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

Monitoring in Distributed Systems

Yigal Hoffner

Abstract

A general model of management is introduced and used in the development of a model of the management of monitoring for object-based federated distributed systems. This model is subsequently used to show how monitoring and its management can be implemented in such systems.

The information and structures necessary for conducting a monitoring session with multiple objects are presented. The problem of managing a monitoring session, where the set of objects under observation changes dynamically, is addressed. Finally, the problem of management across federation boundaries is discussed.

APM.1008.01 **Approved** 25th October 1994

Architecture Report

Distribution:
Supersedes:
Superseded by:

Monitoring in Distributed Systems

Monitoring in Distributed Systems

Yigal Hoffner

APM.1008.01

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 Abstract

A general model of management is introduced and used in the development of a model of the management of monitoring for object-based federated distributed systems. This model is subsequently used to show how monitoring and its management can be implemented in such systems.

The information and structures necessary for conducting a monitoring session with multiple objects are presented. The problem of managing a monitoring session, where the set of objects under observation changes dynamically, is addressed. Finally, the problem of management across federation boundaries is discussed.

1.2 Audience, scope and purpose

This document develops a model of monitoring and its management for object-based federated distributed systems. It is addressed to designers of distributed systems.

The approach described in this document 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].

1.3 Context

This document should be read in conjunction with [TR.39 93] and [TR.41 93]. The documents are related to each other as follows:

- TR.39 explains the philosophy and general approach to management in object-based federated distributed systems
- TR.41 explains the problems with visualizing distributed systems and discusses the requirements such a process poses to the monitoring infrastructure
- this document explains and develops a model of monitoring and its management. It uses the model presented in TR.39 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.

1.4 Overview

Monitoring is the process of obtaining, collecting, and presenting the information required by an observer about the observed system [JOYCE 87], [DOMAINS 92], [MCDOWELL 89], [SAMANI 92], [SLOMAN 89a], [SLOMAN 89b], [LABARRE 91] and [WINTERBOTHAM 87].

Monitoring is always carried out with a purpose in mind. The general aim is to obtain information in order to construct a model of system behaviour or to modify an existing model. The general activity of monitoring a system can be specialized to a particular purpose such as accounting, debugging or testing, among others. The specialization of monitoring to the different purposes determines the type and the way in which information collected.

Distribution and more specifically, dealing with issues such as heterogeneity, autonomy, physical separation and concurrency, complicates the process of monitoring. The design and development of monitoring facilities needs to deal with these problems.

The object-based approach to building distributed systems requires the designers to incorporate monitoring and management facilities in each object. This paper is an investigation of the facilities which should be included in each object if monitoring is to be viable in distributed systems.

Monitoring often involves correlating concurrent events at multiple objects. Management structures are, therefore, necessary to maintain information on the objects participating in a monitoring session, and manage the monitoring facilities in each of them. In addition, appropriate structures for collecting monitoring information are required.

In a distributed system, a monitoring session can evolve dynamically, as activities related to an application spread throughout the system. The management of a monitoring session must therefore be able to extend and contract the set of objects under observation. The distributed system infrastructure must allow access to management functions of an object given a reference to one of the object's service interfaces.

When monitoring across federation boundaries, differences in monitoring and management facilities must be accommodated either by prior agreement to provide common facilities, or by supplying the appropriate translators which allow interworking. There is also a need to provide channels and mechanisms for resolving policy conflicts across federation boundaries. The integration of monitoring facilities of different systems is an important part of the problem of monitoring across boundaries.

2 Monitoring

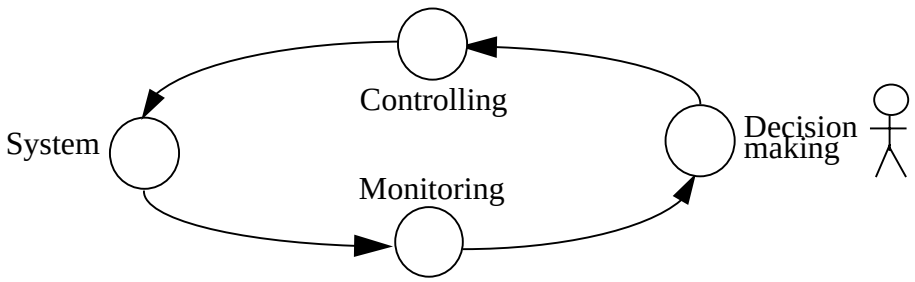
2.1 The purpose of monitoring

Monitoring is carried out in order to obtain information about a system, and in general, monitoring is part of the process of management (Figure 2.1). Among the many activities which involve monitoring we find:

- debugging
- testing
- accounting
- performance evaluation
- resource utilisation analysis
- security
- fault detection
- teaching aid.

Monitoring and its **management** are concerned with providing the necessary information in order to allow the construction of the required model of the observed system and its presentation. It is the purpose of monitoring which dictates what should be observed and also how the information is to be obtained.

Figure 2.1: The relationship between management and monitoring



2.2 Modelling and the level of monitoring

The different purposes for which monitoring is carried out can be executed at different levels. Thus, for example, debugging a single object as opposed to debugging the interactions among multiple objects will require different events to be observed. A language debugger will require events to be generated at a smaller level of granularity than that which is aimed at debugging the interactions between objects.

Some of the models constructed for the purposes listed in §2.1 will require a different model or models of the distributed system. The exact level of

modelling will dictate the granularity of the events the observer wishes to monitor.

2.3 The problems of monitoring

The following is an exposition of the problems encountered when monitoring centralized and distributed computer systems.

2.3.1 Direct and indirect observations

The behaviour of some systems can be directly observed, thereby making the process of monitoring relatively straight forward. In computer systems most events of interest cannot be observed directly without special facilities, thereby requiring the incorporation of a monitoring infrastructure in such systems. The monitoring infrastructure will also facilitate the management of monitoring.

There may be several levels of indirection between the observed system and the observer. Indirection may:

- make changes to the observations which are not related to the behaviour of the observed system, for example, change the order of the messages sent to the observer
- affect the reliability of the observations
- introduce distance between observer and observed system and hence lack of trust
- directly influence the behaviour of the observed system (interference).

Distribution complicates the process of monitoring, introducing additional levels of indirection and subsequently additional problems. These are discussed in Chapter 3.

