Journal of Network and Systems Management, 7(1):9–26, 1999

A Survey of Distributed Enterprise Network and Systems Management Paradigms

Jean-Philippe Martin-Flatin[1,2], Simon Znaty [2,3] and Jean-Pierre Hubaux[2]

ABSTRACT: Since the mid 1990s, network and systems management has steadily evolved from centralized paradigms, where the management application runs on a single management station, to distributed paradigms, where it is distributed over many nodes. In this survey, our goal is to classify all these paradigms, especially the new ones, in order to help network and systems administrators design a management application, and choose between mobile code, distributed objects, intelligent agents, etc. Step by step, we build an enhanced taxonomy based on four criteria: the delegation granularity, the semantic richness of the information model, the degree of specification of a task, and the degree of automation of management.

KEY WORDS: Distributed Network Management, Distributed Systems Management, Integrated Management, Taxonomy, Delegation, Automation of Management.

1. INTRODUCTION

Network and systems management (N&SM) has thrived on either centralized or weakly distributed paradigms for many years. Soon after the advent of open systems, proprietary solutions gradually gave way, in the early 1990s, to two open protocols: the Simple Network Management Protocol (SNMP [1]) and the Common Management Information Protocol (CMIP [2]). These protocols primarily addressed what was then perceived as the most critical feature lacking in existing N&SM systems: interoperability between multiple vendors. SNMP was widely adopted by the Internet Protocol (IP) world to manage local area networks, wide area networks and intranets, and, to a lesser extent, distributed systems. Parallel to this wide-scale deployment, CMIP, richer but more complex than SNMP, found a niche market in the telecommunications world, as the ITU-T (International Telecommunication Union, Telecommunication Standardization Sector) decided to adopt the Open Systems Interconnection (OSI) management model [3, 4, 5] of the International Standards Organization (ISO) as the basis for its Telecommunications Management Network (TMN) model [6, 7].

Despite the lack of competition between these two protocols that looked set to rule their respective markets for many years, both the protocols and their underlying models have been criticized since the mid 1990s. Why is it that more and more network managers are now demanding strongly distributed management technologies, when the same people were happy with centralized or weakly distributed technologies a few years ago?

The answer, in our view, is twofold. First, strongly distributed management paradigms address some of the major shortcomings of traditional paradigms. Beyond mere interoperability, they offer scalability, flexibility and robustness [8]. These three features, identified by Goldszmidt when he proposed his own model (Management by Delegation), actually justify the use of any kind of strongly distributed management technology. Second, much progress has been made in software engineering since CMIP and SNMP were devised, and new technologies suggested new ways of doing N&SM. For example, the architecture of distributed objects offered by the Object Management Group (OMG) with the Common Object Request Broker Architecture (CORBA [9]) proved to be a viable alternative to the traditional client-server paradigm, while intelligent agents [10] started to spread from Distributed Artificial

1. Correspondence should be directed to: J.P. Martin-Flatin, EPFL-DI-ICA, 1015 Lausanne, Switzerland. Email: martin-flatin@epfl.ch. Fax: +41-21-693-6610.
2. Institute for computer Communications and Applications (ICA), Department of Computer Science (DI), Swiss Federal Institute of Technology in Lausanne (EPFL), Lausanne, Switzerland.
3. Intelligent Networks, Services and Applications Group, Department of Multimedia Networks and Services, École Nationale Supérieure des Télécommunications de Bretagne (ENST-Bretagne), Rennes, France.

Intelligence (DAI) to distributed systems. New languages also appeared, such as Java [11], widely adopted by the Web community, and the Knowledge Query and Manipulation Language (KQML [12]), which is well established in Multi-Agent Systems (MASs), a sub-domain of DAI.

As a result, the N&SM community was recently overwhelmed by an avalanche of new technologies. Today, research is going in all directions, and it is increasingly difficult to tell in which one N&SM is currently heading. A designer of N&SM applications is faced with increasingly difficult decisions to make: "What management paradigm should I use to manage my network or my distributed system?"; "Should I choose a new management technology, thereby gaining in functionality, but losing in terms of standardization, and exposing my production network to dreaded heterogeneity problems?"; "Supposing I have decided to adopt a new management paradigm, what technology should I go for, when dozens of them have been prototyped, but hardly any have ever been deployed in real life to manage production networks or systems?"; "Should I use a mix of different paradigms or a single one?"; "What about a single paradigm but multiple technologies?"

Our goal in this paper is to clarify the situation for designers of N&SM applications, by classifying all open (i.e., non proprietary) technologies into a limited set of paradigms, and by proposing criteria to assess and weigh the relative merits of different paradigms or technologies. In particular, we show that there is no win-all solution: different technologies are good at managing different networks and distributed systems.

