Poseidon House
Castle Park
Cambridge CB3 0RD
United Kingdom

TELEPHONE: **Cambridge (01223) 515010**
INTERNATIONAL: **+44 1223 515010**
FAX: **+44 1223 359779**
E-MAIL: **apm@ansa.co.uk**

ANSA Phase III

Management in Object-Based Federated Distributed Systems

Yigal Hoffner

Abstract

This paper examines the philosophy of management in object-based federated systems. The problems introduced by distribution which concern management in object-based federated systems are presented. The approach taken to deal with some of the problems is then outlined.

APM.1018.01 **Approved** 25th October 1994

Architecture Report

Distribution:
Supersedes:
Superseded by:

Management in Object-Based Federated Distributed Systems

Management in Object-Based Federated Distributed Systems

Yigal Hoffner

APM.1018.01

25th October 1994

The material in this Report has been developed as part of the ANSA Architecture for Open Distributed Systems. ANSA is a collaborative initiative, managed by Architecture Projects Management Limited on behalf of the companies sponsoring the ANSA Workprogramme.

The ANSA initiative is open to all companies and organisations. Further information on the ANSA Workprogramme, the material in this report, and on other reports can be obtained from the address below.

The authors acknowledge the help and assistance of their colleagues, in sponsoring companies and the ANSA team in Cambridge in the preparation of this report.

Architecture Projects Management Limited

Poseidon House
Castle Park
CAMBRIDGE
CB3 0RD
United Kingdom

TELEPHONE UK	(01223) 515010
INTERNATIONAL	+44 1223 515010
FAX	+44 1223 359779
E-MAIL	apm@ansa.co.uk

Contents

1 Introduction

1.1 Audience, scope and purpose

This document proposes a model of management for federated distributed systems. It is addressed to the designers of management infrastructures for distributed systems.

The document focuses on the means used to discover and negotiate monitoring interfaces. It is not concerned with the management functions provided at those interfaces: this is a separate topic well-addressed in other places (e.g. OSI management standards).

The approach described requires that the underlying distributed systems infrastructure can represent managed entities as encapsulated objects and can transmit references to object interfaces as parameters of management operations. Such a capability is the basis of the ISO Basic Reference Model for Open Distributed Processing [X.900 92], the OMG Combined Object Request Broker Specification [OMG 91] and the ANSA architecture [AR.001 93].

1.2 Context

This document should be read in conjunction with “AR.010: Monitoring in Distributed Systems” [AR.010 93] and “TR.41: The Visualization of Distributed Systems” [TR.41 93]. The documents are related to each other in the following fashion:

- this document explains the philosophy and general approach to management in object-based federated distributed systems
- AR.010 explains and develops a model of monitoring and its management. It uses the model presented in TR.mgt in order to construct a model of the management of monitoring. The requirements which the visualization of distributed systems impose on the monitoring infrastructure, and which are outlined in TR.41 are also used
- TR.41 explains the problems with visualizing distributed systems and discusses the requirements such a process poses to the monitoring infrastructure.

1.3 Relation to the enterprise projection and model

The enterprise projection and model [TR.038 93] address management in architectural terms. This document addresses the technical issues involved with management and discusses the mechanisms and procedures necessary to facilitate it.

1.4 Overview

The management of a system can be decomposed into three generic sub-processes: control, monitoring and decision making. The model can be applied recursively to its own sub-processes, where the sub-process itself is complex.

The decision making process can be decomposed into a policy and a policy-based decision making process. Moreover, the decision making process can be decomposed into sequences and hierarchies of decision making sub-. processes, going from the general to the more specific. Such decomposition facilitates the automation of certain levels of decision making, and also allows the construction of policy driven systems.

The problems introduced by distribution which concern management in object-based federated systems are presented and discussed. A federated system arises when two or more autonomous systems are interconnected whilst retaining their individual control. The philosophy of management in object-based federated systems is examined in the context of these problems. In this situation, where each system in the federation retains individual control, gaining access to management interfaces must be by discovery and negotiation, as prior agreement may not always be possible. Finally, some conclusions concerning management within the ANSA architecture are drawn.

2 A generic model of management

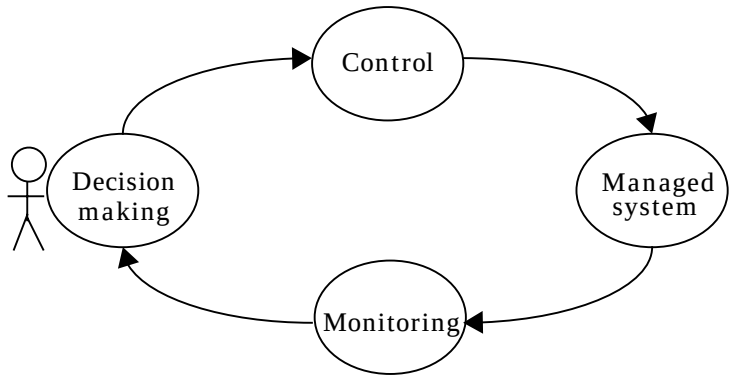
Implied in most, if not all discussions of management, are two fundamental notions:

- that management is a continuous activity
- that the process of management is a closed loop activity [WIENER 75].

These notions necessitate three generic management functions (Figure 2.1):

- *monitoring* - the process of observing the managed system
- *control* - the process by which changes in the managed system are effected
- *decision making* - the process of determining what changes should be effected in the managed system.

Figure 2.1: The generic model of management: monitoring, control and decision making



Where the processes of decision making, control or monitoring are complex, and therefore require management, the model can be applied recursively to parts of itself. The application of the management model to the monitoring process aimed at developing the management of monitoring is presented in [AR.010 93] and in [HOFFNER 93].