2.3.2 Complete and incomplete observations

Completeness and incompleteness refer to whether the information necessary in order to construct a particular model of an observed system is available or not.

It could be argued that any observation of a system only reveals part of the system. This is not a problem when the observer is constructing a particular model of the system and the observation fits this model. However, incompleteness can cause problems when it is not intended or not catered for [TR.41 93]. For example:

- when information cannot be obtained thereby hiding certain aspects of the system from the observer
- when hidden information makes some of the available information non-interpretable by creating the wrong context for its interpretation.

In reality, the two problems may stem from the same cause.

2.3.3 Presentation problems

In many cases it is necessary to modify the information from the observed events in a system in order to overcome the following problems (Figure 2.2):

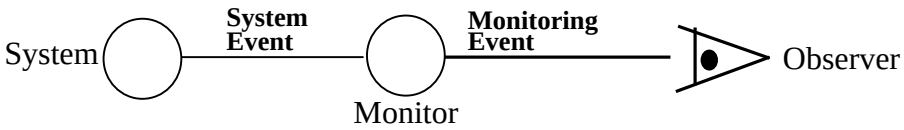
- observed events appear in a **form** which is not amenable for immediate use by the observer
- observed events occur at a **rate** which cannot be easily used by the observer
- the **volume** of observed events may be such that it overwhelms the observer
- in a system in which has no central point of observation, events of interest may occur at different parts of the system. Structures and processes which collect and order the information from the observed events are therefore necessary.

2.3.4 Monitoring and interference

Every system is affected by being monitored. The extent of the influence may or may not be negligible from the point of view of the user(s) or the observer(s) of the system. There is a relation between the flexibility of the monitoring facilities, the cost of implementation, and the extent to which they interfere with the behaviour of the system.

The most general requirement from monitoring, which is independent of the purpose for which it is introduced, is that although the sequence of system events may change as a result of the interference caused by monitoring, it must not result in an illegal sequence of events taking place.

Figure 2.2: Monitoring: transforming the information from the system events



3 Distribution and monitoring

3.1 Introduction

This chapter discusses aspects of **distribution** which affect monitoring: physical separation, concurrency, heterogeneity, federation, scaling and evolution. A summary of the assumptions which are no longer valid when monitoring distributed systems, as opposed to centralised systems, is presented. Some general comments are made about the philosophy of management in an object-oriented distributed system. The chapter also identifies the three major problem areas of providing monitoring in distributed systems: **management** of monitoring, **reconstruction** of the causal flow of events, and **presentation** of monitoring information.

3.2 Aspects of distribution

The following sections discuss the aspects of distribution which have an effect on monitoring: physical separation, concurrency, heterogeneity, federation, scaling and evolution [WARNE 91].

3.2.1 Physical separation

In a distributed system the physical separation of objects is unavoidable. In addition, communication delays among objects are usually variable and unpredictable. As a result there is no single point of reference from which events in the entire system can be directly observed. In order to obtain a global view of the system it is necessary to collect information on local events from several locations, from which a reconstruction of the flow of global events can be made. For example, to determine whether a certain event at one location is causally related to another event at some other location.

There are situations in which it is not possible to monitor events in certain parts of the system. This may be the result of the absence of monitoring facilities, or policy decisions imposed on an object. There are two additional complications in distributed systems:

- failures can occur during communication
- services may partially fail.

Such failures may affect not only the activities being monitored, but also the monitoring of these activities, resulting in incomplete information.

This complicates the reconstruction of the flow of events in the system, and results in an incomplete picture of the system. This problem is addressed in more detail in [TR.41 93].

Distributed systems are characterized by the possibility of partial failures. Partial failures 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 therefore have to

be applied to the management facilities themselves in order to make them more resilient to failure.

The physical separation of systems together with the variable communication delays also means that there is no single point of control in a distributed system. This together with the absence of a single point of observation means that checkpoints, tracing, breakpoints and single stepping of a distributed application are difficult, if not impossible without changing the nature of the system.

Figure 2.1 in Chapter 2 shows the relationship between a system, its monitor and controller, and the decision making process. If the system is distributed, the absence of a single point of control and the absence of a single point of observation implies that the monitor and controller must be distributed as well. Furthermore, in some systems the decision making process may either be distributed and/or have to be carried out in the face of incomplete information

3.2.2 Concurrency

Distributed systems will support multiple objects and activities. Bindings between objects will be set up and discarded, and objects will be able to invoke other objects asynchronously through these bindings. Furthermore, objects will be created and destroyed as the need arises. The dynamic initiation and termination of activities will lead to situations where the activities stemming from an application may spread throughout the system. The extent of the initiated activities may not be known in advance.

From the point of view of monitoring this creates several difficulties. In order to gain sufficient understanding of the flow of events in a system it may not be enough to monitor a single object or simply its interactions with other objects. In fact we may wish to gain information on how activities spread in a system. Thus we may wish to:

- fully activate monitoring of objects with which a monitored object interacts
- follow the chain of activity as it moves from one object to another.

As different combinations of these strategies may occasionally be required, the management of the monitoring activities in such circumstances will be difficult unless extremely flexible management structures can be provided. Together with different monitoring activation strategies, additional event information to allow the observer to follow activities throughout the system is necessary.

3.2.3 Heterogeneity

Large scale distributed systems inevitably include some diversity in their hardware, operating systems and their distributed system infrastructure. It is reasonable to assume, therefore, that this diversity will be reflected in the implementations of monitoring facilities. Distribution does not only refer to the run-time physical separation of components, but also to the possibility of a distributed development environment. In such a case it is possible to have different implementations of monitoring which do not conform to one another.[1]

1. This may happen between heterogeneous systems but may also happen in a homogeneous environment.

In order to make possible monitoring across heterogeneous systems, it is necessary to reach agreement on monitoring conformance issues. These are discussed in Chapter 9.

Standard management facilities cannot be assumed across domain boundaries. Different monitoring and control facilities may exist in different management domains.

The problem of the integration of management infrastructures where different monitoring and control facilities may exist can be overcome through agreement on facilities or by the incorporation of facilities which allow dynamic integration of the different local management facilities

3.2.4 Federation

The existence of centralised ownership and universal and technical control in large scale distributed systems cannot be assumed, and separate sources of authority will inevitably reside side by side. In such systems a “federated” style of interworking will be necessary in which no participant is in control of the others. Each system controls its own services locally according to its policies. Different monitoring policies must be anticipated within federated systems and problems will arise when attempting to monitor across federation boundaries between systems whose monitoring policies clash. Cooperation between systems requires the parties responsible for them to negotiate the use of services either prior to the request for use of service or as a result of such a request.

