Proceedings of the *Sixth IFIP/IEEE International Symposium on Integrated Network Management (IM'99)*, Boston, MA, USA, May 1999

Managing the Management: CORBA-based Instrumentation of Management Systems

A. Keller
Munich Network Management Team
Department of Computer Science, TU München
Arcisstr. 21, D-80333 Munich, Germany
akeller@ieee.org

Abstract

In liberalized (tele)communication markets, the goal of integrated management has become particularly demanding because now an arbitrary number of service providers needs to dynamically exchange customer- and technology-related data. In this context, management systems are crucial for the seamless interworking because they contain the management information that has to be shared between service providers: On the one hand, different service providers have chosen different management systems that are bound to specific, standardized management architectures and, due to the heterogeneity of the underlying frameworks, do not interoperate easily. Solutions for achieving interoperability between heterogeneous frameworks are one key factor towards integrated enterprise management and have been studied in the past. On the other hand, the provisioning of management instrumentation for the management systems themselves (i.e. the problem "how to manage the management") is still an open research question.

The paper presents a novel approach to this problem by defining a MIB for management systems. It can be regarded as a step towards integrated enterprise management and is based on the viewpoint languages and concepts of RM-ODP; generic management models for distributed applications can then be refined to handle the specifics of management systems. A CORBA-based prototype implementation for managing management systems illustrates the applicability of our concepts.

Keywords

Enterprise Management, Architecture Interoperability, Information Modeling, CORBA

1. Introduction and Motivation

Integrated management solutions based on standardized management architectures aim to support the network and service providers' efforts in maintaining a high degree of quality and availability of their services while decreasing the cost of running the information technology infrastructure. Although providers can choose their management solution from a large number of implementations, the increasing complexity and heterogeneity of distributed systems still represents a major challenge for the Operation, Administration, Maintenance and Provisioning (OAM&P) of large-scale public and corporate networks.

From a management point of view, the situation has become more complicated due to the implications of telecom deregulation: Nowadays, an enterprise is not only free to

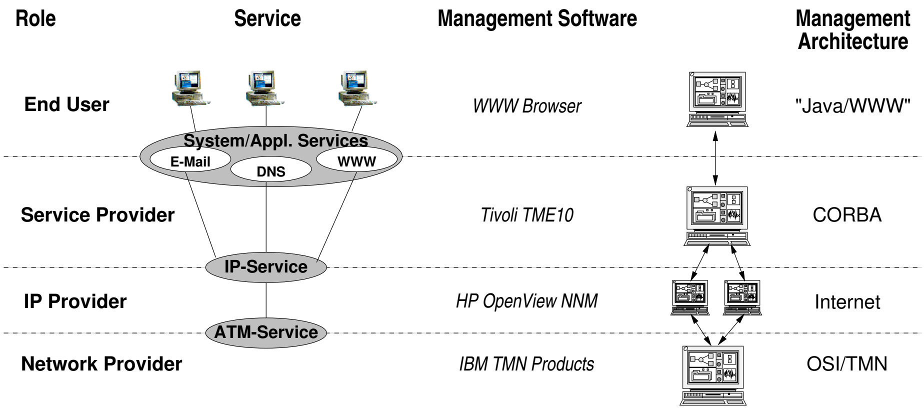

Figure 1: Service provider hierarchies require cooperation of management systems

choose between numerous IT service providers and network carriers but also needs to verify the QoS of the subscribed services. Furthermore, the role of the Internet Protocol (IP) as “lingua franca” for worldwide data communication and the layering of its standardized services and protocols (e.g., EMail (SMTP), Domain Name Service (DNS), World Wide Web (HTTP)) present opportunities of outsourcing IT activities for cost savings. This leads to layered service provider hierarchies as depicted in the left part of Figure 1 where a service provider is at the same time the customer of another provider: The provisioning of end-user services like electronic mail, WWW and DNS therefore depends on the availability of the IP service which, in turn, depends on, e.g., an ATM service. The problem domain dealing with the fulfillment of service-level agreements (SLA) between service providers and their users is usually known as *Customer Service Management* and is currently a subject of ongoing research (see e.g. [14]).

