Proceedings of the *IFIP/IEEE International Workshop on Distributed Systems: Operations & Management*, L'Aquila, Italy, October 1996

Service-based Systems Management: Using CORBA as a Middleware for Intelligent Agents

MNM TEAM
Munich Network Management Team

Alexander Keller*
Department of Computer Science, University of Munich
Oettingenstr. 67, 80538 Munich, Germany
Phone: ++49-89-2178-2167
Fax: ++49-89-2178-2262
E-Mail: keller@informatik.uni-muenchen.de

Abstract

Management of large, heterogeneous information technology infrastructures is only possible if integrated management solutions are applied. In the near future, providers of such distributed environments will have not only the choice between the well-known OSI/TMN- and Internet management frameworks but also have to consider a third alternative: The Common Object Request Broker Architecture (CORBA) which is gaining increasing importance in the field of network and systems management. This paper analyses the suitability of CORBA for integrated systems management and presents an approach for defining management functionality in terms of services accessible across heterogeneous environments. The set of management services represents a kind of middleware for systems management purposes; its existence is a prerequesite for the delegation of functionality from managing systems to managed systems. This delegation can occur either at the compile time of a management agent or during its runtime. Agents of the latter kind are termed intelligent agents in the context of this paper because their management functionality can dynamically be assigned or withdrawn. While the technical aspects of Management by Delegation are clarified, there is currently a lack of understanding *how* the functionality is made available for delegation. The paper presents a methodology for defining and implementing delegetable management services in CORBA environments as a base for intelligent agents.

Keywords: CORBA, Distributed Objects, Distributed Systems Management, Middleware, Service-based Management

1 Introduction

The ever-increasing complexity and heterogeneity of distributed systems represents a major challenge for the Operation, Administration, Maintenance and Provisioning (OAM&P) of large-scale public and corporate networks. Integrated management solutions based on standardized management architectures ([7]) support the network providers' efforts in maintaining a high degree of quality and availability of their services while decreasing the cost of running the information technology infrastructure.

Apart from the well-known OSI/TMN and Internet management architectures, a third alternative is gaining increasing attention: the Common Object Request Broker Architecture ([2])

*The author's research is supported by the IBM European Networking Center, Heidelberg

which has been standardized by the Object Management Group (OMG).[1] Initially developed for distributed object-oriented programming, the advantages of using CORBA in the domain of network and systems management are more and more recognized.

One reason for this is the fact that today the development of management systems is perceived as a special case of developing large-scale information technology applications. The successful application of new principles from the field of software engineering like object-oriented techniques for the analysis, design and programming of complex software systems leads to the development of modular, well-structured management applications built from "off-the-shelf" software components. This yields demand for techniques that protect developers, at least to a certain degree, from the heterogeneity of the underlying operating systems. This is particularly important for the development of powerful management software.

Management in open, heterogeneous environments, in turn, can only be effective on the basis of standardized architectures because these encompass not only the communications infrastructure, but also end-user systems and their software. It is easy to see that in large networks the amount of managed systems can become very high; on the other hand, often enough those networks are managed from a single point of control which is flooded with error messages if several managed systems encounter failures. For this reason (and others given in the second section), it is necessary to extract the functionality from the managing systems and distribute this "management intelligence" between the managing and the managed systems.

This distribution can be done in two ways:

- by transferring management functionality dynamically from managing systems to the managed systems. This approach is termed *Management by Delegation* ([17], [5]). It requires either the implementation of functionality in scripting languages like Java ([6]) or Tcl ([13]) or the use of extensible agents and the existence of a delegation protocol. The drawback of this approach is that implementations of already existing management functionality like the ISO-OSI Systems Management Functions ([9]) cannot be reused.
- by isolating the management functionality from the managing systems, packaging it in the form of objects and distributing it across a CORBA-based environment. This can be done by implementing the management functionality in terms of *Management Services* i.e., a *Middleware for systems management* that can be inherited either by managing systems or managed systems with respect to their needs; already existing implementations of object catalogues can be encapsulated and reused. A special delegation protocol is not required. The major difference to the former approach is that the functionality can not be altered during the runtime of the agent since "raw" CORBA does not support passing objects by value.