Examples of possible areas where negotiation is needed are:

- where the authority allowed to request monitoring may differ
- where the collation and logging strategies may be different causing, for example, security compromise or unacceptable resource usage
- where granting access to monitoring management may be related in different systems to different conditions, e.g. system load, number of users, time of day, etc.

3.2.5 Scaling

As discussed in the section on concurrency, activities in a distributed system can spread and encompass large parts of the system. In cases where monitoring is expected to report on such activities, it is important to note that the monitoring activity itself will have to spread, thus consuming increasing storage, processing and communication resources. It is therefore essential that (the distribution of) monitoring itself scales well.

The requirement for scaling needs monitoring structures which can accommodate distribution, system evolution, and growth of the activity in the face of resource constraints and performance requirements. Both management and collation structures must be designed with scaling in mind; these issues are dealt with in Chapter 7 and Chapter 8.

3.2.6 Evolution

Distributed systems will evolve over time, possibly in an inconsistent manner. If monitoring procedures change over time, there may be clashes between monitoring standards embedded in new components and monitoring standards in existing components. The problems arising in evolving systems

are often similar in nature to those arising in heterogeneous and federated systems.

3.3 Problems and reversed assumptions

Certain problems associated with monitoring in a centralized system are exacerbated when dealing with distributed systems. However, problems also arise because of the reversal of many of the implicit assumptions made when monitoring centralized systems:

- **no central point of control:** not being able to directly control the entire system from any single point requires extensions to sequential techniques involving monitoring, which in a centralized system are based on the existence of a single thread of control. Examples of such techniques are break-points, single stepping and checkpoints
- **no central point of observation:** not being able to directly observe the system in its entirety from a single point of observation requires the collection of locally observed events in order to construct global views. However, this is more complicated than simply collecting the monitoring information. This is due to the fact that coupled with non-deterministic communication delays, collation of monitoring information cannot be based on the assumption that the order in which monitoring information is collected is related to the order in which they occurred. Furthermore, events of interest may consist of sequences of events which occur at different points of observation, thus requiring sequence recognition facilities
- **no central source of monitoring information:** a frequent implicit assumption in a centralized systems is that the source of the monitoring information is a single source, and that error and monitoring messages will be sent to directly to the user's terminal or to a local file. Neither of these assumptions holds in a distributed system. Collation strategies are necessary to cater for multiple sources and destinations
- **no central point of decision making:** the process of making decisions in a distributed system may itself be distributed. This may also be the case with the management of monitoring resulting in more than one manager in a monitoring session or an object participating in more than one monitoring session
- **incomplete observability:** in some cases it is not possible to observe certain parts of the system at all or only partially, resulting in incomplete information about events in that part of the system
- **non-determinism:** distributed, asynchronous systems are inherently non-deterministic. Thus, two executions of the same program may produce different, but nevertheless valid, ordering of events. This makes the reproduction of errors and the creation of certain test conditions difficult, if not impossible, at times (monitoring information can be used to reproduce test conditions if monitoring interference can be minimized sufficiently)
- **monitoring interference:** the dependencies between different processes in a distributed system are such that any change in the behaviour of one process can alter the behaviour of the entire system. The inclusion of

monitoring in a distributed system can alter the behaviour of a program in a manner which is important to the observer

- **replication:** in a distributed system an object may be implemented as a replicated group [AR.002 93]. When wishing to monitor such an object it is necessary to have the appropriate facilities to deal with both cases:
  - a group with replication transparency
  - a group without replication transparency
- **migration:** in a distributed system objects may migrate from one system to another. This will cause difficulties with the control and collation of the monitored objects. The appropriate monitoring and management facilities must deal with such cases
- **passivation:** objects may be passivated [**AR.006**]. It is necessary to decide what the meaning of monitoring a passivated object is and how to notify the monitoring session of such a case
- **objects, encapsulation and security**: One of the problems with monitoring in an object-oriented system is that the notion of monitoring is directly opposed to one of the fundamental characteristics of such systems, namely that of encapsulation.Ensuring that the state of objects and their associated procedures are protected from external observation and interference creates a conflict with the need to monitor those objects. For example, the incorporation and usage of monitoring facilities in an object may clash with security requirements.
- **objects administer their own management:** in contrast to centralized systems which are characterized by a central management entity, management facilities are distributed to the objects. This also applies to facilities for management of monitoring
- **monitoring as a distributed activity:** the monitoring of a distributed system is itself a **distributed** activity and it therefore requires:
  - tools which allow the management of the process of monitoring access and use to remote resources
  - that the monitoring services and their associated management structures do not **interfere** with the performance of the system to an unacceptable degree and that they **scale** well when active in large distributed systems
  - dynamic and selective **monitoring activation:** the ability to define the granularity of monitoring, activate it at run-time and modify it as the need arises without re-compilation. The granularity of monitoring is the level to which a single activity can be monitored in an object without having to activate the entire monitoring in the object
- **agents and roles**: in a distributed system: the assumption that the same agent in the same location may carry out several roles does not hold in distributed systems. For example, the application programmer, application user and the observer roles may be carried out by different agents in different locations
- **visualization and system models:** this concerns the need to present the data produced during a monitoring session to the user in an intelligible form, relating it to known models of the system. Distribution adds an extra level of complexity to intelligible presentation of monitoring

information. Special analysis and visualization tools are therefore essential.

3.4 Conclusions about monitoring in a distributed system

The problems cited above which distribution introduces to monitoring can be grouped together into three major problem areas:

- the definition, design and incorporation of a **monitoring and management infrastructure** to facilitate the dynamic monitoring of distributed systems
- **ordering and reconstruction** of the flow of events in a distributed system from the monitoring information: the transformation of a collection of monitoring information of local events into a global picture. The ability to reconstruct can be seen as a pre-requisite for providing useful presentations of monitoring information
- the **visualization** of monitoring information in order to provide the observer with useful models of the system and the activities in it.