In this paper, we will concentrate on the right part of Figure 1, i.e., the question how the **Management Systems (MSs)** of the service providers and their customers can cooperate effectively. Today, it is very likely that these systems cannot interoperate seamlessly because different service providers usually deploy MSs that differ not only by their vendor (IBM/Tivoli, HP, CA), but also by the underlying management framework (OSI/TMN, Internet, CORBA, Java/WWW, proprietary). Furthermore, when these systems were initially purchased, there was often no need to exchange management information with peer MSs operated by another authority. This situation has changed: End users now check the quality of their subscribed services by retrieving data with WWW browsers from SNMP-based MSs and SNMP-based MSs that initially controlled the local area networks of an enterprise need to exchange management information with TMN-compliant MSs surveying long-haul telephony links etc. As MSs are the point of control for services and networks, management policies have to be enforced and surveyed by them. All this makes MSs crucial for successful enterprise management. It is therefore not only necessary to (re-)configure MSs at runtime but also to control whether they are working properly. We refer to these duties as “Managing the Management”. This paper presents a CORBA-based approach to tackle this problem.

The above discussion shows that two questions must be answered to ensure the successful deployment of MSs in an service markets:

1. How can the **Interoperability** between MSs be achieved, i.e., which mechanisms are needed so that they can exchange information with each other even if they are based upon different management architectures? The integration of management architectures, i.e., establishing a so-called "Umbrella Management" is currently a large field of investigation [4]; section 2 gives an overview on promising approaches.
2. What management information and which services are needed for Managing the Management? What are the management requirements and how does an appropriate model of MSs look like? Section 3 focuses on the aspects of **Interworking** between MSs.

Questions relating to the design of a Management Information Base (MIB) for MSs are: How is the management model structured? Can we reuse object classes specified by standardized reference models as base Managed Object Classes (MOCs) for our model? Section 4 describes the principles that guided us in designing our management model and how we made use of RM-ODP concepts. We also describe the impact of CASE technology for the design of the MIB. Section 5 deals with our implementation and analyses how the prototype uses general-purpose distributed object technologies like CORBA and Java. Section 6 concludes the paper and presents issues for further research.

2. Umbrella Management: Achieving Interoperability

A major objective of **Enterprise Management** consists in presenting a unified, all-encompassing view on networks, systems, services and applications. It is oriented towards the providers' goals and gives them the ability to control any kind of managed resource according to their management requirements, policies and organizational conditions. Enterprise Management follows a top-down approach and should therefore not be constrained by technical specifics. On the other hand, the amount of management frameworks providers have to deal with is already determined by the nature of the managed resources: LAN components usually have SNMP agents while telecommunication components are managed according to the OSI/TMN framework; distributed applications and services might provide a CORBA-based management instrumentation. **Umbrella Management** has to cope with the heterogeneity of Management and focuses on the technology-related aspects of cooperation between entities involved in the management process. Its objective is to abstract from the heterogeneity of the underlying management architectures by defining means of interoperability. It is therefore fundamental for integrated Enterprise Management (see Figure 2).

There are basically three strategies for achieving interoperability between systems located in different architectural domains (see also [10]):

The first approach consists in the integration at the resource level, i.e., the managed systems support more than one management protocol; they are equipped with **Multiarchitectural Agents**. This is usually unfeasible for the following reasons: Often, agents are used to perform monitoring of simple network devices like hubs or bridges. They

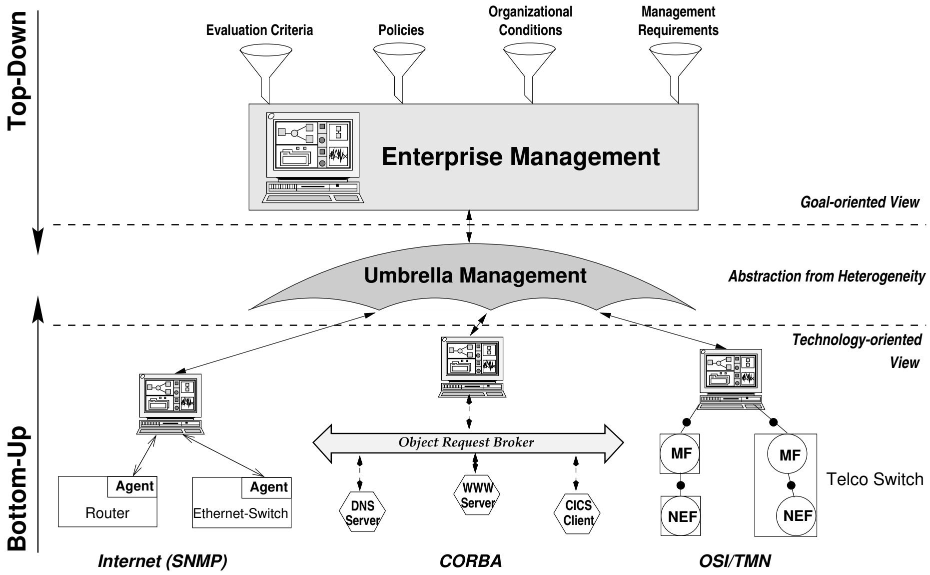

Figure 2: Umbrella Management as a basis for Enterprise Management

should not consume a large amount of resources and are usually built into the firmware of the device; the implications are that these agents can neither be enhanced to support another management protocol nor should they introduce additional overhead.