In contrast to papers focusing on the more technical aspects of Management by Delegation ([1], [4]), the work presented in this paper focuses primarily on the service-based approach. It presents arguments for using "yet another management architecture" and discusses design issues for establishing a middleware accessible for managing systems and managed systems in the form of management services. Finally, a methodology for bringing together the two approaches i.e. implementing intelligent agents on the basis of CORBA is also described. It is therefore important to bear in mind that the two approaches described above are not mutually exclusive.

The paper is organized as follows: section 2 analyses the needs for distributed management and management by delegation; section 3 introduces the concepts for using CORBA as an architecture for systems management. Section 4 discusses some of the more technical considerations to be made when building intelligent agents based on the paradigm of distributed objects. Finally, the last section of the paper concludes and presents issues for further research.

[1] With now over 650 members, OMG is the biggest software industry consortium worldwide.

2 Distributing management functionality

From the structuring point of view, the main components of management systems are as follows:

- *Sensors* collect events occurring on managed systems. In order to achieve this, sensors monitor the changes of management-relevant parameters and emit predefined notifications. The set of all parameters is termed the Management Information Base (MIB) and may be structured in an object-oriented way.
- *Sieves*: They process notifications emitted by the sensors. The main tasks of a sieve consist in filtering the incoming notifications and in transforming them into information by condensing one or more notifications into a single message. In a wider sense, the task of event correlation is also done by sieves. An example for such a transformation is the generation of an alarm if an exceeding of a previously defined threshold on one or more parameters is reported by the sensors.
- A *Management Algorithm* whose purpose is to take decisions and to trigger actions based on the information presented by the sieves. This role might be taken either by a human expert or a knowledge-based system.
- *Actors*, which enforce the actions triggered by the management algorithm. This is usually done by performing modifications on one or more managed systems i.e. setting parameters.

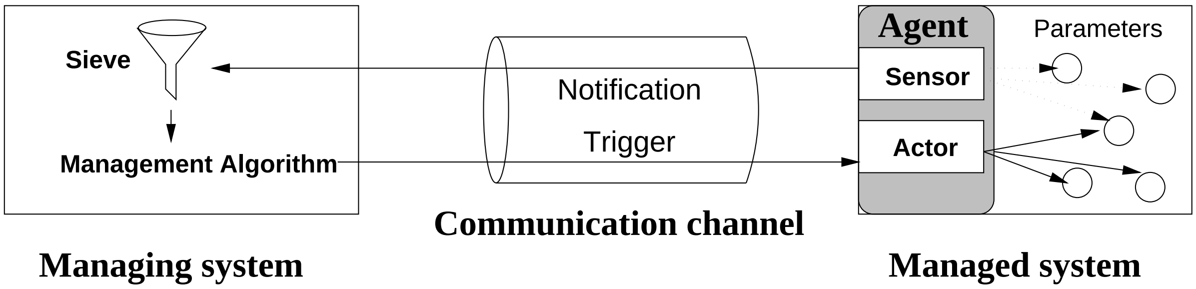

Figure 1: The Management Control Cycle in platform-centric Systems

These four base components implement the management control cycle depicted in figure 1 and are distributed in systems based on a platform-centric paradigm as follows: Sensors and actors reside on the managed system; together, they form the building blocks of the agent. The managing system or "manager" consists, among other components like graphical user interfaces and databases, of sieves and management algorithms. In order to relate events and actions to a specific managed system, the managing system has to store the logical descriptions of all the managed systems it surveys. Managing system and managed system exchange their data via a communication channel termed management protocol.

While the gathering of management-relevant data and the execution of actions is done in a distributed manner, the treatment of the data remains central on the management platform. Currently available state-of-the-art management platforms like *IBM NetView for AIX* or *HP OpenView* are based on this paradigm. As a consequence, these systems are very complex and pose high requirements on the resources of the underlying hardware.

