EIDGENÖSSISCHE TECHNISCHE HOCHSCHULE LAUSANNE
POLITECNICO FEDERALE DI LOSANNA
SWISS FEDERAL INSTITUTE OF TECHNOLOGY LAUSANNE

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Communication Systems Division (SSC)
EPFL CH-1015 Lausanne, Switzerland
http://sscwww.epfl.ch

Annotated Typology of Distributed Network Management Paradigms

Jean-Philippe Martin-Flatin and Simon Znaty

14 March 1997

Technical Report SSC/1997/008

Submitted to DSOM'97, Sydney, Australia, 21-23 October 1997

Annotated Typology of Distributed Network Management Paradigms

Jean-Philippe Martin-Flatin	Simon Znaty
Telecommunication Services Group, TCOM Laboratory, Swiss Federal Institute of Technology (EPFL), 1015 Lausanne, Switzerland	Multimedia Networks and Services Dept, École Nationale Supérieure des Télécommunications de Bretagne, 35512 Cesson Sévigné, France
Email : jpmf@tcom.epfl.ch	Email : znaty@rennes.enst-bretagne.fr

14 March 1997

Abstract

Over the past few years, network management has steadily evolved from a centralized model, where all the management processing takes place on a single network management station, to distributed models, where management is distributed over a number, potentially large, of nodes. Among distributed models, one, the weakly distributed hierarchical model, has been around for several years, whereas a flurry of new ones, based on mobile code, distributed objects or cooperative agents, have only recently emerged. Which of these techniques will eventually win ? Will several ones have to coexist ? How do they compare to each other ? In order to provide a framework to analyze these issues, this paper presents a comprehensive typology of all network management paradigms known to date, whether they have been successfully implemented already or whether they are still confined to the research community. By comparing these models with those used in another research field, enterprise management, we delineate a common trend of evolution, and attempt to predict what the future holds for network management.

Keywords : Distributed Network Management, Organizational Models, Mobile Code, Management by Delegation, Distributed Objects, Intelligent Agents.

1 Introduction

Network management has thrived on centralized or hierarchical models for many years. Soon after the advent of open systems in the second half of the 1980's, proprietary solutions gradually gave way to two open protocols, SNMP [21] and CMIP [10], in the first half of the 1990's. These protocols primarily addressed what was then perceived as the most critical feature lacking in existing network management systems : interoperability between multiple vendors. SNMP was widely adopted by the IP world to manage LANs, WANs, and even systems recently. In parallel to this wide-scale development, CMIP, richer but more complex than SNMP, found a niche market in the telecommunications world, as the ITU-T decided to adopt the OSI model [7, 8, 67] as the basis for its Telecommunications Management Network (TMN) model [32, 51].

Despite the lack of competition between these two protocols, which looked set to rule their separate markets for many years, the use of both SNMP and CMIP has been questioned in the recent past, together with their common underlying models. Why is it that more and more network managers are now demanding Distributed Network Management (DNM), when the same people were happy with centralized or weakly distributed hierarchical models a couple of years ago ? What triggered this sudden and massive shift toward DNM ?

The answer is probably twofold. First, DNM addresses what traditional models fail to provide : scalability, flexibility and robustness. These three features, identified by Goldszmidt [17, 18] to motivate the use of his own model, Management by Delegation (MbD), can actually justify any form of DNM, as we show in section 3. Second, progress in distributed applications design (CORBA, intelligent agents...) and languages (Java, TeleScript, KQML...) since CMIP and SNMP were devised, suggested new ways of organizing network management. Thanks to technology transfer if we have a positive mind, or thanks to opportunism if we feel more ironic (some people assert these were just solutions looking for a problem to solve), the network management community was suddenly overwhelmed by an avalanche of new tools over the span of a couple of years. Research is currently going in all directions, and it is increasingly difficult to tell in which direction network management is currently heading.

The diversity of new potential solutions is a bit confusing, and prompts a number of questions. Will the IP and telecommunications worlds have to deal with multiple management paradigms and protocols in the future, thereby gaining in functionality, but losing out in terms of standardization and contemplating dreaded heterogeneity problems ? Will both worlds continue to evolve separately ? Or will they both eventually converge toward a single solution ?

A good way of unveiling global trends in an apparent chaos of new ideas is through classification. But if the literature offers many examples of typologies of organizational structures in other research fields, oddly enough, fairly little has been published recently in the area of network management. In the past, most authors simply sketched the centralized approach as opposed to hierarchical approaches (see Sloman [55], Leinwand [35], Znaty [71], Bapat [3], etc.), some even in great detail (Boutaba [5]). Others, such as Schönwälder [52] and Baldi [2], studied some of the new paradigms, but none studied the whole range of distributed models, nor analyzed in detail the pros and cons of the various forms of distribution.

The first objective of this paper is therefore to compile a comprehensive typology classifying the whole range of network management paradigms known to date, whether they have been successfully implemented already or whether they are still confined to the research community. To do so, we first define a common terminology in section 2. We then define a simple typology in section 3, and review all network management paradigms. In sections 4 and 5, we draw a parallel between network management and enterprise management, another research field sharing closely related organizational models. By cross-fertilizing these two research fields, we delineate a common trend underlying the evolution of management in both worlds. We then analyze the granularity at which the delegation process can take place, and introduce the notions of micro-task and macro-task. In section 6, we identify 12 criteria on which network managers evaluate the gain that DNM-based solutions give them over traditional ones. Among these criteria, we select the two most discriminating ones to build our enhanced typology in section 7. This typology, as opposed to the simple one presented in section 3, is solution-oriented rather than model-oriented : it describes the different ways of delegating a network management task, and maps them onto the different technologies available to date. We then try to meet the second objective of this paper, which is to try and predict what the future holds for DNM. Based on the common trend identified earlier, we show in section 8 that if network management evolves in the same way enterprise management did, we should be going from traditional paradigms to mobile code in the near future, then evolve to distributed object-oriented management, and finally integrate agent-based cooperative management.

2 Terminology

Before we go in detail into the different network management paradigms known to date, we must first acknowledge that the network management community has not fully converged on a common terminology yet. If most people agree that the centralized model is characterized by a single Network Management Station (NMS), concentrating all the management processing, and a collection of agents limited to the role of dumb data collectors, there are different views on several other definitions. For example, some authors motivate the use of their new highly distributed models by criticizing the centralized model, but overlook hierarchical models ([17, 37] to quote only two) ; and many others simply ignore the cooperative model. To address this confusion, we therefore propose the following terminology.

Decentralized management is to the enterprise world what *distributed management* is to computer science : a management paradigm based on the delegation of tasks to other entities. These entities are persons in the enterprise world, machines or programs in computer science. *Delegation* is a generic word, used in both contexts to embody the process of transferring power, authority, accountability and responsibility [15, 45] for a specific task to another entity.

Delegation generally goes down the enterprise or computer network hierarchy : a manager at level (N) delegates a task, i.e. a management processing unit, to a subordinate at level (N-1) ; this is known as *downward delegation*. *Upward delegation* goes the other way round : in the enterprise world, Mullins [45] gives the example of an employee 'delegating' his tasks to his manager when he is off sick. Downward and upward delegations are two kinds of *vertical delegation*, typical of hierarchical models. A *hierarchical model* is characterized by a multi-layer pyramid, comprising a *top-level manager* (at level 1), several *mid-level managers* (at levels 2, 3, etc.), and *operatives* at the lowest level (this word is used in UK [15], but it may be more politically correct to use another one in other countries !). In network management, NMSs globally refer to the top-level and mid-level managers, whereas operatives are called *agents*. Orthogonally to vertical delegation, we have *horizontal delegation*, between two peers at the same level, typical of *cooperative models* used in Distributed Artificial Intelligence (DAI). DNM relies on either an underlying hierarchical model, a cooperative model, or a combination of the two : indeed, any model outside the realm of centralized models belongs to DNM.

Delegation is normally a *one-to-one relationship*, between a manager and an agent in a hierarchical management model, or between two peers in a cooperative management model. Arguably, delegation may also be considered, in rare cases, as a *one-to-many relationship*, where a task is delegated to a group of entities, collectively responsible for the completion of the task [48]. One-to-many delegation is forbidden by most authors in enterprise management [45, 63, 1, 15]. It could be envisaged in DAI. In network management, we propose to classify it as a form of cooperation, by coupling the hierarchical and cooperative models : a manager delegates a task to an agent, and this agent in turn cooperates with a group of agents to achieve this task. In the case of a *many-to-many relationship*, we are clearly in the realm of cooperation rather than delegation.

Now, let us come back on two acronyms we already defined : DNM and NMS. Why should DNM stand for *Distributed Network Management*, as we said earlier, rather than *Decentralized Network Management*, as Baldi et al. [2] advocate ? Our choice is motivated by usage in other computer science research fields : for years, people have been referring to *Distributed Systems*, *Distributed Artificial Intelligence* and *Distributed Processing Environments*. It therefore makes sense, in our view, to translate the acronym DNM into *Distributed Network Management*. Decentralized management is a notion pertaining to enterprise management and economics rather than computer science.

As for NMS, its meaning has shifted over the years from *Network Management System* to *Network Management Station*. The reason for this is clear : as we show in section 3.2, SNMP, when it was first released, assumed an underlying centralized model, characterized by a single network management station. The whole network management system was made of a management application running on a single workstation. Several years later, SNMP moved to a hierarchical model, a la CMIP, where the network management system actually comprises multiple stations. Since we are now clearly in the days of DNM, we will translate NMS into *Network Management Station* throughout this paper.

To cope with legacy network devices, whose internal SNMP/CMIP agent does not support the capabilities described by strongly distributed models, we assume in this paper that old network devices make use of delegated agents if necessary. A *delegated agent* is a network management gateway, dedicated to a certain network device and external to it, which is located between the manager and the SNMP/CMIP agent. It is transparent to the management processing. It translates the semantics of strongly distributed technologies into SNMP/CMIP protocol primitives, and vice versa. When a delegated agent is used, the SNMP/CMIP agent embedded in the network device is called a *dumb agent*. Throughout this paper, when we refer to an *agent*, we may as well refer to the pair {dumb agent, delegated agent} or simply to the SNMP/CMIP agent.

Finally, since the word *agent* has traditionally a different meaning in DAI and network management, we will speak of *intelligent agents* (IAgs) when we mean an agent in the DAI sense throughout this paper.

