a piece of software with one or more well-defined interfaces that are configurable, integrable and immutable [4]. This definition highlights the immutability of software components. Though it guarantees black-box reuse [2], immutability can lead to static caller relationships between software components, a lack of system configurability, and poor support of object-oriented reuse mechanisms such as implementation inheritance [12]. Thus, the physical shape of software components highly influences their reusability.

The driving force behind the use of pre-fabricated components, be it in computer science or other domains, has always been reuse. Reuse relies a great deal on the availability of descriptions of entities to be reused.

Figure 1 shows a taxonomy of reuse potentials with respect to adaptability and modifiability.

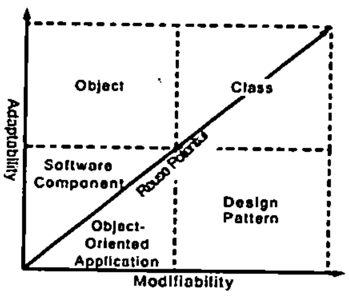


Figure 1 : Taxonomy of Reuse Potentials

At one extreme we have classes, which exist in the form of modifiable source code. The reuse potential of a class is very high as one can always change the class' source code to meet the requirements of the developer. Objects, on the other hand, are adaptable, not modifiable, entities. An object exists only at runtime as an instance of a class and cannot be modified. However, an object's state and its relationships with other objects can be changed at runtime. Design Patterns [2] exist in graphical and/or textual format, and like classes, can be modified. Because a Design Pattern is dedicated to a specific use, it has very limited adaptability.

Object-oriented applications, i.e., applications that have been developed using object-oriented techniques, also have limited adaptability. Moreover, they are not modifiable since they exist in an immutable physical shape. It may come as a surprise to find software components at the lower left end of our taxonomy of reuse potentials. But a closer scrutiny reveals that software components share the same characteristics as object-oriented applications. They come in an immutable physical shape, having been implemented on distributed object-oriented architectures such as CORBA [11] and DCOM [1]. They provide, at most, limited support for the differential reuse mechanisms: inheritance, aggregation, and delegation, and are therefore not very adaptable either.

Software components are reused by composition [10], which means the grouping of a set of components to form a new component or even an application. The components are integrated on a common *composition layer* and communicate by exchanging messages. The interaction of these components is controlled by a *control logic*. Software component composition requires component compatibility. If components that are to be composed are not compatible, i.e., they are not able to collaborate because of interface or semantic reasons, the composition language can be used to realise component compatibility. We call this concept *adaptation*.

Let C_{org} be a component to be adapted that contains the implementations I_j, $)I_j$ the adapting implementation, and operator ρ an operation to combine I_j with $)I_j$. Then the adapted component C_{adapt} can be defined as:

$$C_{adapt} = C_{org}\ \rho\ )I \text{ with })I = \{I_1\ \rho\ )I_1, \dots, I_n\ \rho\ )I_n\}$$

The following criteria for software component reuse and adaptation mechanisms have been identified: transparency of use, black-box characteristics, configurability, reusability and architecture independence and efficiency [6]. As adaptation functionality at the level of a composition language is not reusable, it is not an appropriate mechanism for software component adaptation.

To cope with this problem, we apply the concept of Component Adapters [5]. Component Adapters are software components which represent a specific view of one or more software components to client components and which act as surrogates for these.

The incoming interfaces of a Component Adapter represent a view required by client components while its outgoing interfaces are connected to the incoming interfaces of server components to be used to realise the Component Adapter's functionality. The interface members of the Component Adapter can be mapped to one or more implementations provided by the server components. Depending on the implementation of the Component Adapter, these caller relationships can be changed at runtime or can be statically assigned at the time of development.

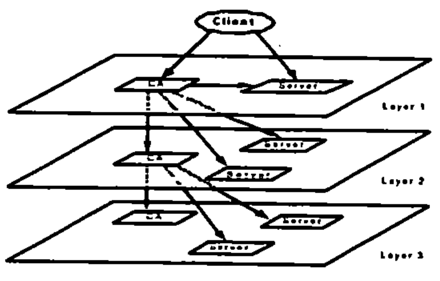


Figure 2: Component Adapter Usage Scenario

The mapping of incoming to outgoing interface members is installed by configuration and includes possible parameter and return type conversions. Though it is possible to do an automatic data conversion for elemental data types such as integer or float, the mapping of complex data types has to be configured by the user. Without any data conversion facilities only server components forming part of a subtype/supertype relation with client components could be used by the Component Adapter. Figure 2 shows a usage scenario of Component Adapters in which the approach is used recursively.

The design of Component Adapters is based on a combination the structural Design Patterns [15] [2]: *adapter* (adaptation of object interfaces), *decorator* (addition of functionality to existing implementations), *facade* (provision of high-level interfaces to sub-systems), and the behavioural Design Pattern *mediator* (centralisation of control). This makes it possible to change communication paths between software components, to define new interfaces to existing implementations, and to centralise the required implementation for the adaptation (control logic) in one place without the necessity of modifying the concerned components.

The integration of the Component Adapter approach with a component management architecture such as discussed in [5] provides flexibility in integrating self-developed software components with pre-fabricated components and supports systems evolution. This means that even software components that were unknown at the time of the management system's development can be slightly integrated with it. The major benefit of using a management system together with Component Adapters is that software components to be managed neither have to provide specific management functionality nor specific management interfaces.

3 Component-Based Integrated Network and Systems Management

The terms network, systems and applications management stand for all precautions and actions taken to guarantee the effective and efficient use of hardware and software resources of distributed systems and their underlying communication networks [3]. Several management architectures have been proposed and standardised as a basis for integrated management (e.g., SNMP based TCP/IP management, Telecommunications Management Network by the ITU). Management platforms that integrate different management applications are available on the market (e.g., IBM Tivoli TME/10 Management Framework). However, the proposed management architectures and platforms do not provide integrated management solutions for network management of LANs and WANs and for their different requirements on systems management. They generally represent management toolboxes with differences in middleware, management protocols or incompatible management applications. Given the diverse technologies and vendor specific implementations in today's heterogeneous networks, the management architecture itself becomes a heterogeneous mixture comprising different management applications and platforms.

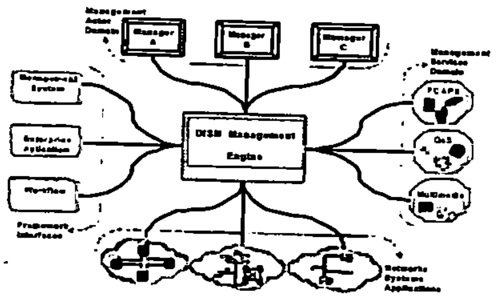


Figure 3: Management Framework

One attempt to address this problem could be to provide a single system capable of providing all Network and Systems Management services. Such a system is referred to as an Integrated Network and Management System. The problem with these is that they are very complex and therefore expensive and processor intensive. The Management Framework itself must be a distributed system with open interfaces, where the required management services are put together to provide the required management functionality (see Figure 3). Furthermore, open interfaces are required to provide interoperability to other management and enterprise applications and to expand the management framework to meet future requirements. Therefore, we propose a new component-based Management Framework [7].

The impact and leverage of distributed systems technology is prevailing not only for design and implementation of applications but also for the deployment of management systems. Thus far, management systems have typically been of two categories. Either specialised along one dimension, e.g., vertically targeting one or a few management aspects, e.g., configuration, billing or security or horizontally dedicated to management of a specific layer, e.g., network elements. Alternatively, they have resembled monolithic "main frames" based to a large extent on proprietary solutions. The presented research uses contemporary componentware technology to leverage a modular approach to the design of management systems, thus facilitating openness and extensibility on one hand and adaptability, i.e., customisation of management services, on the other.

The distribution of component-based systems mirrors the distribution of managed hardware/software components. We have already argued that a component-based approach does not inevitably guarantee reusability. We argue that this deficiency can be overcome by using Component Adapters that also allow for the integration of legacy components/applications into the management system.

As mentioned before, the conventional Management Systems cannot meet the needs of today's rapidly changing network systems. To make the system flexible to change, we propose a new style of the network management system, in both the development and operation phases. We call it "Componentware Integrated Network and System Management" (INSMware), as it uses the component-oriented approach for building and running the Integrated Management System [7] [8]. Our component-oriented Management System solves the problems of heterogeneous management protocols and can help reducing development costs.

The component-oriented software approach frees the developer from cumbersome coding, as the componentware based approach provides integration and customisation of Software components. This enables rapid and efficient system development, since tool software handles and validates much of the integrity of the system, but not a human.

4 INSMware

Limitations and restrictions of existing Network and System Management frameworks such as distribution of the management services and adaptation and integration of new management services can be overcome by providing a component-based approach [7] [8].

INSMware is a componentware-based framework for Integrated Network and System Management. We provide a component-based development approach meeting requirements for integrated management services. There are two versions of INSMware: one implementation is using CORBA [11], the other is based on DCOM[1]. This allowed us to study both middleware architectures in detail.

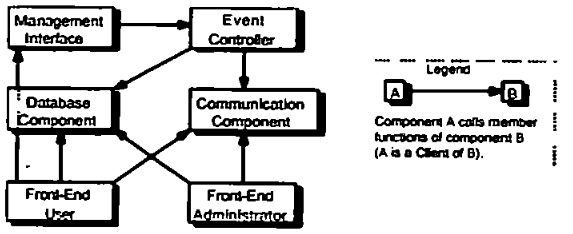


Figure 4: INSMware Components and Connectivity

The design of the individual INSMware components (see Figure 4) is based on a domain specification that subdivides the entire application domain into subdomains. First, the data processing system requires a connection to a data source. This is realised by the Management Interface component that exists in several different forms, similar to the device drivers of an operating system. It is configurable as required for different data sources.

The Management Interface component interprets the received data. It is filtered and analysed, and the component notifies the event controller when particular pre-defined exceptional states occur. Data storage is accomplished by a call to the database component and user notification is effected via communication components. It should be emphasised that all information about the users that need to be notified, e.g., access to user, user's role regarding the monitored processes, are stored within the system. The communication component itself consists of a set of several sub-components that again implement sub-domains, for example, faxes, voice mails, e-mails. The user can visualise system states using the front-end user component and can maintain the system by using the front-end administrator component.

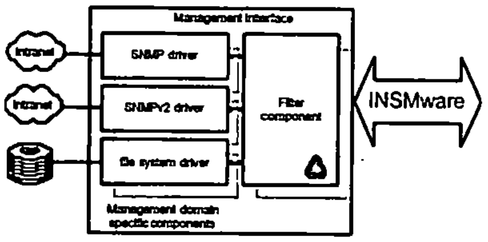


Figure 5: Management Interface Component

In our approach, a new management protocol can be installed by merely adding a new management domain specific component that represents the protocol into the Management Interface. This means that the protocol-handling part of the Management System is usually encapsulated to one component. This strengthens the adaptability of the component-oriented NMS to new management protocols, as the developer only has to develop the communication component specific to the new protocol. Therefore, we use a driver concept for the Management Interface that consists of the specific management domain and generic Filter component as shown in Figure 5.

4.1 Integration of Security Component

In any application domain, all data in the INSMware system is stored using the database component. To prevent unauthorised access to security-relevant data, clients must encrypt data before being transmitted to a database component that then decrypts it. Similarly, the database component must encrypt the data to be transmitted while client components have to decrypt the data. It was decided to add this functionality to the INSMware system, which means that four existing components, namely, the database component, the event controller component, the front-end user component, and the front-end administrator component have to be modified and re-built. These changes are not required if the additional functionality is integrated into the INSMware system in the form of Component Adapters via surrogate substitution. We demonstrate this using the DCOM version of INSMware.

Two approaches are possible:

i. realisation of the required security functionality inside a Component Adapter
ii. realisation of the required security functionality by a separate software component that can be used by a Component Adapter.

Approach (i) implies that the security functionality has to be realised in the programming language in which the Component Adapter is implemented. This restricts the reusability of the security functionality to one particular programming language. Approach (ii) allows to possibly reuse the provided security functionality not only by the Component Adapter, but also by several other software components. Coming from these two scenarios, we chose to realise approach (ii) because of reusability issues.

When realising approach (ii) either a security component can be developed from scratch or a software component can be purchased on the market that meets the requirements. We decided to develop our own security component to be able to reuse it in our various management domains. One demand for the realisation of a security component was the integrability with the development tools used. As these tools were ActiveX-capable, i.e., Microsoft ActiveX components could be integrated with these, we decided to implement the security component as an ActiveX component. ActiveX is a part of the Microsoft DCOM architecture that allows the realisation of scripting-capable software components.

Figure 6 shows the integration of a security component into INSMware. A client component accesses a Component Adapter that encrypts the data and sends it to a Component Adapter located on the same host as the database component. The second Component Adapter decrypts the data and sends it to the database component. After processing the client request, the database component transmits the results to the local Component Adapter that encrypts the data and sends it to the client-located Component Adapter. Finally, the results are decrypted and transmitted to the client component.

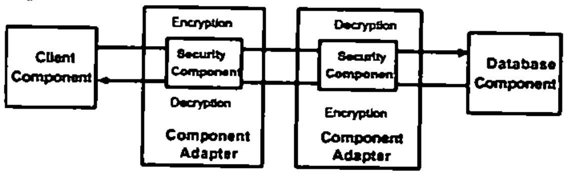


Figure 6: Component Adapter Integration

Both Component Adapters are located locally to the adapted software components. This location constraint is obvious as data transmission between the adapted components and their associated Component Adapter is still insecure. This does not represent a problem to local inter-component communication while this is not acceptable for remote communication.

The integration of Component Adapters with the INSMware system implies that every client of the database component, i.e., the event controller component, the front-end user component, and the front-end administrator component, accesses a client-located Component Adapter instead of directly accessing the database component. This integration is transparent to all client components and can be dropped if necessary.

Implementing a separate security component optimises the reusability of implemented algorithms, but its integration to a system may adversely affect the system's performance since the number of inter-component communications necessarily raises.

The Component Adapters must implement the interface of the software component to be adapted and expose this interface to client components. In the case of the INSMware system, this is the interface of the database component.

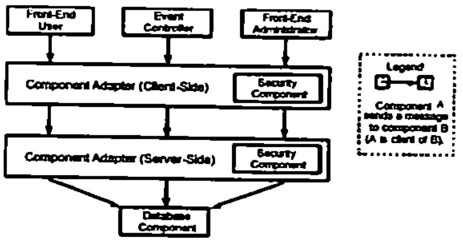

Front-End User
Event Controller
Front-End Administrator
Component Adapter (Client-Side)
Security Component
Component Adapter (Server-Side)
Security Component
Database Component
Legend
Component A sends a message to component B (A is client of B).

Figure 7: INSMware Component Adapter Integration

Two Component Adapters, one for the client and one for the server side, are required. On the client side, data has to be encrypted before being sent to the server-located Component Adapter and data has to be decrypted after results being received from the server-side Component Adapter. The reverse behaviour is required on the server side.

The Component Adapters are integrated into the INSMware system by surrogate substitution and thus are transparent to client components. Each component of the INSMware system that accesses the database component does so through a client-side Component Adapter that communicated with a server-side Component Adapter. This is shown in Figure 7.

4.2 Management of SW Components

Component monitoring provides data about the run-time state of a set of software components. Data such as server host utilisation and memory usage of software components can be monitored. Without the use of Component Adapters, software components would have to be especially designed and implemented for the use within the management framework. This would far limit the components that can be used within our management environment. With Component Adapters, software components, with or without individual monitoring facilities, can be integrated and managed.