4 Approach to monitoring in distributed systems

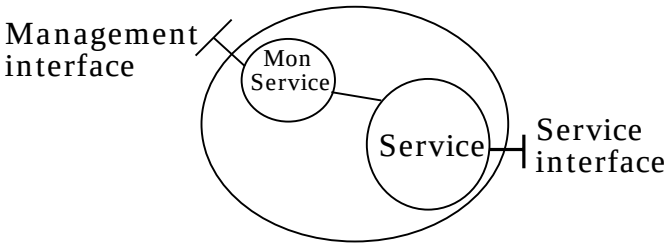
4.1 Introduction

This chapter presents the approach to monitoring and management in object-based federated distributed systems, based on the problems cited in Chapter 3. A more detailed description of the approach and the rationale behind it is given in [TR.39 93].

4.2 Monitoring and its management in object-based federated distributed systems

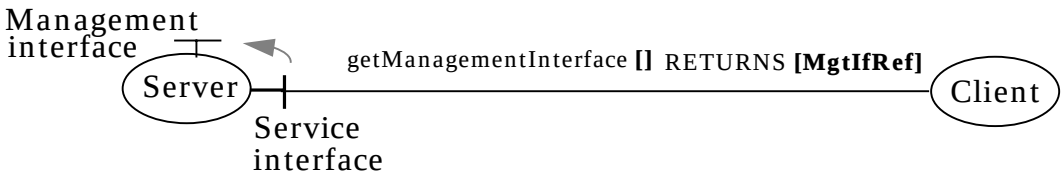
The principles of encapsulation and autonomy of objects means that each object will have its management service and an interface to it (Figure 4.1).

Figure 4.1: Objects have their own monitoring management service and an interface to it



Obtaining a reference to this interface can be done through trading [AR.005 93] [OMG 91], or by providing in each interface in the system an operation: `getManagementInterface [] RETURN [MgtIfRef],` which when invoked, returns the references to the object's management interface (Figure4.2) [TR.39 93].

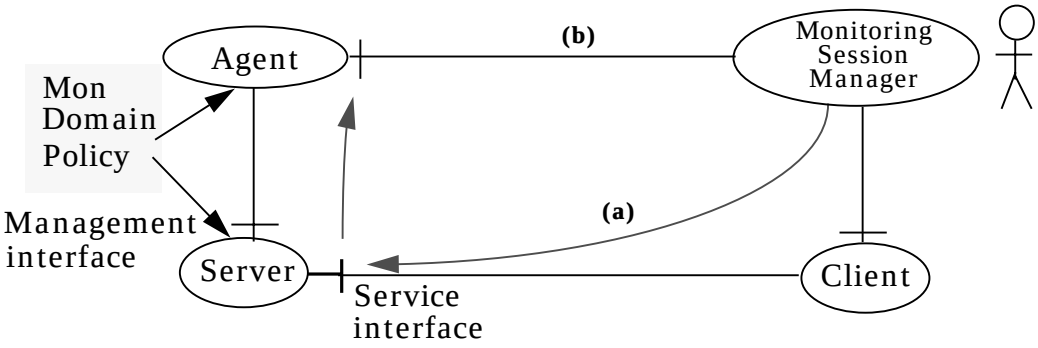
Figure 4.2: Gaining access to management interfaces



Having obtained the management interface, the manager can now invoke the management operations of the server's monitoring service.

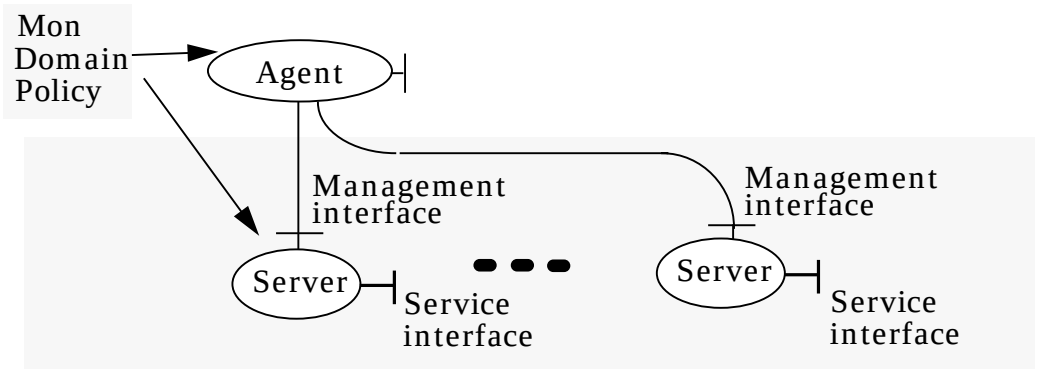
The `getManagementInterface[]` scheme can be used to return an interface to an agent which can be made responsible for the management of that object (Figure 4.3).

Figure 4.3: 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 restricting the access to the management interfaces of the individual objects except through the agent, the agent can be made responsible to ensure that the policy will be applied to all the objects (Figure 4.4). This is a way of creating management domains [SLOMAN 89b] [SLOMAN 89a].

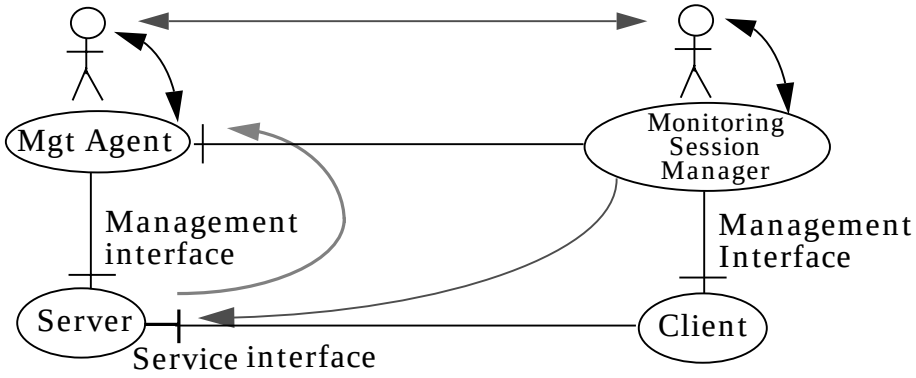
Figure 4.4: Using indirection to construct management domains



Resolving policy conflicts between domains requires a dialogue between the management agents. If the management facility in the server cannot resolve the clash, the client may request access to a management agent of the server. The `getManagementInterface[]` operation can be used to request access to successively higher levels of an object's agents, until the appropriate level for policy clash resolution can be achieved (Figure 4.5). This may also require the client requesting it's own management agents to conduct the dialogue at the appropriate level.