The main drawback of this centralized approach is the fact that it is unscalable because the successful operation of such a system is bound to the following factors: the number of managed systems, the number of the managed system's parameters and the bandwith of the communication channel between managing and managed system. If any of the former ones increases or if the latter decreases, the whole network becomes unmanageable. In summary, by using a platform-centric approach, the management system and the communication channel are likely to become bottlenecks.

Having the above described base components in mind, this situation can only be avoided either by distributing parts of the sieves and the management algorithms across the common run time environment or by explicitly delegating the functionality to the managed systems.

3 Applying CORBA for distributed systems management

Structured according to the four models of the ISO/OSI management framework ([10]), the first part of this section outlines the implications of using CORBA as an architecture for distributed systems management. As the underlying models differ from their OSI and Internet counterparts, some important considerations concerning the managing and the managed systems have to be made. These are treated in the second part of this section.

3.1 Management models in the context of CORBA

3.1.1 Organizational model

CORBA has no notion of hierarchy but assumes symmetric peer-to-peer relationships between objects; in terms of management, this results in an extension of the role model: Not only are manager/agent or manager/manager relationships possible, but also agent/agent relationships. This simplifies the distribution and delegation of management functionality and supports cooperation between different components of the management infrastructure. It is therefore not only possible to delegate functionality from managing systems to managed systems but also between managed systems.

Another point is that the roles of systems participating in management activities are subject to dynamic changes: roles like "managing system" and "managed system" are temporary and bound to the tasks to be done. This has major implications on security (see 3.2.2).

3.1.2 Communication model

In contrast to existing management architectures, CORBA has no need for a separate management protocol as objects cooperate in invoking each others' methods. The communication infrastructure is the *Object Request Broker* (ORB) which transmits a client call to a server object regardless of their actual location i.e., from the clients' perspective there is no difference whether the server object resides on the local or on a remote machine. The prerequisite for achieving location transparency is that the ORBs running on different systems are able to interoperate. The corresponding mechanisms have been specified in the second version of CORBA ([2]).

Client calls are accepted by the ORB either through the *Client Stubs* or the *Dynamic Invocation Interface (DII)*. The flexibility of the DII consists in allowing a client to build its requests in a dynamic way. The implication for management is that no modification has to be applied to the managing system if changes occur at some managed systems.

The CORBA analogon of management operations is the invocation of methods on a remote object. For an acceptable performance, it is necessary that a managing system is able to issue asynchronous requests to a managed system. As asynchronous i.e. non-blocking calls are currently not supported by CORBA, service requests can be either synchronous or use *deferred synchronous* semantics. The latter is much more suitable for management as, upon submission of the call, the managing system obtains a handle to be used later to inquire about the state of the operation. Asynchronous calls can be simulated by deferred synchronous calls in conjunction with *best-effort* semantics.

3.1.3 Information model

As the main focus of CORBA is not bound to network and systems management, ASN.1 is not used for defining management information. Instead, the description of CORBA objects is established by specifying their interfaces in the *Interface Definition Language (IDL)*. IDL is a means of presenting the interfaces of all objects connected to an ORB system in a uniform way. Clients issue requests to servers without the need to take care on which kind of architecture or under which operating system the servers are running. Even the language in which servers are implemented is shielded by IDL. Each managed object class is defined by one IDL interface describing attributes and operations of the managed objects; managed objects can inherit from other objects as CORBA yields all the advantages of object-orientation. If new object implementations are introduced into the ORB runtime system, their interfaces are registered in the *Interface Repository*. The interface repository can be thought of as a distributed database which contains the interface descriptions of any object known throughout the ORB. Together with the DII (see above), the interface repository decouples the managing system from changes occurring at managed systems.

It is therefore not necessary (as it would be with the OSI[2]– and Internet management architectures) to keep a backup image of each managed systems' interface as a MIB in the memory of the managing system. Inconsistency problems in case of a change to an agent are avoided, too.

3.1.4 Functional model

The CORBA functional model is divided in three layers. Due to the object-oriented nature of the OMG framework, services contained in the upper layers are able to inherit functionality from underlying layers.

1. *CORBAservices* ([12]) form the lowest layer in the functional model and are mandatory for the operation of any CORBA-based implementation. The Externalization, Properties, Change Management and Licensing services are examples of CORBAservices particularly useful for management purposes.