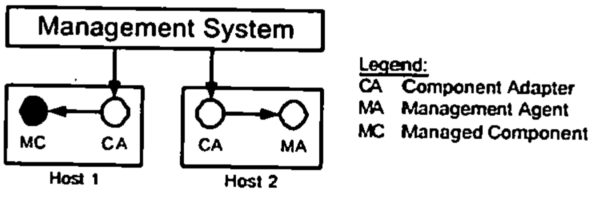


Figure 8: Component Adapters - Component Monitoring

Figure 8 shows two types of software component monitoring. In Host 1, a Component Adapter directly monitors a managed software component (MC). This approach can be applied if a MC supports monitoring functionality or if the Component Adapter gathers information about a component by making calls to its operational interface. In Host 2, the Component Adapter monitors a management agent (MA) that gathers information about a software component's environment, for example, the number of running processes or software component instances on a particular host, the operational status of hardware components (e.g., on/off/stand-by).

Using the Component Adapter approach, monitoring software components can be easily integrated with new or existing management systems. If, for example, Component Adapters implement a Simple Network Management Protocol (SNMP) interface [14], they can be integrated with any SNMP-compliant management system.

Along with the distribution of management components comes a distribution of management knowledge. In particular scenarios, it may be required that elemental management tasks are performed by local management components while complex tasks that might require user interaction may be co-ordinated by a central authority.

5 Conclusions and Outlook

Our work with INSMware has shown that a distribution of management software and thus of management knowledge can help in mastering the inherent complexity of distributed hardware/software systems. The realisation of management systems as componentware, i.e., software entirely composed of immutable pieces of software called "software components", can significantly support software reuse in this domain. In a scenario where normally the management software would have to be adpated to changing requirements, we propose the use of the Component Adapter concept. The latter helps to integrate and configure even incompatible software components and supports a common denominator on the software level.

For particular application domains, pre-fabricated Component Adapters may exist providing specific interfaces to managed components or to management systems. Other scenarios might require custom adapters whose development could be supported by libraries or code skeletons providing basic functionality.

6 References

[1] N. Brown, C. Kindel. *Distributed Component Object Model Protocol - DCOM/1.0.* Microsoft Corporation, Network Working Group, 1996.

[2] E. Gamma, R. Helm, R. Johnson, J. Vlissides. *Design Patterns — Elements of Reusable Object-Oriented Software.* Addison-Wesley, 1994.

[3] H. Hegering, S. Abeck, B. Neumair. *Integrated Management of Networked Systems.* Morgan Kaufmann, 1999

[4] H.D. Hofmann. *Componentware — Integration of Software Components in Distributed Computing Environments.* M.Sc. Thesis, Cork Institute of Technology/Ireland, 1997.

[5] H.D. Hofmann, J. Stynes. *Implementation Reuse and Inheritance in Distributed Component Systems.* Proceedings of Twenty-Second Annual International Computer Software and Applications Conference (COMPSAC'98), Vienna/Austria, 1998.

[6] H.D. Hofmann, J. Stynes, G. Turetschek. Software Reuse by Adaptation. *Proceedings of the Second International Network Conference (INC 2000).* University of Plymouth: Plymouth, UK.

[7] M. H. Knahl, H.D. Hofmann, A. D. Phippen. A Distributed Component Framework for Integrated Network and System Management. Information Management and Computer Security, 7(5), pp. 254-260, MCB University Press, Bradford/UK, 1999.

[8] M. H. Knahl, U. Bleimann, H.D. Hofmann, S. M. Furnell. 2000. An Integrated Management Architecture for Heterogeneous Networks: INSMware. In: Jajszyczyk, Andrzej (Editor), Proceedings of the IEEE Workshop on IP-oriented Operations and Management (IPOM '2000). September 2000. pp. 111-118.

[9] M.D. McIlroy. *Mass-produced software components.* In J.M. Buxton, P, Naur, and B. Randell (editors), *Software Engineering Concepts and Techniques*, pp. 88-98, 1968 NATO Conference on Software Engineering, 1976.

[10] O. Nierstrasz, S. Gibbs, D. Tsichritzis. *Component-Oriented Software Development.* Communications of the ACM, 35(9), pp. 160-165, 1992.

[11] OMG. *The Common Object Request Broker: Architecture and Specification*, Revision 2.2. OMG Document 98-07-01, Object Management Group, Inc., 1998.

[12] M. Sakkinen. *Inheritance and Other Main Principles of C++ and Other Object-oriented Languages.* PhD thesis, University of Jyvaeskylae/Finland, 1992.

[13] J. Sametinger. *Software Engineering with Reusable Components.* Springer, 1997.

[14] Stallings, William. SNMP and SNMPv2: The Infrastructure for Network Management. IEEE Communications: Management of Heterogeneous Networks, 36(3), 1998.

[15] W. Zimmer. *Relationships between Design Patterns.* Proceedings of PLoP '94 - Pattern Languages of Programs, Addison-Wesley, 1995.

An Integrated Management Architecture for Heterogeneous Networks: INSMware

Martin Knahl [◆], Prof. Dr. Udo Bleimann [•], Dr. Holger D. Hofmann[†], Dr. Steven Furnell [◆]

e-mail: knahl@vas.fh-darmstadt.de

Keywords: Network Management, System Management, Componentware, SNMP.

Abstract: The Component-based approach to develop distributed software represents a new paradigm in software engineering. This approach is used to implement a new framework for Integrated Network and System Management for heterogeneous networks. Future Management Systems will be derived from a set of pre-fabricated components rather than being developed from scratch. In this paper, we present research in the area of Componentware based Integrated Network and System Management and the research prototype INSMware, which was built using only component-based techniques and which represents a research prototype that was used for investigating such component-based network management techniques. We describe an approach that integrates the software component paradigm with network and systems management. Its integration upon different networking technologies is outlined. We focus on the monitoring of SNMP-capable network elements.

1 Introduction

Limitations and restrictions of existing Network and System Management frameworks, such as distribution of the management services, adoption and integration of new services can be overcome by providing a component based approach [1], [4]. The impact and leverage of distributed systems technology is prevailing not only for design and implementation of user applications and services but indeed also for the benefit from deploying management systems. Thus far, management systems have typically been of two categories. Either specialised along one dimension (e.g. vertically, targeting one or a few management aspects such as billing or performance, or horizontally, dedicated to management of a specific layer such as network elements) or have they resembled monolithic "main frames" based to a large extent on proprietary solutions.

Existing management solutions, based on the integrated manager or platform approach, do not meet the requirements and are complex. The development and provisioning of integrated management services proved to be too complex. A simpler solution is required and proposed: Management is Lego^{TM}, Management is distributed components - thinner layers, higher reuse potential of existing solutions, improved potential of easy integration of new technologies and therefore less effort to integrate existing and new technologies. Componentware is an enabling technology to meet the requirements and architectural principles of the proposed framework and of the prototype implementation (INSMware). The research uses contemporary distributed technology to leverage a modular approach to design management systems, thus facilitating openness and extensibility on one hand and adaptability, i.e. customisation of management services, on the other.

Distributed object-oriented systems [14] represent the logical development of the object model supporting the distribution of objects to physically distinct locations. Distributed object-oriented architectures, also referred to as middleware architectures, such as

[◆] Network Research Group, Department of Communication and Electronic Engineering, University of Plymouth, Plymouth, United Kingdom

[•] Department of Computer Science, University of Applied Sciences Darmstadt, Darmstadt, Germany

[†] Department of Mathematics and Computing, Cork Institute of Technology, Cork, Ireland

OMG CORBA or Microsoft DCOM form the basis for the development of distributed object-oriented systems. Providing a mediation layer for distributed object communication can be enhanced by the provision of a framework to support reuse. *Software components* are distributed objects that are designed to be reused. They represent physically immutable pieces of software that can be assembled with other components to form new, more complex components or even entire applications. The latter is called *componentware.*

The development and operation of component-based systems requires an architectural basis in the form of a componentware architecture. The main mechanisms that have to be realised by such an architecture are those for component search and description, for component management, for component composition, and for component configuration.

The use of software components on an enterprise level enables the use of distribution, software reuse, security services and other functionality. Distribution allows software components to communicate with each other over electronic networks (such as local area networks or the Internet). Modelling aspects and the functionality of a distributed, component-based Integrated Network and System Management framework will be discussed to prove the relevance to build a novel management framework using software components, namely INSMware.

The presented approach, leading to the componentware based Management Framework INSMware for Integrated Network and System Management, also enables the monitoring of component-based applications. It is a building block of an enormous effort in research and industry towards the integrated management of all resources in a distributed system. The often-cited ITU-T Management Framework that categorises the management tasks into the five classes Fault, Configuration, Accounting, Performance and Security Management, only gives a functional decomposition of this multi-dimensional problem. However, another important problem dimension is the resources or components within a distributed system upon which the management functions are applied. In the past, mostly the logical communication resources (e.g. communication protocol entities) and the physical network components (e.g. repeaters, bridges, routers) have been addressed by the standardisation bodies and industrial products. Consequently, the term network management is commonly used to cover these aspects. In the meantime, hardware and software resources of the end-systems (e.g. disk space, CPU time of applications) are included in the management view expanding the network management to systems management.

Beside the integration of established management procedures, the proposed framework requires functionality to be instrumented in order to enable the monitoring of distributed applications - one step in the direction of application management. However, purely isolated component-oriented management is not what the end-user expects. Integration of management solutions with the aim of one single architecture that may cope with all type of resources is demanded. For instance, by means of commercially available network management systems, network operators get a management view upon the network resources and thus helps to answer questions about the reachability and availability status of network devices and end systems or the load on the network segments. Additional knowledge is necessary in order to close the gap between the applications and the network; knowledge like, which application components are running on which end systems, what is the communication behaviour of the applications or how much load is produced by the individual applications.

To gain this knowledge is currently very tedious, as the mapping relations between applications and network resources is often changing, possibly without the knowledge of the network operator. The situation will become even worse when object migration technologies or automatic fault recovery procedures are widely used. What is required is a common and integrated view upon all resources of a distributed system on the network, (operating) system, middleware and application levels.

This paper presents research leading to a componentware-based framework for Integrated Network and System Management (INSMware). The transferability of INSMware to other management domains is realised because of a consistent component-based development approach to meet the requirements for integrated management services. There are two versions of INSMware: one using the CORBA [2] component model and a DCOM [3] implementation. This allowed us to study both middleware architectures in detail.

2 The INSMware Project

This section presents the INSMware project. The requirements for integrated management services for heterogeneous networks are identified and outlined. Then, the architecture of the INSMware framework is presented, followed by a discussion of the application domains and sub projects.

2.1 Integrated Management Services

The disciplines of Network and System Management encompass all actions taken to enable and guarantee the maintenance and operations of the resources - either hardware or software - in a network (see Figure 1).

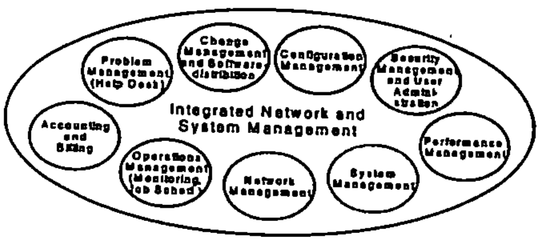

Problem Management (Help Desk)
Change Management and Software distribution
Configuration Management
Security Management and User Administration
Integrated Network and System Management
Accounting and Billing
Performance Management
Operations Management (Monitoring, Job Sched.)
Network Management
System Management

Figure 1: Management Disciplines

Integrated Network and System Management includes the communication network as well as the server and the end-systems in a network. The Telecommunications standardisation sector of the ITU (ITU-T) defined five management categories - Fault, Configuration, Performance, Accounting and Billing, Security Management - that define the different disciplines and requirements for the management of heterogeneous networking environments.

Required management services include configuration of network elements, monitoring and controlling of network elements, software distribution (e.g. software distribution to PCs), user management, security management (e.g. access control), service management (e.g. video) and so on. The complexity of the management services is related to the complexity of the environment to be managed.

The research has shown that management standards, such as SNMP or today's Management-Platforms like Tivoli TME 10 or HP OpenView Network Node Manager, basically cannot handle the end-to-end management of ATM networks [4]. The challenge for the network operators is how to migrate from these restricted systems to an architecture that integrates the different network technologies, i.e. the TCP/IP and the ATM management model, and that will enable them to meet the goal of providing seamless, end-to-end connectivity and management for tens of thousands of users across LANs and WANs. It has been found that this seamless architecture presents problems and raises challenges in just about every area of network management [7].

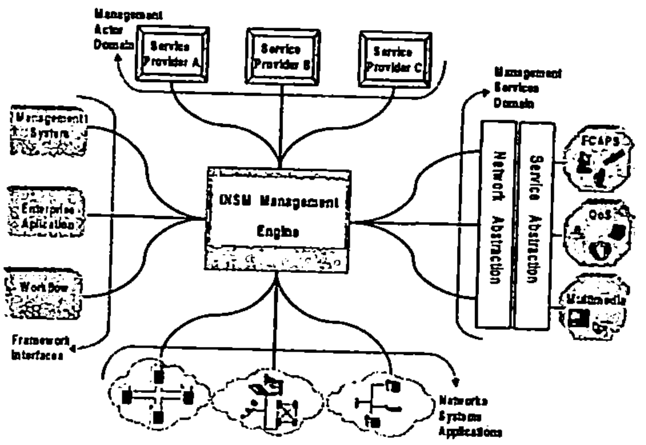


Figure 2: Integrated Management Framework

The aims of INSMware are to hide the complexity of the heterogeneous network environment and underlying technologies and to provide a universal framework for the various management services as illustrated in Figure 2. In addition, INSMware is designed to facilitate the easier addition and integration of new networking technologies and management services by employing component-reuse.

2.2 INSMware Architecture

Distributed systems allow the partitioning of applications into logical and physical self-contained entities: distributed objects. These distributed objects represent a part of the system-global object model. The possibility to use distributed objects for the realisation of different applications makes them into so-called software components. A software component is a piece of software with one or more well-defined interfaces that is configurable, integrable, and immutable. By configurable we mean being able to set parameters affecting the properties of a component without requiring its modification. The integration of software component means the connection of incoming and outgoing interfaces, i.e., interfaces being used in the client role and in the server role of a client/server component communication, while immutability requires a component to be *physically immutable*. Such physically immutable forms of software are, for example, executable files or dynamic link libraries (DLLs).

The most important criterion of our component definition above is immutability (Black-Box Reuse) since it allows dissociation from object-oriented concepts such as Classes and Design Patterns (White-Box Reuse). The functionality of management framework presented consists of the processing, filtering, and analysis of management relevant data, the presentation in a graphical user interface and user notifications at the occurrence of predefined states that represent important network states. The system allows that one or more users may be notified over varying communication channels.