Due to problems of heterogeneity and evolution, not all systems will have a coherent or integrated management functionality. Where autonomous systems will be connected there will be a need to integrate some management functions. The 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. This is discussed in chapter 8.

Figure 4.5: Management dialogues - policy conflict resolution through successive layers of management agents



5 A model of monitoring and its management

5.1 Introduction

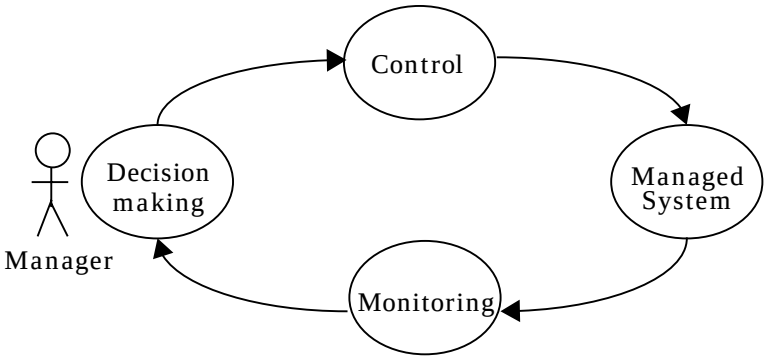
This chapter uses the approach to monitoring and management described in Chapter 4 to develop a model of the monitoring and its management.

5.2 The generic model of management

The generic model of management is introduced in [TR.39 93]. It links three generic management functions together (Figure 5.1):

- *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 5.1: The generic model of management - monitoring, decision making and control



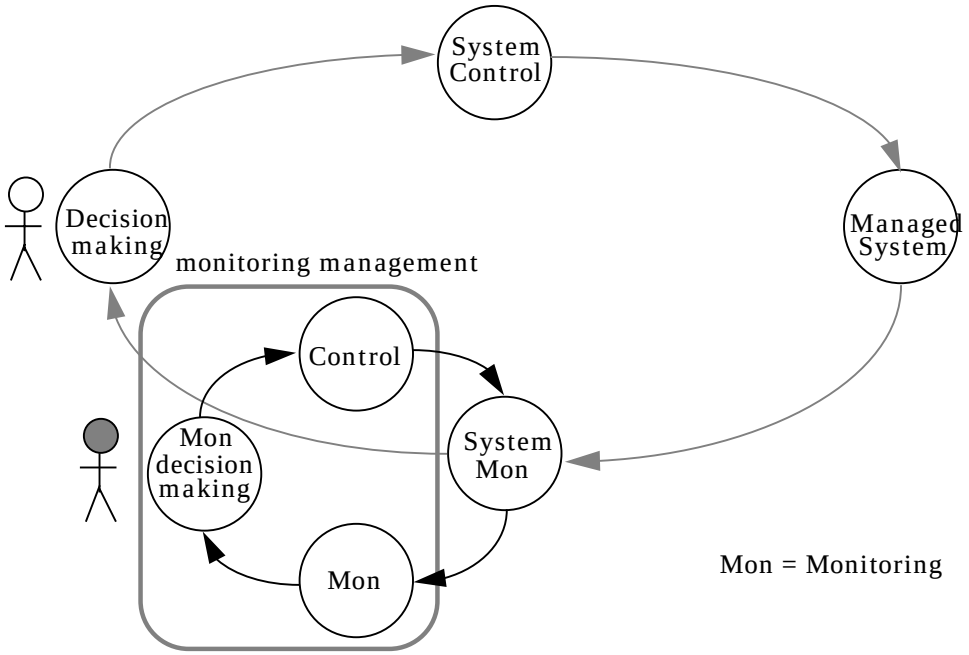
5.3 Applying the model of management to monitoring

When the managed system is large, complex and dynamic, the processes of monitoring and control will themselves be complex activities. They may themselves, therefore, require appropriate management. *Monitoring is a complex process which involves all the epochs in the life of a system and therefore requires appropriate management structures, mechanisms and policies to enable it to be carried out effectively, efficiently and in a timely manner*

The model of management presented in Figure 1 can be applied to the monitoring process in order to show how its management is done, as shown in Figure 5.2.

In Figure 5.2 the roles of the manager of the system who also observes the system through the monitoring facility, and the manager of the monitoring

Figure 5.2: The management of the monitoring process

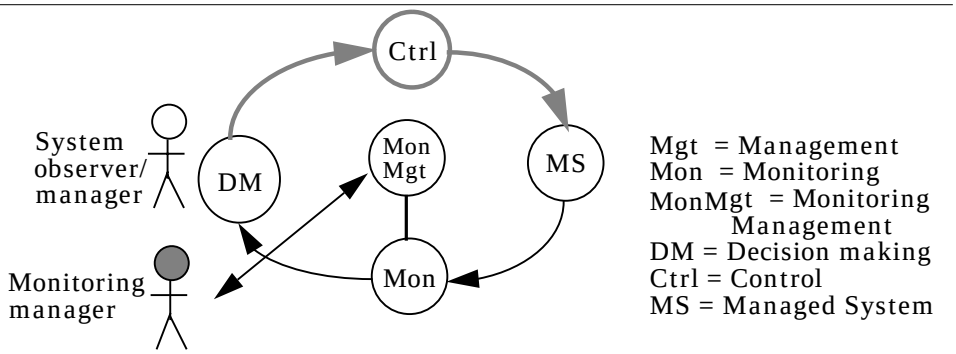


process, are shown separately. In practice, the system manager may also be the monitoring manager.

5.4 Developing the model of management of monitoring

Figure 5.2 can be simplified by combining the control and monitoring of the monitoring process into a single management component (Figure 5.3).

Figure 5.3: Combining the control and monitoring components

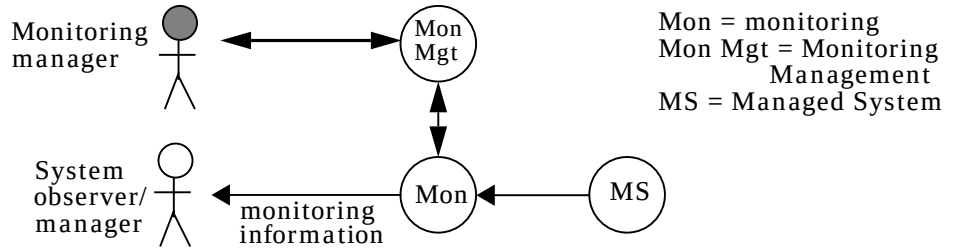


The simplified model shown in Figure 5.4 results from concentrating solely on monitoring whilst ignoring the system control process.