3 Network Management paradigms

3.1 Simple typology

With these definitions in mind, let us specify a first, simple network management typology, to prepare the ground for the richer one presented in section 7. Our approach, based on the underlying organizational model, is threefold. First, we ought to separate centralized models from distributed models, just like traditional models do. Second, we should try to isolate what is inherently different between traditional and new paradigms. Third, we should distinguish models relying on vertical delegation from those based on horizontal delegation.

If DNM encompasses a vast and growing number of technologies, they can all be classified in two broad types: weakly and strongly distributed technologies, implementing respectively weakly and strongly distributed models. *Weakly distributed models* are characterized by the fact that network management processing is concentrated in a handful of NMSs, whereas the numerous agents are limited to the role of dumb data collectors (we typically have 1 or 2 orders of magnitude between the number of agents and the number of NMSs). The typical examples of weakly distributed network management are the hierarchical paradigms underlying CMIP and SNMPv2. *Strongly distributed models*, on the other hand, decentralize management processing down to each and every agent : management tasks are no longer restricted to NMSs, all agents and NMSs take part in the network management processing. Many strongly distributed technologies have been suggested in the recent past, which can be grouped into 3 families : mobile code and distributed objects, based on vertical delegation and thus hierarchical models, and cooperative agents, based on horizontal delegation and cooperative models.

This leads us to the typology depicted in Figure 1, comprising four types of network management paradigms : centralized paradigms, weakly distributed hierarchical paradigms, strongly distributed hierarchical paradigms and cooperative paradigms, which we are now going to review in more detail in this section.

	centralized paradigms	hierarchical paradigms	cooperative paradigms
not distributed	SNMPv1		
weakly distributed		RMON, SNMPv2, CMIP, TMN, TINA	
strongly distributed		mobile code, distributed objects, Web-based management	intelligent agents

Figure 1 : Simple typology of network management paradigms

3.2 Centralized models : SNMPv1

Network management is currently dominated by one organizational model, the centralized model, and one management protocol, SNMP. The first stable release of SNMP, often referred to as SNMPv1, appeared in 1990 [21]. It was tremendously successful, and received a very wide acceptance within a few years : by 1993, virtually any device in the IP world had an SNMP agent embedded. If SNMP-based systems management has been less successful so far than SNMP-based network management, there is a clear move in this direction as well, and several commercial offerings integrate both.

Why has SNMP been so successful in the IP world ? First, to anyone who had to manage a heterogeneous network, open systems and SNMP were a double blessing. Gone were the days of multiple network management platforms : all these IBM, Digital... consoles could be disposed of and replaced with a single IP workstation. SNMP enabled network managers to defeat heterogeneity by standardizing management. Second, the administrative model (to use the IETF jargon) of SNMP is simple, very simple indeed [50] : it is based on a centralized model, the simplest of all organizational models, and the well-known client-server paradigm. On the client side, management data is stored in MIBs, data repositories characterized by a very simple structure, cast in iron at design time. A common kernel, known as MIB II [22], is supported by all SNMP-compliant network devices. It provides for standard management data such as device description, per-interface and per-protocol statistics, etc. MIBs are typically vendor-specific (Cisco, Fore...) or technology-specific (ATM, FDDI...). They are polled on a regular basis by the NMS, which concentrates all the management 'intelligence' such as building usage statistics, taking corrective actions when certain thresholds are reached, etc. The semantics offered by the protocol are very basic, hence simple to implement : the manager can use 3 primitives, Get, GetNext and Set, to access or modify the MIB on the agent (manager-agent request-response), and the agent can spontaneously send a trap to the manager (agent-manager unconfirmed information). And security, finally, is reduced to its simplest expression : a clear text string called the *community*. With such a simple model, it took vendors little effort to offer support for SNMP on each and every network device. In the early 1990's, they all raced against each other, claiming to be more *open* than the others. SNMP became a marketing tool, and vendors used it a great deal.

Three independent evolutions soon exposed a major weakness in the centralized model, and with it SNMP : scalability. First, throughout the 1990's, the IP world has been expanding at a very fast pace. The so-called TCP/IP stack (which also supports UDP, thus SNMP), once limited to Unix machines, became available on most network devices in the early 1990's, and on most PCs and Macintoshes in the mid 1990's (thanks largely to the success of the WWW). Second, the size of networks grew dramatically, as the number of machines installed was growing fast, and the proportion of networked machines (as opposed to standalone) was also increasing quickly. Third, the size of SNMP MIBs increased several times : if IP routers and hubs just had a few FDDI, Ethernet and Token Ring ports to manage in the recent past, ATM switches and intelligent hubs now have many more entities to manage (ports, cross connections...), which requires more data to be brought back to the NMS. So the very success of SNMPv1 was the cause of its decline : if it was good at managing relatively small networks, it could not scale to large networks (e.g. geographically dispersed enterprises), and could not cope with ever more management data. A new paradigm was needed.

3.3 Weakly distributed hierarchical models

CMIP, another protocol used primarily by network operators in the telecommunications world, had already shown how to solve this problem : by using a hierarchical model, with multiple levels of managers. In the remainder of this section, we show how SNMP evolved toward a hierarchical model, and how CMIP proved successful in telecommunications management with TMN and TINA.

3.3.1 RMON

RMON probes, remote devices dedicated to network monitoring, were the first and simplest form of delegation added to SNMPv1. The RMON MIB was issued as early as 1991 [23], and updated in 1995 [28]. RMON probes were designed to monitor network segments, and typically Ethernet segments. By gathering usage statistics in these probes, network managers could delegate simple network management tasks, relieve the NMS from the corresponding CPU burden, and decrease the amount of management data to move about.

The RMON MIB is split into 10 groups: *statistics* is dedicated to making per-segment instantaneous statistics; *history* to compiling per-segment historical statistics ; *alarm* to sending alarms when a threshold is crossed by some usage statistics ; *hosts* to making per-host statistics ; *hostTopN* to identifying the top N hosts with regard to certain statistics ; *matrix to* making usage statistics per {sender, destination} pair ; *filter* to defining filter equations (matched packets may be *captured* or may generate *events*) ; *capture* to storing captured packets in buffers ; and finally *event* to controlling the generation and notification of events (companion of *alarm*, in charge e.g. of sending SNMP traps). For instance, with RMON, a LAN manager can be informed automatically by email when the usage of an Ethernet segment goes beyond 50% on average over the past 5 minutes, and can be beeped when it goes beyond 70% (in practice, the collision rate should also be taken into account).

If the RMON MIB [28] was typically implemented in RMON probes a few years ago, it is generally implemented in intelligent hubs today : there is no need for an extra network device, since the backplane of an intelligent hub is already crossed by all packets. Tasks achieved by RMON are static, in the sense that only the gauges and traps hard-wired in the RMON MIB are available, together with all kinds of combinations thereof via the filter mechanism. If contact is lost between the RMON-capable agent and the central NMS, statistics are still gathered, but no independent corrective action may be undertaken by the agent.

3.3.2 SNMPv2

Support for a fully-fledged hierarchical model was added to SNMP in a new version of the protocol, SNMPv2, released in 1993. Three MIBs are defined as part of the SNMPv2 specification: the SNMPv2 MIB [26], similar to the SNMP group in MIB-II, also contains information about the setup of an SNMPv2 manager or agent ; the Manager-to-Manager MIB [27] supports the distributed management architecture ; and the Party MIB [25] defines the security framework. The key enhancements brought by SNMPv2 to SNMPv1 were fourfold [59]. First, the SNMPv2 structure of management information (SMI) codifies a number of existing practices which were left unspecified by SNMPv1 ; it also adds new object definitions such as 64-bit integers, and a new convention is provided for creating and deleting conceptual rows in a table, a la RMON. Second, two new protocol primitives were added : GetBulk, to efficiently retrieve large blocks of data from a MIB (a la CMIP), and Inform, to enable one manager to send trap-like data to another manager. Third, the manager-to-manager capability was added so as to enable the use of SNMPv2 both as a centralized model or a weakly distributed hierarchical model. Fourth, an enhanced security scheme was adopted, which introduced the notions of MIB view, context and party.

Compared with SNMPv1, the acceptance of SNMPv2 was fairly poor. First, the administrative framework, the security framework and the Manager-to-Manager MIB proved to be unworkable in deployment [41]. Second, it was deemed too complex by non-technical users accustomed to the simplicity of SNMPv1. Third, its security scheme was heavily criticized by advanced users who wanted a truly secure protocol, which SNMPv2 failed to provide. This resulted in the IETF Network Working Group resuming work in 1994, and in a complete rewrite of the SNMPv2 RFCs in 1996. Unfortunately, this working group could not agree on which technology to choose for the administrative and security frameworks [41], so the concept of SNMPv2 party was left out, and the simple community-based SNMPv1 security scheme was chosen as a last-minute compromise [41]. This new release of SNMPv2 is called SNMPv2c. A replacement for the Manager-to-Manager MIB, needed since the 1993 RFC relied on the concept of party, still remained to be published as this paper was written. Therefore, as it currently stands, SNMPv2c only supports a centralized model, and offers slightly improved semantics compared to SNMPv1. Nevertheless, since it was designed to enable distributed network management, we classify both the now obsolete SNMPv2 and the new SNMPv2c in the weakly distributed management paradigms.

3.3.3 CMIP

The Common Management Information Protocol (CMIP) [10] was developed in response to the need to monitor and control the health of OSI communications entities. Its first stable release dates back to 1991. In 1992, it was integrated by ITU-T in telecommunication standards (see section 3.3.4). CMIP is expressed as a set of operations which communicate the service parameters (management information) ; these operations correspond to the Common Management Information Service Element (CMISE) services [9]. The structure and semantics of the management information, modeled as managed objects, comply with the GDMO (Guidelines for the Definition of Managed Objects) [12] semi-formal notation. The information model has a strong influence on CMISE.

CMISE provides a generic set of management services aimed at supporting all management exchanges. These are [56] : M_GET to retrieve the attribute values of one or more managed objects (MOs) ; M_SET to request the modification of the attributes values of one or more MOs ; M_ACTION to perform an action on MOs ; M_CREATE to request creation of a new MO ; M_DELETE to delete MOs ; M_EVENT_REPORT to report the occurrence of an event to a manager ; and M_CANCEL_GET to cancel a previously requested M_GET service for which the response has not been received yet. Generally, changes (e.g. lock, unlock, set_attribute) are performed on the attributes of an object using M_SET. The M_ACTION primitive, which could in principle provide another means of flexibility, is not often implemented. Scoping and filtering are two extra features of CMISE that add a vital dimension of flexibility. *Scoping* is employed to enable multiple objects to be selected for performing a management operation (e.g. M_GET), while *filtering* is used to selectively choose objects from all scoped objects by defining assertions about attribute values.