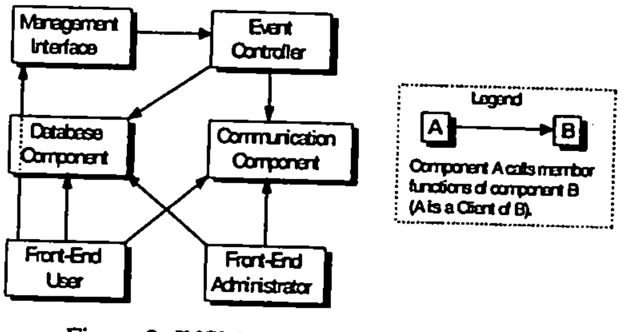


Figure 3: INSMware components and their connectivity

The design of the individual INSMware components (see Figure 3) is based on a domain specification that subdivides the entire application domain into subdomains. First, the data processing system requires a connection to a data source (physically existing system). This is realised by the Management Interface component that exists, similar to device drivers of an operating system, in several different forms and is configurable, as required, for different data sources.

The Management Interface component interprets the received data, filters and analyses it, and notifies the event controller component when particular predefined exception states occur. Data storage is accomplished by a call to the database component and user notification is effected over the communication components. It must be emphasised that all information about the users that need to be notified (e.g., access to user, user's role regarding the monitored processes) is stored in the system. The communication component itself consists of a set of several components that again implement subdomains, e.g., sending of faxes, voice mails, e-mails.

The user can visualise system states by using the front-end user component and can maintain the system by using the front-end administrator component.

2.3 Application Domains and Sub Projects

INSMware was originally conceived to monitor the various network elements and to provide management services in heterogeneous TCP/IP and ATM network environments. With INSMware, a timely user intervention in the running of the network processes is made possible when required (e.g. user notification when a network critical condition occurs).

To test the workability of the component-based approach and to examine aspects of reuse at component level, INSMware was applied to the application domain of monitoring SNMPv1 and SNMPv2 based managed objects. Two of the components, namely the Management Interface component and the front-end user component, have to be modified to integrate new management services (e.g. SNMP) into the management framework. The database structure has to be adapted to the management relevant information, while the remaining four components can be reused with no modifications. Adaptation is required in the Management Interface component since the data acquisition mechanism differs between the different physical and logical managed objects with different management protocols. The application domain implemented by the Management Interface component is actually very limited and a universal Management Interface component was developed which can be adapted to different systems by configuration [6].

The graphical data representation supported by the front-end component to visualise the management information has to be adapted to the respective application domain and remain user-friendly and simple.

By generating source code responsible for intercomponent communication, a tremendous reduction in the development time for front-end components was possible. Using a CORBA/ DCOM bridge, a platform-independent connection of partially generated front-end components is achieved.

3 INSMware Implementation

This section presents details of the prototype implementation of the INSMware concept.

3.1 SNMP

SNMP is a set of network and system management standards that describe the asynchronous requests and responses for the exchange of management data between SNMP management objects [8]. SNMP was originally developed by the IETF to manage TCP/IP networks and network elements such as routers and hubs [9]. The 'basic' SNMP (SNMP version 1) is now in widespread use and the de-facto industry standard. Virtually all major vendors of end-systems, workstations and network devices such as routers and switches offer SNMP support. The widespread acceptance of SNMP has resulted in adoptions such as the use of SNMP over OSI and other non-TCP/IP protocol suites. In addition, enhancements to the initial SNMP have been pursued in a number of directions (e.g. RMON, SNMP Version 2 and 3, ATM Management).

The SNMP entities are an SNMP Manager, an SNMP Agent and a Management Information Base (MIB). The SNMP Manager is typically network management software that implements the SNMP protocol. The SNMP Agent resides in a managed network element, such as a router or switch or in an end-system, such as a PC or Unix Workstation. The agent stores management information and processes SNMP requests from the SNMP Manager and responses from the agent itself. The MIB is the database for management information that is stored in ASN.1 syntax format [10].

SNMP is a polling-based protocol. The manager sends a request for information to the agent (Get, Get Next operations) periodically and the agent responds to that request and the manager can modify and manipulate SNMP values in the agent (Set operation). Beside that, an agent can send an event to indicate an alarm or specified condition (Trap operation) to the manager to mark an important event, such as a critical network error.

There are two basic approaches to coexistence in a multi-lingual network (several SNMP versions), namely multi-lingual implementations and proxy implementations [11]. Proxy implementations provide a mechanism for translating between SNMP versions using a third party network element, and hence add complexity into the management-services due to the translation services required. The proposed INSMware framework is based on a multi-lingual SNMP implementation. The multi-lingual implementation of the management interface supports the different SNMP versions and enables the seamless integration of the (typically monolingual) SNMP-based managed objects. Besides that, additional protocols or access policies can be integrated into the Management Interface component.

3.2 Management Interface component

This section describes the Management Interface component. The Management Interface component connects the framework to the underlying networking technologies as illustrated in Figure 4.

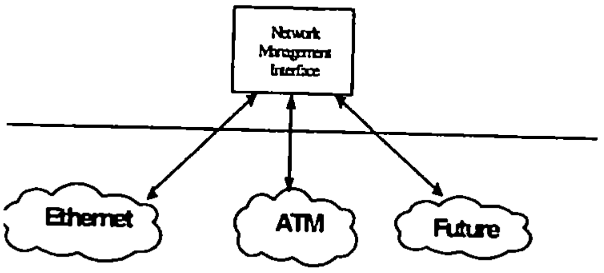


Figure 4: Network Management Interface

3.2.1 Architecture of Management Interface component

One advantage of component oriented software is the relatively easy reuse of individual parts of the system. It became clear after the first stages of the project, that major parts of INSMware within the application domain integrated network and system management will be reusable with no or relative little changes to the components. Modifications to the system to integrate new services and reuse existing components are required on the interface (gateway) to the user (Front-End component for user interaction) and the interface to the managed network (Management Interface component for the communication with the managed objects).

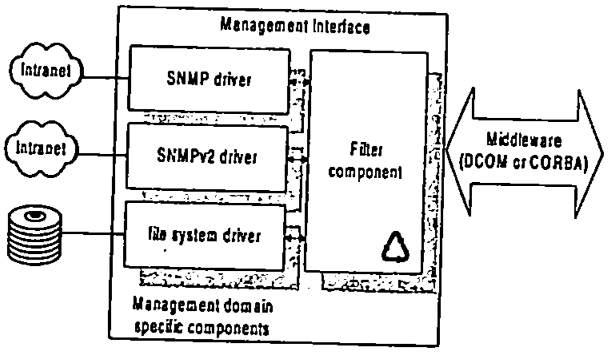


Figure 5: Architecture of Management Interface

To improve and optimise the reusability of the Management Interface component it was further abstracted – therefore, the functionality of the Management Interface component was split into several smaller components (see Figure 5).

The general functions for filtering of management information offer a high potential for reuse. This part analyses the information from the managed objects and decides whether a critical value has been exceeded or whether a critical event has occurred. This evaluation can proceed without further knowledge of the underlying technology (e.g. whether SNMPv1 or SNMPv2) of the managed objects because solely the protocol independent data stream (from the managed objects) has to be analysed. Therefore (because of this high potential for reuse), the filtering was encapsulated and implemented into an individual component within the Management Interface. Hence the problem occurs of how to bring the data from the various kinds of managed objects into a syntax that can be understood and proceeded from the filtering component. The initialisation and de-initialisation of a connection with a managed object is individual for every kind of managed object with different management functionality (management protocols), e.g. the initialisation of a connection and the access to a SNMP managed object very much from the access to an ODBC database or a file system. The potential for reuse in this domain is relatively small and the component cannot integrate new access technologies through component configuration but through re-implementation and integration of the new access technologies. One exception are data sources that are similar concerning their access, e.g. different ODBC databases where relative little adjustments are required. To be able to integrate different access technologies within the management domain a concept was developed that is similar to the driver concept of operating systems. For each access technology (e.g. SNMP), a specific component is developed that implements a specific defined interface and which can be used from the general Filter component.

This specific component is responsible for the communication with the managed object and transforms/translates the received data into a format that can be used by the Filter component. Whilst developing different management domains, it has been experienced, that a few parts – particularly in the area of the interface implementation – are similar. For these similarities, source code can be reused. Beside that, the Management Interface offers a high degree of flexibility and provides good reuse for additional and future management domains.

The introduction of an additional abstraction layer within the Management Interface leads to components with a very small level of granularity. To minimise the associated disadvantages (e.g. loss of performance and stability) no middleware for distribution was introduced but the concept of Dynamic Link Libraries (DLL) was used. Due to the fact that it is an In-Process-Server, a complex binding process is not required and the communication can be performed very fast and without changing tasks.

3.2.2 SNMP-Interface

The analysis and design stage of the SNMP-Interface was based on the layered model of the driver-components of the Management Interface (Figure 1). Dynamic access mechanisms have been implemented to read the configuration from an ODBC database (MS Access is used as database for the prototype). The prototypical implementation of the Management Interface component is written in C++. The SNMP functions have been implemented using two different SNMP frameworks based on C++: the SNMP interface of the Microsoft Foundation Classes (MFC), which only implements SNMPv1 and the SNMP++ framework from Hewlett Packard which offers support for SNMPv1 and SNMPv2c [12].

Another advantage of the design that is focused on integration of new management services and managed objects is the easy adoptability of the INSMware management framework for new developments, e.g. for new versions of SNMP. The integration of the SNMP++ framework required only minor changes to the source code of the management interface. New SNMP functionality, such as SNMPv3, start to occur and can be implemented into the SNMP++ framework [13] and, therefore, with minor modifications into the INSMware framework. Figure 6 illustrates the layers of the Management Interface and the integration of the different SNMP frameworks within it.

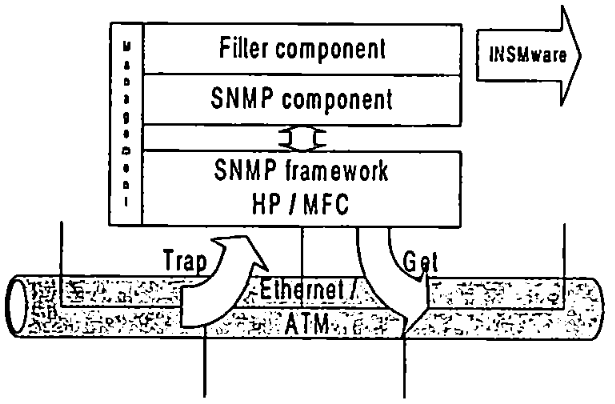


Figure 6: Management Interface layers

The INSMware management framework offers management services for the collection and monitoring of SNMP-Traps and offers services for SNMP Get and GetNext operations. This enables INSMware to act as a SNMP Manager. Traps, which are sent from SNMP managed objects such as routers or even from software in the network, are collected and further processed for the Filter component. The SNMP component then forwards the data to the Filter Layer for further analysis. The Filter component then analyses the data using the configuration parameters from the INSMware Framework. The filtering uses the source address as well as the Object ID (OID) of the trap, e.g. to discover a cold-start of a managed object.

Contrary to the passive management based on traps, where the managed object itself is initiating and sending management data, the Management Interface also enables active monitoring based on SNMP Get and GetNext operations. Here, the Management Interface reads management relevant data from the MIB of the managed objects to monitor and discover status changes or network conditions. The SNMP++ C++ libraries from Hewlett Packard support the different data types (Integer, Unsigned Integer32, Octet, OID, IP-Address, Counter32, Gauge32 and Timeticks) which are defined for SNMP. This MIB management information is compared against predefined values using different operators (e.g. <, >, =) to discover status or administrational changes such as change of System Administrator or critical network conditions, such as high collisions on an Ethernet or the failure of a WAN connection.

When the Management Interface discovers - due to an incoming Trap or a MIB value - that a critical event occurred it protocols the event and arranges the notification of the related INSMware user on its GUI or forwards the notification to a manager via an external communication (e.g. telephone or e-mail). The protocolisation and messaging functions are provided by the Event Controller and the Communication Component.

The specification of the relevant data for the managed objects and events - for the configuration of the SNMP managed objects (e.g. IP-Address, SNMP Version and Community) and the related Get/GetNext and Trap events (e.g. OID, Value, Operant) - is fully implemented via the GUI and saved in the management database (MS Access). The type of notification can be dependent on different parameters (e.g. dependent of day or time) and can be configured for each individual event that occurs. Furthermore, it is possible to set the intervals for the polling and controlling of events according to the requirements and to individually add/delete intervals according to specific management requirements.

3.3 Front-End

As mentioned before, the second component of INSMware that has to be adapted for new services in the domain of Integrated Network and System Management is the user and administrator front end. Two different design approaches were considered: A pure graphical approach, which uses overview maps to represent the network structure (as known from commercial Network and System Management platforms such as HP OpenView), and a more Microsoft Windows Explorer like view showing the network structure as tree. The first approach visualises the network structure very clearly and several network management tools (e.g. HP OpenView) use such an interface. Therefore, they are typically well known for experienced users of management tools, but they are also very space consuming and become unclear for really large networks, e.g. if an internetworking map consists of 30 routers and 50 networks.

The tree view is less space consuming and, for this reason, more suitable for large networks. A problem of this approach is that the network structure is not always hierarchical (e.g. cross-links) and, therefore, the tree representation might not represent the logical and physical network connections as clearly as the established network maps.

The main user interface of INSMware is shown in Figure 7. The network structure is represented by tree at the left-hand side. At the right side, four tabs show information about the node that is currently activated. If desired the user can demand an overview map by pressing the "Network View" button which then shows a network map for the actual network or domain.

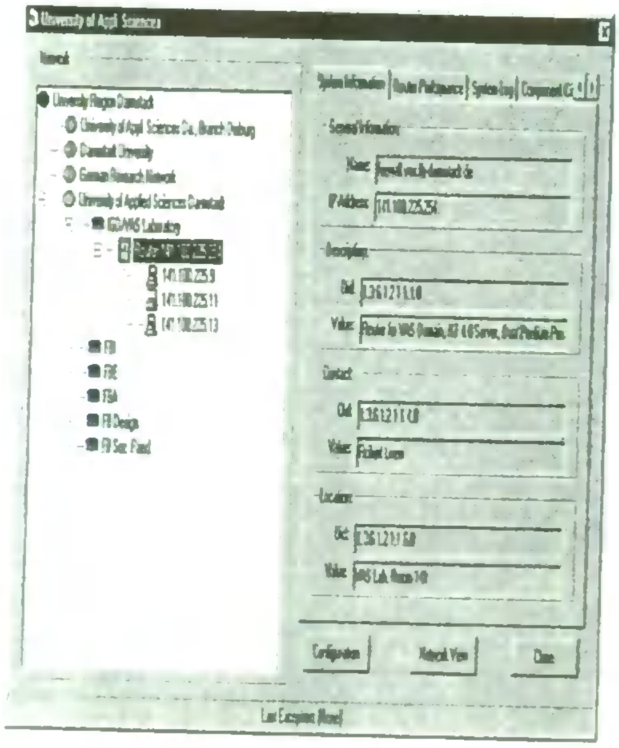


Figure 7: INSMware user interface

The view of the network is configurable for every single user. Every user can name the nodes in their preferred way and the part of the network, which is shown by the tree, is determinable by the user. In order to enable such flexibility, all the data that is presented by the front-end is stored in the system database. This concerns the entire network structure – represented by the tree – as well as the user settings for the configuration.

The four tabs at the right hand side of the GUI provide quick access to the most important information of every network node. The "Information" tab displays the basic information like node name, description, location and contact person at a first sight. The second tab visualises the performance of the network node. The network traffic, utilisation and error rates of the whole network or every single port in case of a router or a switch are presented. The "Log" tab displays an event log for every node. Restarts, port failures and breakdowns of the system will be registered with date and time. The 'Component configuration' tab will enable the user to reconfigure the managed object, e.g. to reset a network interface. All the configuration data – including the settings for the notification events – are stored in the system database.