5.4.1 The generic model

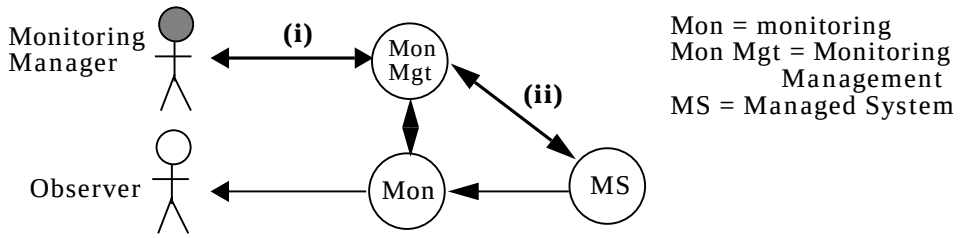
The model of monitoring and its management shown in Figure 3 assumes that all management is directed from outside the system being monitored. However, the monitored system may use the monitoring facilities for obtaining information about itself. Moreover, some of the management of monitoring

Figure 5.4: Concentrating on monitoring and its management



may be automated and embedded in the observed system. This is shown in Figure 5.5.

Figure 5.5: The generic model of the management of monitoring



A generic model of monitoring and its management comprises four components:

- the system under observation: this could be a single service, a collection of services or part of a service
- the monitoring component
- the management component
- the controller/observer

Also shown in the model are the monitoring information channels and two monitoring management channels:

1. external management channel: interaction between observer and the monitors can either take place in an interactive manner with the observer, or by a monitoring manager with embedded decisions concerning monitoring.
2. internal system management channel: this is a form of management embedded in the system so that certain conditions will allow the system to modify the process of monitoring. This type of management is of particular interest when application dependent monitoring is necessary.

The application of this model in the development of monitoring in object-oriented distributed systems ensures that the implementation of monitoring will be uniform both inside objects and for collections of objects. The monitoring and management facilities mentioned in Chapters 5, are investigated and explained in the following chapters in order to develop the facilities in an individual object as well as that of a collection of objects

6 The management of monitoring

6.1 Introduction

This chapter examines the processes of monitoring which require management in order to be able to conduct dynamic monitoring sessions in a distributed system.

6.2 The monitoring process

The process of monitoring can be broken down into the following basic sub-processes:

- **generation**: the generation of a monitoring message as a result of a monitoring event. Any physical system has a variety of interfaces through which different **system events** can be observed, although this cannot always be done directly. Where direct observation of system events is not possible, it is necessary to generate a related secondary event which can be observed. Such a secondary event is referred to as a **monitoring event**, to be distinguished from a system event
- **forwarding** or **distribution**: the transportation of monitoring messages from one or more sources to one or more destinations
- **collation**: when monitoring messages arrive from different parts of the system it is necessary to collect them to enable subsequent logging and processing
- **logging**: storing the monitoring message for deferred processing
- **processing**: the transformation of individual or collections of monitoring messages. Processing may, for example, consist of combining different messages and correlating them to each other to detect global events, filtering, translation of message format and encoding, analysis, etc.
- **presentation** or **visualization**: the transformation of collections of monitoring messages into a form amenable to observer interpretation.

A monitor is a component which provides any or all of the sub-processes of monitoring.

6.3 Areas of management

The **management of monitoring** is concerned with the **organization** (setting up the necessary facilities) and **control** (using the previously set up facilities) of the monitoring process throughout the different epochs of a system's development and use. Decisions have to be made in order to ensure that the monitoring carried out satisfies the purpose of monitoring in an efficient and timely manner. The decisions concern the following main areas:

- organization and control of the generation of monitoring messages

- organization and control of the distribution, collation and logging processes
- organization and control of processing of monitoring messages
- organization and control of the processing and the presentation/ visualization of monitoring information.

The issues in each of these areas are not independent of each other. For example, the choice of monitoring configuration will dictate set of objects which participate in the monitoring session; this in turn will determine whether collation is necessary or not. In all cases resource utilisation and performance considerations are likely to influence the decisions.

6.4 Management of generation of monitoring events

The generation of monitoring events and the subsequent messages is largely concerned with:

- which objects and/or activities in a system should be monitored
- what and when events should be monitored in each of these objects and/or activities
- how are the facilities required to facilitate the control of the above set up.

The answer to these questions in a particular system will depend largely on:

- what model of the system the observer wishes to construct from the observed events. Thus, the monitoring required for accounting may be different from that required for security, fault tolerance or debugging. This has implications on the definition of events and also indicates the need to facilitate extending monitoring to include additional events to those defined at earlier epochs
- the structure and the implementation of the system. In a **centralised** system we are likely to be interested in:
  - — events which occur in an application
  - — the interaction between the application and the user
  - — the interaction between the application and service libraries or between the application and the operating system
  - — events which occur in the operating system

  Whilst in a **distributed** system, in addition to the above, the observer is likely to be interested in:
  - — interaction between parts of the application
  - — interaction between the application parts and their distributed infrastructure
  - — events which occur in the distributed infrastructure.

6.5 Management of distribution, collation and logging

Collation is defined as the process of collection, forwarding and storing (logging) information for subsequent use.

The collation process is largely concerned with questions such as:

- where is the monitoring information collated
- whether there is more than one collation location (no distinction is made here between post run-time collation and the case of multiple observers, for example)
- how are the facilities required to facilitate the control of the above set up.

There are a number of choices depending on the type of the system being monitored:

- an application can hold the information internally in a volatile form (this may mean that when the application terminates it is lost)
- the information can be saved in a file
- the information can be sent to another part of the system
- the information can be displayed to an observer in real-time.

In addition, there will be cases where the monitoring information will be sent to more than one destination.

The decision on what form of collation and logging is necessary will depend on:

- run-time or post run-time analysis
- storage, processing and communication resources
- performance considerations.

There are different ways in which the collation can be set up. When monitoring a small and localized set of objects it may be possible to use a single collator, whilst for a large number of objects distributed widely it may be expedient to provide a hierarchy of local and global collators.