2. The next building blocks in the hierarchy are *CORBAfacilities*; they consist of multi-purpose services useful for a large part of applications. Typical kinds of CORBAfacilities are: Information Management, Task Management and Systems Management. The latter provides the basis for introducing management functionality into the OMG framework and is of big interest for the definition of delegatable functionality as described in section 4 of this paper.

3. *Domain Interfaces*, finally, are high-level services suitable for specific application domains. As systems management is an important issue in all kinds of application domains, this layer does not define any management functionality. Instead, it is inherited from the underlying CORBAservices and CORBAfacilities.

[2]The *repertoire objects* specified in the current version of the OSI Management Knowledge Management Function [8] alleviate this problem.

Application objects reside on top of the three layers and obtain an important part of their functionality by inheriting from the three layers below. A typical example for an application object is a distributed management application.

3.2 Aspects of modularisation

3.2.1 Implications on managing systems

On the side of the managing systems, management platforms relying on standardized architectures are considered today as a sound basis for presenting heterogeneous resources in a uniform way to network operators. In order to reduce their complexity and the resulting high hardware requirements, a first step consists of dividing the management platform and the management applications running on top of it into smaller pieces.

Those smaller parts can either be encapsulated or re-implemented in an object-oriented manner; they are then interconnected by means of an ORB. Note that the use of an ORB needs not to be visible outside of the managing system; it is therefore possible to use management protocols like SNMP or CMIP for exchanging information with managed systems. This can be considered as an intermediate step towards a purely CORBA-based management solution. The *Tivoli Management Environment* [15] is an example of a commercial management implementation built according to this idea.

3.2.2 Implications on managed systems

The development principles and implementation methods for the agent remain the same as for the application itself because, using CORBA as underlying software platform, the development of management software becomes a special case of developing information systems. As outlined in the previous subsection, there is neither a specific management protocol nor a specific kind of information representation.

It is therefore more likely that, in the future, an application development team will not only implement the application but also the corresponding management part. The problem of undocumented or even non-existing management interfaces of systems and applications could then be solved. Also, off-the-shelf CASE-tools may be used for the analysis, design and programming for the agent part of an application.

Another issue is security: While in traditional client/server architectures a client could always trust the server, this is no more true for distributed objects due to the frequent role changes of CORBA-clients and -servers. It is therefore important to provide mechanisms on either sides (client and server) to prevent unauthorized execution of functionality. An object implementation must have the ability to check whether a client is authorized to perform functions on the object: With every method call, the ORB runtime system delivers information from which client a service request has been issued. A managed system can therefore authenticate the managing system and decide whether the delegated functionality should be executed or not. As it is impossible for a managed system to determine the semantics of delegated functionality, access control lists have to be maintained by the managed system.

3.2.3 Cooperation with existing management architectures

Considering the large base of installed systems based on the Internet and OSI/TMN architectures, using OMG technology does not imply a replacement of well-known standards; in fact, it is likely that some managed resources remain either based on the traditional management frameworks or even use proprietary protocols. It is therefore necessary to provide smooth

transitions that enable not only the coexistence, but also the cooperation between different management architectures. This can be achieved by *management gateways* that translate not only the communication models by mapping protocol data units to ORB-requests/-replies and vice-versa but also the structure of management information i.e. the information models. Bearing in mind the efforts of the *OMNIPoint* initiative ([3]) conducted by the Network Management Forum, it is obvious that specifying management gateways between the OSI/TMN, Internet and OMG architectures is a challenging task.
As this paper focuses mainly on a CORBA-based environment, this topic is out of scope.

4 A service-based approach for implementing intelligent agents

As mentioned in section 3, the *Systems Management Facilities* provide a means of integrating management functionality into the OMG framework. This section discusses different ways of building those facilities in order to extend CORBA by services suitable for systems management.

The service-based approach assumes that management functionality is not bound to the managing system but allows, if it is defined in terms of *Management Services*, being distributed and delegated: These services can then be used either by managing or by managed systems.