The first prototype of the front-end component is realised as a Visual Basic program, but there are thoughts to produce a web based front end – using HTML and ASP – in order to allow management and configuration from everywhere without installing client software.

4 Summary and Outlook

This paper has presented an integrated approach to Network and System Management and its prototypical implementation. The use of componentware technology provides flexibility and enables distribution of the management system. The usage of component technology allows the reuse of existing management, messaging and control functionality to integrate new management services.

Our future INSMware research aims at the following goals:

(i) development of extended and new forms of user interaction;

(ii) extension of INSMware to provide unified, integrative end-to-end management services for the different High-Speed Networking technologies and distributed applications.

Goal (i) includes the integration of Web-based management services and the integration of handheld devices, such as the Palm Pilot, to realise ubiquitous computing facilities. As no component models currently exist for such handheld systems, one task will be to develop appropriate integration mechanisms. Another effort in this area will be the integration of speech input and output with INSMware, which is to facilitate system operation in scenarios in which no computing device is available. It also provides a more intuitive way of INSMware utilisation to the user. Another area for future research is a Web based GUI.

In relation to goal (ii), INSMware's current shaping focuses on the management of hardware components. However, this represents only a reduced view to real world information systems, since a correlation exists between managed hardware components (e.g. server systems) and software components operated on the basis of that hardware. Hence, future versions of INSMware will also integrate the management of software components and thus provide integrative management facilities. To overcome architecture-inherent limitation of componentware, we integrate the Component Adapter approach to INSMware, which allows the integration of even semantically incompatible software components.

References

[1] Knahl, Martin; Hofmann, Holger D.; Phippen, Andy. A Distributed Component Framework for Integrated Network and System Management. Information Management and Computer Security, 7(5), pp. 254-260, MCB University Press, Bradford/UK, 1999.

[2] OMG. The Common Object Request Broker: Architecture and Specification, Revision 2.2. OMG Document 98-07-01, Object Management Group, Inc., 1998.

[3] Brown, N.; Kindel, C. Distributed Component Object Model Protocol - DCOM/1.0. Microsoft Corporation, Network Working Group, 1996.

[4] Knahl, M. et al. Integration of ATM management procedures into native integrated network and systems management architectures. Proceedings of the International Network Conference 1998, Plymouth, pp. 91-97. July 1998.

[5] Hofmann, Holger D., Stynes, J. Implementation Reuse and Inheritance in Distributed Component Systems. 22nd Annual International Computer Software and Applications Conference (COMPSAC'98), pp. 496-501, Vienna/Austria, IEEE Computer Society, 1998.

[6] Amrhein, Matthias. Wiederverwendung von Softwarekomponenten — dargestellt am Beispiel eines Überwachungssystems mit Sprachausgabe (Reuse of software components - described by way of example of a monitoring system with speech output). BSc thesis, University of Applied Sciences Darmstadt/Germany, 1998.

[7] CEK-97, Cekro, Z. (1997). Using the SunNet Manager Platform to monitor European ATM activities. 2nd ATM Symposium, Brussels, November 21, 1997.

[8] Stallings, W. (1998). SNMP and SNMPv2: The Infrastructure for Network Management. IEEE Communications Magazine. March 1998.

[9] Case, J. D., Darwin, C., Fedor, M., Schoffstall, M. L. A Simple Network Management Protocol (SNMP). Request for comments (Standard) RFC 1157. Internet Engineering Task Force. May 1990.

[10] ISO-87, ISO (1987). ISO 8824-1987: Information Processing Systems – Open Systems Interconnection – Specification of the Abstract Syntax Notation One (ASN.1). 1987.

[11] LEV-99, Levi, D. et al. (1999). 'Coexistence between SNMP Versions'. SNMPv3 Working Group. Internet Draft, 21 May 1999.

[12] Mellquist, Peter E., SNMP++: An Object-Oriented Approach to Developing Network Management Applications, Prentice Hall, 1997.

[13] Katz, Jochen. SNMPv3 Support for SNMP++. The Simple Times. Volume 7, Number 1. March 1999.

[14] Zitterbart, Martina (ed.). Kommunikation in Verteilten Systemen (Communication in distributed systems). Informatik aktuell, Springer, Berlin, Heidelberg, New York, 1997.

Management of Service Level Agreements using INSMware

M.H.Knahl and S.M.Furnell

Network Research Group, University of Plymouth, Plymouth, United Kingdom
e-mail: knahl@jack.see.plym.ac.uk

Abstract

The paper presents a component-based approach to implement a new framework for Integrated Network and System Management and its application on management services to implement Service Level Agreements (SLAs). Applications are assembled from a set of pre-fabricated components rather than being developed from scratch and are being incorporated into new management domains. In this paper, we describe a project, componentware based Integrated Network and System Management (INSMware), which was built using only component-based techniques. We describe an approach that integrates the software component paradigm with network and systems management and its application upon SLAs. We focus upon the monitoring of SNMP-capable network elements. The requirements for new management services implementing SLAs are outlined and a prototypical implementation is presented.

Keywords

Service Level Agreements, Network Management, System Management, Componentware, SNMP Networks

1. Introduction

The disciplines of Network and System Management encompass all actions taken to enable and guarantee the maintenance and operations of the resources - either hardware or software - in a network. This includes the communication network as well as the server and the end-systems in a network. ITU-T defined five management categories (namely Fault, Configuration, Performance, Accounting and Billing, Security Management) that define the different disciplines and requirements for the management of heterogeneous networking environments.

This paper presents research leading to a component-based framework for Integrated Network and System Management (INSMware) and its application upon the management of Service Level Agreements (SLAs). The transferability of INSMware to other management domains is realised because of a consistent component-based development approach to meet the requirements for integrated management services (Knahl et al. 1999). There are two versions of INSMware: one using the CORBA (OMG, 1998) component model and a Microsoft DCOM (Brown and Kindel, 1996) implementation. This allowed us to study both middleware architectures in detail.

The aims of INSMware are to hide the complexity of the heterogeneous network environment and underlying technologies and to provide a universal framework for the various management services. In addition, INSMware can be applied to several application domains within the INSM area. In this paper, we present the application of INSMware to the management of SLAs.

2. Integrated Management of Service Level Agreements

Limitations and restrictions of existing Network and System Management frameworks, such as distribution of the management services, adoption and integration of new services can be overcome by providing a component-based approach (Knahl et al. 1998; Knahl et al. 1999). Furthermore, the management must be configurable to enable the provisioning and management of Service Level Agreements.

2.1 Service Level Agreements

Service Level Agreements (SLAs) are formal negotiated agreements that help to identify expectations, clarify responsibilities and facilitate communication between a service provider and its customer (Karten, 1998). In a typical customer / provider relationship, a customer demands (and pays for) specified services and for a Quality of Service (QoS) that are defined in the SLAs. To enable (and prove) the fulfilment of those SLAs it is necessary for the provider to have management services that can monitor and control the service status.

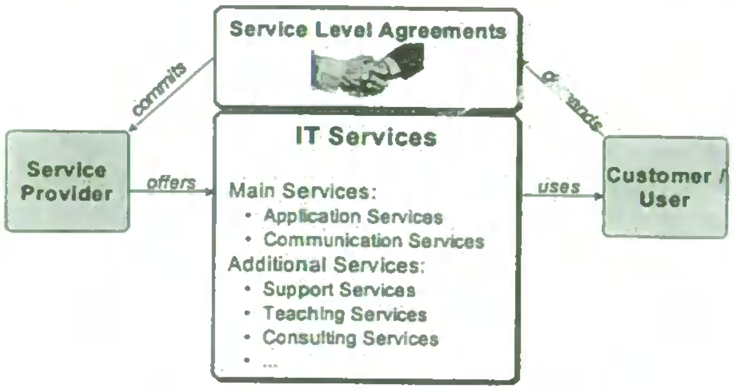


Figure 1: Service Level Agreements

In order to observe the quality of delivered services it is necessary to negotiate QoS parameters as well as modes for the measurement and evaluation of these parameters. QoS parameters must reflect the expectations of the customer and they are part of the SLAs between the customer and the provider. SLAs are for several purposes, e.g. if a service is not delivered with the specified quality the customer may get a discount. Therefore, the SLAs and QoS parameters have to be supervised by the management system of the provider. Further, the provider is obliged by the SLA to report the compliance with agreed QoS parameters. Customer Service Management (CSM) offers a management interface between customer and network provider which enables the customers to monitor and control their subscribed services (Langer et al. 1998).

Examples for SLAs customer/provider relationships could include:

- Network provider (e.g. Deutsche Telekom, a network operator who is acting as a service provider) and customer (e.g. PanDacom, a system integrator with branches all over Germany who is using the services offered by this or another service provider);
- System Integrator (e.g. PanDacom as a system integrator that is offering remote management services based upon QoS parameters to its customers) and customer with service contract.

One example for such QoS parameters is the response time which implies the connectivity or the availability of a certain service where these parameters have to be measured and valued from the customer point of view when the customer uses a specified service, which requires parts of the management system to be installed at the customers side (e.g. intelligent agents or management gateway that report to the provider's management framework). This allows the monitoring and measurement of the services from the customers side, e.g. from an end-system to monitor end to end-connectivity.

SLAs will always change in the course of time due to new requirements for services that demand modified or new QoS parameters. It must be possible to extend the management framework with additional functionality, e.g. for the configuration and monitoring of a new QoS parameter. It is, therefore, important that the architecture is flexible and that it enables fast and cheap integration of these new services. For large scale, distributed networks it is also essential that the management system scales well. Furthermore, the variety of the different hardware and software in existing and future networks requires a high-degree of platform independence for the management system.

2.2 Integrated Management Architecture

Distributed systems allow the partitioning of applications into logical and physical self-contained entities: distributed objects. These distributed objects represent a part of the system-global object model. The possibility to use distributed objects for the realisation of different applications makes them into so-called software components. A software component is a piece of software with one or more well-defined interfaces that is configurable, integrable, and immutable (Langer et al. 1998, Hofmann 2000). By configurable we mean being able to set parameters affecting the properties of a component without requiring its modification. The integration of software component means the connection of incoming and outgoing interfaces, i.e., interfaces being used in the client role and in the server role of a client/server component communication while immutability requires a component to be *physically immutable*. Such physically immutable forms of software are, for example, executable files or dynamic link libraries (DLLs).

The most important criterion of our component definition above is immutability since it allows a dissociation from object-oriented concepts such as Classes and Design Patterns. The functionalities of presented management framework consist of the processing, filtering, and analysis of management relevant data, the presentation in a Graphical User Interface (GUI) and user notifications at the occurrence of predefined states that represent important network states. The system allows that one or more users may be notified over varying communication channels.

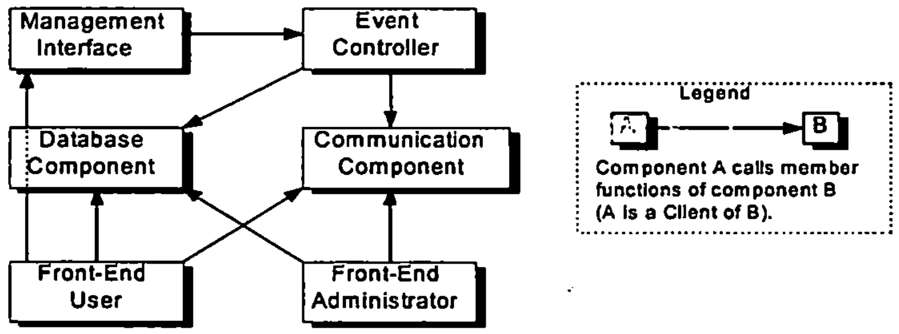


Figure 2: INSMware Components and their Connectivity

The design of the individual INSMware components (see Figure 2) is based on a domain specification which subdivides the entire application domain into subdomains. The data processing system requires a connection to a data source (physically existing system). This is realised by the Management Interface component which exists, similar to device drivers of an operating system, in several different forms and is configurable, as required, for different SLAs. The Management Interface component is installed at the customer side to monitor the specified SLA. The filtered service level data is then forwarded to the SLA provider's management framework using the services of the underlying middleware technologies. The Management Interface component interprets the received data, filters and analyses it, and notifies the event controller component when particular pre-defined exception states occur. Data storage is accomplished by a call to the database component and user notification is effected over the communication components. It must be emphasised that all information about the users that need to be notified (e.g., access to user, user's role regarding the monitored processes) are stored in the system. The communication component itself consists of a set of several components that again implement subdomains, e.g., sending of faxes, voice mails, e-mails. The user can visualise system states by using the front-end user component and can maintain the system by using the front-end administrator component.

2.3 Application Domains and Sub Projects

INSMware was originally conceived to monitor the various network elements and to provide management services in heterogeneous network environments. With INSMware, a timely user intervention in the running of the network processes is made possible when required (e.g. user notification when a network critical condition occurs).

INSMware was applied to the application domain of monitoring SLAs. SNMPv1 and SNMPv2 based managed objects at the customers side can be monitored and relevant management data is then forwarded to the management framework. Two of the components, namely the Management Interface component and the front-end user component, have to be modified to integrate SLA services (e.g. SNMP) into the management framework and the database structure has to be adapted to the service relevant information while the remaining four components can be reused with no modifications. The application domain implemented by the Management Interface component is actually very limited and a universal Management Interface component was developed which can be adapted to different systems by configuration (Amrhein, 1998). The graphical data representation supported by the front-end component to visualise the management information has to be adapted to the respective service domain and remain user-friendly and simple (Vierow, 1998). By generating source code responsible for inter-component communication, a tremendous reduction in the development time for front-end components was possible. Using a CORBA/ DCOM bridge, a platform-independent connection of partially generated front-end components is achieved.

3. INSMware

3.1 Customer scenario

The scenario provides an insight into a whole range of different Management and SLA requirements (see Figure 1). First of all, each provider must manage its own network. An integral part of this is network element management, which concerns the supervision of the availability, capacity utilisation and fault-free operation of the network elements. Added to this is the functioning of the network as a whole. At the access to a network, the providers aim to offer their customers services with a certain Quality of Service based on an SLA. The constant monitoring of service quality is a management task. The management of the customer / provider interface also includes procedures for fault-reporting and for service adaptation or service provisioning. It is essential that customers have access to specific management information (e.g. service quality, service availability) because this is the information they need if they themselves want to develop added value and other new services based on the network services they are already using. For customers, it is the service related information based on the customer SLA that is generally interesting rather than the 'raw' data from the component management of their providers.

In principle, SLAs should exist for all the services of a provider's service offering and used by customers. The SLA contains an exact description of what is offered in a service and defines which costs are applied when a customer uses the service. Since providers of networked systems (e.g. of an enterprise network) are just now slowly being accepted as IT service providers, a large number of services in the IT area are still being used without explicit SLAs. Furthermore, customers require these specifications for planning the use of the IT services in their own business processes (Corsten, 1997).

The current framework implements the Simple Network Management Protocol (SNMP) (Case et al. 1990) to monitor and control managed objects for the provisioning of the SLAs. SNMP is a set of network and system management standards that describe the asynchronous requests and responses for the exchange of management data between SNMP management objects (Stallings, 1998). Virtually all major vendors of end-systems, workstations and network devices such as routers and switches offer SNMP support. In addition, enhancements to the initial SNMP have been pursued in a number of directions (e.g. RMON, SNMPv2, SNMPv3).