The remainder of this paper is organized as follows. In section 2, we first define a terminology, because there is a great deal of inconsistency in the jargon used by the N&SM, software engineering and DAI communities (e.g., what is an agent?). In section 3, we present a simple taxonomy of N&SM paradigms based on a single criterion: the organizational model. In this taxonomy, paradigms are grouped into four broad types: centralized, weakly distributed hierarchical, strongly distributed hierarchical and cooperative paradigms. We then outline some of the limitations of this simple taxonomy, and the need for a richer classification. In sections 4 to 6, we analyze successively the four criteria retained for our enhanced taxonomy: the delegation granularity, the semantic richness of the information model, the degree of automation of management, and the degree of specification of a task. Finally, we conclude with our enhanced taxonomy, detailed in section 7.

2. TERMINOLOGY

Before we proceed with the review of distributed N&SM paradigms, we must first acknowledge that the N&SM research community has not fully converged on a common terminology yet. Most people agree that the centralized paradigm is characterized by a single N&SM station, concentrating all the management application processing, and a collection of agents limited to the role of dumb data collectors. But there are different views on several other definitions. For example, some authors advocate the use of their new distributed technology by criticizing centralized technologies, but overlook hierarchical technologies that already exist [8, 13]; and most authors simply ignore cooperative technologies [13, 14, 15, 16, 17, 18, 19]. To address this confusion, we therefore propose the following terminology.

An N&SM application is composed of *managers*, running in N&SM Stations, and *agents*, running in managed devices (which can be network devices, systems, or components of a distributed system). In practice, an N&SM application may actually consist of several independent applications running on one or several hosts. For example, network management and systems management may rely on different software. But for the sake of clarity and simplicity, we will consider it as a single application. Similarly, an N&SM *station* will be either a Network Management Station, a Systems Management Station, or both.

By extension, the *managers* sometimes refer to the N&SM stations, and the *agents* refer to the managed systems or network devices. These are clearly misnomers, but these terms are seldom ambiguous once placed in context. To avoid any confusion between programs and humans when we use the term *managers*, the people in charge of managing networks or systems will be called *administrators*.

The meaning that we retained for the word *agent* is standard for the N&SM community. It comes from the manager-agent paradigm, one of the building blocks of OSI management and all SNMP frameworks (SNMPv1, SNMPv2 and SNMPv3). But we experienced that it is confusing to people coming from the software engineering or DAI communities. To avoid any confusion, we will speak of an *intelligent agent* when we mean an agent in the DAI sense, whereas a *mobile agent* will refer to one of the vectors of mobile code (software engineering). These concepts are presented in great detail in [20].

In earlier work [21], we compared the different types of organizations in the enterprise and networking worlds (that is, organization charts vs. computer and telecommunication networks). To summarize, distributed management is to computer science what decentralized management is to the enterprise world: a management paradigm based on the delegation of tasks to other entities. These entities are people in the enterprise world, and machines or programs in computer science. *Delegation* is used in both contexts as a generic word to describe the process of transferring power, authority, accountability and responsibility [22, 23] for a specific task to another entity. In distributed N&SM, delegation always goes down the management 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*. In enterprises, we can also find *upward delegation*; for example, an employee delegates his tasks to his manager when he is out due to illness [23]. Downward delegation and upward delegation are two kinds of *vertical delegation*, typical of hierarchical paradigms. In the enterprise world, organization charts generally follow a *hierarchical paradigm*. They are characterized by a multi-layer pyramid, comprising a *top-level manager* (at level 1), several *mid-level managers* (at levels 2, 3...), and *operatives* at the lowest level [22]. In distributed N&SM, we also have one top-level and several mid-level managers, but operatives are called *agents*. Orthogonally to vertical delegation, we have *horizontal delegation*, between two peers at the same level, typical of *cooperative paradigms* used in DAI. Distributed N&SM may rely on a hierarchical paradigm, a cooperative paradigm, or a combination of the two. Indeed, any paradigm outside the realm of centralized paradigms belongs to distributed N&SM.

Delegation is normally a *one-to-one relationship*, between a manager and an agent in a hierarchical management paradigm, or between two peers (be they managers or agents) in a cooperative management paradigm. Arguably, delegation may also be considered, in some 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. One-to-many delegation is forbidden by most authors in enterprise management [22, 23, 24, 25]. It can be considered in DAI though. In distributed N&SM, we propose to classify it as a form of cooperation, by coupling hierarchical and cooperative paradigms: 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.

To conclude with the terminology, some people confuse *management paradigms* and *management technologies*. A typical example is CORBA: in the literature, we find it referred to indistinctly as a paradigm, a technology, or even as a framework, to avoid choosing between the two. In the tradition of software engineering, and especially object-oriented analysis and design, we consider that technologies implement paradigms [26]. At the analysis phase, network and systems administrators select a management paradigm (e.g, distributed objects). At the design phase, they select a management technology (e.g., Java or CORBA). At the implementation phase, they use that technology to program the N&SM application.