2.1 The monitoring process

The monitoring process serves management by providing information about the history of the system. It is this information which enables the decision making process to determine how to act in order to narrow the gap between the actual and desired state of the system.

If the monitoring is to be of use to the decision making process, then it must be capable of providing directly or indirectly the information which allows the comparison between the actual and desired state of the system. Thus, what should be monitored is determined by the management policy.

2.2 The control process

The control process provides management with the facilities which effect the required changes in the system. The control facilities will have an affect on the range of policies which can be carried out in a managed system as will be explained later.

2.3 The decision making process

The activity of decision making closes the loop of the management activity and ties the processes of monitoring and control together. This is usually done by a sequence of steps:

- comparison of observed state of system with the desired state
- reaching a decision based on this comparison, the management policy and the model of the managed system
- effecting the changes decided upon.

3 Policies and management

3.1 What is a policy

Literature on the subject of management [SLOMAN 89], [MOFFETT 92a] and [MOFFETT 92b] discusses the relation between notions related to policy such as:

- directing, managing, and executing
- policy and plan
- strategy and tactics.

There is general agreement that policies involve a notion of direction or aim, a plan of how to achieve the aim, and some statement about the resources and constraints within which the plan can be carried out. More specifically a policy will have [TR.038 93]:

- objective - the definition of a future state of affairs desired and believed to be feasible
- missions - a set of statements which define a set of actions intended to achieve the objective
- a set of resources and constraints.

3.2 Policy, plans and models of the managed system

An important aspect of policy is that it either contains a plan, or that it is possible to derive from it a plan of action concerning what to do about the difference between the desired and actual state of the managed entity. In order to close the gap between the desired and actual state of the managed entity, the policy must have some model of the managed system, either in an implicit or explicit form. For example, the model could be expressed as a stimuli-response behaviouristic model [GREENE 86], or as a transfer function [ATKINSON 68].

An important conclusion drawn from the relation between policy, plan and the model of the managed system is that: **embedded in every policy, either in an explicit or implicit form, is a model of the managed system on which the policy is based**.

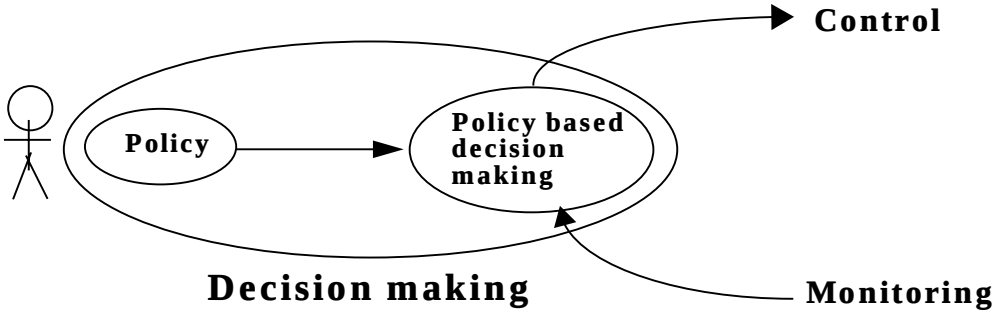
3.3 Policy and measurability

The ability to compare the desired and actual state of the system implies the ability to measure the state of the system in the same terms as that of the policy. This places a requirement that the policy should be stated in an implementable form.

3.4 Policy and decision making

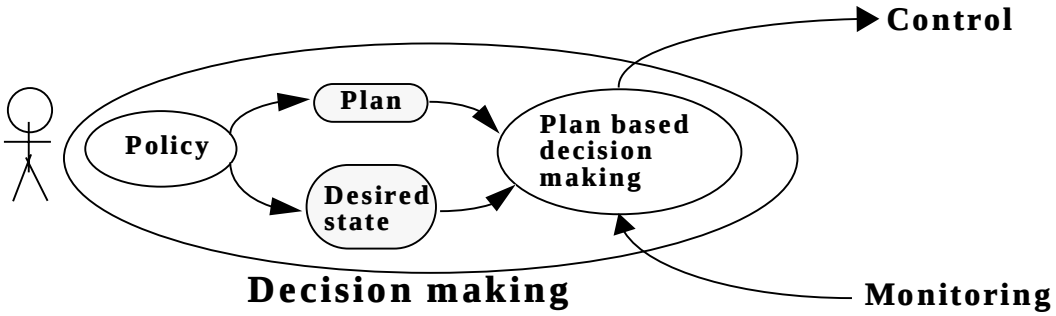
If a measurement of the difference between a desired and actual state of a system is possible, and a plan can be derived from the policy, then it is usually possible to divide the decision making process into a policy and a policy driven decision making process (Figure 3.1).

Figure 3.1: Policy and policy-based decision making



More precisely, a plan derived from a policy should provide the decision making mechanism with an indication of the desired state of affairs and the actions to be taken when this differs from the actual state of affairs (Figure 3.2).

Figure 3.2: Policy, plans and decision making



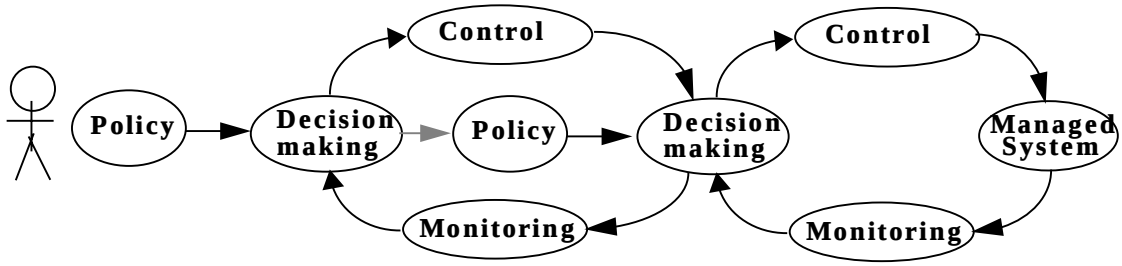
The changes which can be effected in a managed system depend not only on the constraints imposed by the policy but also on resources, including the control facilities. This has to be taken into account when ensuring the fitness of policies to the systems in which they are applied.