An alternative is to place the burden of integrating the different architectures on the MS. Such a **Multiarchitectural Manager** supports a set of management protocols which are implemented onto the platforms' communication stack. Thus, conversions between different management protocols are not necessary. The transformation of the management information descriptions is often handled by tools bundled with an MS like MIB compilers and therefore need not be handled by the developer. In addition, the service APIs of MSs can easily be accessed thus yielding the opportunity of reusing the large amount of platform services (see also section 5). On the other hand, these APIs are specific to a concrete MS product: the portability of interoperability solutions based on this integration paradigm is often restricted.

The third solution is the **Management Gateway (MG)** approach. It is then possible to manage services, systems and networks in different management architectures from a single point of control as demonstrated in [15] and [21]. It is even possible to apply the power of a management architecture with rich functionality to any resource in different architectural domains. In management architectures having no notion of a management functional model such as the Internet (SNMP) framework, the application of management functionality "borrowed" from other architectures is particularly useful. Standardized mappings between the OSI, Internet and CORBA information models have been developed by the *Joint Inter-Domain Working Group* of Opengroup and NM-Forum [9]; the appropriate interworking architecture has been recently submitted to the

OMG in response to the *CORBA/TMN Interworking RFP*. Our experiences with building MGs [12] have shown that they represent powerful instruments for bridging the gaps between different management architectures as neither the managing nor the managed systems need to be modified. This is a crucial feature w.r.t. the service provider scenario described in section 1. While design principles of MGs are today reasonably well understood, our experiences have also shown that they are complex systems and – due to their high relevance for enterprise management – need to be managed as well; in the next sections, we will describe why we consider MGs as a special case of MSs and show what their management instrumentation may look like.

3. Managing Management Systems

This section analyses the properties of MSs and the requirements concerning the necessary management information and the management services, respectively. It provides the conceptual background and helps to identify important features needed for the definition of the object model for MSs described in section 4.

3.1 Characteristics of Management Systems

The way how MSs are deployed today is by establishing well-defined, central points of control, termed *management platforms*. As outlined in several publications (e.g. [3], [22]), platforms tend to become bottlenecks in large networks and a solution to this problem is to distribute their functionality (e.g., topology functions, event filtering mechanisms, logging facilities) in order to reduce the load on the central MS. This can be done by introducing MSs acting in the role of **Mid-Level Managers (MLMs)** that are located closer to the managed resources and execute management functions on behalf of other MSs. Their main purpose is to condense raw data stemming from the resources into meaningful management information that can then be used by another MS. This may also include caching of frequently accessed information like topology data of the underlying systems. These properties obviously reduce the amount of overall traffic that is sent through the network for management purposes. MLMs are an effective mechanism for structuring networks into domains and can be used for establishing management hierarchies. An extension of this concept is to delegate (and withdraw) management functionality to MLMs at runtime. This delegation may either be initiated by the MLM itself (pull model) or by another MS (push model). The interworking between MSs of different service providers as described in section 1 can be seen as a special case of interactions between different MLMs. Consequently, the above mentioned principles apply also to independent MSs operated by different authorities.

As motivated in section 2, further complexity is introduced by the heterogeneity of the deployed management architectures. MGs are a convenient mechanism for achieving interoperability between heterogeneous MSs. They are obviously located on the boundaries of different (architectural) management domains and act in the role of an MLM w.r.t. other MSs. As MGs are aware of the architectural specifics of the underlying resources, it therefore makes sense to provide them with the same management functionality as MLMs. Such an enhanced MG could e.g., condense several SNMP-traps into one CORBA-event instead of translating every trap into one event and leaving the filtering of these events to the MS one level above in the service provider hierarchy. This implies that providers need an instrumentation for these systems in order to present a unified view on their management data to customers; the amount of accessible information may be specified in a service contract.

3.2 Requirements Analysis

Although the different management frameworks provide mechanisms of inter-manager-communication (OSI/TMN: x reference point, SNMP: inform-pdu), it is currently not clear what the shared management knowledge between MSs should look like and what actions may be initiated by an MS on behalf of another one. We will now first describe what kind of management information ought to be present in an object model for managing MSs and will then move on to the required management services, i.e., the functionality that MSs may provide for interacting with peers. The identification of management information and management functionality was based on a use case analysis [7] applied to typical management scenarios. For the sake of brevity, we can only describe a small part of the amount of management instrumentation identified in our analysis. The complete set is described in [11].