3. A SIMPLE TAXONOMY OF ENTERPRISE NETWORK AND SYSTEMS MANAGEMENT PARADIGMS

With these definitions in mind, let us now introduce our simple taxonomy of N&SM paradigms. This taxonomy is presented in detail in [27, 20], where we define the concepts of mobile code (Remote Evaluation, Code On Demand, Mobile Agents), distributed objects (CORBA, Distributed Java, Java Management Application Programming Interface —JMAPI—, Web-Based Enterprise Management —WBEM—, Open Distributed Management Architecture —ODMA—), and intelligent agents. We simply summarize it here.

When we first built this taxonomy [21], we had six objectives in mind: (i) it should be a first, intuitive categorization of management paradigms; (ii) for the sake of clarity, it should comprise a limited number of types; (iii) it should clearly separate centralized paradigms from distributed paradigms; (iv) it should highlight the inherent differences between traditional paradigms and new paradigms; (v) it should distinguish paradigms relying on vertical delegation from those based on horizontal delegation; and (vi) it should enable designers of N&SM applications to find at a glance the paradigm implemented by a given technology.

To keep this taxonomy simple, we decided to base it on a single criterion: the organizational model. This is the approach taken by most authors [20]. To meet the third objective, we started with two types: centralized paradigms and distributed paradigms. To meet the fourth objective, we had to further split up distributed paradigms. By studying the different technologies that implement distributed management, we found that despite the wide spectrum of approaches taken, they could all be classified into two broad types, according to the role played by agents in the N&SM application. We called them weakly and strongly distributed technologies; they implement weakly and strongly distributed paradigms.

Weakly distributed paradigms are characterized by the fact that the N&SM application processing is concentrated in a few managers, whereas the numerous agents are limited to the role of dumb data collectors (in an intranet, we typically have one or two orders of magnitude between the number of agents and the number of managers). A typical example of weakly distributed hierarchical management is the OSI management framework.

Strongly distributed paradigms, on the other hand, decentralize management processing down to each and every agent: management tasks are no longer confined to managers, all agents and managers take part in the N&SM application processing. Many strongly distributed technologies have been suggested in the recent past. We showed in previous work [27, 20] that they can be grouped into three types. The first two, mobile code and distributed objects, implement vertical delegation, and are based on hierarchical paradigms. The third type, intelligent agents, implements horizontal delegation, and is based on a cooperative paradigm.

The simple taxonomy that we propose consists of four types:

- centralized paradigms
- weakly distributed hierarchical paradigms
- strongly distributed hierarchical paradigms
- strongly distributed cooperative paradigms

Our sixth and last objective is met by Fig. 1.

	centralized paradigms	hierarchical paradigms	cooperative paradigms
not distributed	SNMPv1, SNMPv2c, SNMPv2u		
weakly distributed		SNMPv1 + RMON, SNMPv2p + M2M, SNMPv3 + DISMAN, OSI management	
strongly distributed		mobile code, distributed objects	intelligent agents

Fig. 1. Simple taxonomy of enterprise N&SM paradigms

The main advantage of this taxonomy is that it highlights some similarities between apparently very different approaches. Despite the fact that new technologies appear every month, network and systems administrators are no longer overwhelmed by the variety of approaches offered to them: they have a simple way to analyze them, which reduces the scope of their investigation.

Its main disadvantage is that it is more oriented toward academia than industry. By focusing on the sole organizational model, it remains fairly theoretical, and does not really say what paradigm or what technology to use in the context of a given enterprise. Administrators, especially designers of N&SM applications, need more pragmatic criteria. They have some tough software engineering problems to solve at the analysis and design levels, and find it difficult to make the right decision when so many possibilities are offered to them. They need to think carefully before choosing expensive technologies such as CORBA, or before embarking for unexplored lands like intelligent agents, perhaps more adapted to pioneering start-ups than corporate organizations. The purpose of our enhanced taxonomy is to fulfill this need. In the next sections, we will study four criteria, and show the software engineering trade-offs of the different approaches with respect to each criterion.

4. DELEGATION GRANULARITY

In [21], we studied the features that designers wanted from strongly distributed N&SM. We showed that besides interoperability and scalability, which are already addressed by weakly distributed management paradigms, the four most important and discriminating criteria were the granularity of the delegation process, the semantic richness of the information model, the degree of specification of a task, and the degree of automation of management. In this section, we analyze the first criterion, delegation granularity. We clarify the concepts of delegation by domain and delegation by task in N&SM, and explain why we need to distinguish between micro-tasks and macro-tasks.