3.5 Decomposing the decision making process

The transition from the general to the specific, mentioned earlier as “policy and plan“, “strategy and tactics“, “directing, managing, and executing“, manifests itself in the ability to use the plans at one level of management as the objectives of another. It is in this manner that the process of decision making can be decomposed by breaking it down into sub-processes going from the general to the particular (Figure 3.3).

Although the decomposition shown in Figure 3.3 is linear, this may not always be the case. Complex hierarchical structures are common, sometimes with peer interaction as well. Often, informal channels of communication and control exist as well [HERBERT 89].

Figure 3.3: Decomposing the decision making process

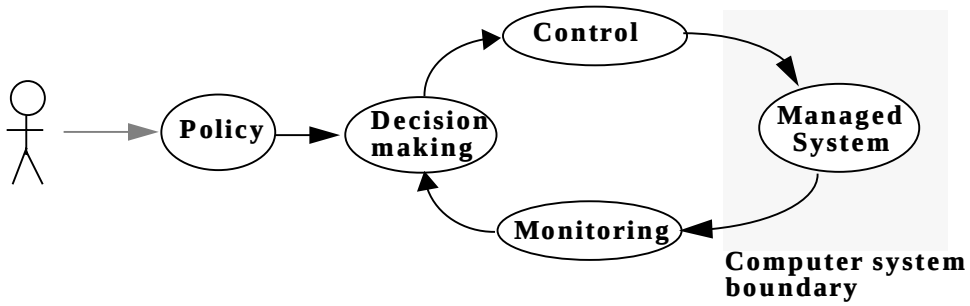


3.6 Incorporating management with the managed system

It is often the case, that the lower level decision making processes can be expressed in more concrete terms than high-level ones. It is therefore often possible to formalize the lower level decision making-processes, making them candidates for automation.

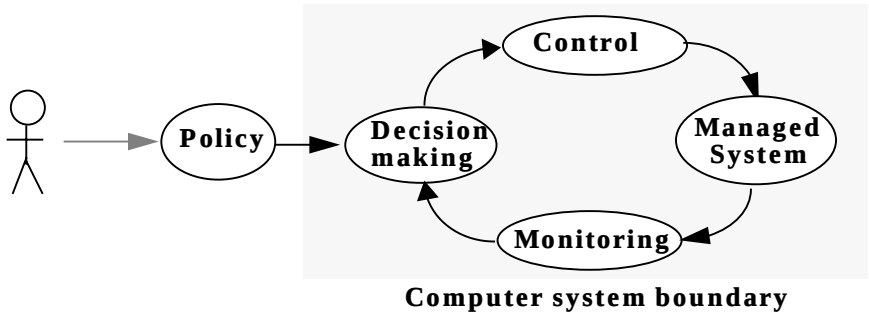
Figure 3.4 delineates the boundary separating the managed system from its the associated management.

Figure 3.4: Management and computer system boundaries



In practice, some of the process of decision making can be automated and embedded in the system itself. Thus the situation illustrated in Figure 3.4 is modified to encompass this situation as shown in Figure 3.5.

Figure 3.5: Automation of management



Automating the decision making process may be expedient for various reasons:

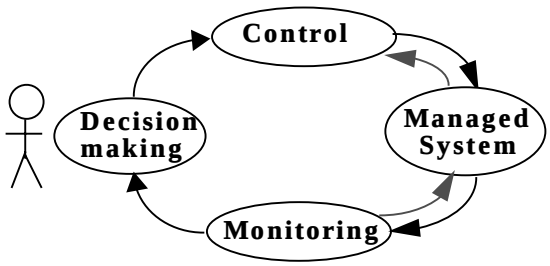
- *repetitive nature of management*: where the same policy and decision making process can be applied many times it is usually better to automate the process rather than rely on human intervention

- *speed of reaction*: where management decisions have to be taken within a time span smaller than the human capacity, allowing management to be carried out in a timely fashion
- *cost*: being able to automate certain activities may be cost effective in the long run
- *flexibility*: where embedded management can be designed and implemented so that it will be easy to modify existing policies or incorporate new policies in the automated decision making process.

3.7 Using the management facilities

If monitoring and control are incorporated into the system, the managed system can use the facilities to monitor its own behaviour and even to manage itself. For example, an application may wish to find out about the state of its own infrastructure by using query operations provided by the monitoring facilities. Alternatively, an application may wish to modify certain aspects of its own environment such as the length of its queues, its priorities or its resource allocation. This is shown in Figure 3.6.