The SNMP Manager is network management software that implements the SNMP protocol and is represented in our case by the Management Interface component. The SNMP Agent resides in a managed network element, such as a router or switch or in an end-system such as a PC or Unix Workstation. The agent stores management information in the Management Information Base (MIB) and processes SNMP requests from the SNMP Manager and responses from the agent itself. The proposed INSMware framework is based on a multi-lingual SNMP implementation (Levi et al. 1999). The multi-lingual implementation of the management interface supports the different SNMP versions and enables the seamless integration of the (typically mono-lingual) SNMP based managed objects. Besides that, additional protocols or access policies can be integrated into the Management Interface component.

3.2 Management Interface component

This section describes the Management Interface component for the management of SLAs. The Management Interface component can be installed at the customers side to collect and analyse the management information. It then forwards the required management data to the SLA provider's management framework where it is processed and the required management tasks are undertaken.

3.2.1 Architecture of Management Interface component

One advantage of component-oriented software is the relatively easy reuse of individual parts of the system to adapt the framework for different SLAs, e.g. Network Element Monitoring or Service Management (see Figure 3) . Hence, the SLA provider can reuse the existing systems to offer services to different customers. Modifications to the system to integrate new services and reuse existing components are required on the graphical user interface to represent the service states and the interface to the managed network (Management Interface component for the communication with the managed objects).

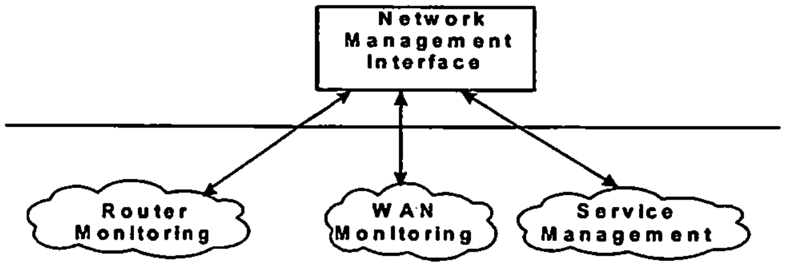


Figure 3: Management Interface Application Domains

To improve and optimise the reusability of the Management Interface component it is split into several smaller components. The filtering functionality analyses the data from the managed objects and decides whether a critical value has been exceeded or whether a critical event has occurred. This evaluation can proceed without further knowledge of the underlying technology (e.g. whether SNMPv1 or SNMPv2) of the managed objects because solely the protocol independent data stream (from the managed objects) has to be analysed. The SNMP component provides the initialisation and de-initialisation of a connection with a managed object. This is individual for every kind of managed object with different management protocols, e.g. differs the initialisation of a connection and the access to a SNMP managed object very much from the access to a file system and has, therefore, be abstracted. For each access technology, a specific component is developed that implements a specific defined interface and which can be used from the general Filter component. This specific component is responsible for the communication with the managed object and transforms/translates the received data into a format that can be used by the Filter component. Whilst developing different management domains it has been discovered that a few parts – particularly in the area of the interface implementation – are similar. For these similarities, source code can be reused. Beside that, the Management Interface offers a high degree of flexibility and provides good reuse for additional and future management domains.

3.2.2 Distribution of Management Interface

The INSMware Architecture allows the distribution of its components over the network. The separate components can also be used for the different tasks involved in SLA Management. The collection and filtering of SLA relevant data at the remote locations reduces the management related traffic over the network. The distribution of management procedures means the realisation of the Client/Server principle on the software level. The different software components of a distributed software system may act as a client or a server, or also as a client and a server if the application requires this.

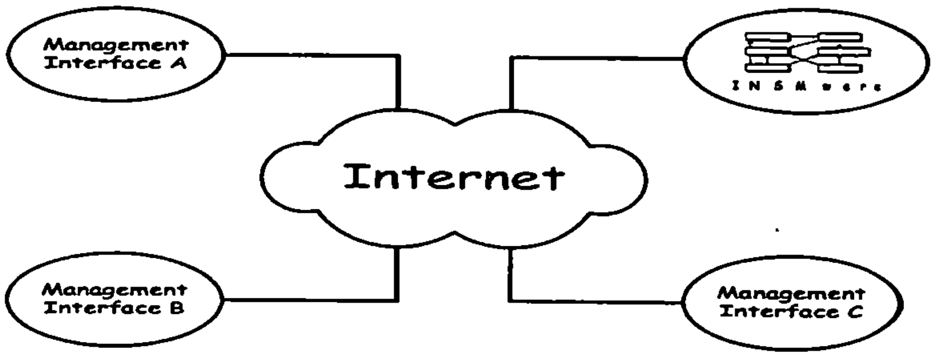


Figure 4: Distribution of Management Interface via Internet

The realisation of distributed software systems is made possible by the increase of low-cost bandwidth on Wide Area Networks (for example the Internet, a new generation of network-enabled desktop operating systems and middleware technologies, such as CORBA and DCOM, that enable the seamless distribution of software components). The possibility to access remote resources (e.g. via the Internet) enables software developers to distribute their software components, even over the boundaries of a corporate network (see Figure 4).

3.2.3 SNMP-Interface

The analysis and design stage of the SNMP-Interface was based on the layered model of the Management Interface. Dynamic access mechanisms have been implemented to read the configuration from an ODBC database. The prototypical implementation of the Management Interface component is written in C++. The SNMP functionalities have been implemented using two different SNMP frameworks based on C++: the SNMP interface of the Microsoft Foundation Classes (MFC) which only implements SNMPv1 and the SNMP++ framework from Hewlett Packard which offers support for SNMPv1 and SNMPv2c (Mellquist, 1997). Functionalities of the implementation include the Monitoring and collection of SNMP Traps and monitoring of SNMP managed objects using SNMP Get / GetNext. This makes it possible to actively monitor and control changes within the SNMP agents.

New SNMP functionalities such as SNMPv3 start to occur. These can be implemented into the SNMP++ framework and, therefore, with minor modifications into the INSMware framework (Katz, 1999). Figure 5 illustrates the layers of the Management Interface and the integration of the different SNMP frameworks into the Management Interface.

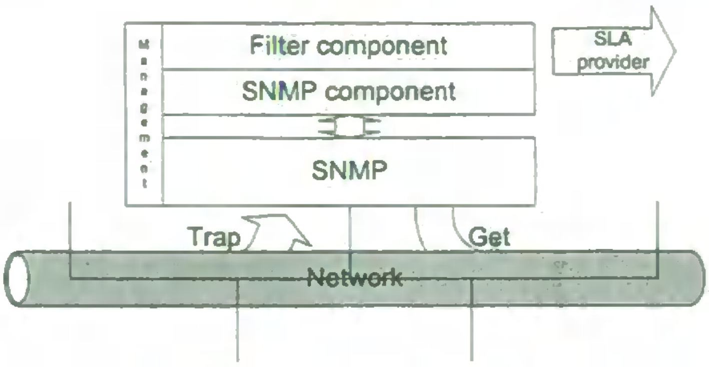


Figure 5: Management Interface layers

The INSMware management framework offers management services for the collection and monitoring of SNMP-Traps and offers services for SNMP Get and GetNext operations. This enables INSMware to act as an SNMP Manager. Traps, which are sent from SNMP managed objects such as routers or even from software in the network, are collected and further processed for the Filter component. The SNMP component then forwards the data to the Filter Layer for further analysis. The Filter component then analyses the data using the configuration parameters from the INSMware Framework. The filtering uses the source address as well as the Object ID (OID) of the trap, e.g. to discover a cold-start of a managed object.

The collected management data is compared against predefined values using different operators (e.g. <, >, =) to discover status or administrational changes such as change of System Administrator or critical network conditions such as high collisions on an Ethernet or the failure of a WAN connection. When the Management Interface discovers - due to an incoming Trap or a MIB value - that a critical event occurred it protocols the event and arranges the notification the related INSMware user on its GUI or forwards the notification to a manager via telephone or E-Mail. The protocolisation and messaging functions are provided by the Event Controller and the Communication Component.

The specification of the relevant data for the managed objects and events - for the configuration of the SNMP managed objects (e.g. IP-Address, SNMP Version and Community) and the related Get/GetNext and Trap events (e.g. OID, Value, Operator) - is fully implemented via the GUI and saved in the management database (implemented in Microsoft Access). The type of notification can be dependent on different parameters (e.g. dependent of day or time) and can be configured for each individual event that occurs. Furthermore, it is possible to set the intervals for the polling and controlling of events according to the requirements and to individually add/delete intervals according to specific management requirements.

3.3 Visualisation of Network infrastructures and service states

As previously mentioned, the second component of INSMware that has to be adapted for new services in the domain of Integrated Network and System Management is the user and administrator front end. Two different design approaches were initially considered: A pure graphical approach that uses overview maps to represent the network structure as known from commercial Network and Sytsem Management platforms such as HP OpenView (see Figure 6) and a more Microsoft Windows Explorer-like view, showing the network structure as a tree (see Figure 7). The first approach visualises the network structure very clearly, particularly for experienced users of management platforms such as HP OpenView. A lot of network management tools use such a map-based interface and for this reason they are well known for experienced users, but they are also very space consuming and become unclear for really large networks, e.g. if an internetworking map consists of 30 routers and 50 networks.

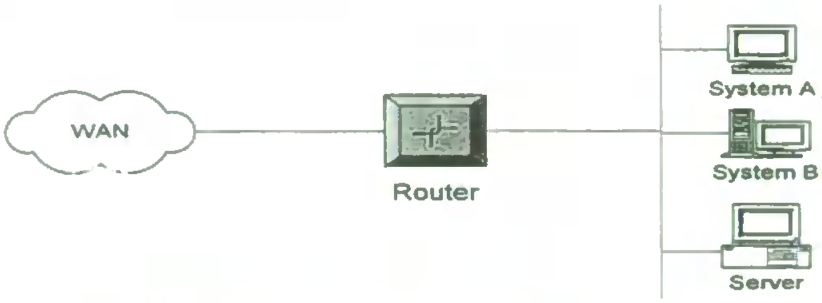
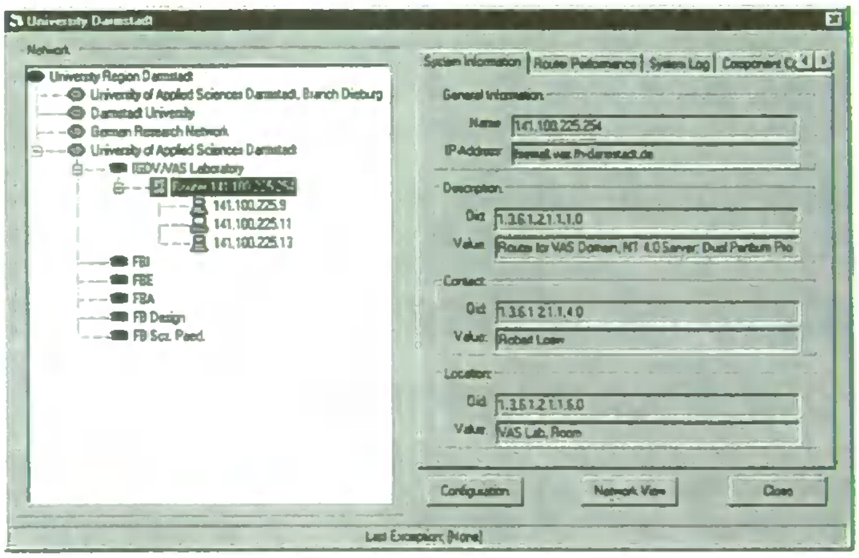


Figure 6: Map representing Network Infrastrucuture

The tree view is less space consuming and, for this reason, more suitable for large networks. A problem of this approach is that the network structure is not always hierarchical (e.g. cross links) and, therefore, the tree representation might not represent the logical and physical network connections as clearly as the established network maps.

To get the best of both worlds the implemented front end is a mixture of both approaches. The main user interface is shown in Figure 7. The network structure is represented by a tree on the left hand side. On the right side, four tabs show information about the node that is currently activated. If desired, the user can demand an overview map by pressing the "Network View" button which then shows a network map for the actual network or domain.



University Region Darmstadt
- University of Applied Sciences Darmstadt, Branch Dieburg
- Darmstadt University
- German Research Network
- University of Applied Sciences Darmstadt
  - IGDV/VAS Laboratory
    - Router 141.100.225.254
      - 141.100.225.9
      - 141.100.225.11
      - 141.100.225.13
  - FBI
  - FBE
  - FBA
  - FB Design
  - FB Soz. Paed.

Name: 141.100.225.254
IP Address: [illegible]
Oid: 1.3.6.1.2.1.1.1.0
Value: Router for VAS Domain, NT 4.0 Server; Dual Pentium Pro
Oid: 1.3.6.1.2.1.1.4.0
Value: Robert Loew
Oid: 1.3.6.1.2.1.1.6.0
Value: VAS Lab. Room

Last Exception: [None]

Figure 7: User interface representing network strucures and states

The tree view is configurable for every single user. Every user can name the nodes in his preferred way and it is also intended to make the part of the network that is shown by the tree determinable by the user. In order to enable such flexibility, all the data that is presented by the front-end is stored in the system database. This concerns the entire network structure – represented by the tree – as well as the user settings for the configuration.

The four tabs at the right hand side of the GUI provide quick access to the most important information of every network node. The "Information" tab displays the basic information like node name, description, location and contact person. The second tab visualises the performance of the network node. The network traffic, utilisation and error rates of the whole network or every single port in case of a router or a switch are presented. The "Log" tab displays an event log for every node. Restarts, port failures and breakdowns of the system will be registered with date and time. 'Component configuration' allows configuration of managed objects, e.g. to reset a network interface.

The information presented (except Component Configuration) provides the management data and are for monitoring and controlling the managed Service Level Agreements. The configuration facilities enable the user to configure the management services. The selection and monitoring configuration of specific MIB variables specifies the management services for specific managed objects. In addition, SNMP traps can be configured to monitor and configure events reports from the SNMP entities themselves. The user can define notification events by specifying threshold, notification channel (voice, fax, pager, email) and priority. All the configuration data – including the settings for the notification events – are stored in the system database.

4. Summary and Outlook

Our future INSMware research aims at the extension of INSMware Management Services to provide a unified, integrated management framework. INSMware's current shaping focuses on the management of hardware components. However, this represents only a reduced view of real world information systems, since there is a correlation between managed hardware components and software components operated on the basis of that hardware. Hence, future versions of INSMware will also integrate the management of software components and thus provide fully integrated management facilities.

Furthermore, the development of extended and new forms of user interaction, including the integration of Web-based management services and the integration of handheld devices such as the Palm Pilot to realise ubiquitous computing facilities will be considered. As no component models currently exist for such handheld systems and their operating systems, one task will be to develop appropriate integration mechanisms. The first prototype of the front-end component is realised as a Visual Basic program, but there are thoughts to produce a web-based front end in order to allow management and configuration from everywhere without installing client software. Another effort in this area will be the integration of speech input and output with INSMware, which is to facilitate system operation in scenarios in which no computing device is available. It also provides a more intuitive way of INSMware utilisation to the user.

5. References

Amrhein, M. (1998), *Wiederverwendung von Softwarekomponenten — dargestellt am Beispiel eines Überwachungssystems mit Sprachausgabe (Reuse of software components - described by way of example of a monitoring system with speech output).* Diploma Thesis, University of Applied Sciences Darmstadt/Germany.