4.1. Delegation by domain, delegation by task

In order to better understand the process and scope of delegation, we compared in [20] the organizational models encountered in N&SM with businesses' organizational structures. We highlighted strong similarities between the two, and demonstrated that the current evolution of N&SM from centralized to weakly distributed to strongly distributed paradigms follows the track of an enterprise organizational evolution. We compared the granularity of the delegation process in businesses [23, 24] and in N&SM [28], and proposed to group all possible delegation policies into just two types: 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 (N=1 for the top-level manager, N=2,3,4... for the mid-level managers). In today's networks, delegation by domain typically translates into delegation by geographical domain, to manage geographically dispersed enterprises. For instance, let us suppose that the headquarters of a multinational company are located in Sydney, Australia. This company cannot afford to manage its large subsidiaries in the United States of America (USA), Asia or Europe over expensive and relatively slow transcontinental wide area network links. Let us consider its European subsidiary, located in Geneva, Switzerland. The manager in Sydney delegates the whole management of the Swiss subsidiary to the manager located in Geneva, and expects it not to report when a local printer goes down, but to report when the number of errors per minute exceeds a given threshold on the Switzerland-Australia link. The point here is that the Australian manager does not tell the Swiss manager 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 administrator, hard-coding in the Swiss manager what to report back to Sydney, and how to manage the rest of the local network. There is no mechanism for the Australian manager to alter the way the Swiss manager manages its domain: it is a *carte blanche* type of delegation, where the Geneva-based manager has total control over its own local network. Network management is not automated, and there is no way for the Australian network administrator to enforce a management policy over all its subsidiaries. Clearly, these are serious limitations.

Delegation by task, conversely, offers a finer grained vision at level (N) of the management processing occurring at level (N+1). As a result, 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 to be static, and hard-coded in every manager: they may as well be modified dynamically. This idea was first applied to N&SM when Management by Delegation was devised: Goldszmidt departed from the well-established notion of static

tasks underlying the centralized paradigm, and introduced the notion of dynamic tasks, transferable from the manager to its subordinate agents. This paradigm was soon generalized by others to transfer dynamic tasks from a manager at level (N) to a manager at level (N+1).

4.2. 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 SNMPv1, 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 Management Information Base (MIB) variables, which are statically defined when the MIB is designed. With large MIBs or large networks, this leads to the micro-management syndrome [8], which entails a significant network overhead, and a poor use of the resources of N&SM stations, managed devices, and managed systems.

Recent approaches avoid this syndrome by splitting the N&SM application into many different units, or *tasks*, and by distributing these tasks over a large number of managers and agents, while still letting the manager at level (N) in control of what subordinates at level (N+1) do. The underlying mechanism of this distribution is independent of the tasks being delegated: it can rely on program transfer, message passing, Remote Procedure Calls (RPCs), etc. The focal point for the N&SM application is the granularity of the delegation, that is, the way the work is divided. Clearly, there is a wide spectrum of task complexities, ranging from the mere addition of two MIB variables to the whole management of an Asynchronous Transfer Mode (ATM) switch. We propose to distinguish only two levels in our enhanced taxonomy: micro-tasks and macro-tasks.

A *micro-task* (μ-task) simply 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 to be aggregated by the manager one level up. If contact with the manager 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 manager one level up, corrective actions can be automatically undertaken.

5. SEMANTIC RICHNESS OF THE INFORMATION MODEL, AND DEGREE OF SPECIFICATION OF A TASK

The semantic richness of the information model of an N&SM application is an indication of the expressive power of the abstractions used in this model. It measures the facility for designers of N&SM applications to specify a task to be executed by a manager or an agent. The higher the level of abstraction used to model an N&SM application, the higher the semantic richness of the information model, and the easier it is for someone to build and design an N&SM application.

In N&SM, people like to think at a high level of abstraction, particularly those in charge of the analysis and design of large and/or complex applications. But management frameworks have traditionally offered fairly poor Application Programming Interfaces (APIs), constraining designers to model N&SM applications with low-level abstractions. This limitation has been addressed recently by some of the new management paradigms, as we show in this section. Today, designers of N&SM applications have the choice between three types of abstractions to build their information model:

- managed objects, offering low-level abstractions;
- computational objects, offering high-level abstractions; and
- goals, offering very high-level abstractions.

Let us review these three types of abstractions. We will introduce and compare the concepts of protocol API and programmatic API, and will identify a new criterion for our enhanced taxonomy: the degree of specification of a task.