CMIS/CMIP can support either a centralized model, with a NMS concentrating all the management processing, or a hierarchical model, with a vertical distribution of NMSs to cope typically with geographical dispersion. Moreover, CMIS allows cooperation between network operators, or between network operators and service providers. Such a cooperation requires an X interface, which is nothing else than CMIS/CMIP augmented with security features.

CMIS/CMIP has an important role to play in the end-to-end management of emerging broadband communications services and applications involving interoperability between customers, third party service providers and public network operators. Although powerful, the CMIS service is often considered complex. This is why most TMN platforms support TMN object-oriented APIs operating at a higher level than CMIS [42]. CMIS/CMIP is typically designed to run over a fully implemented OSI protocol stack, which leads to vast amounts of overhead. Attempts to implement it over other protocol stacks such as TCP/IP have been less successful.

3.3.4 TMN and TINA

The telecommunications market has been shaken by the advent of new multimedia services, which resulted in the arrival of new business partners : between a network operator and an end-user suddenly appeared service providers, service brokers, content providers, etc. This rendered network management more complex, and substantially increased the amount of management data to move about, in-band or out-of-band. Moreover, telecommunication network infrastructures are increasingly large, complex and sophisticated, and new telecommunication services require more flexibility for their management. The above considerations have led to the definition of an object-oriented management framework, the Telecommunications Management Network (TMN) [32]. The management functionality is vertically distributed over a number of components, organized in a hierarchy of management layers. There are different object models at different layers of the hierarchy. Information flows across the hierarchy in both directions. Cascading is achieved through an ordered sequence of management tasks, in the form of CMIP operations performed on managed objects at various management layers.

In 1993, the TINA Consortium defined the Telecommunications Information Networking Architecture (TINA) [4]. It is based on a Distributed Processing Environment (DPE) and on service delivery technologies that enable the rapid and flexible introduction of new telecommunications services, and integrate the management of both these services and the underlying network infrastructure. Management in TINA follows the ITU-T and OSI management concepts. Protocols such as CMIP can be considered the "assembly language" in TINA. The TINA DPE, based on CORBA and ODL (Object Definition Language), offers a programming language level of abstraction. Inherently, the TINA management concepts and principles lead to the paradigm of distributed object-oriented management.

3.4 Strongly distributed hierarchical models

Although weakly distributed hierarchical models address several shortcomings of the centralized models, they are not the panacea. They often prove to be too limited in practice : they do not cope well with mobile computing and mobile telecommunications ; they only partially address the need for scalability ; and they lack flexibility and robustness, two features that network managers, learning from experience, have now come to demand. To address this, a new breed of technologies emerged, based on strongly distributed hierarchical models. The full potential of distribution was first demonstrated in network management by Goldszmidt with his Management by Delegation (MbD) framework [17, 18], which set a milestone in this research field. MbD triggered a lot of research work, and its success was leveraged by three other promising technologies which emerged at about the same time : mobile code languages, initially devised for large-scale distributed systems ; OMG CORBA [54], devised by the object-oriented community to address the interoperability of distributed objects ; and Intelligent Agents (IAgs) [66], coming from the Distributed Artificial Intelligence (DAI) community. We will come back on the latter in section 3.5, since IAgs are based on a different organizational model.

3.4.1 Mobile code

Before it entered the network management arena, mobile code was initially conceived for building distributed applications. To the public at large, it was first demonstrated on a large scale with Netscape applets, which appeared on the WWW in 1995. But Stamos and Gifford [60] defined the Remote Evaluation scheme as early as 1990 ; and in 1991, Goldszmidt laid the bricks of the MbD framework [68], which was fully specified in 1995 [17]. In 1996, along the same line, Tennenhouse [62] proposed a new paradigm : active networks. The Script MIB [36] and active networks can be considered as a generalization of the RMON MIB, i.e. a way to do with any MIB what could only be done with the RMON MIB : move the programs into the MIB itself.

3.4.1.1 Distributed applications

Carzaniga et al. [6] made a good review of mobile code paradigms used by distributed applications. Interest in code mobility, according to them, was raised by the recent advent of a new family of programming languages known as Mobile Code Languages (MCLs), and the marketing impetus given by the industry. Java, promoted by Sun, Netscape and the success of the WWW, implements a simple form of weak mobility, whereas Telescript, by General Magics, is a sophisticated incarnation of strong MCLs. The authors define *strong mobility* as the ability of an MCL to allow an execution unit (i.e. a Unix process or a thread) to move both its code and its execution state to a different host : the execution is suspended, transferred to the destination host, and resumed there. Examples of strong MCLs are Telescript, Tycoon, Agent Tcl and Emerald. *Weak mobility*, on the other hand, is the ability of an MCL to allow an execution unit on a host to bind dynamically code coming from another host. Examples of weak MCLs are Java, MOLE, TACOMA, M0, Facile, Obliq and Safe-Tcl. Details about these MCLs can be found in [14]. These languages are no longer limited to the distributed applications world, and people like Baldi [2] or Magedanz [37] are now trying to use them to manage networks.

By analyzing existing MCLs, Carzaniga et al. classified all distributed applications based on mobile code in 3 categories :

- *Remote Evaluation* (REV) : when a client invokes a service on a server, it does not only send the name of the service and the input parameters : it also sends the code along. So the client owns the code needed to perform the service, while the server owns the resources and provides an environment to execute the code sent by the client.
- *Code on Demand* (COD) : a client, when it has to perform a given task, contacts a code server, downloads the code needed from that server, links it in on the fly and executes it. So the client owns the resources and the server owns the code.
- *Mobile Agent* (MA) : an MA is an execution unit able to migrate autonomously to another host and resume execution seamlessly. Conceptually, an MA can migrate its whole virtual machine from host to host : it owns the code, not the resources.

The REV paradigm, inspired by the REV system [60], can be seen as an extension of the Unix command `rsh`. The COD paradigm, conceptually, looks very much like Video on Demand. As for the MA paradigm, the choice of its name is unfortunate, but alas reflects a clash in the terminologies used by the distributed applications and DAI communities. This clash has confused a number of people ([34] to quote just one), who liken the concepts of mobile code, mobile agent and intelligent agent. In DAI, a mobile agent is a full-blown intelligent agent, as we define it in section 3.5, with an extra property : mobility. In this sense, there is much more to a mobile agent than just a mobile program and a mobile state. In network management, we propose to give to the expression ‘mobile agent’ the meaning it has in DAI (see our analysis in section 8).

3.4.1.2 MbD

Management by Delegation was designed to address perceived shortcomings in the centralized model. It can be summed up in Goldszmidt's motto : "delegation can be used to move management functions to the data rather than move data to these functions" [18]. To the micro-management syndrome, MbD answered with large-scale distribution. To rigid servers, i.e. servers offering services defined once and for all at design time, it brought dynamic extensibility (*flexibility* as Goldszmidt puts it), i.e. the ability to dynamically extend services by remote applications. The delegation process is fairly simple : the client sends a program, the *delegated agent*, to the server, the *elastic server*, using the Remote Delegation Protocol (RDP) ; this delegated agent is dynamically linked in by the elastic server ; then its execution by the server is either immediate, or delayed and controlled via a scheduling system. This is made possible by a multi-threaded run-time environment providing a "software backplane where delegated programs are loaded and execute as threads in a shared address space" [18]. Processes running in this environment are known as *elastic processes*. An elastic process is "an incarnation of a program that can be modified, extended and/or contracted during its execution" [18]. There is no fixed format for delegated agents : they may be scripts, binary programs or anything.

RDP allows code to be transferred from a client to an elastic server, and its execution to be remotely controlled. If we try to integrate this scheme into Carzaniga's classification, we see that MbD follows the REV paradigm. But unlike the REV system [56], MbD offers remote control capabilities. The execution of the delegated agent by the server is not necessarily synchronous with the invocation by the client : the program can be downloaded in advance, and its execution controlled remotely (scheduled, suspended, resumed, cancelled...). This feature renders MbD-capable network devices more robust than traditional models ; if contact is lost between the agent and the manager (e.g. due to a network link going down), the agent has local controls (programs already downloaded) which enable it to take corrective action in case of emergency.

The novelty of this work stems on the simple, yet powerful idea that with the constant increase in processing power in each and every network device, network management no longer has to be restricted to a limited set of powerful management stations : all network devices can get involved, and become active in the network management process. For the first time with MbD, the full potential of large-scale distributed network management was demonstrated : from dumb data collectors, network devices were suddenly promoted to the rank of managing entities. In the course of 1996, two working groups were created, one by IETF and another by ISO [52], in order to integrate MbD in their respective frameworks. So far, this has resulted in an Internet draft defining the Script MIB [36].

With hindsight, the terminology used by MbD seems a bit confusing. First, the meaning of the word *agent* is different here from what the network management and the DAI communities are used to. *Mobile program* or *mobile code* would probably have been more appropriate than *delegated agent*. Second, there is much more to the concept of delegation than Goldszmidt's MbD framework, which is just one example of delegation, and does not encompass all the ways of delegating a management task (see section 5). But let us remember that 6 years ago, when the MbD framework was devised, the concepts of mobile code and IAgs were less mature than today : hindsight analysis is an easy art...

3.4.1.3 Active networks

An active network is a network where nodes can perform computations on, and modify, the packet contents [61]. Active networks are based on the mobile code paradigm. Two approaches to active network technology are possible. The evolutionary path called the *programmable switch approach* keeps the existing packet format and provides a mechanism for the downloading of programs to dynamically programmable nodes [57, 69]. The revolutionary path, also known as the *capsule approach*, considers packets as miniature programs that are encapsulated in transmission frames and executed at each node along their path [62]. Tennenhouse's approach in active networks is similar to Morgenstern's approach in active databases [43] : in active networks, the code is moved into the network device MIB or run-time environment, and controlled either remotely or locally ; in active databases, programs are moved into the database, and rely upon internal or external triggers to get executed.

3.4.2 Distributed objects

The Object Management Group (OMG), faced with the issue of interoperability in the object-oriented world, addressed it by standardizing the Common Object Request Broker Architecture (CORBA). If early versions showed a number of problems, version 2.0 [46], released in 1995, fixed many interoperability issues (IIOP), defined a dynamic interface on the server side, and added many CORBA services. CORBA 2.0 has now received a wide acceptance, and is gradually becoming the *de facto* standard for objects distribution. Since OSI is object-oriented and SNMP entities map easily onto objects, it took little time for researchers to start integrating CORBA with existing network management environments.