Brown, N. and Kindel, C. (1996), *Distributed Component Object Model Protocol – DCOM/1.0.* Microsoft Corporation, Network Working Group.

Case, J. D. , Darwin, C., Fedor, M. , Schoffstall, M. L. (1990), *A Simple Network Management Protocol (SNMP).* Request for comments (Standard) RFC 1157. Internet Engineering Task Force. May 1990.

Corsten, H. (1997), *Management von Geschäftsprozessen: Theoretische Ansätze – Praktische Besispiele.* W. Kohlhammer GmbH. Stuttgart, 1997.

Hofman, H.D. (2000), *Software Component Reuse by Adaptation.* PhD Thesis. Institute Of Technology, Cork, Ireland. March 2000.

Karten, N. (1998), "How to Establish Service Level Agreements". Karten Associates. Randolph, MA, USA, 1998.Katz, J. (1999), "SNMPv3 Support for SNMP++". *The Simple Times.* Volume 7, Number 1. March 1999.

Knahl, M. , Bleimann, U. , Furnell, S. M. , Sanders, P. W. (1998), "Integration of ATM management procedures into native integrated network and systems management architectures". *Proceedings of the International Network Conference 1998*, Plymouth, UK, July 1998, pp91-97.

Knahl, M. Hofmann, H. D. and Phippen, A. (1999), "A Distributed Component Framework for Integrated Network and System Management". *Information Management and Computer Security,* Vol. 7, No. 5, pp254-260.

Langer, M., Loidl, S. and Nerb, M. , (1998), "Customer Service Management : A More Transparent View to your subscribed services". *Proceedings of the 9th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM 98).* Newark, USA. October 1998.

Frye, R. , Levi, D. , Routhier, S. , Wijnen, B. (2000). "Coexistence between Version 1, Version 2 and Version 3 of the Internet Standard Network Management Framework". Internet Engineering Task Force, SNMPv3 Working Group, RFC 2576. March 2000.

Mellquist, P.E. (1997), *SNMP++: An Object-Oriented Approach to Developing Network Management Applications.* Prentice Hall, 1997.

OMG. (1998), *The Common Object Request Broker: Architecture and Specification, Revision 2.2.* OMG Document 98-07-01, Object Management Group, Inc.

Stallings, W. (1998), "SNMP and SNMPv2 : The Infrastructure for Network Management". *IEEE Communications Magazine.* March 1998.

Vierow, T. (1998), *Entwicklung eines Generatorwerkzeuges zur Unterstuetzung der Gestaltung von graphischen Benutzeroberflaechen (Development of a generator tool for supporting the design of graphical user interfaces).* Diploma Thesis, University of Applied Sciences Darmstadt/Germany.

A distributed component framework for integrated network and systems management

M. Knahl
Network Research Group, School of Electronic, Communication and Electrical Engineering, University of Plymouth, Plymouth, UK
H.D. Hofmann
Department of Mathematics and Computing, Cork Institute of Technology, Ireland
A. Phippen
Network Research Group, School of Computing, University of Plymouth, Plymouth, UK

Keywords
Systems management, Distributed data processing, Computer networks, Automated teller machines, Computer software

Abstract
It is proposed that future work should move on from existing network and system management methodologies to consider enhancing the management methodology for ATM and other networking technologies to meet existing and future requirements. This paper outlines an area where significant potential for further research exists and proposes a component-based management architecture. The discussion indentifies the technological limitations and architectural drawbacks of current solutions and proposes the extension of existing services and an enhanced management framework to overcome the current restrictions.

Introduction

Today's growing IT and telecommunication networks, and the demand for expanded bandwidth to enable real-time multimedia applications, has made network and systems management the primary enabler for effective deployment of these complex and heterogeneous environments. Recent research has concentrated upon the identification of existing requirements, systems and future developments in the fields of asynchronous transfer mode (ATM) and integrated network and systems management. The requirements from the users and the technological point of view have been analysed. The research has been based on theoretical study and practical industry requirements and challenges. Management frameworks and applications that solve dedicated management disciplines have been implemented.

The main focus of the research presented is the integration of ATM management procedures into a novel integrated network and systems management framework that leads to a new management system. Research to date has identified that essential ATM management services are not specified and implemented to allow integrated management services. ATM is going to be one networking technology that has to coexist and interoperate with other switching or non-switching based networking technologies, such as Ethernet, and new services will be integrated that add further complexity. The integration of local area network (LAN) and wide area network (WAN) services adds further complexity that has to be considered whilst implementing management services. This paper presents a novel component-based network and systems management framework that is capable of integrating the various management services.

Requirements for further optimisation of management procedures

The research has identified that ATM management differs from today's network and systems management procedures (Willets and Adams, 1996; Ahrens, 1994). ATM and other networking technologies will coexist in networks and have to interoperate to guarantee existing services and allow future services (Data Communications International, 1997). Ongoing research is investigating the developments in network and systems management and will focus on native technologies, ATM and directions for future technologies. This process leads to an ever increasing number of management services and new networking applications (e.g. Internet) add further requirements to the network and systems management frameworks.

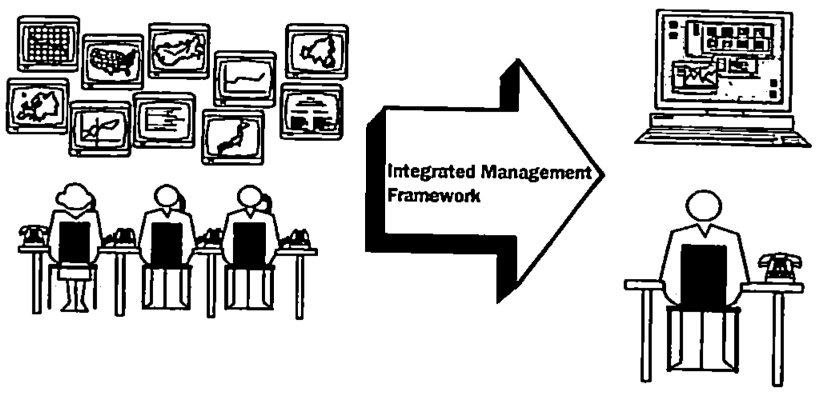
A primary goal of an integrated network and systems management framework is to have an integrated system that allows the monitoring and management of the various services as illustrated in Figure 1. Existing management architectures typically consist of various dedicated management applications for specific management tasks, e.g. router management applications for router configuration and management or switch management applications for switch management. Furthermore, management tasks such as router management are in most cases vendor specific, e.g. Cisco routers require CiscoView (CiscoWorks) for configuration and management tasks whereas BAY Networks routers require Site Manager (Optivity). Typically, large enterprise software networks are built with equipment from different vendors. Beside that, system management is typically based on other applications. The management architecture presented provides a platform to integrate the different services into one integrated management framework.

The challenges of ATM and native network management have been identified (Knahl *et al.*, 1998) and solutions offered by optimised management procedures will be outlined. Integrated management procedures are required that enable interoperable ATM deployment. The proposed management solution will provide an enhanced methodology that outlines the integration of networking technologies such as the different Ethernet implementation in the LAN and its management requirements.

Existing and future applications have crucial requirements on the communication infrastructure.

Therefore, management services have to be able to provide a certain quality of service (QoS) to the users. An application such as video conferencing handles time critical information (video and audio) and requires appropriate support from the system components and especially from the network. New management services have to be developed to enable and offer integrated telecommunication services in heterogeneous environments that were formerly performed by separate architectures, such as the telephone network and the data network. Beside that, the Internet, and therefore TCP/IP with its rapid growth of new services and its rapidly growing number of users, is one main player. All these technologies have different technological characteristics and QoS architectures that need to be integrated (Wolf, 1998). Management services, e.g. for real-time multimedia applications like video conferencing in heterogeneous network environments with ATM and IP subnets, involve communication-interaction and QoS requirements over the end-to-end communication path. The provision of the end-to-end service requires the provision and integration ATM and IP QoS architectures (Borden *et al.*, 1995).

Figure 1
Consolidation of management services



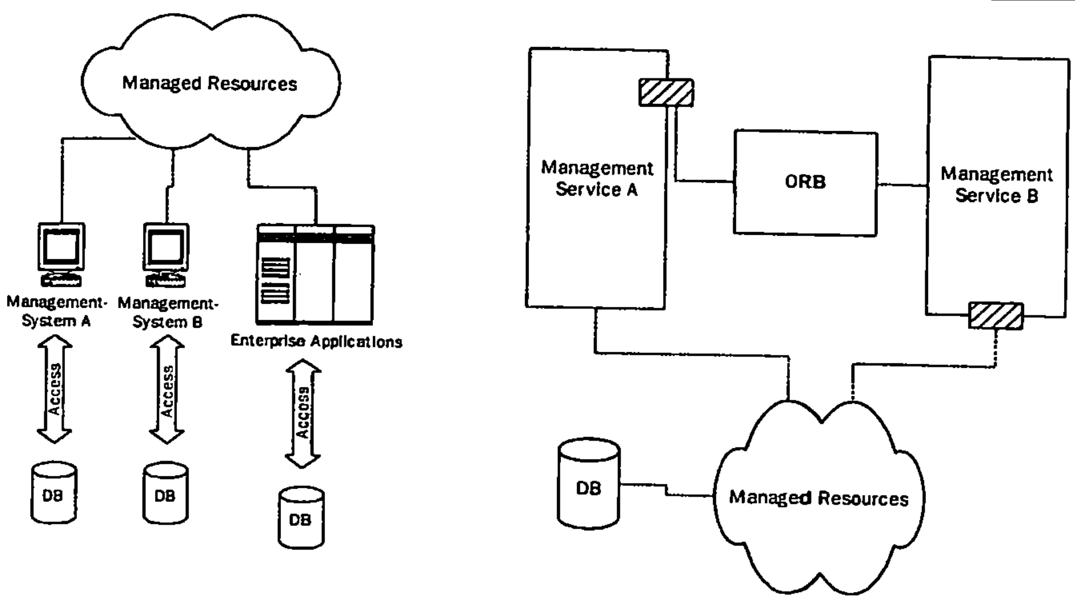
Development of management services

Standardisation and development in the fields of ATM and integrated network and systems management is an ongoing process and the required management services have been identified (Emanuel, 1994). Most of the management architectures implemented today consist of different dedicated management systems, such as stand-alone management applications and network/system management platforms. Existing interfaces for these products are typically proprietary, product-specific interfaces to access the GUI or the underlying database. Therefore, additional or new functionality is difficult to integrate due to the interfaces' proprietary nature. Typically, large enterprise networks consist of multiple locations and the network management system is also distributed. Enterprise networks can span time zones and continents (Strauss, 1995). Besides the standard requirements for integrated network and system management, different organisations have additional, or even unique, management requirements. Existing management architectures typically consist of different management systems and databases (e.g. network management or systems management) with architectural limitations such as restricted (or no) distribution and integration of management services.

If we consider possible scenarios to improve the proprietary nature of existing management systems, one scenario could be that a central site wants to have the ability to manage the switches and active network components at all sites, but give each site the ability to manage a particular functional or geographic domain from that site (distributed network management). Another scenario could be that a central site wants to manage the network elements and the critical server systems of the central site and the remote sites (distributed network and systems management). The management system must have the flexibility to accommodate different access and control functionality. Distribution of management tasks have to allow the backup and recovery of all sites within the WAN. Figure 2 compares the different approaches for dedicated management systems and the proposed distributed system that provides integrated management services (Bleimann, 1996).

We identify a requirement for a management architecture which provides the functionality of existing management architectures, but with additional flexibility and distribution to accommodate new functionality or scaling management requirements in an expanding

Figure 2
Dedicated management systems and CORBA-based management systems



network. The management architecture presented is based on a distributed system and provides a common management database.

Emerging component software standards such as the "Common object request broker architecture" (CORBA) (OMG, 1995) or the "Distributed component object model" (DCOM) (Brown and Kindel, 1996) enable us to develop software components which expose their functionality through interfaces based on a standard. Therefore, any other piece of software written to the same standard can, in theory, use the services offered by developed component. Additionally, these standards provide common functionality for the communication and distribution of software components.

By developing the management architecture on top of one of these standards (CORBA), we can develop management services that offer functionality through standard interfaces. Therefore, an optimised management system can be developed through the integration of a number of standard management components. New functionality can also be added to the system by the development of new components to the same standard. Additionally, different implementation of the same interface enables us to interchange components based on the management requirements of a given system, without having to change the core architecture. Distribution of component such that management distribution can match equipment distribution is also possible using the component standard.

New services that require ATM QoS are starting to occur. These services need to be managed and integrated into today's environments and new management strategies are required. The concept and a model framework of an integrated services network provide the foundation for an integrated management model. The service model of communication technologies, such as IP or ATM, must be incorporated into the management architecture to enable the management of integrated services via heterogeneous networks (Bordon *et al.*, 1995). Thus, the framework will be based on different management protocols and specifications, such as SNMP (Stallings, 1998) and TMN (Fowler, 1995).

An integrated network and systems management framework must allow the description and implementation of the relevant aspects of any management procedure in a way that is suitable as a basis for all areas of management. For example, it has to enable the monitoring of an Ethernet switch port as well as the monitoring of an ATM switch port. Beside that, the management framework has to integrate the different management disciplines (i.e. network and systems management). For example, it has to allow the implementation of software management and distribution procedures (systems management) on the one hand and procedures that allow the network manager to monitor important network links in the LAN and WAN (network management) on the other.

Guidelines for a management methodology

The proposed system provides guidelines for a management methodology. These will incorporate the latest developments in the standardisation and industry implementation of ATM management and integrated network and systems management. A robust, scalable and open end-to-end interactive management architecture that is – to the extent possible – independent of the details of the physical layer of the underlying network infrastructure is required.

The architecture will incorporate ATM and non ATM networking technologies. Guaranteeing and monitoring the QoS for specified applications, which involves determining the QoS parameters demanded by applications to satisfy user-perceived QoS requirements and appropriately translating them at the interface units to the underlying networking and transport layers (e.g. IP and ATM) will be integrated into the system. This will enable the network manager to implement services that require ATM QoS parameters over heterogeneous network infrastructures.

For an interaction between ATM and other technologies, the seamless interworking and management of the heterogeneous network infrastructure is important. That means that the provision of management guidelines and services for the various objects in the network have to be applicable regardless of what is inside the network and whether both ends are located in the same technology domain or not. It must be possible to enable the required end-to-end management, e.g. providing a homogenous service over a heterogeneous network.

Using components to enable integrated management services

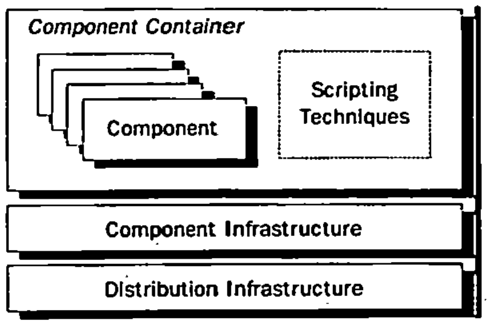
The research to date has identified that existing frameworks and implementations do not meet the identified requirements due to architectural limitations. The implementation technologies used require the development of new management applications that operate beside existing applications if new networking specifications, technologies and products have to be incorporated into existing management systems. One example for technology evolution is the Ethernet. Originally it was specified to provide 10Mbit/sec. shared media. It has evolved to Fast-Ethernet (100Mbit/sec.) and Gigabit Ethernet. All these different specifications can be found in today's networks and the proposed integrated management system will be able to integrate them. The usage of software components based in an integrated management infrastructure will allow the reuse of existing components and the integration of new components into the management system.