5.1. Managed objects (low-level abstractions)

Both the SNMP and the OSI management frameworks offer what is called a *protocol API*. In these frameworks, there is a one-to-one mapping between the communication model and the information model, to use the ISO/ITU-T terminology [4]. In other words, the abstractions defined in the information model, which constitute the building blocks for the designer of an N&SM application, are identical to the communication protocol primitives used underneath. The protocol is not transparent to the application; this breaks a well-established rule in software engineering. For instance, in the different SNMP frameworks, programmers of N&SM applications have to think in terms of SNMP `get` and `set` when they write applications.

We call this the managed-object approach, as both the Internet Engineering Task Force (IETF) and the ISO use the phrase *managed object* to describe a basic unit of the information model in the SNMP and OSI management frameworks. This identity between the communication model and the information model has nothing to do with the protocols themselves: it is implicit in the management frameworks. When an N&SM application is designed with managed objects, a protocol is automatically imposed, the managed objects must live in full-blown agents (in the case of TMN, these agents need to implement a large part of the OSI stack, including the Common Management Information Service —CMIS— and CMIP), and the manager-agent style of communication is imposed. These are very strong constraints imposed on N&SM application designers. Today, most technologies implementing centralized or weakly distributed hierarchical paradigms are based on this managed-object approach.

5.2. Computational objects (high-level abstractions)

Protocol APIs for distributed systems are based on ideas which began to be criticized in the late 1970s and early 1980s, especially by the software engineering research community which was then promoting the new concept of *objects*. Since the mid 1980s, this community has been advocating the use of *programmatic APIs* instead, which have been one of the selling points of the object-oriented paradigm for distributed systems. With such APIs, any object belonging to a distributed system is defined by the interface it offers to other objects. The distributed object model is independent of the communication protocol: it only defines a programmatic interface between the invoker and the operations (methods) supported by the invoked object. This programmatic API relies on a protocol at the engineering level, but this protocol is completely transparent to the N&SM application designer.

We call this the computational-object approach, with reference to the terminology used by the ISO for the Open Distributed Processing (ODP) framework and ODMA. In this approach, designers of N&SM applications can rely on rich class libraries, offering high-level views of network devices and systems. Few constraints are imposed on the design: objects may be distributed anywhere, they need not live in specific agents that implement specific protocol stacks. The only mandatory stack is the one that implements the distributed processing environment. No specific organizational model is imposed or assumed: the N&SM application relies solely on object-to-object communication. The administrator may define his own site-specific classes, and use them in conjunction with libraries of classes that implement standard MIBs, such as Sun's transcription of MIB-II [29] in JMAPI.

The computational-object approach is the main strength of many new management technologies which have recently appeared. In N&SM, it accounts to a large extent for the recent success of CORBA in telecommunications, and Distributed Java in the IP world (Internet, intranets, extranets).

5.3. Goals (very high-level abstractions)

The third type of abstractions that may be used in information models is the *goal*. In [27], we explain that cooperative paradigms are goal-oriented: the N&SM application is split into tasks, which are modeled with very high-level abstractions, and partially specified with goals. Once these goals have been sent by the manager to the agent, it is up to the agent to work out how to achieve these goals.

This approach is fundamentally different from the one taken by weakly or strongly distributed hierarchical paradigms, where the N&SM application is broken down into fully specified tasks. Whether the

implementation of the task relies on calls to communication protocol primitives, or method calls on objects, the agent is given by the manager a step-by-step *modus operandi* to achieve its task. With cooperative paradigms, it is not.

Goals may be specified via a programmatic API, a protocol API, or both. They do not require an object-oriented distributed system to be used underneath, but the coupling of agents and objects seems promising in N&SM. Knapik and Johnson [30] describe different styles of communication between intelligent agents: object-oriented agents rely on remote method calls, whereas plain agents rely on communication languages such as KQML [12]. The primitives (performatives) of KQML are considerably richer than those of SNMP and CMIP, so goals are less limited by protocol APIs than managed objects.

Goals represent the highest level of abstraction available to date to N&SM application designers. They rely on complex technologies, based on intelligent agents, supporting some kind of inference engine and pattern learning, which are generally not available on managed systems or network equipment. So there is still a large market for simpler technologies that support computational objects, or even simpler technologies that support only managed objects. But goals are a type of abstraction that makes it possible to manage very complex networks, systems or services, for which simpler abstractions are not suited. They are particularly well suited to support negotiation (e.g. to get the best deal for a cross-Atlantic videoconference from competing service providers), load balancing, or resource usage optimization. In [20], we give practical examples of the use of goal-oriented intelligent agents for multimedia services and Virtual Private Networks (VPNs).

Managed objects and computational objects rely on fully specified tasks, whereas goals rely on partially-specified tasks. In other words, the semantic richness of the information model and the degree of specification of a task are tightly coupled. We decided to retain the latter as a criterion for our enhanced taxonomy, since it shows two very different ways of specifying tasks in N&SM. But we stress that these two criteria are not independent.