The Joint Inter-Domain Management (XoJIDM) group, jointly sponsored by X/Open and the Network Management Forum (NM Forum), was created precisely for that : to provide tools that enable interworking between management systems based on CMIP, SNMP and CORBA. The SNMP/CMIP interoperability has been addressed by the ISO-Internet Management Coexistence (IIMC) group of the NMForum with the translation between the SNMP and CMIP services, protocols and information [47]. Thus CMIP/CORBA and SNMP/CORBA [38] interworking is tackled by XoJIDM, whose approach consists in specification translation and interaction translation. The specification translation covers the process by which information specifications are converted. Algorithms are defined for the mapping between GDMO/ASN.1 and CORBA IDL [39] and between SNMP MIBs and CORBA IDL [40]. Interaction translation consists in either the mapping of CMIP PDUs into one or more requests or replies on CORBA IDL interfaces, or the translation between SNMP PDUs and CORBA IDL requests/replies. The XoJIDM mappings allows CORBA programmers to write OSI or SNMP managers and agents without any knowldege of GDMO, ASN.1 and CMIP, and conversely GDMO, CMIS or SNMP programmers to access IDL-based resources, services or applications without knowing IDL.

3.4.3 Web-based network management

To advocate Web-based management, Wellens and Auerbach [64] denounce two myths in IP networks management : the myth of the collapsing network and the myth of the dumb agent. First, they assert that “nearly 100% of network management occurs when networks are not failing” [64, p. 2]. And when networks do fail, i.e. when network management comes down to troubleshooting, “SNMP is, at best, a tertiary level tool with value rather below that of ‘ping’, ‘traceroute’, ‘nslookup’ and ‘mtrace’” [64, p. 2]. So the motivation for using a UDP-based protocol like SNMP rather than a TCP-based protocol like HTTP is weak. Second, they argue, like Goldszmidt, that “today’s network devices are capable of managing themselves, if given the opportunity” [64, p. 2], except for a certain class of bottom-of-the-range, price-sensitive devices.

When they investigated network management platforms on the market, they realized that besides the support for a few generic MIBs such as MIB II, most platforms are just collections of add-ons, offering nice device-specific GUIs as the result of peer-to-peer agreements between vendors (which implies that some platforms lack some device-specific GUIs). So they argue that “those add-ons could be just as easily created by having a device export highly device-specific Web pages with controls and user interface paradigms” [64, p. 3]. This argument leads them to their first vision of Web-based management, called the *embedded management application*. Inside their equipment, vendors sell device-specific, self-contained Web servers providing all management functions they deem useful ; and to manage this equipment, rather than an expensive network management platform which may or may not support it, network managers just need a WWW browser. If this solution can seem attractive at first sight, it actually has a large number of shortcomings. The main ones are that it does not automate management (a human being need stare at the browser), and there is no possible site customization since, so to say, ‘vendors think for you’. In practice, such a scheme could only work if it was integrated with an existing network management platform : only the vendor-specific add-ons would then be managed via a WWW interface. And this is indeed the route several platform vendors are taking.

The second vision they give is to use HTTP instead of SNMP, especially HTTP 1.1 [31] which supports long-lived TCP connections. MIB data can then be encoded in a new MIME type, or embedded in HTML structured documents. Network management security can rely on the WWW security, which a lot of people are currently working on. They acknowledge that this scheme would be inefficient with the short-lived TCP connections that

HTTP 1.0 [30] currently offers. But with HTTP 1.1, which is gradually replacing HTTP 1.0 throughout the WWW, "as MIB retrieval size increases, TCP becomes more efficient than UDP-based SNMP" [64, p. 4].

As far as the industry is concerned, two consortia are trying to come up with real products for Web-based management. One, promoted by Microsoft, Intel, Compaq and others, is based on the Web-Based Enterprise Management (WBEM) framework. It relies on a new object model, the HyperMedia Management Schema (HMMS), a new protocol, the HyperMedia Management Protocol, and the Microsoft-specific HyperMedia Object Manager (HMOM), based on OLE/COM. The second one is more open, and supported by Javasoft (Sun), Cisco, Bay Networks, 3Com and many other network equipment vendors. It is based on the Java Management API (JMAPI), and is a mix of object-orientation, mobile code and Web GUIs. It relies on a managed object framework based on the Java Remote Method Invocation (RMI) scheme.

3.5 Cooperative models

The cooperative models that are coming to be used more often in network management come from the DAI community. DAI is traditionally concerned with two kinds of issues : Distributed Problem Solving (DPS) and Multi-Agent Systems (MAS). DNM is primarily interested in MAS, i.e. large groups (societies) of IAgs. If IAgs became a major research topic in artificial intelligence (AI) in the late 1980's, large-scale research on the use of IAgs in DNM only started in 1995. Since this research field is so recent and so hyped, the terminology has not converged yet, which is a major problem. Wooldridge's attempt to address this issue is interesting : he defines a core of properties shared by all IAgs, but also allows any other property as application-specific. This approach is less limitative than others, and is getting increasing acceptance, especially outside the DAI community. We therefore propose to adopt it in DNM. The 4 properties that any IAg should offer are [66] :

- *autonomy* : an IAg operates without the direct intervention of humans, and has some kind of control over its actions and internal state
- *social ability* : IAgs cooperate with other IAgs (and possibly humans) to achieve their goals, via some kind of agent communication language
- *reactivity* : an IAg perceives its environment and responds in a timely fashion to changes that occur in it
- *pro-activeness* : an IAg is able to take the initiative to achieve its goals, as opposed to solely react to external events.

Other optional application-specific properties include mobility, veracity (IAgs do not knowingly communicate false information), and rationality (IAgs act so as to achieve their goals) [19]. Pro-activeness is an important and very discriminating property : if most IAg implementations are indeed reactive, few are pro-active outside the AI community, and consequently only few can truly pretend to the designation of IAg.

When applied to network management, IAgs offer a level of automation that other DNM paradigms cannot meet. They also enable to tackle the growing complexity of networks, particularly multimedia networks. Many researchers are now trying to use this technology to manage IP or telecommunications networks : Mountzia with flexible agents [44], Post with the manager/agency paradigm [48], Somers with the HYBRID system [58], Zhang who proposes to extend TMN [70], etc...

The definition of agency given above is known as the *weak agency*. Although it generally suits the expectations of people outside the AI community, it is too restrictive for AI researchers, used to describing IAgs with human-like mental states such as knowledge, belief, intention and obligation [53]. These properties have been formalized in theories of agency, and characterize the *strong agency*. One of the most influential contributions was Cohen and Levesque's pioneering work known as the *theory of intention* [13], based on beliefs and goals, which proved useful to analyze conflicts and cooperation in MAS. Possibly the most sophisticated to date is Rao and Georgeff's belief-desire-intention (BDI) model : to simplify things, beliefs represent the information an IAg has about its environment, desires are the tasks allocated to it, and intentions represent desires that it has committed to achieving [65]. To the best of our knowledge, nobody so far has tried to use the BDI model, or indeed any implementation of strong agency, to manage networks.

The area of agent languages is very active as well, especially since Shoham [53] proposed a new programming paradigm, Agent-Oriented Programming (AOP). In AOP, IAgs are directly programed "in terms of the mentalistic, intentional notions that agent theorists have developed to represent the properties of [intelligent] agents" [66, p. 18]. The rationale for defining mentalistic language primitives is to offer "a familiar, non-technical way to talk about complex systems" [65, p. 2]. Many languages have emerged, either based on AOP (AGENT0, PLACA), derived from it (AgentSpeak, DAISY), or based on other paradigms (KQML, Concurrent MetateM... [66]). KQML [16] currently seems to be the most popular cooperation protocol for knowledge sharing used in DNM. People also use mobile code languages such as Telescript.

4 Organization structures in enterprise management

Mullins [45] distinguishes 8 ways of dividing work in an enterprise :

- by function (one department per function : production, R&D, marketing, finance, sales... all staff share a common expertise within a department)
- by product (autonomous units, all functions are present in each unit)
- by location (multi-site companies, subsidiaries abroad)
- by nature of the work to be performed (e.g. by security clearance level)
- by common time scales (e.g. shift work vs office hours work)
- by common processes (e.g. share production facility in manufacturing industry)
- by the staff employed (e.g. surgeons, doctors and nurses in a hospital)
- by type of customer or people to be served (e.g. home vs export sales).

He also argues that delegation can take place at two levels : enterprise or individual. At the enterprise level, it is depicted in the organization chart (or at least should be !), and relies on federal or functional decentralization. He defines *federal decentralization* as "the establishment of autonomous units operating in their own market with self-control and with the main responsibility of contributing profit to the parent body" [45, p. 276]. As for *functional decentralization*, it is "based on individual processes or products" [op. cit.]. At the individual level, delegation is "the process of entrusting authority and responsibility to others" [op. cit.] for a specific task.

Weinshall [63] identifies 3 basic managerial structures : entrepreneurial, functional and decentralized. The *entrepreneurial structure* is typical of an organization recently created, fairly small and growing fast. It must be managed in an informal and centralized fashion in order to survive. Everything is centered on one person, the entrepreneur who created the company. When organizations grow beyond a certain size, they must go through a major transformation : a whole set of rules by which the work is managed and carried out need be formalized, "in order to cope with the growing quantities of product and services, their variety, and the complexity of the organization" [63, p. 55]. This is called the *functional structure*. The chief executive directly controls the various functional heads, such as the production manager, the marketing manager, the sales manager... The formalized and centralized nature of the functional structure must, at some point, give place to the *decentralized structure* : as a result of expansion, the number of managers grows far beyond the number that can efficiently report to a single person. At this stage, the organization must slow down its growth, and introduce a new formal and decentralized structure, organized by product/service line or by geographical area.

These 3 structures are uniform, in the sense that "all subordinates of the chief executive are structured either in an entrepreneurial, or a functional, or a product line or area structure" [63, p.188]. Beyond a certain size, this uniformity cannot be maintained : the decentralized structure need change into a *multistructure*, i.e. a federated managerial structure where "different building blocks may be combined into different kinds of structures" [63, p. 189]. The Japanese, according to Weinshall, were the first to operate their large organizations in multistructures, in a type of organization known as *zaibatsu*. The multistructure is inherently flexible, in that it enables changes in the composition of the federated basic structures. This natural evolution as enterprises grow from the entrepreneurial structure to the multistructure is depicted in Figure 2 [63, p. 56]. The time slots on the abscissa depend on the sector of activity of the company (5 to 20 years). Similarly, the values in ordinate only give an order of magnitude, and ought to be correlated to the sector of activity.