As already mentioned before, delegation of management functionality can happen at different stages in the agent's lifecycle: Functionality can either be assigned at compile time[3] or during the runtime of the agent (Management by Delegation). While management functionality cannot be withdrawn from an agent during its lifetime using the former approach, this is feasible in the latter case.

Subsection 4.1 deals with definition aspects of management services; these aspects are independent of the mechanisms how implementations handle delegation (static or dynamic). Subsection 4.2 analyses how management functionality already defined in other management architectures can be accessed by CORBA-based management systems. The last subsection, 4.3, covers the case of dynamic delegation of functionality and presents an approach how to implement intelligent agents in CORBA.

4.1 Defining Management Middleware

There are two main reasons for the definition of pre-defined, commonly usable and delegatable functionality for realizing management services in distributed heterogeneous environments i.e. *management middleware*:

- Functionality needed by several management applications should only be defined and implemented once. It can then be used collectively by all sorts of management applications. This can be done in object-oriented environments by inheriting the desired functionality from appropriate services. Examples of this kind of functionality are services for creating, deleting and grouping managed objects or the application of management policies.
- As mentioned in section 2, scalability issues require the delegation of specific management functions to the managed systems. Appropriate examples are threshold monitoring, event filtering and forwarding or the scheduling of periodic management tasks.

The adequate means of implementing management middleware in object-oriented environments is by defining the functionality in terms of *services* and making them accessible to all

[3]This is the usual way how the OSI Systems Management Functions are included in an agent.

the components of a management system. The Interface Repository and the DII in conjunction with CORBA 2.0 interoperability provide a well-suited infrastructure for these purposes as the interfaces of newly introduced managed objects are implicitly propagated throughout the ORB environment.

Figure 2 illustrates how management services can be shared between managing and managed systems. As these services are implemented as objects, any managing or managed system is able to derive the needed functionality from these services.

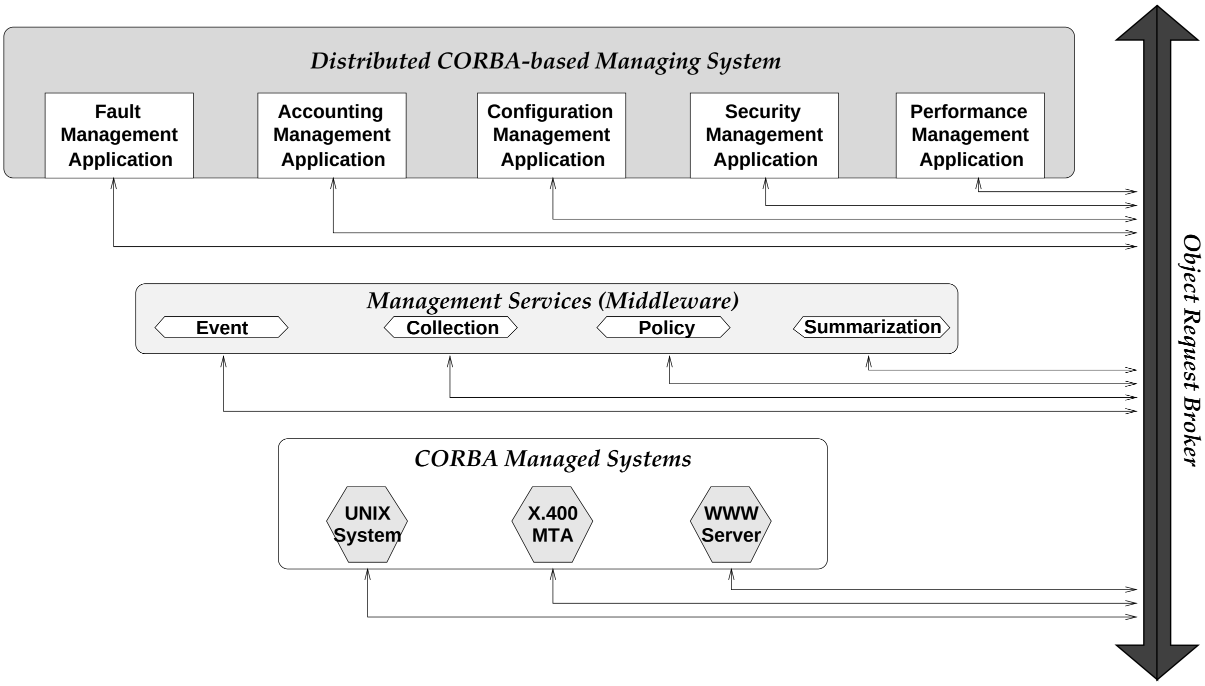

Figure 2: Sharing Management Services between Managing System and Managed Systems