6. DEGREE OF AUTOMATION OF MANAGEMENT

Until a few years ago, the main purpose for the automation of management was to relieve network and systems support staff from the burden of constantly monitoring visually a Graphical User Interface (GUI), and of solving problems manually as they occur. As systems and networks grow by the year in size and complexity, administrators become more and more eager to automate their management: ad hoc manual management is not coping anymore. While advocating the use of Management by Delegation, Yemini claimed a few years ago that “management should pursue flexible decentralization of responsibilities to devices and maximal automation of management functions through application software” [5, p. 28].

Today, the need for more automation of management is also determined by two factors: the deregulation of the telecommunications industry worldwide, and the explosion of new services offered to end-users, especially multimedia services. There is more and more competition in the telecommunications market: monopolies (or near monopolies) gave way to a plethora of competing network operators, service providers, service traders, content providers, etc. So any service provision today is likely to cross several networks, managed by different companies, with equipment from several suppliers [31].

More and more services are being offered as well: mobile telephony, electronic commerce, video on demand, videoconference, teleteaching, telemedicine, etc. Videoconferences, for example, used to be booked by fax on an ad hoc basis. End users would contact support staff several days in advance; support staff would fax the single provider on the market (the local network operator); they would receive a reservation confirmation and an invoice within a day or so, sometimes less; and finally, they would inform the end-user that the booking had been made. This process was time-consuming, very inefficient, and error prone. Today, end-users want to deal directly with a service trader via a user-friendly GUI, get the best possible deal for a videoconference scheduled in a couple of hours, and make an electronic transaction with a click of a mouse. Such demands are much more stringent than they used to be, and

require considerably more work than mere faxes. As the number of such transactions grows (from once a month to once an hour), and as the demands become more stringent ("I do not want to book a videoconference for next week, but for this afternoon"), manual handling becomes less often an option. Service management must be automated, to offer the on-line GUI that the end-user expects. Network management also has to be automated, for instance, to handle resource reservations and potential rerouting. Eventually, systems management must also be automated, for example, to provide for automatic failover (so-called hot stand-by) for video-on-demand servers.

As we show in Fig. 2, micro-tasks poorly automate distributed N&SM, but macro-tasks are very good at it, because they enable remote agents to take corrective actions independently from the manager. Intelligent agents are typically used in negotiation, but they are also good at dealing with the dependencies between service management, network management, and systems management. To summarize the need for automation, the larger and the more complex the networks or the systems, the more automated the N&SM application should be.

7. AN ENHANCED TAXONOMY OF ENTERPRISE NETWORK AND SYSTEMS MANAGEMENT PARADIGMS

We have now completed our enhanced taxonomy. It is based on the four criteria presented in the previous sections:

- delegation granularity
- semantic richness of the information model
- degree of automation of management
- degree of specification of a task