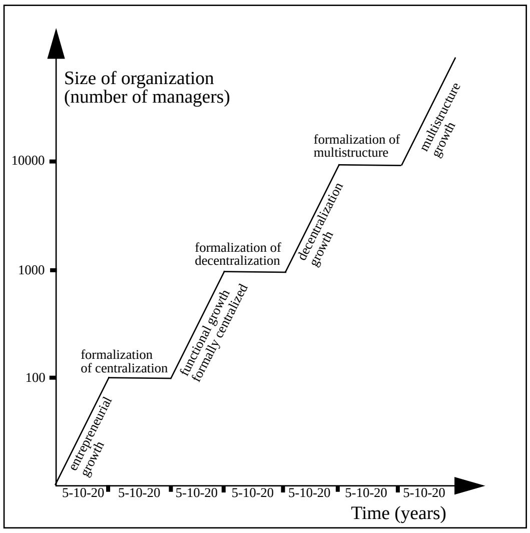

Figure 2 : The effect of size on the enterprise structure (Weinshall)

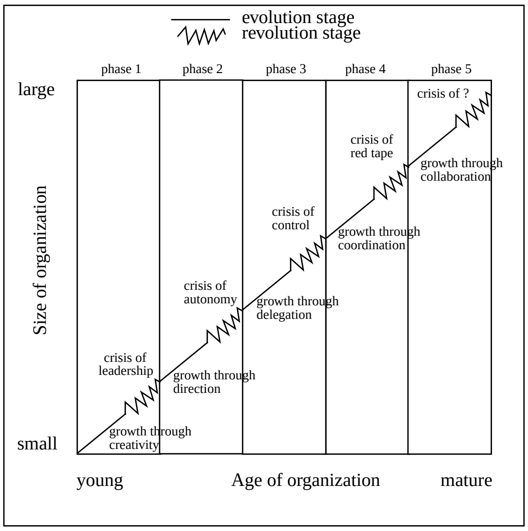

Figure 3 : The five phases of organization development (Greiner)

To conclude with enterprise management, let us take a look at the *evolutions and revolutions* trend depicted in Figure 3 [20, p. 39], that Greiner identified way back in 1972. With hindsight, it is amazing to see how this cycle of evolutions and revolutions, devised for enterprise management, suits network management well. If the first three phases look similar to those identified by Weinshall, the last two, coordination and collaboration, are incredibly visionary, and predicted 25 years ago what intelligent agents are now striving to achieve in network management ! It is also interesting to notice that the last crisis is left as unknown. Would the combination of hierarchical and cooperative models be the ultimate solution ? Or was the idea of collaboration so new in the enterprise world that Greiner, lacking hard evidence, did not want to speculate on what could go wrong then ?

5 Division of work : delegation granularity

5.1 Comparison with enterprise management

The topology of an enterprise computer network tends to be modeled after its organization chart. The main reason for this is that the persons accountable for the smooth operation of these networks belong to this chart, and it makes their life a lot easier if different managers have a hold on different computer and network equipments. Besides, such a topology is also often justified in terms of budget, and sometimes it also makes technical sense.

As a consequence, network management and enterprise management are not orthogonal, and lessons learned from enterprise management may benefit network management. What do Mullins, Weinshall and Greiner tell us, and how does it translate in network management terms ? First, all underlying models are hierarchical, except for the last two phases described by Greiner, characterized by a mix of hierarchical and cooperative models. Likewise, most DNM models are hierarchical, and cooperative models have only just started to appear in hybrid structures similar to Weinshall's multistructures. Second, delegation schemes should evolve as enterprises grow in size, otherwise they become inefficient. Similarly, network management should rely on different DNM paradigms as networks grow in size and complexity. In this respect, the recent explosion of new distributed paradigms seems justified, since networks have grown by at least an order of magnitude in terms of size and complexity since SNMP and CMIP were devised.

Third, if there are many ways of dividing up enterprise organization structures, depending on the granularity of the analysis, most authors agree with Weinshall that they all coalesce in three broad types : by function, by product or service, and by geographical area. There are strong similarities between enterprise and network management : the division of management domains by geographical area, e.g., makes sense in both worlds. But there are clear discrepancies too, since the basic entities are human beings in one case, and machines or programs in the other. The division by function only makes sense in the enterprise world : it takes long for a person to become an expert in accounting or electrical engineering, and an accountant cannot be turned into an engineer overnight ; conversely, a computer can be equipped with new competencies in a matter of minutes or hours, by simply transferring a few programs and testing them on their new host. In section 5.2, we show that Mullins' federal decentralization and Weinshall's decentralized structure map onto our *delegation by domain* scheme in network management, whereas Mullins' functional decentralization and Weinshall's functional structure map onto our *delegation by task* scheme.

The fourth and most important thing we can learn from enterprise management is this common evolution trend, of which Greiner and Weinshall give two different, but compatible, flavors. There is a natural evolution of companies from centralized structures to decentralized ones, and from lightly decentralized structures (organized by function) to more decentralized ones, with independent units (organized by product/service or by geographical area), to even more decentralized ones (based either on federation or on cooperation). In network management terms, these 4 stages nicely map onto the different types presented in section 3 in our simple typology. The evolution which occurred in enterprise management over the 20th century suggests that the same evolution may take place in network management in the next decade or so, even though the time scales are different.

5.2 Domains and tasks

The division of work, as we have seen, is fairly similar in network management and in enterprise management. Bearing their idiosyncrasies in mind, we see that all the possible enterprise organization structures (by geographical area, by function, by product, by customer...), when applied to the network management world, can be grouped in two families : delegation by domain and delegation by task.

Delegation by domain relies on static tasks : the manager at level (N) assumes that the manager at level (N-1) knows all of the management tasks to be completed within its domain. In today's networks, delegation by domain typically translates into SNMPv2 and CMIP to manage geographically dispersed enterprises. For instance, if the headquarters of a multinational company are located in Sydney, Australia, it cannot afford to manage its large subsidiaries in the USA, Japan or Europe over expensive and relatively slow transcontinental WAN links. Let us consider its European subsidiary, located in Geneva, Switzerland. The NMS in Sydney delegates the whole management of the Swiss subsidiary to the NMS located in Geneva, and expects it not to report a local printer going down, but to report that the number of errors per minute has exceeded a critical threshold on the Switzerland-Australia link. The point here is that the Australian NMS does not tell the Swiss NMS what to report : it expects it to be able to work it out by itself. In practice, this translates into a human being, the local network manager, hard-coding in the Swiss NMS what to report back to Sydney, and how to manage the rest of the local network. There is no mechanism for the Australian NMS to alter the way the Swiss NMS manages its domain : it is a white-card type of delegation, where the Geneva-based network manager has total control over its own local network. This scheme requires a network manager be based in each and every large subsidiary : network management is not automated, and there is no way for the Australian network manager to enforce a management policy over all its subsidiaries. Clearly, these are serious limitations.

Delegation by task, on the contrary, offers a finer vision at level (N) of the management processing occurring at level (N-1). A task is a network management unit. It can be viewed at different scales : we show in section 5.3 that it makes sense to distinguish micro-tasks from macro-tasks. Before we go further on this, it is important to realize that an important property derives from the fact that the manager at level (N) can see the different tasks at level (N-1), as well as other tasks of its peers at level (N) : tasks no longer need be static, hard-coded in every NMS. They may as well be dynamic, and modified on the fly. This idea was first applied to network management when the MbD framework was devised : Goldszmidt departed from the well-established notion of static tasks underlying the centralized model, to introduce the notion of dynamic tasks, transferable from the NMS to its subordinate agents. This model was soon generalized by others to transfer dynamic tasks from a manager at level (N) to a manager at level (N-1).

5.3 Protocol primitives, micro-tasks and macro-tasks

A manager at level (N) has several ways of driving a subordinate at level (N-1). With traditional approaches such as SNMP, the basic unit in the manager-agent dialog is the protocol primitive : the manager issues a series of Get and Set requests to the agent. The data manipulated are MIB variables, which are statically defined when the MIB is designed. Recent approaches, conversely, attempt to avoid the micro-management syndrome described by Goldszmidt [18], while still letting the manager at level (N) in control of what subordinates at level (N-1) do. This is achieved by splitting the whole management application into many different units, or *tasks*, and by distributing these tasks over a large number of NMSs and agents. The underlying mechanism of this distribution is independent from the tasks being delegated : it can rely on program transfer, message passing, RPCs, whatever... What is relevant for the management application is the granularity of the delegation, i.e. the way the work is divided. Clearly, there is a whole range of tasks complexity, ranging from the mere addition of two MIB variables to the whole management of an ATM switch. We propose to distinguish only two levels in our enhanced typology : micro-tasks and macro-tasks.

A *micro-task* (μ-task) just performs preprocessing on static MIB variables, typically to make statistics. It is the simplest way of managing site-specific, customized variables. There is no value in these data *per se*, they still need be aggregated by the NMS one level up. If contact with the NMS is lost, statistics are still gathered, but there is no way for the subordinate to take corrective action on its own. In the case of a *macro-task* (M-task), the entire control over an entity is delegated. A macro-task can automatically reset a network device, or build an entire daily report, etc. If contact is lost with the NMS one level up, corrective actions may be automatically undertaken.

Delegation by domain with dynamic tasks is considered in this paper as a particular case of delegation by M-task : the domain then describes the scope of the task, and the tasks are explicitly defined (as opposed to the delegation by geographical area, where tasks are implicitly defined).

6 What do network managers expect from DNM ?

Goldszmidt, as he was working on his MbD framework [18], identified 4 areas where MbD proved superior to centralized models : distribution, scalability, flexibility and robustness. What he means by *distribution* is twofold ; first, since the processing (CPU cycles) is distributed over the NMS and all agents, NMSs no longer need be very powerful and expensive machines ; second, the micro-management phenomenon is avoided, and bandwidth is conserved by decreasing the amount of management data traffic (this is particularly crucial in the IP world, where this data is moved about in-band). *Scalability* is the ability to manage small, medium-size or large networks indiscriminately. *Flexibility* is the ability to dynamically update management programs, by downloading them from the NMS to the remote agents. This feature is often called *dynamic code binding*. *Robustness*, finally, is the ability for an agent to take independent corrective action, should a problem arise, when contact is lost between the NMS and the agent (e.g. due to a network outage).