6.6 Management of processing

This involves the management of any processes which transform the monitoring information into forms which are more amenable for use by the observer. One example of such processing concerns reconstruction of event sequences and is explained in the next section.

6.7 Management of processing and presentation processes

The re-presentation is largely concerned with the question of what form do we want the presentation of monitoring information to take. In other words, how to transform the monitoring information to fit the purpose of monitoring. The re-presentation of monitoring information can be divided into two areas:

- reconstruction: an attempt to construct the causal flow of events in a system from information collected from its various parts. Reconstruction is essentially a problem of recognising sequences in a set of given events. This is particularly difficult when monitored events may occur concurrently. The topic of reconstruction is discussed in detail in [TR.41 93] and [HOFFNER 92]
- system modelling using monitoring information: an attempt to present the monitoring information in the context of a specific model or view of the system [JOYCE 87] [ZERNIK 91] [ZERNIK 92]. In some cases it will be necessary to provide different views of the system and to switch between

them and combine them to create more complex views. This will dictate which events are identified as being of interest.

6.8 Management facilities

From the discussion in the previous sections it is concluded that he management facilities provides the monitoring manager with:

- query facilities:
  - — a view of what monitoring facilities are available, e.g. what is the set of monitoring events defined in the system
  - — a view of the state of the monitoring facilities, e.g. are they on/off, where is the monitoring information sent to, whether the system is participating in more than one monitoring session
- activation facilities:
  - — control facilities for changing the:
  - — state of monitoring
  - — the destination of the monitoring messages.

6.9 Monitoring, management and system development epochs

Monitoring can be incorporated by using tools such as:

- transformers - which convert application and infrastructure code to include monitoring
- libraries - allow conditional inclusion depending on whether monitoring is to be added or not

7 Monitoring and management in objects

7.1 Introduction

The computational model[AR.001 93] is an application writer's view of a distributed system without the concerns for the engineering required to implement it. The engineering model [X.903 92] is concerned with the entities, and the relations between the entities, which animate computational model objects, and allows resource issues to be discussed. Capsules are engineering entities which provide processing, storage and communication facilities for one or more objects [ARM 89].

In the following sections the model of management will be applied recursively to developing the structures necessary for management of a monitoring session:

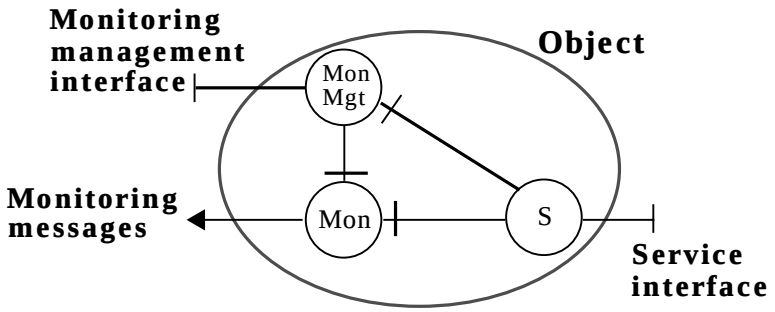
- within an individual object
- of a collection of objects inside a capsules.

7.2 Model of monitoring facilities in an object

In a distributed object-based system each object will have its monitoring and management facility with individual management interface [TR.39 93].

The model of monitoring in objects follows from the general model of monitoring and its management, developed in Chapter 5, and is shown in Figure 7.1.

Figure 7.1: Objects have their own monitoring management service and interface



7.3 Monitoring and management facilities

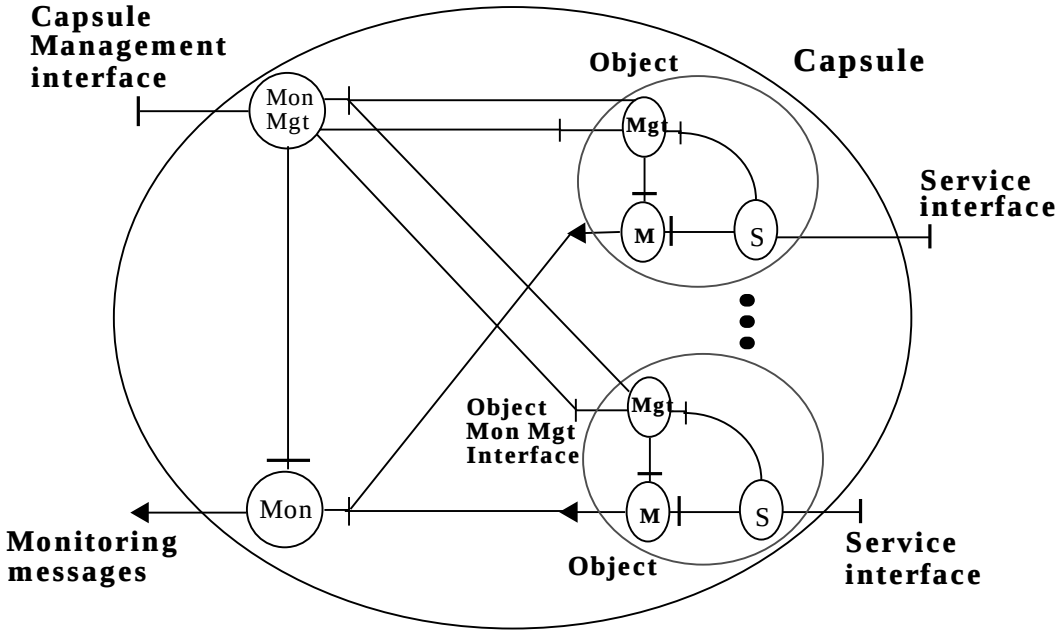
The monitoring facilities in objects are responsible for:

- generating monitoring messages when certain events take place within an objects
- filtering the generated messages if necessary

- forwarding the monitoring messages to the appropriate destination.

To achieve this, the monitoring management service has to provide the information and operations at object level, as discussed in the following section.

Figure 7.2: Monitoring in a capsule consisting of several objects



7.3.1 Object query operations

The query operations on the monitoring management interface provide information on:

1. *what* monitoring facilities does an object have
2. *which* collation destinations are specified to object
3. *whic*h server objects and client objects the object under observation interacting with. The related problem of finding out about the clients of an object is explained in [TR.39 93].

7.3.2 Object activation operations

The following operations can be invoked on the monitoring management interface of an object:

1. modify the state of the monitoring facilities
2. modify the collation destination.