Management Information

As with any other kind of managed resource, generic configuration management information (manufacturer, product name and version, installation date, etc.) and properties relevant for fault management (support contact, time since last restart and its usage, operational and administrative states) are relevant for managing MSs. If an MS supports delegation, information about the supported scripting languages or the version of the execution engines should be available. From the inventory perspective, it is also necessary to provide an overview about the installed components, their versions and their states (e.g., the installed system modules, the database and the management applications, the type of available delegated management functionality) and the corresponding process names.

The gathering of licensing data relevant for accounting management like the kind of product license (nodelocked, domain-bound, floating), the maximum and the actual amount of MS users is necessary because many commercial MS products are equipped with license servers. A high amount of rejected requests due to an insufficient number of licenses indicates that additional licenses should be purchased. This may concern either the number of concurrent users of the MS or the number of managed nodes. Appropriate counters need to be provided.

Security is a major concern in distributed environments where several MSs are acting as peers: there is a need to provide information about the management domains an MS is involved in and the agents, MLMs and MGs it is responsible for. If an MS is able to configure another one, every involved system needs to maintain on the one hand view-related access control mechanisms for protecting itself against unauthorized access and on the other hand information about its own capability set, i.e., what kind of interactions with peer MSs are permitted.

Performance-related information includes parameters for configuring MLM caching properties like the cache size or the maximum aging time for the cached data. Counters for the time an MS takes for serving a request and the number of requests per interval (minutes, hours) may indicate performance bottlenecks requiring additional MLMs.

Additional information needed to control MGs encompasses the architectures the MG translates, the names and versions of the management information models and protocols supported. Of equal importance are counters for (successful, erroneous) translated PDUs and information related to the mapping of managed object references.

Management Services

This section will sketch a subset of the overall management functionality that is needed for MSs in order to interwork properly.

Configuration management services provide the ease of introducing new MSs into distributed environments by dynamically assigning them configuration profiles covering domain affiliation and polling intervals. A MS therefore needs to provide functionality for downloading initial configuration profiles to other MSs. If delegation is supported, the usual operations that apply to delegated management functionality (start, stop, suspend, resume etc.) should be present.

Of major importance for fault management are basic operations like verifying whether all the components of an MS are running and services for (re-)starting or stopping either the complete MS or single components should be available. Checking the consistency of the MS database and the execution of maintenance (e.g., backup and restore) and error detection tasks need also to be done at regular intervals that may require scheduling mechanisms. In case of errors, the event and error logs maintained by an MS provide a rich source of information for determining the probable cause of failures; consequently, facilities for searching these logs according to different criteria should be available.

The configuration of event forwarding mechanisms, the initiation of management operations on peer MSs and the registration of an MS for specific events require security services to ensure that no security policy is violated.

Services related to performance and accounting management include the generation of reports about resources administered by a specific MS (statistics, system parameters filtered according to different criteria like throughput, delay, load and usage) and should therefore be made available to network administrators.

3.3 Relationships of MSs to the System and Network Infrastructure

MSs usually run on general-purpose operating systems like UNIX or WindowsNT and are therefore sensitive to errors that occur in the underlying operation system and the communication infrastructure provided by networked services (e.g., the Domain Name System (DNS), the Network File System (NFS) and the Network Information System (NIS)) and protocols. Consequently, commercial MS implementations like the one described in [5] dedicate a high degree of their management instrumentation to the monitoring of the underlying system. While this may be necessary, in our opinion it is not the primary concern of a MIB for MSs to control not only the MS itself but also the underlying environment. Thus, we restrict ourselves to the instrumentation of the MS itself and consider the instrumentation of operating systems (CPU usage count, user quota, disk and paging space etc.), networks (connectivity checks, timeout errors, network buffer size, packet and frame errors) and underlying networked services like DNS, NFS and NIS as out of scope for this paper. Nevertheless, we have designed object models and implemented the corresponding agents for UNIX systems and networked services. Readers interested in these agents are referred to [16]. The development of these agents has been done according to the approach described in section 4, too. On the other hand, operating system configuration tools like IBM AIX SMIT provide a vast amount of static management information pertaining to MSs such as the names and versions of the software modules, their installation paths, their prerequisites and dependencies. As these tools are accessible through APIs we were able to retrieve this management information.

We consider management itself as a distributed application; the consequence that the distributed application "management" itself needs to be managed has already been motivated in section 1. Another implication is that we can apply already defined and standardized models for (general-purpose) distributed applications like the *Reference Model of Open Distributed Processing (RM-ODP)* to the management of MSs. We will show in the following section how we used concepts from RM-ODP to obtain generic management object classes for application management; we will use these as base classes for building the inheritance hierarchy of our object model.