Clearly, these 4 criteria may apply to any form of DNM, not just MbD : distribution and scalability are addressed by all DNM paradigms ; flexibility is offered by all paradigms providing for dynamic code binding; and robustness, depending on the design, may optionally be implemented. So we keep them for our classification. But in order to assess the relative performance of different DNM paradigms, we need to add several other criteria to this list, particularly those mobile code technologies are not so good at.

In our view, the two most important features that network managers are after are semantic richness and automated management. The *semantic richness* measures how easy it is for a network manager to specify a task to be executed by an NMS or an agent. SNMP and CMIP protocol primitives offer fairly poor semantics, constraining network managers to program (typically Perl scripts in the IP world) at a low-level of abstraction (e.g. SNMP Get and Set primitives) when they actually think at a high level. CORBA-based distributed objects, on the contrary, offer rich semantics, with a direct mapping with OSI and SNMP managed objects. Agent languages such as KQML can also offer very rich semantics, and there is still a lot of research to be done in this area to take full advantage of it.

The automation of management requires the distributed network management application be coded so as to relieve network managers and operators as much as possible from the burden of constantly monitoring visually a GUI and fixing problems manually as they occur. If µ-tasks poorly automate network management, M-tasks are very good at it, since they enable remote agents to take corrective action independently from the NMS. The larger and the more complex the network, the more automated the management application should be.

Interoperability is the ability to easily support any kind of platform, be it a Unix workstation, a PC running Windows, or a Macintosh. This issue is critical for mobile code, where the price to pay for dynamic code binding may be the loss of interoperability : in this case, the network manager needs to bother about the platforms as much as the management application itself. In the same line, DNM paradigms may attempt to fulfill a long-standing expectation : to ally at last the IP and ISO communities, i.e. more or less the Internet/ Intranet world and the telecommunications world, by paving the path to a unified network management solution. We call this *convergence*, a companion criterion to interoperability.

Let us come back to the term *flexibility*, which is often abused since it is so general. In object-oriented terms, the flexibility referred to by Goldszmidt is the ability to dynamically modify the behavior of an object, i.e. its methods. In this respect, this dynamic code binding can also be called the *dynamic behavior definition*. Similarly, one can define the *dynamic state definition* as the ability to modify dynamically the structure of a MIB, i.e. the object attributes. SNMP MIBs are not flexible, since they are cast in iron at design time, so it is the responsibility of DNM paradigms to cater for it, e.g. by coupling SNMP with CORBA (see section 8).

The *delegation granularity* was presented at length in section 5, and *security*, in its usual sense, deals with authentication, authorization, access control and privacy policies. Finally, *mobility* is the ability to cope with mobile network devices, such as portable PCs or mobile phones, and thus mobile agents and mobile managers. If managers are generally expected to be static (although they do not have to), more and more agents will be mobile in the future. This is a research area where mobile code proponents have been very active.

In summary, the 12 criteria that we have identified to assess and compare different DNM paradigms are:

- scalability
- distribution of CPU cycles
- conservation of bandwidth
- robustness
- dynamic state definition (dynamic MIB definition)
- dynamic behavior definition (dynamic code binding)
- semantic richness
- automation
- interoperability and convergence
- delegation granularity
- security
- mobility

It is not possible to take into account all of these criteria for our enhanced typology, if we want to keep it readable. We have therefore retained only two, semantic richness and delegation granularity, which we believe are the most discriminating criteria among DNM paradigms. In next section, when multiple DNM technologies compete to offer similar levels of semantic richness and delegation granularity, we also weight them with their ability to provide for dynamic state and behavior definitions.

7 Typology of network management paradigms

In section 3, we presented a simple typology dividing up network management paradigms into 4 broad types, according to their underlying organizational model, and further split up each category according to the kind of technology being used. The limitation of this approach is that to solve the problem, network management, we have just listed the tools available to date. A better approach, in our view, is to explore the solution space of this problem, and then investigate the existing tools. This is what we strive to achieve in this section. Based on what we learned from the enterprise management world, and the criteria we identified in the previous section, we propose the following enhanced typology. All solutions are grouped into 7 broad types, characterized by the delegation granularity, the degree of specification and the semantic richness :

- No delegation
- Delegation by domain
- Delegation by µ-task fully specified with low-level semantics
- Delegation by µ-task fully specified with high-level semantics
- Delegation by M-task fully specified with low-level semantics
- Delegation by M-task fully specified with high-level semantics
- Delegation by M-task partially specified with high-level semantics

No delegation

Network devices are dumb data collectors. The management data is retrieved by the central NMS using low-level protocol primitives. This is the traditional centralized model, epitomized by SNMPv1, and heavily criticized by all proponents of the DNM school. It lacks lacks all desired properties listed in section 6 (except interoperability !), and is exposed to the micro-management syndrome.

Delegation by domain

The management of an entire geographical area is delegated by a manager at level (N) to a manager at level (N-1), regardless of the actual tasks to be completed to manage this domain. There is no way for the manager at level (N) to enforce a management policy at level (N-1). Tasks are static, hard-coded in each NMS, and need be defined explicitly by the network manager at each level. The typical example is the weakly distributed hierarchical model, underlying both CMIP and SNMPv2.

Delegation by µ-task fully specified with low-level semantics

This is the simplest task-oriented paradigm. It allows for the delegation of the computation of simple data not directly available from a MIB, and can be considered as preprocessing. Tasks are dynamic, and may be updated at any time. This scheme is more flexible than the previous two, but still offers poor semantics to the manager, who must think in terms of protocol primitives. Since only µ-tasks are delegated, most management ‘intelligence’ is still in the hands of the NMS, e.g. the making of a complete daily report. RMON is a typical example, but the manager is restricted to using RMON statistics, RMON alarms or RMON events. Mobile code generalized this concept by offering dynamic state and behavior definitions : tasks can then use any variable from any MIB.

Delegation by µ-task fully specified with high-level semantics

Distributed objects also generalized the RMON concept by offering dynamic state and behavior definitions, but the semantics used by the manager are at a higher level than the mere protocol primitives used in mobile code. The price for that is that these high-level semantics need be transparently translated into low-level protocol primitives via some kind of gateway : the management application manipulates objects, and the subsequent calls to protocol primitives are encapsulated by these objects. There is no reason why mobile code should not incorporate high-level semantics as well, but current technologies show a preference for manipulating protocol primitives directly.

Delegation by M-task fully specified with low-level semantics

This type of delegation was the initial idea behind MbD, and with it most DNM paradigms : the programs that used to micro-manage agents from the NMS are downloaded to the agents, where they run locally. They can be remotely controlled, so the NMS still has a say. But the semantics of the programs do not change : the network manager still manipulates protocol primitives. Most mobile code paradigms are based on this type of delegation.

Delegation by M-task fully specified with high-level semantics

This is typically the realm of distributed objects. This adds high-level semantics to the advantages of the previous type of delegation : the network manager can now specify high-level tasks with a high-level language, and is no longer contrived to think at a low level of abstraction.

Delegation by M-task partially specified with high-level semantics

This represents the highest abstraction level of all delegation types presented here. It requires IAgs, and capitalizes on their autonomy and their ability to cooperate to solve complex management tasks. Such tasks are not described in a procedural language, specifying step by step how to achieve them (*full specification*). Rather, they are described in terms of goals in a high-level language, (e.g. KQML in the synthetic figure below), what we call a *partial specification*. A mobile IAg may optionally be created to achieve such a task. This type of delegation enables a high degree of management automation.

Semantic richness

Delegation granularity

	none	SNMP/CMIP primitives	CORBA	KQML
none	SNMPv1			
all		SNMPv2, CMIP, TMN, TINA		
µ-task		RMON (static), Web-based management (static), mobile code (dynamic),	distributed objects	
M-task		mobile code	distributed objects	IAg (partial specification)

Figure 4 : Enhanced typology of network management paradigms

8 Analysis of the potentials of strongly distributed paradigms

Comparing the relative performances of all DNM paradigms would be a long and risky exercise : all these technologies are evolving so quickly that any comparative analysis would soon become obsolete. In this last section, we therefore opted to discuss the potentials of the 3 strongly distributed paradigms presented in section 3, based on to the 12 criteria identified in section 6. We then try to derive from this analysis what network management could be in the future.

By their very design, all strongly distributed paradigms address the demand for scalability and distribution of CPU cycles. They also provide for dynamic state and behavior definitions. Conservation of bandwidth does not easily go with management automation : IAgs exchange a lot of data while they cooperate ; but they are also the only ones that can cope with complex tasks, and therefore relieve the manager from having to split these complex tasks into lots of simple tasks. In this respect, weakly and strongly distributed hierarchical models are better at conserving bandwidth, but cooperative models are better at automating management. Robustness is probably better tackled by mobile code paradigms and IAgs than distributed objects. CORBA offers a software bus for clients to contact servers, and is not really meant for servers to run standalone. Security issues go far beyond the scope of this paper. Although security is equally necessary whatever the DNM paradigm, network management applications often leave a lot to be desired in this respect, especially in the IP world. The telecommunications world is probably the one where this concern has been tackled with most success so far.

Interoperability is best addressed by distributed objects, since we assume the same DPE, CORBA, to be available everywhere. It is a problem for some mobile code technologies like MbD, which do not mandate a single scripting language : the network manager, when he writes the management application, has to bear in mind the differences in the run-time environments between the different network devices. For large networks, this can be a major problem. Interoperability between IAgs is still an open question : KQML could become a *de facto* standard, but the competition is fierce.

Mobility is probably best addressed by IAgs : they are independent from the NMS, and can migrate if necessary. Mobile code paradigms, by design, can also cope with mobile network devices. But a way to inform an NMS that a network device has emigrated from its management domain need be devised, since NMSs have to know what agents belong to their domain. Distributed objects are probably less inclined to migrate, but CORBA could deal with it.

The last two criteria, the delegation granularity and the semantic richness, are very discriminating between strongly distributed paradigms. The delegation granularity was already analyzed at length in section 7. The semantic richness offered by IAgs is the highest of all, but is overkill if M-tasks are not complex. The semantic richness offered by distributed objects is sufficient for most management tasks : the management application directly deals with managed objects, and the use of protocol primitives is encapsulated in these objects. This enables network managers to think with high-level abstracts. Mobile code is comparatively poor in this respect, since mobile code languages force network managers to think in low-level terms.