Figure 3.6: The managed system using available management facilities

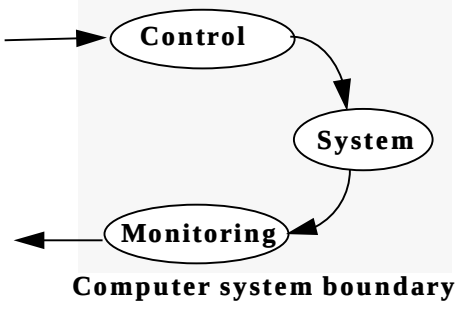


3.8 Policy driven systems

The incorporation of management with the managed system leads to the notion of policy driven system where decision making processes, and monitoring and control facilities can accommodate a variety of policies [TR.038 93]. From the discussion presented above it is possible to derive two different meanings to the notion of policy driven systems:

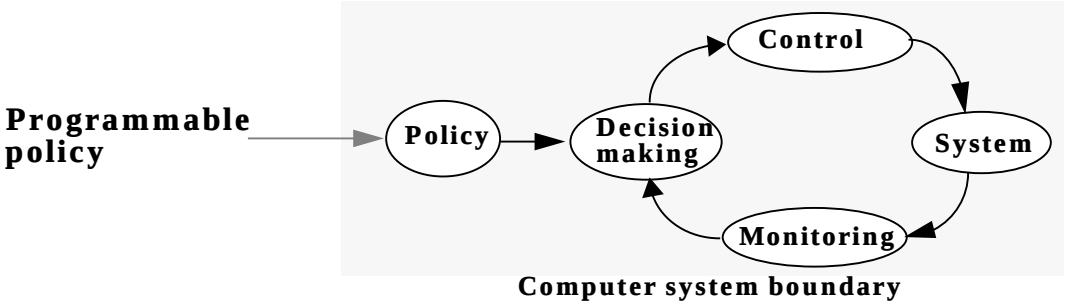
- where the monitoring-control loop is left open, to be closed outside the computer system (Figure 3.7)

Figure 3.7: Policy driven systems - leaving the loop open



- where the decision making process is automated but is made programmable to cater for a variety of policies (Figure 3.8).

Figure 3.8: Policy driven systems - automating the decision making with programmable policy



4 Distribution and management

Distribution makes many of the explicit and implicit assumptions behind the design and operation of centralized systems either invalid or incorrect [WARNE 91]. The following is an examination of how distribution affects management in the light of the main differences between centralized and distributed systems: autonomy, heterogeneity, physical separation and concurrency.

4.1 The autonomy of objects

In centralized system management, single management entity dictates to the rest of the system, for example, when to start or abort activities.

In a distributed system, by contrast, functionality is distributed and as a consequence the management of that functionality will itself be distributed until objects ultimately control themselves. Objects can request management functions to be performed on others, but cannot force this to happen.

4.2 Autonomy and multiple authorities

A distributed system may consist of a heterogeneous collection of physically separate systems, of any scale, grouped into different administrations applying different management policies at different places. In such a situation, it would be impractical to impose either a world-wide management policy or a common management infrastructure. When managing a federated distributed system it will be necessary to accommodate and integrate different policies and various forms of local management infrastructures. Also, channels for querying policies and facilities for negotiating with policy enforcers and makers have to be made available in order to resolve policy conflicts.

4.3 Heterogeneous management facilities

Due to the heterogeneous nature of distributed systems, standard management facilities and procedures cannot be assumed across domain boundaries. Different monitoring and control facilities may exist in different management domains.

The problem of interworking of management infrastructures can be overcome through a prior agreement on common facilities and procedures. Alternatively, it can be overcome by incorporating the means which allow dynamic integration of the different local management facilities.

4.4 Lack of centralized observation, control and decision making point

With monitoring and the control facilities being distributed, no centralized observation or control point exist. This will introduce delays when obtaining monitoring information, and also when trying to effect changes consistently across a number of physically separate objects. To overcome these problems it is necessary to have:

1. facilities for obtaining a consistent view of the system. This has implications for the monitoring infrastructure in distributed systems
2. facilities for exercising control of the system in a consistent manner. This has implications for the control infrastructure in distributed systems.

The decision making process may itself be distributed and carried out at more than one place by more than one person. This may result in conflicting decisions or in attempts to exert control in an un-synchronized manner. Procedures and channels for resolving such cases will be necessary.

4.5 Management and instability

Stability is a characteristic of a process or system whose response to an external or internal disturbance dies down when the stimulus is removed. Alternatively, a process is unstable if its response to a disturbance continues after the disturbance is removed. The use of feedback to manage a system may cause problems as feedback loops are subject to instabilities. In a computer system, for example, if unnecessary long delays are involved in the decision-making process and in the application of remedial action, and if excessive remedial action is taken, there is a fair probability that instability will arise.

In a distributed system, delays may be the result of differences associated with obtaining monitoring information, ordering it and ensuring a consistent view of the system, as well as the time required to synchronize control activities.

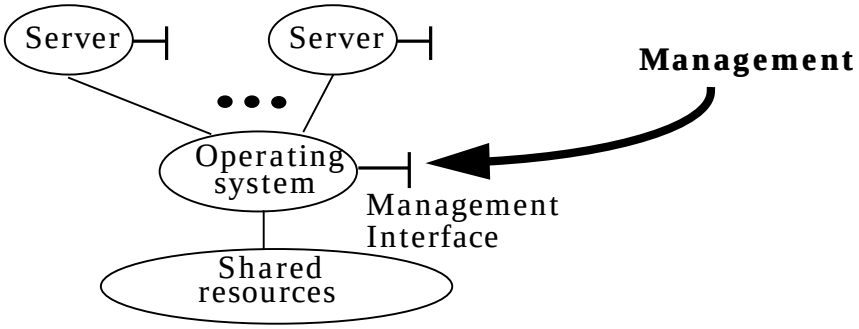
4.6 Complex failure modes

Partial failures, a characteristic of distributed systems, may lead to situations where some but not all of the managed objects in a system can be accessed. Moreover, some of the management infrastructure itself may fail. Fault tolerant techniques may need to be applied to the management facilities themselves in order to make them more resilient.

4.7 Resource allocation

Traditional centralized system management is based on keeping control of shared resources through the operating system (Figure 4.1). In this approach the operating system is in complete control of allocation and revocation of resources. For example, dictating to applications when to start or abort activities, when to relinquish storage facilities, etc.