There are two ways of introducing management services in CORBA:

- The first possibility is to implement new services "from scratch" and introduce them as new CORBAfacilities. This approach has been chosen by the X/Open Systems Mananagement Task Group. A first set of management facilities has been submitted to the OMG and is described in detail in [16]. It encompasses the following services: Policy-driven Base, Instance Management, Policy Management and Managed Sets.
- The second way consists in reusing management functionality defined in the traditional management architectures and making it accessible to CORBA-objects. This approach is described in subsection 4.2.

It is important to consider that the boundary between CORBAservices and CORBAfacilities is not static but is able to change over time. The consequence is that services for systems management are not necessarily bound to the CORBAfacilities but can also be defined as CORBAservices or even be provided as part of the ORB runtime system.

The following example describes functionality defined as a CORBAservice: As in traditional client/server architectures, accounting plays an important role for CORBA-based applications. The main difference is that these applications consist of a multitude of fine-grained software components; current accounting mechanisms like node-locked or application-bound, floating licenses are therefore not appropriate for performing accounting functions in distributed environments. This problem has been recognized by the OMG and is addressed by the *Licensing* service.

An example of management functionality being part of the ORB and therefore not being defined in terms of services is given below: In CORBA-based systems, objects are identified by an object reference; this reference hides the objects' properties like its location. While this may look unsuitable for systems management because it is important to determine *where* to look in case of failures, this restriction is compensated by the methods of the Interface Repository in cooperation with the ORB Interface. Given an object class, some ORB implementations are able to determine the location of currently instantiated objects; this is comparable to a management auto-discovery service.

The conclusion to be drawn is that in CORBA environments, management functionality is not specified in terms of a dedicated model (as it is the case with the OSI/TMN architecture), but occurs at different levels of the OMG framework.

4.2 Accessing existing management functionality from a CORBA environment

Expressed in terms of traditional management architectures ([14]), management middleware lies between the communication infrastructure and the management applications. The degree of interoperability between different management architectures depends on how similar their functionality has been defined; an important requirement is that middleware defined in one architecture can also be used from another one. On the one hand, this implies that middleware needs to be defined in a precise and open manner. On the other hand, it is necessary to provide the communication infrastructure in order to access the middleware from another architecture; this aspect of interoperability is covered by the management gateways mentioned in 3.2.3.

An example may illustrate this: The OSI management architecture presents with the Systems Management Functions (SMF) a rich set of management functionality specified in terms of object classes. As CORBA possesses, at its current stage, only few similar management middleware, it is important to provide a means of using the SMFs from a CORBA environment. Therefore, it is necessary to map the OSI middleware to CORBA services. Examples of particular good candidates for such a mapping are the *Metric Objects and Attributes* or the object classes defined in the *Summarization* function.

The mechanisms of mapping middleware specified in different management architectures are encapsulation and data abstraction. The purpose of encapsulation is to hide implementation- and architecture-specific details: CORBA has, for example, no notion of OSI distinguished names as it uses a completely different naming scheme based on the *Object Naming Service*. For accessing OSI middleware from a CORBA object, it is important that the interfaces of the OSI middleware are given in IDL. This can be done by encapsulating the OSI SMFs with *wrapper objects* having an IDL interface. The fact that OSI relies on the object-oriented paradigm eases the transition of the middleware interfaces into IDL. It is therefore possible to reuse the knowledge and specifications of existing standards without needing to modify their implementation because it is irrelevant whether the implementation of a system is object-oriented; the important thing is that the interfaces of its components make them appear as objects ("Perception is reality"). The restriction is that such management functionality has to be defined at compile time of a CORBA agent.