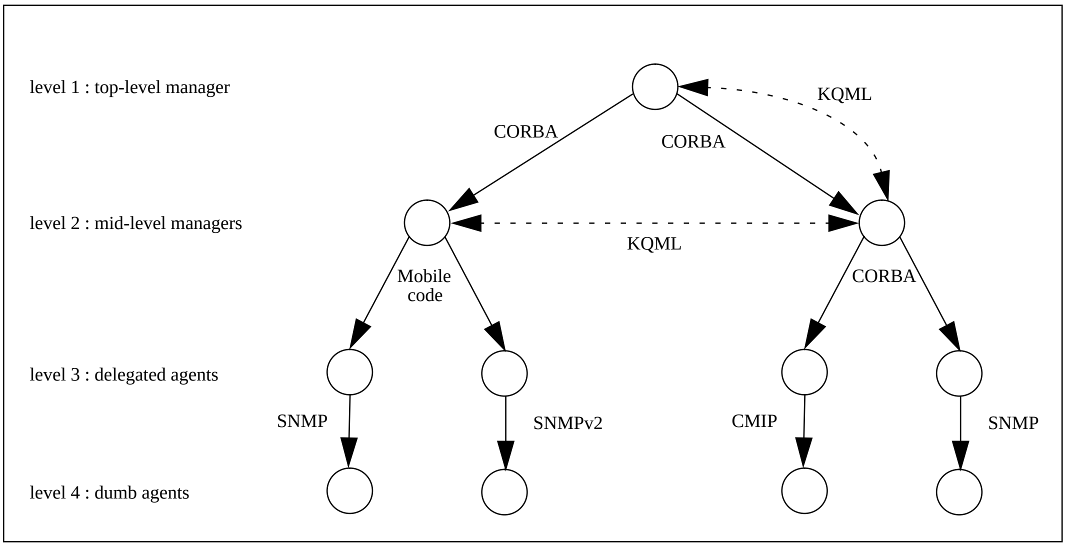
level 1 : top-level manager
level 2 : mid-level managers
level 3 : delegated agents
level 4 : dumb agents
CORBA
CORBA
KQML
KQML
Mobile code
CORBA
SNMP
SNMPv2
CMIP
SNMP

Figure 5 : Distributed network management: the integration model

We have left out one criterion in this review : convergence. This is because no single paradigm is particularly good at everything : there is no clear winner. On one hand, to preserve the huge investments that have already been made in network management, telecommunication networks will be managed with CMIP, TMN and TINA for many years to come, and likewise the Internet and Intranets will be managed with SNMP or its derivatives for many years. On the other hand, network managers come up every day with new demands : they want richer semantics and more automated management, but they also make their networks more and more complex to manage, and keep inventing ever more complex network services (e.g. setting up a multimedia service session at the cheapest price, which requires negotiations with multiple service providers, brokers, etc.). How can we cope with these opposing constraints ? By analogy with Weinshall's multistructures in enterprise management, we predict that the convergence will emerge from the combined use of multiple network management paradigms and multiple network management protocols, as depicted in Figure 5. At the lowest level, network devices are managed with SNMP or CMIP, as usual. One level up, depending on the semantic requirements and on the technology available, CORBA or a mobile code language can be used. At the highest level, we only find languages offering high-level semantics, such as KQML to solve complex management tasks and CORBA to solve normal ones. We call this the *integration model*. The XoJIDM efforts are also going in this direction, except they do not take IAgs into account yet.

We expect mobile code paradigms to be widely adopted and implemented in network management applications very soon. CORBA-based network management should start to spread in a few years from now, as CORBA gets a wider acceptance and CORBA commercial offerings start to implement the whole set of services and facilities set forth in the specification. Some mobile code languages are now object-oriented (most were just scripting languages initially), so mobile code and distributed objects paradigms could soon work together very easily. Eventually, we expect mobile code and distributed objects to converge. In the meantime, IAgs should slowly start to be used, as people understand better how to use them. But we do not expect them to replace other solutions based on hierarchical models : cooperative and hierarchical models will coexist, since they address very different needs.

9 Conclusion

To address the lack of classification of DNM paradigms, which have been evolving at a very fast pace for the past 3 years, we first presented a simple typology dividing up all network management paradigms into different categories, according to their underlying organizational model. We identified 4 broad types : the centralized paradigm, the weakly distributed hierarchical paradigm, the strongly distributed hierarchical paradigm and the cooperative paradigm. We then further broke down each type according to the technology being used.

In order to refine this typology, we reviewed all network management paradigms and enterprise organization structures known to date, to the best of our knowledge. We compared them, and identified some similarities and discrepancies. This enabled us to define different levels of delegation granularity, and to introduce the notions of micro-task and macro-task. Building upon Goldszmidt's work, we identified 12 criteria on which network managers judge all DNM technologies. We selected the most discriminating ones to build our enhanced typology : the delegation granularity (by domain, by micro-task or by macro-task), the degree of specification of the task (fully specified or partially specified), and the richness of the semantics used (protocol primitives, low-level abstracts or high-level abstracts).

Finally, based on the common trend underlying enterprise and network management, we attempted to predict where network management is currently heading. We explained why multiple solutions will likely coexist in the future, taking advantage of the different technologies. We expect mobile code and distributed objects to eventually converge, and hierarchical and cooperative models, which address different needs, to coexist.

In the future, we intend to implement paradigms based on mobile code, distributed objects and cooperative agents, make them work together, and demonstrate that the coupling of hierarchical and cooperative models can indeed address network managers' perennial quest for ever richer semantics and ever more flexibility. We also plan to investigate whether the use of Rao and Georgeff's BDI model can help manage telecommunication services, particularly multimedia services.

Acknowledgements

This research was partially funded by the Swiss National Science Foundation (FNRS) under grant 5003-045311. The authors wish to thank Jean-Pierre Hubaux for his valuable suggestions and comments, and Shawn Koppenhöfer for proofreading this paper.

Bibliography

[1] M. Armstrong. *A Handbook of Personnel Management Practice*. 4th edition, Kogan Page, London, UK, 1991.

[2] M. Baldi, S. Gai and G.P. Picco. “Exploiting Code Mobility in Decentralized and Flexible Network Management”. Accepted for publication in *Proc. 1st Int. Conf. on Mobile Agents (MA’97), Berlin, Germany, April 1997*.

[3] S. Bapat. *Object-Oriented Networks : Models for Architecture, Operations and Management*. Prentice-Hall, Englewood Cliffs, New Jersey, USA, 1994.

[4] W. J. Barr, T. Boyd and Y. Inoue. “The TINA Initiative”. In *IEEE Communications Magazine*, 31(3):70-76, 1993.

[5] R. Boutaba. *Une architecture et une plate-forme distribuée orientée objet pour la gestion intégrée de réseaux et de systèmes* (in French). PhD thesis, Pierre & Marie Curie University, Paris, France, March 1994.

[6] A. Carzaniga, G.P. Picco and G. Vigna. “Designing Distributed Applications with Mobile Code Paradigms”. To appear in *Proc. 19th Int. Conf. on Software Engineering (ICSE’97), Boston, Massachusetts, USA, May 1997*.

[7] CCITT (now ITU-T). *Recommendation X.700. Data Communication Networks - Management Framework for Open Systems Interconnection (OSI) for CCITT Applications*. ITU, Geneva, Switzerland, September 1992.

[8] CCITT (now ITU-T). *Recommendation X.701. Data Communication Networks - Information Technology - Open Systems Interconnection - Systems Management Overview*. ITU, Geneva, Switzerland, January 1992.

[9] CCITT (now ITU-T). *Recommendation X.710. Data Communication Networks : Open Systems Interconnection (OSI) ; Management. Common Management Information Service Definition for CCITT Applications*. ITU, Geneva, Switzerland, March 1991.

[10] CCITT (now ITU-T). *Recommendation X.711. Data Communication Networks - Open Systems Interconnection (OSI) ; Management. Common Management Information Protocol Specification for CCITT Applications*. ITU, Geneva, Switzerland, March 1991.

[11] CCITT (now ITU-T). *Recommendation X.720. Data Communication Networks - Information Technology - Open Systems Interconnection - Structure of Information Management : Management Information Model*. ITU, Geneva, Switzerland, January 1992.

[12] CCITT (now ITU-T). *Recommendation X.722. Data Communication Networks - Information Technology - Open Systems Interconnection - Structure of Information Management : Guidelines for the Definition of Managed Objects*. ITU, Geneva, Switzerland, January 1992.

[13] P.R. Cohen and H.J. Levesque. “Intention is choice with commitment”. In *Artificial Intelligence*, 42:213-261, 1990.

[14] G. Cugola, C. Ghezzi, G.P. Picco and G. Vigna. “Analyzing Mobile Code Languages”. To appear in C. Tschudin and J. Vitek (Eds.), *Mobile Object Systems*. LNCS, Springer-Verlag, Berlin, Germany, 1997.

[15] David Evans. *Supervisory Management : Principles and Practice*. 2nd edition. Cassell Educational Ltd, London, UK, 1986

[16] T. Finin, R. Fritzson, D. McKay and R. McEntire. “KQML as an Agent Communication Language”. In N.R. Adam, B.K. Bhargava and Y. Yesha (Eds.), *Proc. 3rd Int. Conf. on Information and Knowledge Management (CIKM’94), Gaithersburg, Maryland, USA, November 1994*, pp. 456-463. ACM Press, 1994.

[17] G. Goldszmidt. *Distributed Management by Delegation*. PhD thesis, Columbia University, New York, NY, USA, December 1995.

[18] G. Goldszmidt and Y. Yemini. “Distributed Management by Delegation”. In *Proc. 15th Int. Conf. on Distributed Computing Systems (ICDCS’95), Vancouver, Canada, May 1995*. IEEE Press, New York, NY, USA, 1995.

[19] R. Goodwin. *Formalizing properties of agents*. Technical Report CMU-CS-93-159, School of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, USA, 1993.

[20] L.E. Greiner. “Evolution and Revolution as Organizations Grow”. In *Harvard Business Review*, 50(4):37-46, 1972.

[21] IETF Network Working Group. *RFC 1157. A Simple Network Management Protocol (SNMP)*. J. Case, M. Fedor, M. Schoffstall and J. Davin (Eds.), IAB, May 1990.