Figure 4.1: Traditional approach to management in centralized systems



4.8 Replication, passivation and migration

Distribution, as well as posing many problems also allows the exploitation of replication, passivation and migration of objects, for example. Because of the extra requirements placed on the management of such structures, special management facilities must be developed to deal with them.

4.9 Security and management

There are several aspects to the relationship between management and security in order to conduct secure management in a distributed system [AR.008 93]:

- security facilities will have to be used by the management infrastructure. Thus, authentication facilities will be required, for example
- the security facilities themselves will require management
- management and security may conflict. For example, the need to monitor objects (violates the principle of encapsulation) and may in some cases contradict security requirements.

The principle of devolving management responsibility to objects can also be applied to security and the management of security.

4.10 Locality of computation and state

In centralized system the locality of computation and data is assumed so that usually it is not necessary to provide additional information in a sub-routine call about the identity of the caller. This is based on compile-time, link-time or run-time binding to local libraries which in most case do not carry state between computations or invocations from different clients.

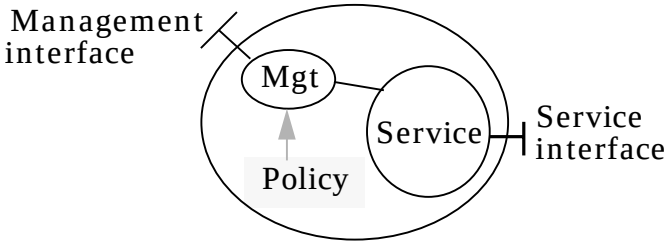
In a distributed object-based environment this assumption does not hold. Objects which maintain state from previous invocations can be shared. Also, the identity of the invoker cannot be assumed to be known as in the centralized case and therefore information concerning the client has to be obtainable where necessary.

5 Managing in object-based federated distributed systems

5.1 Management interfaces

The principles of encapsulation and autonomy of objects means that each object will have its own management service and an interface to it (Figure 5.1).

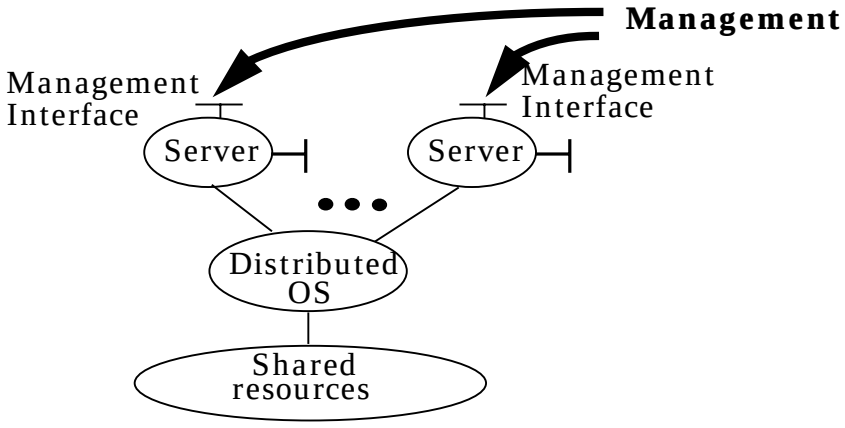
Figure 5.1: Objects have their own management service and an interface to it



5.2 Exercising management

The object-based management approach means that management can only be requested through management interfaces (Figure 5.2). The distributed operating system can refuse to allocate resources to an object but it cannot re-allocate them without agreement from the object.

Figure 5.2: Client-server model of management: each object is responsible for their own management



5.3 Indirection and access to management

Providing a management interface containing management operations in each object means that facilities are necessary to obtain a reference to the management interface.

5.3.1 Getting hold of management interfaces

In a distributed system, a single name space can not be assumed, and it may not always be possible to know in advance the management interface of an object. Access to the object's management facilities will have to be indirect, that is obtained from what is already known about the object. This will usually be a reference to one of the service interfaces provided by the object. Thus, there is a need to be able to associate different interfaces of the same object to one another. This would allow, given a service interface to an object, to obtain the management interface of the same object. Including information which associates different interfaces to each other when trading [AR.005 93], [OMG 91] is unlikely to scale, particularly in cases where objects may provide large numbers of interfaces. In addition, using trading for searching for management interfaces does not solve the problem entirely since not all interfaces are traded.

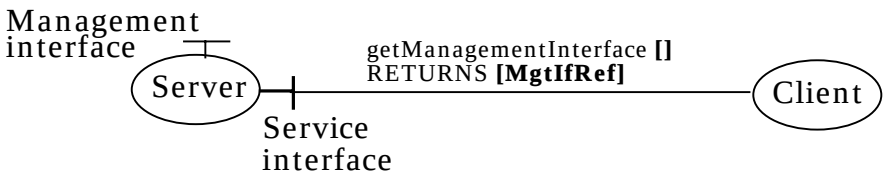
5.3.2 The getManagementInterface operation

A possible solution is to have every interface in the system contain an operation that, when invoked, returns the references to the associated management interface [AR.010 93] (Figure 5.3). In practice this means that any service interface of an object, will in addition to its service operations provide an agreed operation. This operation can be used to obtain a reference to the object's management interface, and will be of the form:

getManagementInterface [] RETURN [MgtIfRef]

This scheme requires agreement either through standards or by mutual consent. It cannot, however, be universally guaranteed

Figure 5.3: Gaining access to management interfaces using the getManagementInterface[] operation



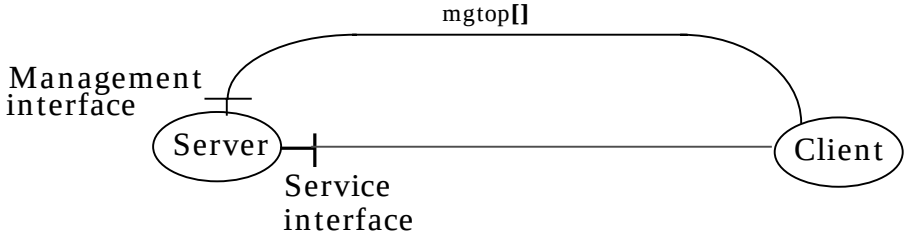
Having obtained the management interface, the client can now invoke the management operations of the server (Figure 5.4).