In the Internet management architecture, management functionality is implicitly contained in object-based MIBs. Furthermore, as the Internet SMI has no notion of inheritance, it is difficult to reuse its functionality in object-oriented environments.

Another issue is the portability of the language in which object implementations are written. Software written in programming languages that need to be compiled is, generally spoken, unsuitable for Management by Delegation because the generated binaries are bound to the architecture of the target system. A solution for this problem is to provide management service implementations (generated through cross-compiling) for each architecture that needs to be supported. Due to the high degree of heterogeneity in todays' networked environments, this approach is unfeasible. The other possibility is to implement management services in scripting languages that are interpreted on the managed system or in a portable binary code as it is done with PERL or Java. This issue is covered in the following subsection.

4.3 Implementing Management by Delegation in CORBA

Management by Delegation permits downloading management functionality to managed systems at runtime. Its major advantage lies in the fact that managed systems need not store extensive management functionality in advance but can be provided with functionality only *on demand*. The corresponding agents are therefore quite small. Using the paradigm of distributed objects, management by delegation is feasible between any kind of object; it is therefore possible that agents cooperate without the participation of the managing system.

A typical problem domain where the use of inter-agent cooperation seems promising is fault management. The following example shows the use of this concept: In many cases, applications do not work properly because of errors in an underlying layer. To avoid network traffic caused by the exchange of data between a managing and a managed system, it is desirable to run detailed diagnostic and error correlation routines already on the system where the error occurred: Upon detection of a fault in an application, the agent in charge of the application may request the testing of the underlying communication system from another agent managing these parts of the system. The agent then has to perform a lookup to determine which testing services are available in the CORBA environment; this is done by querying the interface repository. Having obtained information about the interfaces of an object containing adequate functionality, a call to this object is issued through the DII requesting the delivery of the desired error detection services. These are then sent via the ORB to the agent and executed.

Further error detection, error correlation and even troubleshooting functionality may be downloaded to the requesting agent in the same way. A large part of the management tasks is therefore performed already on the managed system. The managing system shall then be provided with a summary of the results and can, if necessary, perform some additional actions.

For implementing Management by Delegation, it is crucial to by-pass the CORBA restriction of not allowing objects to be copied by value between different systems. The main idea of the work-around lies in the fact that a CORBA client is able to pass parameters in calling a method of a server object. Therefore, if it is feasible to insert an object as a parameter to a remote method call, its reference is marshaled and converted to a proxy object at the agent's side. After this has happened, the methods of the delegated object can be accessed by the agent.

The remaining question is how objects can be flattened into a data stream in memory. This can be accomplished by using the CORBA *Externalization Service*, whose purpose is to enable the internal state of an object to be obtained in an external representation. [11] states that "objects which support the appropriate interfaces and (...) can be externalized to a stream (in memory, on a disk file, across the network, etc.) and subsequently be internalized into a new object in the same or a different process". Apart from the Externalization Service, there is an alternate way to transfer objects in ORB environments by using either the move or the copy operation from the *Object LifeCycle Service*. The main difference between these services is that the Externalization Service breaks the task of copying an object into two parts and is therefore more flexible than the copy operation of the LifeCycle service: First, the data is flattened into a stream by calling the *externalize_to_stream* operation on the target object. After the stream object has been transferred across the ORB, a new object is built from it by invoking the *internalize_from_stream* method of the stream object. The main advantage of splitting the copy operation in two parts is that using this feature, any object can be exported and imported from the ORB to other systems and vice-versa.

The conclusion to be drawn is that it is possible to implement Management by Delegation in CORBA-based environments. Following this concept, one can combine the advantages of both the service-based and the management-by-delegation approach: While CORBA assures the availability of uniform service interfaces and transparent access to their functionality, scripting languages as Java guarantee portability of the service *implementations*. As OMG is going to standardize an IDL/Java language mapping in 1996, it will be possible to implement delegatable distributed management services in a portable, object-oriented interpreted language.