The system has to provide distributed management capabilities to meet the organisational and functional requirements. Management services, such as network and system management, have to be integrated into one management architecture that provides interfaces for the required management services, data storage and a common GUI. The system has to be able to integrate new services and new technologies in a practical and economic manner.

The suggested system will be based on a componentware architecture as illustrated in Figure 3. Componentware is a new paradigm in software engineering that enables high-level software reuse (Gruender *et al.*, 1996). A component is defined as a piece of software with one or more well-defined interfaces that is configurable, integrable and not modifiable (Hofmann, 1997). Software components have to be combined in a componentware architecture that consists of several elements providing different services for the realisation of componentware. Each software component is embedded in a component container and uses services of the component-infrastructure. The ccmponent infrastructure, for its part, uses services of the distribution infrastructure. The component infrastructure offers mechanisms for component management and component communication. This includes the migration of components to dedicated hosts and the realisation of runtime component information (Hofmann, 1998). The distribution infrastructure incorporates the communication and distribution features of different middleware technologies (e.g. CORBA or DCOM).

Using a component-based approach for the implementation of a novel management system offers several advantages. Componentware provides open, flexible and modifiable application structures that can be adapted to the identified requirements. It allows the reuse of existing components that offer

Figure 3
Componentware architecture



management services and reduces redundant application functionality through integration of different management disciplines into one management system. The usage of middleware technologies allows a highly flexible and distributed management solution to be built.

Deliverables and future outlook

Deliverables and the potential of the presented research and the proposed integrated management system are summarised below:

1 Extension of the theoretical aspects of current research, with a far more detailed examination of the current state of ATM and integrated network and systems management and its impacts on future developments.
2 To look beyond the present day and how management procedures may progress, in terms of technologies and environments, and also how these developments can be brought, ultimately, to the real world network environments, to the network operators and network planners who need integrated management procedures to enhance management capabilities. Central to this is consideration and guidelines to enable network and system operators to exploit the promise of new integrated management procedures.
3 Different approaches for an integrated network and systems management architecture have been examined that led to the design of a novel framework. Protoypical implementation of the proposed system is currently being undertaken, leading to a novel software product that exploits the promise of integrated management procedures through a novel methodology. The demonstration platform includes the ability to integrate different management services and shows the capacity to fulfill future management requirements and to meet today's and future requirements.

Network management procedures in existing and future networks

Network management procedures in existing and future networks have to be discussed to identify required management services and architectures. This includes interaction approaches for the management of ATM and existing technologies. A discussion of application scenarios to see what kind and amount of interaction is required and the results will provide further knowledge for new management services.

A highly flexible and transparent network management architecture that extends the functionality and scalabitilty of existing implementations has been designed and provides the foundation for the prototypical implementation. An integrated management methodology and management framework for the adoption of management services for ATM networks is going to be developed which meets the identified requirements.

Software solution

Research to date has shown that an integration of ATM management procedures into native network and systems management architectures is required to enable integrated management services. The prototypical implementation of the described management methodology and the outlined management framework based on components has been developed and will be further expanded and evaluated.

The research has identified that a management architecture has to enable distributed management services and should provide open interfaces that are based on industry accepted standards. Figure 4 illustrates the architecture of the proposed component based management framework. This framework is based on a management architecture discussed in (Hofmann, 1997; Muench, 1998).

The management interface implements the connection between the management system and the managed objects. It can be compared to a device driver used in operating systems. There can exist one management interface that can be used with different managed resources, or one for each particular resource. The event manager deals with events and notifications received by the management interface and assigns them to a set of actions. This set of actions can, for example, be simple user notifications or direct interventions to actions performed by managed resources. The database component serves as a common data repository and this component will be implemented on the basis of an industry standard database product. The messaging handler is used to distribute and forward the required management information via e-mail, fax, voice-mail, short message service (SMS) and pager services. The front-end user components integrate the management users into the system. These access the database component and display information about the current system state and the system's state history graphically to the user and allow a direct notification of other users via the messaging handler. Users that are only allowed to monitor do not have direct access to the management interface, whereas users with further permissions and access rights can access the managed objects via the management interface, e.g. to perform an action via SNMP at a managed object.

Note that the management framework described above does not commit a user of this framework to a particular distribution scenario since our component based management framework is fully scalable. This is due to the fact that its components communicate via mechanisms offered by the distribution infrastructure. Another advantage of our approach is that components can be easily replicated or migrated in order to meet changing systems or user requirements. Management interfaces can, for example, be located physically near to the managed resources whilst more than one event manager component can be installed with the system by replication in order to be able to process a great amount of incoming events.

The presented management framework will enable the integration of existing management procedures as well as the integration of future management procedures that may be required to integrate as yet unknown technologies. Furthermore, the suggested management framework will ensure that multiple software developers are able to integrate different management services for the framework using its core services. The prototypical implementation will demonstrate its architectural principles and management capabilities to provide end-to-end management services. ATM and non ATM managed objects will be incorporated via the common network management interface into the system to demonstrate the integrated management architecture (Figure 5). The network management interfaces will access the managed objects such as switches and routers via the implemented and provided management and control interfaces. Today, most network elements have SNMP interfaces that provide a set of standardised management and control informations and

Figure 4
Component-based management framework

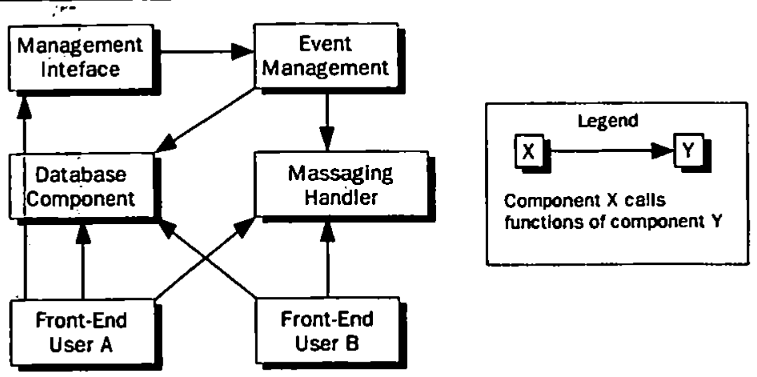
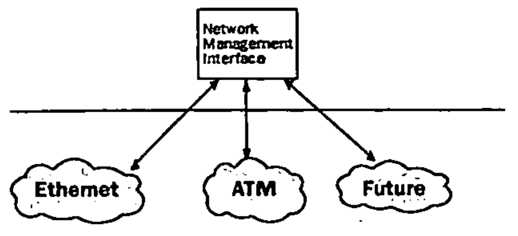


Figure 5
Network management interface



vendor specific extensions. SNMPv2 and other interfaces (e.g. JAVA-based management and control interfaces) are starting to be specified and implemented into network elements. The distributed component framework enables implementation of the appropriate and required interface for the various network elements that have to be integrated via the management interface. The management interface can be expanded to implement additional and future access interfaces (e.g. SNMPv2, JAVA). These interfaces are transparent for the user because the same management services will manage objects with SNMP or other interfaces.

Middleware technology such as CORBA or DCOM and object-oriented programming languages (C++ and Java) are used for the implementation of the components and the management services. This component based architecture allows the abstraction and reuse of implemented services and, therefore, enables the integration of new services and technologies.

Conclusions

This paper examined the current state of ATM and integrated network and systems management and its impact on future developments. Existing limitations and restrictions have been examined and new approaches and solutions to overcome the current limitations are outlined.

Different approaches for an integrated network and systems management architecture have been examined and a novel, component based management framework that provides integrated management services has been introduced. The presented management framework exploits the promise of integrated management procedures through a novel methodology and new software technology. This management architecture has the capacity to meet today's and future management requirements and includes the ability to integrate different management services.

References

Ahrens, M. (1994), "Key challenges in distributed management of broadband transport services, *IEEE Journal – Selected Areas on Communications*, August.

Bleimann, U. (1996), "Project systems engineering: integrated network and systems management ("Projekt Systementwicklung: Integriertes Netzwerk und System Management"), Technical report, University of Applied Sciences, Darmstadt, Germany, March.

Bordon, M. *et al.* (1995), "Integration of real-time services in an IP-ATM network architecture", *RFC 1821*, August.

Brown, N. and Kindel, C. (1996), *Distributed Component Object Model Protocol – DCOM/1.0*", Microsoft Corporation, Network Working Group, http://www.microsoft.com

Data Communications International (1997), *Market forecast 1998*, Data Communications International, December.

Emanuel, (1994), "Open management – addressing real business needs, *Proceedings IEEE Network Operations and Management Symposium*, February.

Fowler, H. (1995), "TMN-based broadband ATM network management, *IEEE Communications Magazine*, March.

Gruender *et al.* (1996), "Reuse and inheritance in distributed object systems", *Proceedings of the International Workshop on Trends in Distributed Systems*, Aachen, Germany.

Hofmann, H.D. (1997), "Componentware – integration of software components in distributed computing environments", M.Sc. thesis, Cork Institute of Technology, Ireland.

Hofmann, H.D. (1998), "Implementation reuse and inheritance in distributed component systems", *Proceedings of Twenty-Second Annual International Computer Software and Applications Conference (COMPSAC'98)*, 19-21 August, IEEE Computer Society, Vienna, Austria, pp. 496-501.

Knahl, M. *et al.* (1998), "Integration of ATM management procedures into native integrated network and system management architectures", *International Network Conference 98*, July, Plymouth, pp. 91-7.

Muench, V. (1998), "Tool-based individual software development in distributed computing environments", M.Sc. thesis, Cork Institute of Technology, Ireland.

Object Management Group, Inc. (OMG) (1995), *The Common Object Request Broker: Architecture and Specification. Revision 2.0*, Object Management Group, Inc., http://www.omg.org

Stallings, W. (1998), "SNMP and SNMPv2: the infrastructure for network management", *IEEE Communications Magazine*, March.

Strauss, P. (1995), "At last: Net managers that distribute the load", *Datamation*, 15 February.

Willets, K. and Adams, E. (1996), "Managing a broadband environment", *IEEE Communications Magazine*, December.

Wolf, L. (1998), "Quality of service in heterogeneous networks", *International Network Conference 98*, July, Plymouth.

Integration of ATM Management Procedures into Native Integrated Network and System Management Architectures

M.Knahl[†], U.Bleimann[‡], S.M.Furnell[†], P.W.Sanders[†]

[†] Network Research Group, School of Electronic, Communication and Electrical Engineering, University of Plymouth, Plymouth, United Kingdom.

[‡] Fachhochschule Darmstadt, Haardtring 100, D-64295 Darmstadt, Germany.

e-mail : mknahl@plymouth.ac.uk

Keywords : ATM, Integrated Network and System Management, SNMP, CMIP

Abstract:

The paper considers the requirements for the integration of Asynchronous Transfer Mode (ATM) management procedures into native network and system management architectures. Other than the research into network and system management, the standardisation and realisation of management frameworks and the industry realisations of open integrated management platforms, network and system management remains mainly a collection of single solutions aimed at handling a limited amount of specific areas. Today the management standards, procedures and implementations for Local Area Networks (LAN) and Wide Area Networks (WAN) are completely different. There is no End-To-End Management for the interconnected systems that communicate via heterogeneous networks. Beside that, connections and applications handling delay-sensitive traffic require management procedures that are not required for asynchronous data applications. Asynchronous Transfer Mode uses one technology for transmitting diverse traffic types and for a number of various applications (e.g. data, voice or multimedia) across LANs and WANs. Beside pure ATM Networks, ATM is designed to operate as backbone technology or High-Speed Technology for interconnecting Server or workgroups with other networking technologies such as Ethernet or Token Ring. The paper shows that integrated management procedures for the various networking technologies are required to operate and manage these networks in an efficient and practicable way to ensure the different communication services and an appropriate solution is described.

ATM Management

Research and industry are currently in the process of developing a multi-layer (five-layer) ATM management model and an Operations, Administration and Maintenance (OAM) interface. The model consists of services and interfaces for the LAN as well as for the WAN and enables the distribution of management intelligence through the network. The problem is that there is still a long way to go before a final and suitable solution will be available. Even with ATM Networks being implemented in the Local Area Networks as well as in the Wide Area Networks, private and public network management systems will continue to maintain separate views into the network with different services and different functionality (Hershey,1997). Implementing a network management scheme that will enable both views involves the building of gateways between SNMP systems at customer sites and the CMIP and proprietary protocol systems used by carriers (Ahrens,1994). Standardisation groups, including the Internet Engineering Task Force (IETF), Network Management Forum and the International Telecommunication Union (ITU), have been working with the ATM Forum to negotiate interworking standards acceptable to both sides.

ATM Management Model

Research and industry (in the form of the Network Management Working Group of the ATM Forum) are investigating and developing an end-to-end management model that includes management services for LAN and WAN Networks and that lays out standards for the interworking between the two. The model will also define gateways between Simple Network Management Protocol (SNMP), Common Management Information Protocol (CMIP) and proprietary systems. Five management interfaces are defined to meet ATM management requirements (Varadajan,1997). All are essential to end-to-end monitoring and control. LANs are addressed by M1 and M2, which define the interface between the network management system and an ATM end station or an ATM switching system. M3 is the Customer Network Management interface. The merger of private and public networking technologies begins at M4, while M5 is the management interface between a carrier's own network management systems.

M1 and M2 embrace SNMP-based specifications defined by the Internet Engineering Task Force (IETF). These include Management Information Bases (MIB) II and relevant standard MIBs for DS-1, DS-3, and SONET connections. Also included is the experimental AToM MIB (RFC 1695) which, in particular, is expected to reduce the proliferation of widely varying vendor-specific ATM MIBs. M3 is the Customer Network Management (CNM) interface. M3 describes the interface between the customer and carrier management systems that gives the customer a view into the carrier's network, so that network managers will have real-time control over the services they use to be able to monitor the services end-to-end. The merger of public and private networking technologies begins at M4. This is the management interface enabling Network Management Level (NML) views and Element Management Level (EML) views to the carrier's network management system and the public ATM network. Both the private network manager and the carrier have to be able to monitor the WAN services, what is new for enterprise network management systems. The manager of the private enterprise wishes to take advantage of public services yet retain control over them to guarantee services for the enterprise network users. The carrier, on the other hand, requires an overview of customers' networks - which would then give it the ability to offer network management as a value-added service. The protocol-independent MIB for M4 supports SNMP objects defined in accordance with the SNMP Structure of Management Information (SMI), as well as CMIP objects that conform to GDMO (Guidelines for Development of Managed Objects). M5 is the management interface between a carrier's own network management systems and is the most complex interface. More research has to be done to finalise specifications defining interworking between SNMP and CMIP (Fowler, 1995).

Operations and Maintenance Services

Operations and Maintenance (OAM) cells will provide extensive management functionality. Services will be dynamically reconfigured in the event of failure to meet service requirements and network devices and end-stations will be able to negotiate and reconfigure themselves to guarantee service-level objectives. These architectures will be distributed, in the sense that management intelligence and functionality will be distributed throughout the network infrastructure. The OAM Flow Reference Architecture, also known as the Management Plane Reference Architecture, defines aspects of ATM point-to-point Virtual Circuit that can be monitored using specialised OAM cells. The Architecture divides a VC into five distinct layers, labelled F1 through F5 and defines the flow of ATM cells through these layers.