The main advantage of this solution is that it scales well and allows each object the discretion of whether to respond to any management requests or not.

5.3.3 The getManagementInterface operation and the type system

This scheme has implications concerning the type system. It raises the question of the type of interface to which an interface reference is given: does the client always have to know the type of the service in advance, or is it

Figure 5.4: Invoking management operation having obtained access to the management interface



possible just to provide a query operation in the given interface to find out what operations are available.

5.3.4 The getManagementInterface operation and security

In general, a service with a reference to any interface of an object will be able, by using this operation, to request the object to provide it with a reference to its management interface. A refusal to return references to management interfaces is a valid response. Furthermore, handing out the management interface does not guarantee that the object will have to perform the management operation invoked on that interface. This is a policy issue.

For example, if object A holds a reference to the management interface of object B, it can request object B to perform management operations on itself. Object A can not force object B to do anything Object B does not want to. Self management includes the right to pass the responsibility for management to another entity. It will be at the discretion of the managed object to accept or reject requests for performing management operations on it. The decisions will depend on the management policy of the object. Security aspects such as establishing the authenticity of the requester will be necessary. Information necessary to determine whether the requester is authorized to request management operations to be activated on the requesters behalf, will also be required.

5.3.5 Finding out about the client of a server

In a distributed object-based environment it is sometimes necessary for a server to be able to find out who its clients are. This is necessary for purposes such as accounting, security, auditing, etc. The problem of gaining access to the management interface of a client of a server (§5.3.6) can be regarded as a special case of this problem.

5.3.6 Finding out about the management interface of a server's client

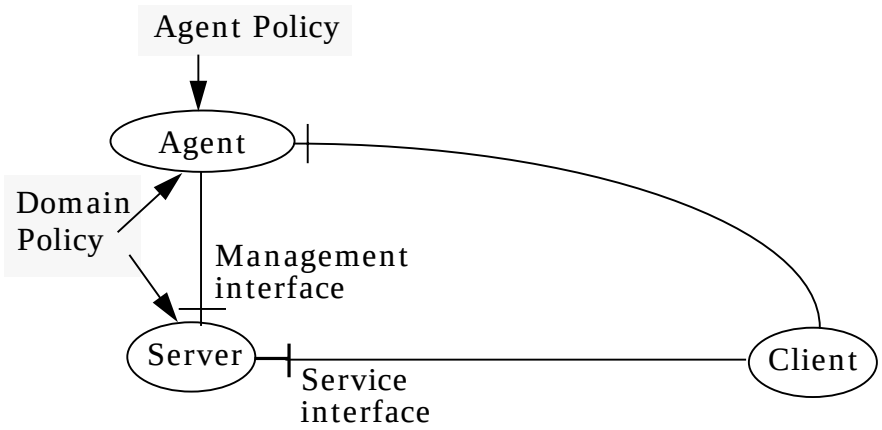
One problem with this solution is that it is necessary to have access to one of the object's service interfaces. However, there may be clients in the system who do not create any interfaces, other than their management interfaces, and who do not export them to a Trader. Having gained access to a server in the system, we may find that it is being used by a client. Gaining access to the client's management interface will constitute a problem unless we can provide the facilities for a server to find out about, or gain access to its clients. This can be done by:

- making the return path of the clients invocation through an interface reference. This requires further investigation of the computational model and the notion of symmetry in invocation send and receive phases
- providing the facilities in the engineering infrastructure for a server to find out about, or access its clients.

5.4 Using indirection to construct management domains

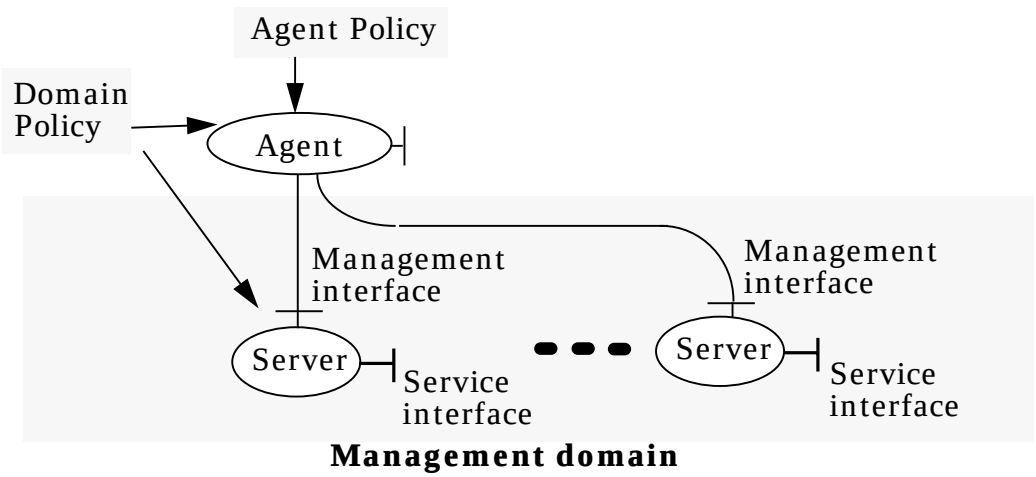
The getManagementInterface[] scheme can be generalized so that the result from its invocation may be an interface to some other object. Such an agent can be made responsible for the management of all the object in a management domain (Figure 5.5).