5 Conclusions and Future Work

In the paper the suitability of CORBA for implementing intelligent agents has been analyzed. Together with its associated CORBAservices and CORBAfacilities, CORBA provides a good enabling technology for integrated distributed systems management but currently lacks of services providing powerful systems management functionality. It is therefore necessary to introduce this functionality in a way that it can be shared between managing and managed systems.

The approach described in this paper relies on defining management functionality in terms of *management services*; the set of all the services yields a *middleware for systems management* purposes. In CORBA-environments, this middleware is implemented in terms of distributed objects which can be accessed in a location-transparent fashion. It is even possible to reuse non-CORBA-conforming management functionality being part of already existing management architectures.

Issues of major importance for further research include:

- The definition of a base object class for defining management functionality in CORBA. Experiences from our work have shown that there is a strong necessity for defining a common base of minimum functionality guaranteed for all managed systems. Examples of such information about the class and its instances are the description of relationships to other object classes or the location of an object instance.

- The implementation of a CORBA-based intelligent agent prototype; it is then possible to examine the behaviour and the performance of intelligent agents.

The recent emerging of services allowing to distribute copies of objects across an ORB provides the enabling technology for CORBA-based intelligent agents. As implementations of these services (and sometimes even CORBA 2.0-compliant interoperability) are at the moment not contained in ORB development toolkits, the concepts remain currently rather theoretic. This situation may change in the near future: All major manufacturers have announced to make their ORB implementations fully conform to the OMG framework.

Acknowledgements

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 Munich University of Technology, and the Leibniz Supercomputing Center of the Bavarian Academy of Sciences.

References

[1] L. Conradie and M.-A. Mountzia. A Relational Model for Distributed Systems Monitoring using Flexible Agents. In *Proceedings of the Third International Workshop on Services in Distributed and Networked Environments*, Macau, June 1996. IEEE Computer Society Press.

[2] The Common Object Request Broker: Architecture and Specification. OMG Specification Revision 2.0, Object Management Group, July 1995.

[3] Network Management Forum. Omnipoint integration architecture. FORUM Technical Report TR114, 1995.

[4] G. Goldszmidt and Y. Yemini. Distributed management by delegation. In *Proceedings of the 15th International Conference on Distributed Computing Systems*, June 1995.

[5] German Goldszmidt. *Distributed Management by Delegation*. PhD thesis, Columbia University, 1996.

[6] J. Gosling and H. McGuilton. The java language environment. White paper, Sun Microsystems, Inc., October 1995.

[7] H.-G. Hegering, B. Neumair, and M. Gutschmidt. Cooperative Computing and Integrated System Management — A Critical Comparison of Architectural Approaches. *Journal of Network and Systems Management*, 2(3), October 1994. M. Malek (ed.), Plenum Publishing Corp., New York.

[8] Information Technology – Open Systems Interconnection – Systems Management – Part 16: Management Knowledge Management Function. DIS 10164-16.2, ISO/IEC, July 1995.

[9] Information Technology – Open Systems Interconnection – Systems Management – Management Functions. IS 10164-x, ISO/IEC, 1991-94.

[10] Information Processing Systems – Open Systems Interconnection – Basic Reference Model – Part 4: Management Framework. IS 7498-4, ISO/IEC, November 1989.

[11] Object Services Architecture. Document 95-1-47, Revision 8.1, Object Management Group, January 1995.

[12] CORBAservices: Common Object Services Specification, Volume 1. Omg specification, Object Management Group, March 1996.

[13] J. K. Ousterhout. *Tcl and the Tk Toolkit*. Addison-Wesley, 1990.

[14] Morris Sloman, editor. *Network and Distributed Systems Management*. Addison-Wesley, June 1994.

[15] Tivoli Management Environment 2.0: Technology Concepts and Facilities. Technology white paper, Tivoli Systems Inc., 1994.

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

[17] Y. Yemini, G. Goldszmidt, and S. Yemini. Network management by delegation. In I. Krishnan and W. Zimmer, editors, *2nd International Symposium on Integrated Network Management*. Elsevier Science Publishers B. V. (North Holland), April 1991.