4. An Object Model for Managing Management Systems

Seamless interworking between different MSs requires that the management information and the appropriate management services identified in the previous section are defined in a way that they can be accessed by peer MSs. Consequently, this section will focus on establishing an object model (a MIB) for MSs. In section 4.1, we show how we achieved a unified representation of management information common to all different kinds of entities involved in the management process. We then describe in section 4.2 how this management information can be refined w.r.t. the needs of specific MSs and how this process is eased by CASE tools.

4.1 Defining the Base Classes of the Inheritance Hierarchy

The inheritance hierarchies of today's object-oriented management frameworks are usually not very deep and the degree of reuse is limited because the predominant part of relevant management information is often specified in resource-specific classes. Consequently, the base classes do not contain a large amount of information. On the other hand, one must recognize that – in general – different distributed applications share a lot of common properties: e.g., they are delivered as packages that consist of modules implementing a service which is instantiated as a process. What we need is a framework that allows us to describe the relevant aspects of any kind of distributed application (in our case: management systems) in a way that it is suitable for management purposes. As described in [13], RM-ODP [6] provides a standardized framework that covers the different aspects of distributed applications and therefore has been taken as the basis for deriving generic application management instrumentation.

For the purpose of technical management, the *computational* and *engineering viewpoint languages* fit best because they standardize descriptions of the resources that we are interested in; the reasons for the restriction on two out of five viewpoints are detailed in [17]. However, the viewpoint language concepts only make assertions about object classes and do not provide information about the class properties or its operations. Thus, parts of the management information and services described in section 3 become attributes and methods of these enhanced base MOCs. Some of these abstract base MOCs are depicted on the highest level of Figure 3 and reflect the characteristics of distributed applications w.r.t. configuration and fault management. For the ease of understanding, we now describe the amount of management information covered by these generic MOCs by applying them to the components of a specific network management platform, namely *IBM NetView for AIX* (the appropriate component names are given in parentheses):

A **Computational_Object** stands for a running service (e.g., a platform topology service) that is instantiated as a process or – in RM-ODP terminology – **Capsule** (e.g., "gtmd"). The static configuration information about this service, i.e., information about the software module is described in the **Computational_Object_Template** (e.g., "Generalized Topology Manager"). In order to reflect the structure of large software packages, we have defined an additional abstract class **Package** (e.g., "IBM NetView for AIX") that serves as a container for the different software modules.

Although the base MOCs are abstract classes that must be further refined, one has to recognize that these MOCs based on RM-ODP concepts make that many required attributes and methods are already defined at the top of the inheritance hierarchy: Important characteristics relevant to software packages are present; in addition, instrumentation for installing and updating them is specified. Descriptions of services and their technical realization (as processes) are available. Services and applications can be started, stopped, suspended and resumed. This covers a large amount of MIB-instrumentation that is usually defined only on lower, i.e., resource-specific levels (e.g. every SNMP group or table contains `Name` and `Description` attributes). Consequently, we can then guarantee a minimum amount of instrumentation commonly applicable to all kinds of distributed applications.

4.2 Representation of MOCs for Managing Management Systems

The abstract base MOCs described in section 4.1 do not contain application specifics. In order to support the management of specific applications (here: MSs), they have to be refined to more application-specific MOCs. These can then serve as a starting point for the design and implementation of appropriate management agents.

For the sake of clarity, the following kinds of management information and services are not present in the figure: Object classes for the maintenance and control of event and alarm logs, instrumentation for traces and definitions of value constraints like thresholds. The definitions of the attributes (data types, default value, hints whether they are read-only) and methods (arguments, return type) have also been suppressed.

It can be seen that MOCs for generic clients and servers are derived from the abstract base MOCs; as MSs act in a client and managed systems act in a server role, we can derive them from the generic MOCs, respectively. The model also reflects the fact that MLMs are dual role entities and consequently inherit properties and operations from