Figure 5.5: Access to management interfaces can be in-direct through agents



An agent can be provided with the management interfaces of several objects which share a common policy, so that by forcing the access to the management interfaces of the individual objects through the agent, the agent can be made responsible for ensuring that the policy will be applied to all the objects (Figure 5.6). This is a way of creating management domains, where a **domain** consists of a set of objects which have some attribute in common or which are managed by the same management agent [SLOMAN 89].

Figure 5.6: Using indirection to construct management domains



5.5 Management across domain boundaries

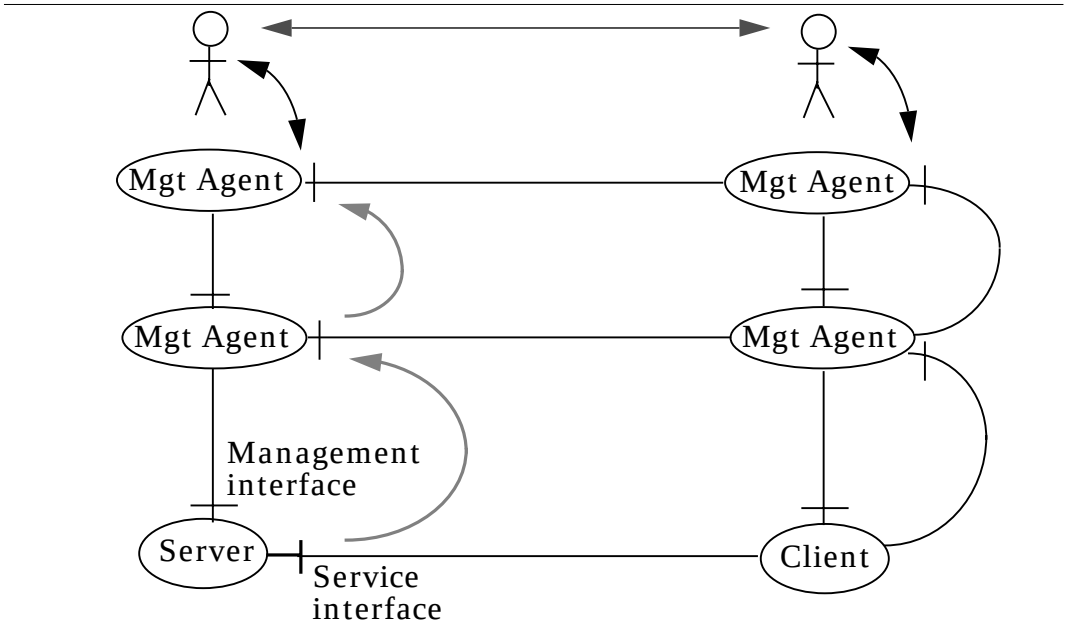
Large distributed systems will inevitably consist of multiple domains. **Boundaries** delineate domains and allow the necessary transformations to take place in order to allow activities to cross from one domain to another. Another way of looking at boundaries is as a point where differences between domains have to be dealt with explicitly, either by prior agreement or by providing the necessary mechanisms to overcome the differences. Agreement

The differences between domains can be characterized as technical, managerial, organizational, economic, contractual, social, political, etc. [AR.005 93]. Examples of technical differences are: naming, type system, data representation and interconnection differences.

5.5.1 Resolving policy conflicts

Resolving policy conflicts between domains requires a dialogue between the client and its management, and the management of the server. If the management facility in the server cannot resolve the clash, the getManagementInterface[] operation can be used to request access to successively higher levels of an object's agents, until the appropriate level for the resolution of a policy clash can be achieved (Figure 5.7). This may also require the client requesting it's own management agents to conduct the dialogue at the appropriate level.

Figure 5.7: Management dialogues - policy conflict resolution through successive layers of management agents



5.5.2 Integration of management facilities

Not all systems will have a coherent or integrated management functionality, due to heterogeneity and evolution. Where autonomous systems will be connected there will be a need to integrate some management functions. An approach to the integration problem is to provide agreed management interface operations which return a description of the available management functions. This scheme can be extended to facilitate the integration of

application specific management facilities with generic distributed infrastructure facilities.

The integration and use of management facilities in a distributed system is necessary at different levels [AR.010 93]:

- use of existing management services which may not have been implemented in an object based manner
- integration in an object based environment:

1. between the application specific management as dictated by the application programmer and the generic management facilities provided in the distributed infrastructure
2. between different services in the same domain
3. between services residing in different domains.

In federated systems, agreement concerning the following issues need to be addressed:

- what set of basic monitoring and control facilities should be available in each object
- what set of basic management facilities are necessary in order to manage monitoring and control facilities, e.g. how are the facilities queried and activated
- the set of procedures and mechanisms necessary in order to gain access to the monitoring and control facilities
- how is monitoring information to be collected.