[22] IETF Network Working Group. *RFC 1213. Management Information Base for Network Management of TCP/IP-based internets : MIB-II*. K. McCloghrie and M.Rose (Eds.), IAB, March 1991.

[23] IETF Network Working Group. *RFC 1271. Remote Network Monitoring Management Information Base*. S. Waldbusser (Ed.), IAB, November 1991.

[24] IETF Network Working Group. *RFC 1445. Administrative Model for version 2 of the Simple Network Management Protocol (SNMPv2)*. J. Galvin and K. McCloghrie (Eds.), IAB, April 1993.

[25] IETF Network Working Group. *RFC 1447. Party MIB for version 2 of the Simple Network Management Protocol (SNMPv2)*. K. McCloghrie and J. Galvin (Eds.), IAB, April 1993.

[26] IETF Network Working Group. *RFC 1450. Management Information Base for version 2 of the Simple Network Management Protocol (SNMPv2)*. J. Case, K. McCloghrie, M. Rose and S. Waldbusser (Eds.), IAB, April 1993.

[27] IETF Network Working Group. *RFC 1451. Manager-to-Manager Management Information Base*. J. Case, K.

McCloghrie, M. Rose and S. Waldbusser (Eds.), IAB, April 1993.

[28] IETF Network Working Group. *RFC 1757. Remote Network Monitoring Management Information Base*. S. Waldbusser (Ed.), IAB, February 1995.

[29] IETF Network Working Group. *RFC 1909. An Administrative Infrastructure for SNMPv2*. K. McCloghrie (Ed.), IAB, February 1996.

[30] IETF Network Working Group. *RFC 1945. Hypertext Transfer Protocol -- HTTP/1.0*. T. Berners-Lee, R. Fielding and H. Frystyk (Eds.), IAB, May 1996.

[31] IETF Network Working Group. *RFC 2068. Hypertext Transfer Protocol -- HTTP/1.1*. R. Fielding, J. Gettys, J. Mogul, H. Frystyk and T. Berners-Lee (Eds.), IAB, January 1997.

[32] ITU-T. *Recommendation M.3010. Principles for a Telecommunications management network.* ITU, Geneva, Switzerland, May 1996.

[33] F.J. Kauffels. *Network Management : Problems, Standards and Strategies*. Addison-Wesley, Wokingham, UK, 1992.

[34] S. Krause and T. Magedanz. “Mobile Service Agents enabling ‘Intelligence on Demand’ in Telecommunications”. In *Proc. IEEE Global Telecommunications conference (GLOBECOM’96), London, UK, November 1996*. IEEE Press, New York, NY, USA, 1996.

[35] A. Leinwand and K. Fang. *Network Management : a Practical Perspective*. Addison-Wesley, Reading, Massachusetts, USA, 1993.

[36] D.B. Levi and J. Schönwälder. *Script MIB. Definitions of Managed Objects for the Delegation of Management Scripts*. IETF Internet-Draft, 1st version, November 1996.

[37] T. Magedanz and T. Eckardt. “Mobile Software Agents : a new Paradigm for Telecommunications Management”. In *Proc. 1996 IEEE Network Operations and Management Symposium (NOMS’96), Kyoto, Japan, April 1996*. 2:360-369. IEEE Press, New York, NY, USA, 1996.

[38] S. Mazumdar. “Inter-Domain Management between CORBA and SNMP”. In *Proc. 7th IFIP/IEEE Int. Workshop on Distributed Systems : Operations & Management (DSOM’96), L’Aquila, Italy, October 1996*.

[39] S. Mazumdar and T. Roberts (Eds.). *Translation of GDMO Specification into CORBA-IDL*. Report of the XoJIDM task force, August 1995.

[40] S. Mazumdar (Ed.). *Translation of SNMPv2 Specification into CORBA-IDL*. Report of the XoJIDM task force, September 1996.

[41] K. McCloghrie. “The SNMP Framework”. In *The Simple Times*, 4(1):9-10, 1996.

[42] P. Miller. "A platform for the rapid development of CMIP agents and objects". In *Proc. 1994 IEEE Network Operations and Management Symposium (NOMS'94), Kissimmee, FL, USA, February 1994*, pp. 107-118. IEEE Press, New York, NY, USA, 1994.

[43] M. Morgenstern. “Active Databases as a Paradigm for Enhanced Computing Environments”. In M. Schkolnick and C. Thanos (Eds.), *Proc. 9th Int. Conf. on Very Large Data Bases (VLDB 1983), Florence, Italy, October 1983*, pp. 34-42. Morgan Kaufmann, 1984.

[44] M.A. Mountzia. “Intelligent Agents in Integrated Network and Systems Management”. In *Proc. 1996 EUNICE Summer School on Telecommunications Services, Lausanne, Switzerland, September 1996*.

[45] L.J. Mullins. *Management and Organisational Behaviour*. 2nd edition, Pitman, London, UK, 1989.

[46] OMG. *The Common Object Request Broker : Architecture and Specification*. Revision 2.0. July 1995.

[47] L.A. Phifer. “Tearing Down the Wall : Integrating ISO and Internet Management”. In *Journal of Network and Systems Management*, 2(3):317-322, 1994.

[48] M. Post, C.C. Shen and J. Wei. “The Manager/Agency Paradigm for Distributed Network Management”. In *Proc. 1996 IEEE Network Operations and Management Symposium (NOMS’96), Kyoto, Japan, April 1996*, 1:44-53. IEEE Press, New York, NY, USA, 1996.

[49] A.S. Rao and M.P. Georgeff. “Modeling rational agents within a BDI-architecture”. In R. Fikes and E. Sandewall (Eds.), *Proc. Knowledge Representation and Reasoning (KR&R-91), San Mateo, CA, USA, April 1991*, pp. 473-484. Morgan Kaufmann, 1991.

[50] M.T. Rose. *The Simple Book : an Introduction to Internet Management*. 2nd edition. Prentice Hall, Englewood Cliffs, NJ, USA, 1994.

[51] V. Sahin. “Telecommunications Management Network : Principles, Models and Applications”. In S. Aidarous and T. Plevyak (Eds.). *Telecommunications Network Management into the 21st Century : Techniques, Standards , Technologies and Applications*. IEEE Press, New York, NY, USA, 1994.

[52] J. Schönwälder. “Network Management by Delegation : from Research Prototypes towards Standards”. Submitted to the *8th Joint European Networking Conference (JENC8), Edinburgh, Scotland, UK, May 1997*.

[53] Y. Shoham. “Agent-Oriented Programming”. In *Artificial Intelligence*, 60(1):51-92, 1993.

[54] J. Siegel. *CORBA Fundamentals and Programming*. Wiley, New York, USA, 1996.

[55] M. Sloman. *Network and Distributed Systems Management*. Addison-Wesley, Wokingham, UK, 1994.

[56] M. Sloman and J. Kramer. *Distributed Systems and Computer Networks*. Prentice-Hall International, London, UK, 1987.

[57] J.M. Smith, D.J. Farber, C.A. Gunter, S.M. Nettles, D.C. Feldmeier and W.D. Sincoskie. *SwitchWare : Accelerating Network Evolution*. Technical Report MS-CIS-96-38, CIS Dept, University of Pennsylvania, USA, 1996.

[58] F. Somers. “HYBRID : Unifying Centralised and Distributed Network Management using Intelligent Agents”. In *Proc. 1996 IEEE Network Operations and Management Symposium (NOMS’96), Kyoto, Japan, April 1996*. 1:34-43. IEEE Press, New York, NY, USA, 1996.

[59] W. Stallings. *SNMP, SNMPv2 and CMIP : the Practical Guide to Network Management Standards*. Addison-Wesley, Reading, Massachusetts, USA, 1993.

[60] J.W. Stamos and D.K. Gifford. “Remote Evaluation”. In *ACM Transactions on Programming Languages and Systems*, 12(4):537-565, 1990.

[61] D.L. Tennenhouse, J.M. Smith, W.D. Sincoskie, D.J. Wetherall and G.J. Minden. “A Survey of Active Network Research”. *IEEE Communications Magazine*, 35(1):80-86, 1997.

[62] D.L. Tennenhouse and D. Wetherall. “Towards an Active Network Architecture”. *ACM Computer Communication Review*, 26(2):5-18, 1996.

[63] T.D Weinshall and Y.A. Raveh. *Managing Growing Organizations : a New Approach*. Wiley, Chichester, UK, 1983.

[64] C. Wellens and K. Auerbach. “Towards Useful Management”. In *The Simple Times*, 4(3):1-6, 1996.

[65] M. Wooldridge. “Issues in Agent-Based Software Engineering”. In P. Kandzia and M. Klusch (Eds.), *Proc. 1st Int. Workshop on Cooperative Information Agents, Kiel, Germany, February 1997*. LNAI 1202:1-18, Springer-Verlag, Berlin, Germany, 1997.

[66] M. Wooldridge and N.R. Jennings. “Agent Theories, Architectures and Languages : a Survey”. In M. Wooldridge and N.R. Jennings (Eds.). *Intelligent Agents. Proc. ECAI-94, Workshop on Agent Theories, Architectures and Languages, Amsterdam, The Netherlands, August 1994*. LNAI 890:1-39. Springer-Verlag, Berlin, Germany, 1995.

[67] Y. Yemini. “The OSI Network Management Model”. In *IEEE Communications Magazine*, 31(5):20-29, 1993.

[68] Y. Yemini, G. Goldszmidt and S. Yemini. “Network Management by Delegation”. In I. Krishnan and W. Zimmer (Eds.), *Proc. IFIP 2nd Int. Symposium on Integrated Network Management (ISINM’91), Washington, D.C., USA, April 1991*, pp. 95-107. North-Holland, Elsevier, Amsterdam, The Netherlands, 1991.

[69] T. Yemini and S. da Silva. “Towards Programmable Networks”. In *Proc. 7th IFIP/IEEE Int. Workshop on Distributed Systems : Operations & Management (DSOM’96), L’Aquila, Italy, October 1996*.

[70] T. Zhang, S. Covaci and R. Popescu-Zeletin. “Intelligent Agents in Network and Service Management”. In *Proc. IEEE Global Telecommunications Conference (GLOBECOM’96), London, UK, November 1996*. IEEE Press, New York, NY, USA, 1996.

[71] S. Znaty. *La qualité de service d’un multi-réseau : du modèle à sa mise en oeuvre* (in French). PhD thesis, École Nationale Supérieure des Télécommunications, Paris, France, 1993.