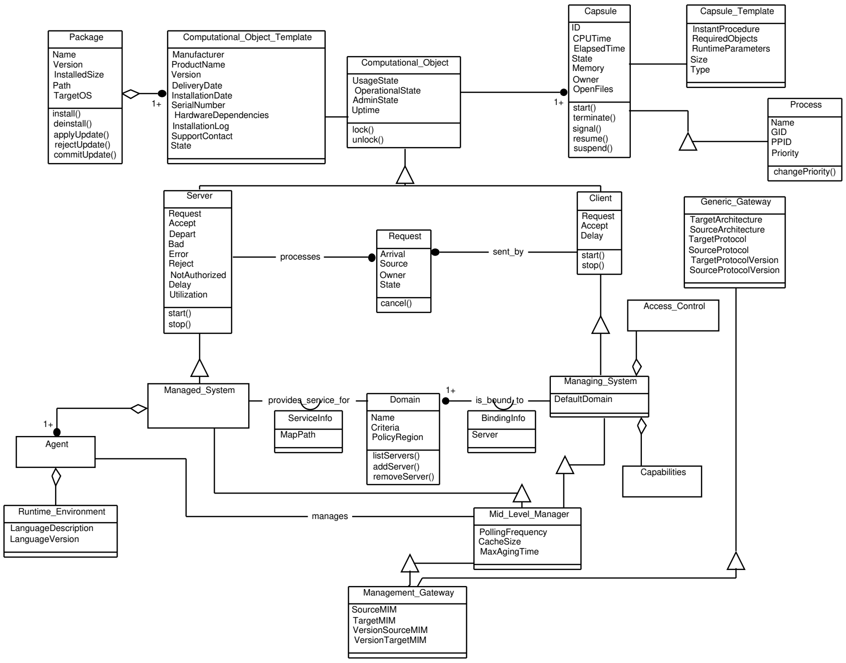

Figure 3: Object model of management systems (partial view)

both of the above MOCs. MGs, finally, are – as described in section 3.1 – a special case of MLMs that additionally perform transformation activities.

The design of the management object model is usually done with object-oriented design methodologies such as the *Object Modeling Technique (OMT)* [20] or the *Unified Modeling Language (UML)* [18] that not only allow the specification of MOCs, their properties and operations but also of the relationships between different MOCs. Furthermore, it should be possible to transform the resulting MOCs straightforward into the description languages of standardized management frameworks (e.g., OSI GDMO, OMG IDL). For our modeling, we have used an OMT-compliant CASE tool [1] that fulfills the above requirements; the object model can be transformed either into C++ or Java classes or OMG IDL. By introducing the OMT interface descriptions into the CASE tool repository we are able to follow a rapid-prototyping approach for implementing appropriate agents.

5. Prototype Implementation

For demonstration purposes, we have instrumented a commercial network management platform (*IBM NetView for AIX 4.1* with *TMN Support Facility*) and a CMIP/SNMP MG developed by us [12] with CORBA-based management agents implemented in Java. The left part of Figure 4 depicts a CORBA/Java-based management environment:

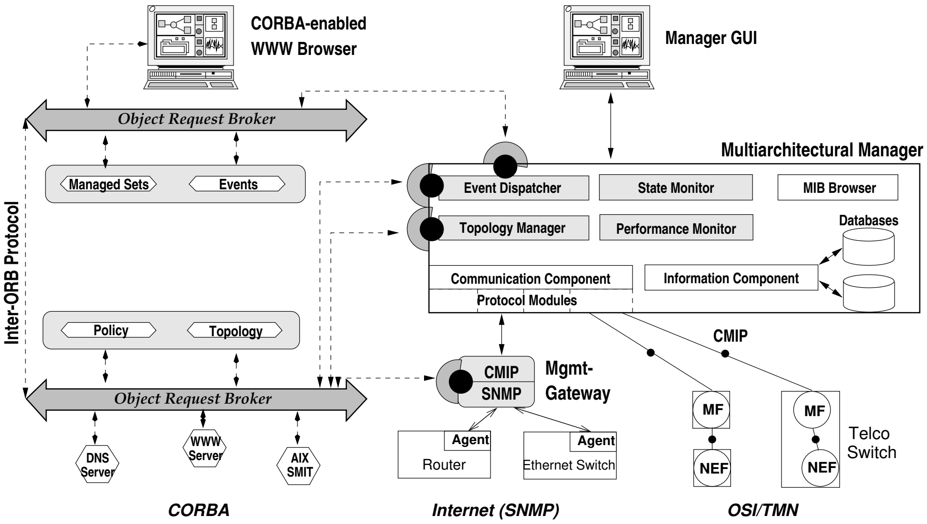

Figure 4: CORBA-based instrumentation of management systems and gateways

CORBA agents (here: for DNS and WWW servers and AIX SMIT) are remotely administered by means of a CORBA-enabled WWW browser. CORBAservices such as Events, Notification, Topology, Managed Sets and Policy (see e.g. [23]) provide the required management functionality in a location-independent way. However, only a small part of these important management services have been implemented until now. Fortunately, current management platforms (right part of Figure 4) contain components that have a very similar functionality to the CORBAservices mentioned above. We therefore decided to make the main platform components (Event Dispatcher, Topology Manager, State and Performance Monitors) accessible from CORBA and use them as a *temporary replacement* for currently specified, but not yet available CORBAservices. We have encapsulated the APIs of the MS with IDL wrappers (dark grey shaded arcs with black dots in Figure 4) which gives us the opportunity of reusing a large part of the MS functionality (e.g., event filtering, topology). By doing so, we create a conceptually integrated management information base on the platform where management-related information is collected and evaluated independent of the originating base architecture. This multiarchitectural manager is able to access managed objects in OSI/TMN, SNMP and CORBA environments.