Research and industry are developing OAM cells for fault management, performance management, and activation/deactivation (for starting and terminating fault and performance management functions). OAM cells will give ATM network devices the ability to gather information about end-to-end connections, reduce the need to distribute MIBs throughout the network, and cut the amount of management-related traffic on the network. The ATM Forum Network Management working group has specified several OAM cells targeting fault management, including Alarm Indication Signal (AIS) cells and Far End Reporting Failure (FERF) cells, which communicate failure information throughout the network. The working group also has specified an OAM loopback capability, which uses a special loopback cell, for verifying connectivity and diagnostic problems that AIS or FERF cells cannot. There will also be a continuity check cell that ascertains whether idle connections are still up or have failed. Whenever an ATM switch fails and a virtual path or virtual connection is interrupted, each adjacent switch in the network automatically generates an AIS cell and sends it to all downstream switches. The AIS cell alerts the other switches in the network of the failure and gives them the opportunity to devise alternate routes for virtual connections that would normally cross the failed switch. When failure disrupts only one-half of the full-duplex ATM connection that traverses several switches, FERF cells are generated. If there is a failure of this kind, traffic can still move through the network, but only in one direction. In this instance, the switch closest to the failure will generate an AIS cell and transmit it back down the network to alert the source of the failure. Then the source switch sends a FERF cell to the destination via the remaining half of the connection. This alerts the destination that its traffic is no longer getting through and an alternate connection route should be set up.

In fault situations where an AIS or FERF cell would not indicate a problem, such as when a VC has been misconfigured, there would be no actual failure in the network, and traffic would get through to both source and destination-but the connection would not be between the right end points. The loopback cell goes from source to destination and requires that the destination switch or end-station mark the cell and return it. The ATM Forum also has defined a fault management cell for continuity checking. When a VC is idle for a certain period of time, end-stations or switches involved in the connection can send a continuity check cell to verify that the connection is still up.

OAM capabilities still have to be developed and built into ATM equipment. Beside that, ATM management will need to interoperate with native network and system management. Today, vendors are rolling out management applications based on proprietary solutions or implement current proposals and interim specifications. Some vendor-specific management implementations might always be necessary to cover areas not addressed (such as application management and interfaces between ATM and legacy technology).

Challenges and Integration of ATM and native Management

Today's Network and System Management Architectures enable an integrated view of the systems and the networking elements such as routers and switches through management consoles that implement different services and management disciplines (Figure 1). The major network management standards include Simple Network Management Protocol (SNMP), Common Management Information Protocol (CMIP) and Telecommunication Management Network (TMN). Network Management standards and protocols such as SNMP or CMIP or proprietary implementations enable the management of the elements in the network (Stratman, [illegible]4).

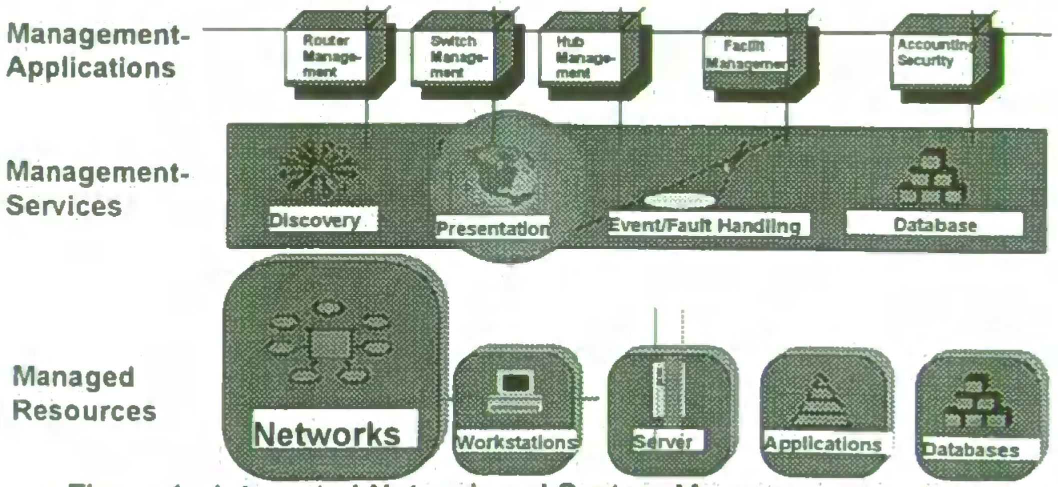


Figure 1 : Integrated Network and System Management Architecture

Implemented on industry-standard platforms, the network management system should also utilise an architecture with a flexible database system to ensure data integrity and to access, manipulate and display data from various management applications for ATM and non-ATM network elements or even for enterprise applications such as SAP R/3 (Figure 2). This is also essential for presenting a consistent view of network views and maps.

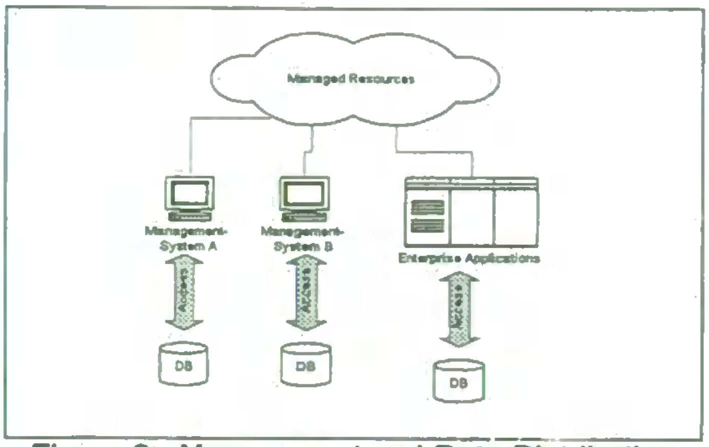


Figure 2 : Management and Data Distribution

Multi-technologies and multi-vendor environments also have the need for standards compliance. Standard protocols and standard interfaces from the network management system will help ensure that the system can continually support changes and enhancements to the underlying hardware. Different management services have to be integrated. The following picture shows the integration of two industry-leading network and system management platforms (Figure 3).

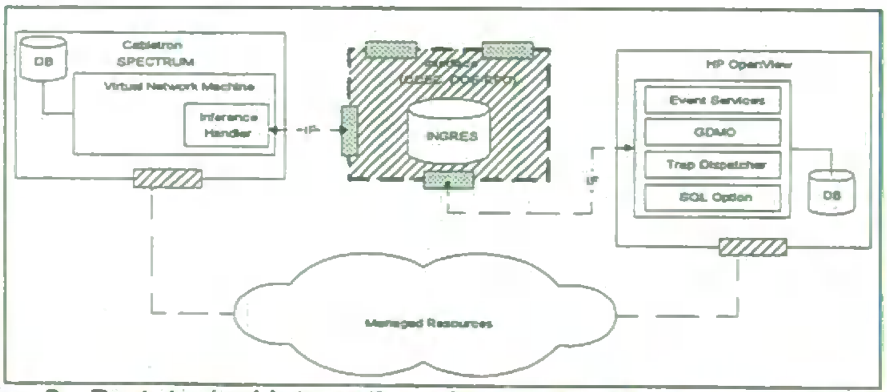


Figure 3 : Prototypical Integration with leading Management Platforms

Most networks consist of multiple locations and the network management system is also distributed. Typically, enterprise networks span time zones and continents (Strauss, 1995). Beside the standard requirements for integrated network and system management, different organisations have additional or even unique management requirements. One scenario could be that a central site wants to have the ability to manage the switches and active network components at all sites but give each site the ability to manage a particular functional or geographic domain from that site. The management system must have the flexibility to accommodate different access and control functionality. The integration of Management Systems could be built on CORBA services. Distribution of management tasks have to allow the backup and recovery of all sites within the WAN. Figures 4 and 5 illustrate different approaches for integrating management services (Bleimann, 1996).

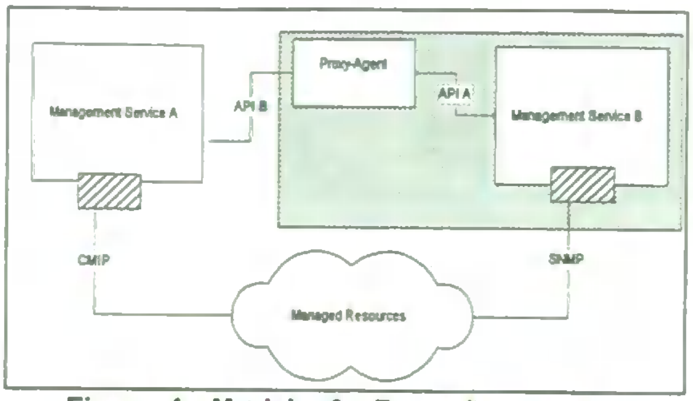


Figure 4 : Model of a Proxy Integration of different Management Services

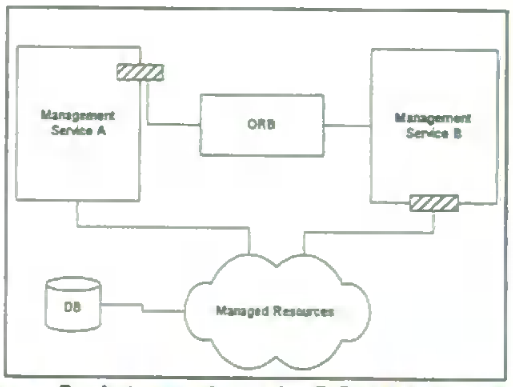


Figure 5 : Integration via CORBA Services

Challenges and Impacts of Integrated Management Services

Our research has shown, that management standards such as SNMP or SNMPv2, or today's Management-Platforms like SunNet Manager, Tivoli TME 10 or HP OpenView Network Node Manager basically cannot handle the end-to-end management of ATM networks. The challenge for the network operators is how to migrate from these restricted systems to an architecture that integrates an ATM management model (Pathak, 1994) and that will help them meet the goal of providing seamless, end-to-end connectivity and management for tens of thousands of users across LANs and WANs (figure 6). We found that this seamless architecture presents problems and raises challenges in just about every area of network management (Cekro, 1997).

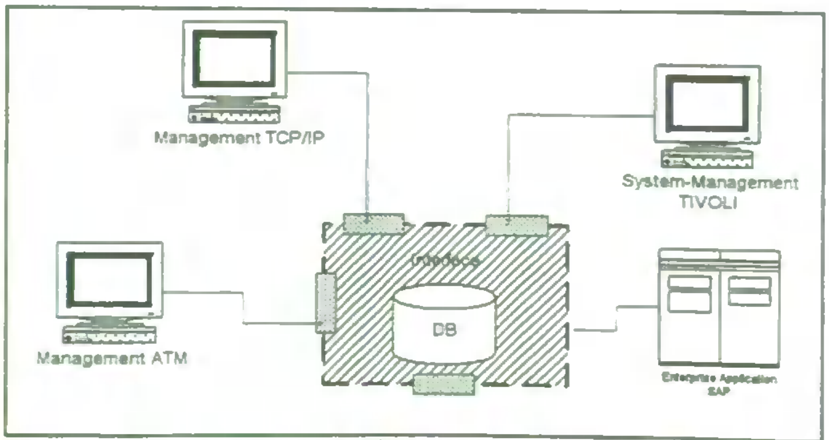


Figure 6 : Integration of different Management Services

Configuration management has to support the various networking equipment. A mechanism should be available that allows the sharing of configuration management data between different network management applications. The network management system must allow the set-up of end-to-end connections in response to user service requests and according to Quality of Service (QoS) parameters such as throughput and delay. The configuration services should incorporate a single view for easy browsing and changing of the various configuration parameters through a Graphical User Interface (GUI) for the different networking equipment. The network management system has to provide the tools to define routing tables to support the connections and to ensure that these tables in different switches and routers are synchronised. Existing tools and services do not provide network configuration that guarantees service provisioning involving ATM and native networking equipment. Four major classes of service are defined to choose from in ATM: Available Bit Rate (ABR), Constant Bit Rate (CBR), Unspecified Bit Rate (UBR) and Variable Bit Rate (VBR). Guaranteeing the desired service level for the users involves configuring optimal connection parameters on each switch in the network - which requires setting traffic priorities, choosing routes, and taking into account Wide Area lines availability and other factors describing the current network state.

ATM high capacity and ability to handle high amounts of different data types adds another dimension to fault management, network monitoring and performance management. For a network manager to keep information on what is happening, the ATM management architecture must enable the collection of a wide variety of statistics, reporting to class of service, fault, performance, and usage. Furthermore, carriers and end-user organisations require fault management services that provide network monitoring, fault detection and correlation, diagnostics, multiple views of the network and helps administrators analyse and resolve fault conditions. The integration of a new technology into an existing network such as ATM is complicating the fault management procedures. Because ATM is more complex than conventional TDM (time-division multiplexing) networks or LANs, network managers must have both a physical and logical (virtual) view of the network service. Complex network structures require dynamic visualisation tools instead of the static topology views of the physical network that are a feature of today's management platforms. Operators should be able to obtain different focused on network problems. Expert systems should guide the operators in diagnosing and resolving a problem.

The new end-to-end architecture of services and applications challenges the requirements of cost allocation and billing. The complexity of these procedures in a heterogeneous network incorporating ATM services and ATM WAN makes a powerful, reliable cost allocation and billing solution essential but complex. The billing system must give carriers and corporate networkers a way to identify the appropriate class of service for each connection and then measure usage. Usage-based billing will drive the cost of ATM down and make the service affordable for end-users, for whom the opportunity to buy ATM bandwidth according to actual usage rather than at a flat rate is interesting. Organisations with private networks need to determine how much traffic comes from one department and how much from another.

Complex networks require design tools to incorporate simulation and modelling exercises based upon real-world data capable of network modelling. These tools could run "what if" scenarios based on real-world data such as line cost, bandwidth availability and bandwidth utilisation, allowing customers to prototype new services and analyse how growth will affect the network and to control network planning.

References

Ahrens, M. (1994), "Key Challenges in Distributed Management of Broadband Transport Services", IEEE Journal selected areas on communications, August 1994

Bleimann, U. (1996), "Prokekt Systementwicklung: Integriertes Netzwerk und System Management", Fachhochschule Darmstadt, March 1996

Cekro, Z. (1997), "Using the SunNet Manager Platform to monitor European ATM activities", 2nd ATM Symposium, Brussels, November 21, 1997

Fowler, H. (1995), "TMN-Based Broadband ATM Network Management", IEEE Communications Magazine, March 1995

Hershey, P. (1997), "A New Approach to Telecoms Network Management", Telecommunications, August 1997

Pathak, G. (1994), "Integrated Network and Service Management for the North Carolina Information Highway", IEEE Network, November/December 1994

Stratman, R. (1994), "Development of an Integrated Network Manager for Heterogenous Networks Using OSI Standards and Object-Oriented Techniques", IEEE Journal on selected areas in Communications, August 1994

Strauss, P. (1995), "At Last: Net Managers that Distribute the Load", Datamation, February 15, 1995

Varadarajan, S. , "Frame-Level Performance Management Requirements for ATM Networks", ATM Forum/97-0610R1, September 1997