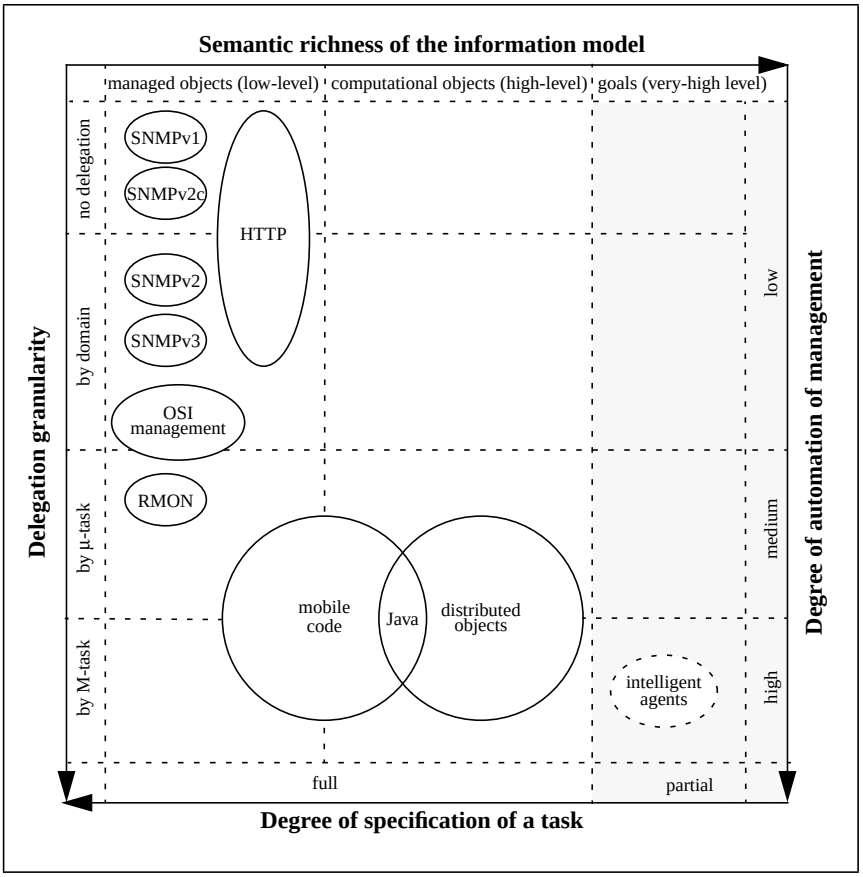

Fig. 2. Enhanced taxonomy of enterprise N&SM paradigms

These criteria are not independent: the semantic richness is closely linked to the degree of specification of a task, as is the delegation granularity to the degree of automation. In Fig. 2, the axes take discrete values, not continuous values: the relative placement of different paradigms in the same quadrant is not meaningful.

If we count the quadrants which are populated in Fig. 2, we see that our enhanced taxonomy comprises seven types:

- No delegation
- Delegation by domain
- Delegation by micro-task with low-level semantics
- Delegation by micro-task with high-level semantics
- Delegation by macro-task with low-level semantics
- Delegation by macro-task with high-level semantics
- Delegation by macro-task with very high-level semantics

8. CONCLUSION

In order to help designers of distributed N&SM applications select the right paradigm and technology for a given enterprise, we proposed two taxonomies, grouping all management technologies into two small sets of management paradigms.

First, we introduced a *simple taxonomy*, based on a single criterion: the underlying organizational model. This taxonomy consists of four types: centralized paradigms, weakly distributed hierarchical paradigms, strongly distributed hierarchical paradigms, and cooperative paradigms. It is well suited to classify the multiple N&SM technologies available today, with new ones appearing constantly.

Second, we presented an *enhanced taxonomy*, based on four criteria: the granularity at which the delegation process takes place (by domain, by micro-task, or by macro-task); the semantics of the information model (managed object, computational object, or goal); the degree of automation of management (high, medium, or low); and the degree of specification of a task (full or partial). We studied the different mechanisms of delegation, distinguished delegation by domain and delegation by task, and introduced the concepts of micro-tasks and macro-tasks. We also explained the need for richer semantics than those traditionally proposed by the SNMP frameworks or OSI management, and illustrated how new technologies could alleviate some deficiencies in the traditional approaches to N&SM. We demonstrated that the degree of specification of a task is closely linked to the semantic richness of the information model, and justified the need for automated management in complex settings. This enhanced taxonomy complements the previous one by being more practical. It gives some arguments for designers of N&SM applications to select one paradigm or another, based on the issues they face during the analysis and design phases.

In the future, we intend to develop a prototype demonstrating the distribution of network management in the IP world. We are currently assessing the relative merits of different mobile code, distributed objects and intelligent agents technologies, and will study whether the coupling of hierarchical and cooperative paradigms can lead to more effective ways to managing multimedia networks.

ACKNOWLEDGMENTS

This research was partially funded by the Swiss National Science Foundation (FNRS) under grant SPP-ICS 5003-45311. The authors wish to thank H. Cogliati for proofreading this paper.