On the other hand, the IDL wrappers also provide us with the management instrumentation of the MS and the MG via CORBA. Furthermore, the representation of AIX SMIT as CORBA objects gives us access to the static management information described in section 3.3. Together, they represent the managed objects identified in the previous sections. Our approach is in accordance with the TINA (*Universal Service Component Model (USCM)* [2]. The core of the MS and MG access layers are accessible either through the usage sector or through the server interfaces of the management sector; the latter has been provided by us. We are then able to manage the MS and MG via CORBA.

As we wanted to manage these systems from a web-based interface, we implemented the management application prototype as Java applets and used a commercial Java-ORB, *VisiBroker for Java v. 3.0* (by Visigenic/Inprise). VisiBroker is also contained in the *Netscape Communicator v. 4.5* web browser; we therefore expected that the application would load reasonably fast due to its modular design and the small size of the resulting Java classes (less than 10 kbyte per class). Unfortunately, Netscape Communicator contains only an earlier version of the Java-ORB we used, namely VisiBroker 2.5. Consequently, the Java classes for the ORB itself and its bundled CORBAservices (in total: a Java archive of 2.5 MB code size) always have to be loaded once per session into the browser first *before* any applet can be started.

On the other hand, the access to IBM NetView is only possible through C Application Programming Interfaces. We therefore had to build wrappers based on the *Java Native Interface (JNI)* around the C APIs in order to access them from Java. JNI has been developed for that purpose, thus enabling programs running under the control of the Java Virtual Machine to access other programs or libraries that have been written in C. The reason for choosing Java instead of C++ stems from the fact that we intend to copy parts of the management instrumentation between different MS at runtime.

6. Conclusions and Areas of further Research

In this paper, a CORBA-based approach for instrumenting management systems has been presented. The need for managing management systems arises because management systems of different authorities have to seamlessly interoperate with each other, even if they are based upon different management architectures. We have analyzed and evaluated three mechanisms for achieving interoperability. On the other hand, management systems are also very complex; it is thus essential to provide appropriate management information and services. Our aim was to define such an open, common set of management information that should be applicable to a wide range of management systems and management gateways. We have developed management models for these entities by analyzing which kinds of management information and services are needed to enable an integrated management of these systems.

Key to our approach is the perception of management as a distributed application that itself needs to be managed. We are then able to apply standardized frameworks for distributed applications (such as the RM-ODP) for the definition of generic MOCs that already contain a large set of management instrumentation at the top of the inheritance hierarchy. This management model was refined into more specific object classes that reflect the structure and functionality of management platforms available on the marketplace. We have implemented the object model described in this paper for instrumenting the commercial network management platform *NetView for AIX* by Java/CORBA-based management agents so that it can be managed from a web-based interface. We experienced that, apart from determining dependency relationships between instantiated components, almost the complete set of required management information can be provided by this management system. However, this information is scattered across different sources such as platform APIs, configuration files and configuration tools of the underlying operating system. A large part of our implementation work consisted in encapsulating all this information under a single type of interface.

With CORBAservices and CORBAfacilities, CORBA provides a very promising technology for integrated enterprise management since more and more services useful for management purposes are being standardized. The approach described in this paper has also been successfully applied to several cooperation projects with industrial partners for the management of networked services like NFS and NIS and for the re-engineering of agents for open CORBA-based management of ATM switches.

However, the mechanisms for transferring management services between management systems are yet unsatisfactory. We are now working on the following issues:

- The deployment of the OMG *Mobile Agent System Interoperability Facility (MASIF)* [19] to provide facilities for CORBA-based mobile management agents.
- The evaluation of the *Java Dynamic Management Kit (JDMK)* [8] to support the delegation of management functionality at runtime in Java/CORBA-based environments.

ACKNOWLEDGMENT

The author wishes to thank the members of the Munich Network Management (MNM) Team for helpful discussions and valuable comments on previous versions of the paper. The MNM Team directed by Prof. Dr. Heinz-Gerd Hegering is a group of researchers of the University of Munich, the TU München, and the Leibniz Supercomputing Center. Its webserver is located at http://wwwmnmteam.informatik.uni-muenchen.de.