5.6 Distribution of policy enforcement

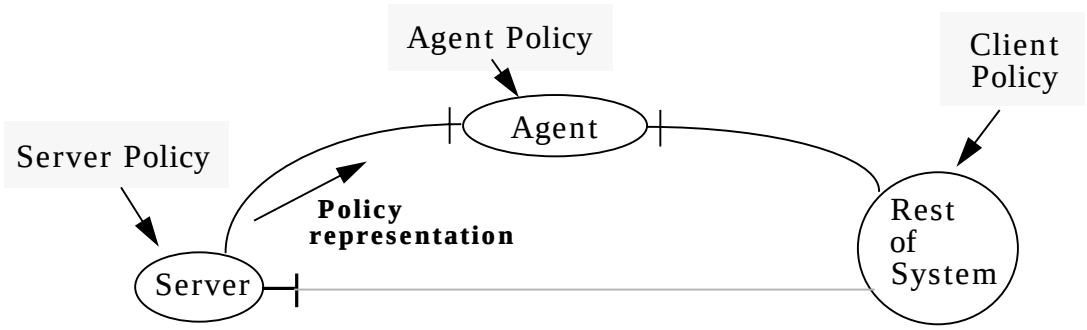
Ultimately, the responsibility for managing an object in a distributed system lies with the object itself. However, policy enforcement can be distributed in order to improve performance, scaling characteristics and protection.

The distribution of policy enforcement can be done by using agents (Figure 5.8). This requires the object to be able to send a representation of the constraints or requirements it wishes the agents to enforce on its behalf. The subject of policy representations is beyond the scope of this work.

An example of the use of this principle is trading [AR.005 93], [OMG 91]. Both the server and the client ask the Trader to act on their behalf in the process of match making. In order to do so they both provide the Trader with information concerning their policy with regard to service offer and usage. Other examples concern authentication servers, factories and node managers.

The policy can also be distributed to clients and adherence to it can be encouraged through law, policing, remuneration or good will.

Figure 5.8: Objects delegate responsibility to agents



6 Management and ANSA

6.1 Management and the computational model

The problem of obtaining access to management interfaces discussed in section 5.3 necessitates the development of new architectural concepts or the provision of appropriate facilities in the engineering infrastructure to allow a server to find out information about its clients.

Editorial: This topic requires further work.

6.2 Management and the engineering model

In order to provide the necessary management facilities in a distributed system, it will be necessary to develop the appropriate structures and to incorporate them into the distributed system infrastructure. This can be done in different ways depending on the distributed application development process and the support given by the run-time environment.

For example, in the present ANSA tool set <10pt>[AIM 93], the appropriate facilities will have to be incorporated through the use of PREPC, STUBC and the use of libraries, as well as the provision of run-time management agents.

References

[AIM 93]
ANSAware 4.1 Implementation Manual, APM., Poseidon House, Castle Park, Cambridge CB3 0RD, UK (1993).

[AR.001 93]
Rees, R.T.O., "The ANSA Computational Model", Architectural Report 001, Architecture Projects Management Ltd., Poseidon House, Castle Park, Cambridge CB3 0RD, UK (1993).

[AR.005 93]
Deschrevel, J. P. "The ANSA Model for Trading and Federation", Architectural Report 005, Architecture Projects Management Ltd., Poseidon House, Castle Park, Cambridge CB3 0RD, UK (1993).

[AR.008 93]
Rees, R.T.O., Bull, J., AR.008: "A Framework for Federating Secure Systems", Architectural Report 008, APM,Architecture Projects Management Ltd., Poseidon House, Castle Park, Cambridge CB3 0RD, UK (1993).

[AR.010 93]
Hoffner, Y. "AR.010.00: Monitoring in Object-Based Federated Distributed Systems", Architecture Projects Management Ltd, Posiedon House, Castle Park, Cambridge CB3 0RD, UK (January 1993).

[ATKINSON 68]
Atkinson, P. "Feedback Control Theory for Engineers", Heinemann Educational Books (1968).

[GREENE 86]
Greene, J. "Language Understanding: A Cognitive Approach", Open Guide to Psychology, Open University Press (1986).

[HERBERT 89]
Herbert, A. H. & Monk, J. M. Editors: "The ANSA Reference Manual, Release 00.03, The Advanced Networked Systems Archiecture Project, Poseidon House, Castle Park, Cambridge CB3 0RD, UK (1989).

[HOFFNER 93]
Hoffner, Y. "The Management of Monitoring in Object-Based Distributed Systems", To appear in the Proceedings of the Third International Symposium on Integrated Network Management, San Franscisco (April 1993).

[MOFFETT 92a]
Moffett, J. D. "Policy Conflict Analysis in Distributed System Management", Imperial College of Science and Technology, Department of Computing, 180 Queen's Gate, London SW7 2BZ, UK (1992).

[MOFFETT 92b]
Moffett, J. D. & Sloman M. S. "Policy Hierarchies for Distributed Systems Management", Imperial College of Science and Technology, Department of Computing, 180 Queen's Gate, London SW7 2BZ, UK (1992).

[OMG 91]
Object Management Group, "The Common Object Request Broker: Architecture and Specification", OMG Document 91.12.1 (1991).

[SLOMAN 89]
Sloman, M. "A Framework for Distributed System Management", Distributed Processing, ed. by Barton, M. H. et al. IFIP (1989).

[TR.038 93]
Iggulden, D. "Architecture and Frameworks", Technical Report 038, Architecture Projects Management Ltd., Poseidon House, Castle Park, Cambridge CB3 0RD, UK (1993).

[TR.41 93]
Hoffner, Y. "TR.41.00: The Visualization of Distributed Systems", Architecture Projects Management Ltd, Posiedon House, Castle Park, Cambridge CB3 0RD, UK (January 1993).

[X.900 92]
ISO 10734/CCITT X.900 Basic Reference Model for Open Distributed Processing Parts 1-5, ISO/IEC JTC1/SC21/WG7 (June 1992).

[WARNE 91]
Warne, J. P. " A System Designer's Introduction to the Architecture", Architecture Projects Management Ltd, Posiedon House, Castle Park, Cambridge CB3 0RD, UK (1991).

[WIENER 75]
Wiener, N. "Cybernetics: or Control and Communication in the Animal and the Machine", The Massachusetts Institute of Technology Press, Cambridge, Massachusetts, USA (1975).