REFERENCES

1. J. Case *et al.* (eds.), *RFC 1157. A Simple Network Management Protocol (SNMP)*, IETF, May 1990.
2. 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.
3. 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.
4. CCITT (now ITU-T), *Recommendation X.701. Data Communication Networks — Information Technology — Open Systems Interconnection — Systems Management Overview*, ITU, Geneva, Switzerland, January 1992.
5. Y. Yemini, The OSI Network Management Model, *IEEE Communications Magazine*, Vol. 31, No. 5, pp. 20–29, 1993.
6. ITU-T, *Recommendation M.3010. Principles for a Telecommunications management network*, ITU, Geneva, Switzerland, May 1996.
7. 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, pp. 72–121, 1994.
8. 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.
9. OMG, *The Common Object Request Broker: Architecture and Specification*, Revision 2.0, July 1995.
10. 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, Springer-Verlag, Berlin, Germany, pp. 1–39, 1995.
11. J. Gosling and H. McGilton, *The Java Language Environment: a White Paper*, Sun Microsystems, October 1995.
12. T. Finin *et al.*, KQML as an Agent Communication Language. In N.R. Adam *et al.* (eds.), *Proc. 3rd Int. Conf. on Information and Knowledge Management (CIKM'94), Gaithersburg, MD, USA, November 1994*, ACM Press, New York, NY, USA, pp. 456–463, 1994.
13. 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*, IEEE Press, New York, NY, USA, Vol. 2, pp. 360–369, 1996.
14. G. Goldszmidt, *Distributed Management by Delegation*, Ph.D. thesis, Columbia University, New York, NY, USA, December 1995.
15. H.G. Hegering and S. Abeck, *Integrated Network and System Management*, Addison-Wesley, Wokingham, UK, 1994.
16. A. Leinwand and K. Fang Conroy, *Network Management: a Practical Perspective*, 2nd edition, Addison-Wesley, Reading, MA, USA, 1996.
17. K. Meyer *et al.*, Decentralizing Control and Intelligence in Network Management. In A.S. Sethi *et al.* (eds.), *Integrated Network Management IV, Proc. 4th IFIP/IEEE Int. Symp. on Integrated Network Management (ISINM'95), Santa Barbara, CA, USA, May 1995*, Chapman & Hall, London, UK, pp. 4–16, 1995.
18. M. Sloman (ed.), *Network and Distributed Systems Management*, Addison-Wesley, Wokingham, UK, 1994.
19. W. Stallings, *SNMP, SNMPv2 and CMIP: the Practical Guide to Network Management Standards*, Addison-Wesley, Reading, MA, USA, 1993.
20. J.P. Martin-Flatin and S. Znaty, Two Taxonomies of Distributed Network Management Paradigms. In P. Ray and S. Erfani (eds.), *Network and Systems Management: Emerging Trends and Future Challenges*, Plenum Press, New York, NY, USA, 1999.
21. J.P. Martin-Flatin and S. Znaty, *Annotated Typology of Distributed Network Management Paradigms*, Technical Report SSC/1997/008, SSC, EPFL, Lausanne, Switzerland, March 1997.
22. D. Evans, *Supervisory Management: Principles and Practice*, 2nd edition, Cassell Educational Ltd., London, UK, 1986.
23. L.J. Mullins, *Management and Organisational Behaviour*, 2nd edition, Pitman, London, UK, 1989.
24. T.D. Weinshall and Y.A. Raveh, *Managing Growing Organizations: a New Approach*, Wiley, Chichester, UK, 1983.
25. M. Armstrong, *A Handbook of Personnel Management Practice*, 4th edition, Kogan Page, London, UK, 1991.
26. A. Fuggetta *et al.*, Understanding Code Mobility, *IEEE Trans. on Software Engineering*, Vol. 24, No. 5, pp. 342–361, 1998.

27. J.P. Martin-Flatin and S. Znaty, A Simple Typology of Distributed Network Management Paradigms. In A. Seneviratne *et al.* (eds.), *Proc. 8th IFIP/IEEE Int. Workshop on Distributed Systems: Operations & Management (DSOM'97), Sydney, Australia, October 1997*, pp. 13–24.
28. R. Boutaba, *Une architecture et une plate-forme distribuée orientée objet pour la gestion intégrée de réseaux et de systèmes* (in French), Ph.D. thesis, Pierre & Marie Curie University, Paris, France, March 1994.
29. K. McCloghrie and M. Rose (eds.), *RFC 1213. Management Information Base for Network Management of TCP/IP-based internets: MIB-II*, IETF, March 1991.
30. M. Knapik and J. Johnson, *Developing Intelligent Agents for Distributed Systems*, McGraw Hill, New York, NY, USA, 1998.
31. S. Aidarous and T. Plevyak, Principles of Network Management. 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, pp. 1–18, 1994.

Jean-Philippe Martin-Flatin is currently preparing for a Ph.D. thesis at EPFL. From 1990 to 1996, he was with the European Centre for Medium-Range Weather Forecasts in Reading, England, where he worked in network and systems management, security, Web management and software engineering. From 1988 to 1990, he worked on the Geographic Information System of a large city in France. In 1986, he received an M.Sc. in EE and ME from ECAM, Lyon, France. His main research interest is in distributed network management. He is a member of the IEEE and the ACM.

Simon Znaty is a professor at ENST-Bretagne in Rennes, France, where he teaches and does research in telecommunication services engineering. From 1994 to 1996, he was a senior researcher with the Telecommunications Laboratory at EPFL in Lausanne, Switzerland. From 1993 to 1994, he worked on service creation and management with the Network Architecture Laboratory at NTT in Tokyo, Japan. In 1993, he obtained his Ph.D. degree in computer networks from ENST in Paris, France. He is a member of the IEEE, and is the author of three books on networking.

Jean-Pierre Hubaux joined EPFL in 1990, where he is now a full professor and co-director of the Institute for computer Communications and Applications, which employs 20 Ph.D. students. His areas of interest include service engineering and multimedia services. Prior to this, he spent 10 years in France with Alcatel, where he was involved in R&D activities, mostly in the area of switching systems architecture and software. He was also in charge of the introduction of artificial intelligence techniques, especially for system troubleshooting. He is a senior member of the IEEE and a member of the ACM.