References

[1] Aonix. *Software through Pictures/Object Modeling Technique: Creating OMT Models*. Aonix, Inc., 1997. Release 3.4.

[2] M. Chapman and S. Montesi. Overall Concepts and Principles of TINA. TINA Baseline TB-MDC.018_1.0_94, TINA Consortium, February 1995.

[3] G. Goldszmidt and Y. Yemini. Delegated agents for network management. *IEEE Communications Magazine*, 36(3):66–70, March 1998.

[4] H.-G. Hegering, S. Abeck, and B. Neumair. *Integrated Management of Networked Systems — Concepts, Architectures and their Operational Application*. Morgan Kaufmann, 1999.

[5] IBM Corporation, International Technical Support Organization, Research Triangle Park, NC 27709-2195. *IBM Systems Monitor: Anatomy of a Smart Agent*, December 1994. Order Number: GG24-4398-00.

[6] Open Distributed Processing – Reference Model. IS 10746, ISO/IEC, 1995.

[7] I. Jacobson, M. Christerson, P. Jonsson, and G. Övergaard. *Object-Oriented Software Engineering: A Use Case Driven Approach*. ACM Press / Addison-Wesley, 1993.

[8] JDMK. *Java Dynamic Management Kit 3.0 (beta) Programming Guide*. Sun Microsystems, Inc., August 1998. Part No. 805-4620-05.

[9] JIDM. Inter-Domain Management: Specification Translation. Open Group Preliminary Specification P509, Open Group, March 1997.

[10] P. Kalyanasundaram and A. Sethi. Interoperability Issues in Heterogeneous Network Management. In Manu Malek, editor, *Journal of Network and Systems Management*, volume 2, pages 169 – 193. Plenum Publishing Corporation, June 1994.

[11] A. Keller. *CORBA-basiertes Enterprise Management: Interoperabilität und Managementinstrumentierung verteilter kooperativer Managementsysteme in heterogener Umgebung*. PhD thesis, Technische Universität München, December 1998.

[12] A. Keller. Tool-based Implementation of a Q-Adapter Function for the seamless Integration of SNMP-managed Devices in TMN. In *Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS 98)*, pages 400–411, New Orleans, USA, February 1998. IEEE Press.

[13] A. Keller and B. Neumair. Using ODP as a Framework for CORBA-based Distributed Applications Management. In J. Rolia, J. Slonim, and J. Botsford, editors, *Proceedings of the Joint International Conference on Open Distributed Processing (ICODP) and Distributed Platforms (ICDP)*, pages 110–121, Toronto, Canada, May 1997. Chapman & Hall.

[14] M. Langer, S. Loidl, and M. Nerb. Customer Service Management: A more transparent view to your subscribed services. In *Proceedings of the 8th IFIP/IEEE International Workshop on Distributed Systems: Operations & Management (DSOM 98)*, Newark, DE, USA, October 1998.

[15] S. Mazumdar. Inter-Domain Management between CORBA and SNMP. In *Proceedings of the IFIP/IEEE International Workshop on Distributed Systems: Operations & Management*, L'Aquila, Italy, October 1996.

[16] T. Müller. CORBA-basiertes Management von UNIX-Workstations mit Hilfe von ODP-Konzepten. Master's thesis, Technische Universität München, February 1998.

[17] B. Neumair. Distributed Applications Management based on ODP Viewpoint Concepts and CORBA. In *Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS 98)*, pages 559–569, New Orleans, USA, February 1998. IEEE Press.

[18] Unified Modeling Language Summary version 1.1. OMG Specification ad/97-08-03, Object Management Group, September 1997.

[19] Mobile Agent System Interoperability Facilities Specification. OMG TC Document orbos/97-10-05, Object Management Group, November 1997.

[20] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Object-Oriented Modeling and Design*. Prentice-Hall International, Inc., 1991.

[21] N. Soukouti and U. Hollberg. Joint Inter-Domain Management: CORBA, CMIP and SNMP. In A. A. Lazar and R. Saracco, editors, *Proceedings of the 5th International IFIP/IEEE Symposium on Integrated Management (IM)*, pages 153–164, San Diego, USA, May 1997.

[22] C. Wellens and K. Auerbach. Towards Useful Management. *The Simple Times*, 4(3):1–6, July 1996.

[23] Systems Management: Common Management Facilities Volume 1. Preliminary Specification P421, X/Open Ltd., June 1995.

Biography

Alexander Keller received his Diploma and a Ph.D. in Computer Science from Technische Universität München, Germany, in 1994 and 1998, respectively. He does research on distributed systems and application management, emphasizing on management system instrumentation and aspects of interoperability between management architectures. He is a member of GI and IEEE.