7.4 Model of monitoring facilities in a capsule

When implementing objects it is often necessary, for resource utilization reasons, to aggregate several objects into a single capsule. The model of the management of monitoring can be applied recursively to provide the monitoring of each object as well as the integrated monitoring of the capsule in which they reside (Figure 7.2).

7.5 Monitoring and management facilities

The monitoring facilities in capsules are responsible for:

- managing the monitoring of object inside the capsule or at least providing access to them
- generating monitoring messages when certain events take place within the capsule and the objects inside it. This may be at capsule level so that when events which are global to the capsule occur, a monitoring message is generated
- filtering the generated messages if necessary
- aggregating and forwarding the monitoring messages from the objects to the appropriate destination either inside or outside the capsule.

To achieve this, the monitoring management service has to provide the information and operations, as discussed in the following section.

7.5.1 Capsule query operations

The additional query operations to those supplied by objects in a capsule is:

1. *which* objects exist in the capsule? The response to this query is a list of management interfaces of the objects contained within the capsule being queried.

7.5.2 Capsule activation operations

These operations are basically the same as for objects but can address all the objects in a capsule.

7.6 The granularity of monitoring

By *Granularity of monitoring* we mean the level at which monitoring is activated. For example, whether all monitoring events of a certain type at different entities are activated, or whether a specific instance is activated.

The relationship between capsules, objects and interfaces (and the plugs and sockets associated with them) is useful for dealing with the problem of the granularity of monitoring. An observer in a monitoring session may wish to:

- activate, at a specific level, the monitoring intrinsic to that level. For example, capsule level monitoring which pertains to resource allocation issues, as capsules are the containers of resources such as threads, tasks, plugs, sockets and memory. Cluster level monitoring relates to transparency issues such as migration and passivation which are cluster wide
- zoom to a higher or lower level of encapsulation in order to activate monitoring specific to a level of encapsulation. For example, to activate the monitoring in a specific object or even interface, thereby being able to relate to application specific events.

The management of monitoring must therefore provide the facilities which enable management to relate to each one of these levels separately, thus providing different levels of monitoring granularity.

8 Managing a monitoring session

8.1 Introduction

Distributed applications consist of multiple objects configured in different ways, interacting with each other. This chapter describes the necessary structures and procedures to facilitate monitoring a configuration of objects dynamically.

8.2 Monitoring a configuration of multiple objects

8.2.1 Managing a dynamic monitoring session

Most applications in a distributed system can be characterised as having multiple concurrent activities where:

- the activities may be distributed over different physical locations; and the activities dynamically:
  - — interact with other activities
  - — initiate or terminate other activities
  - — -suspend or terminate themselves
- The dynamic initiation and termination of activities leads to situations where the activities stemming from an application may spread throughout the system. The extent of the initiated activities may not be known in advance.

Thus, monitoring a configuration of objects will require the appropriate structures and procedures to find out about, obtain access to, and manage objects. Management facilities to extend and contract the set of objects being monitored will also be required. In addition, structures for collecting and displaying the monitoring information generated by the objects will be necessary.

8.2.2 Scope of monitoring

The *Scope of monitoring* consists of the set of objects:

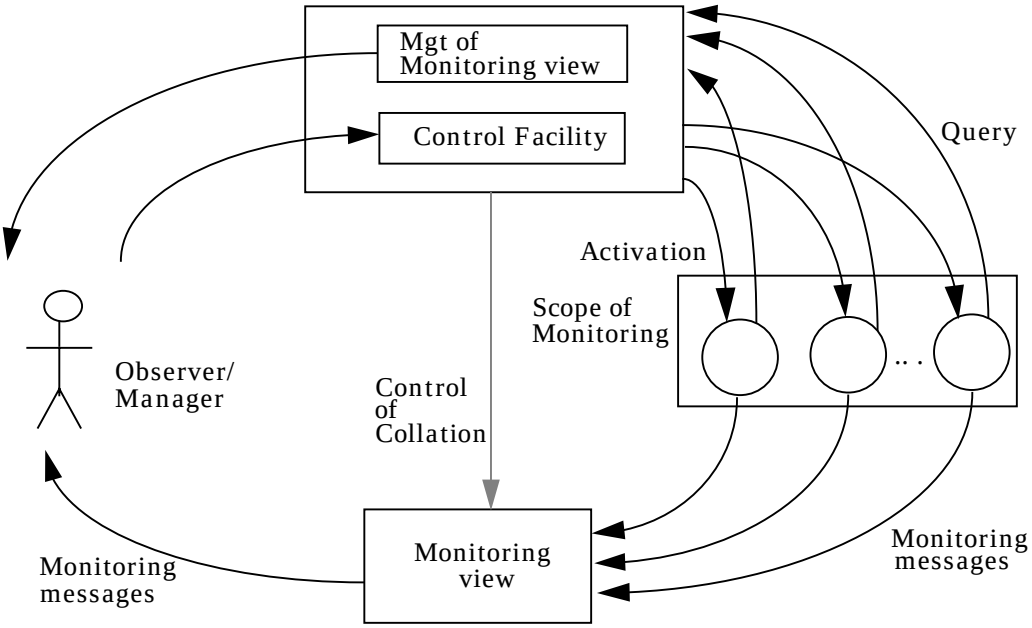
- which are known to the manager of the monitoring session
- whose monitoring management service can be accessed
- whose monitoring information can be directed to the appropriate point of collation.

8.2.3 Views and information in a monitoring session

In order to manage a monitoring session dynamically it is necessary to provide the manager of a monitoring session with the following views (Figure 8.1):

- a *management of monitoring view* consists of the information which is necessary in order to manage the monitoring session, for example, information on the scope of monitoring and the monitoring facilities available in each one of the objects within it
- a *monitoring view* based on the monitoring messages collected by the collation setup from the objects under observation
- a *control facility* to allow effecting changes in the monitoring facilities.

Figure 8.1: Management and monitoring views in a monitoring



8.3 Assumptions about the monitoring session

It is assumed that each monitoring session has its own monitoring management and collation configuration. This assumption does not contradict the option of a single object participating in more than one session, but ensures that outside an object, the monitoring management and collation management is on a “per monitoring session” basis.

In addition, though the case of an object participating in more than one monitoring session does not contradict any of the following discussions, it is considered to be outside the scope of this paper.

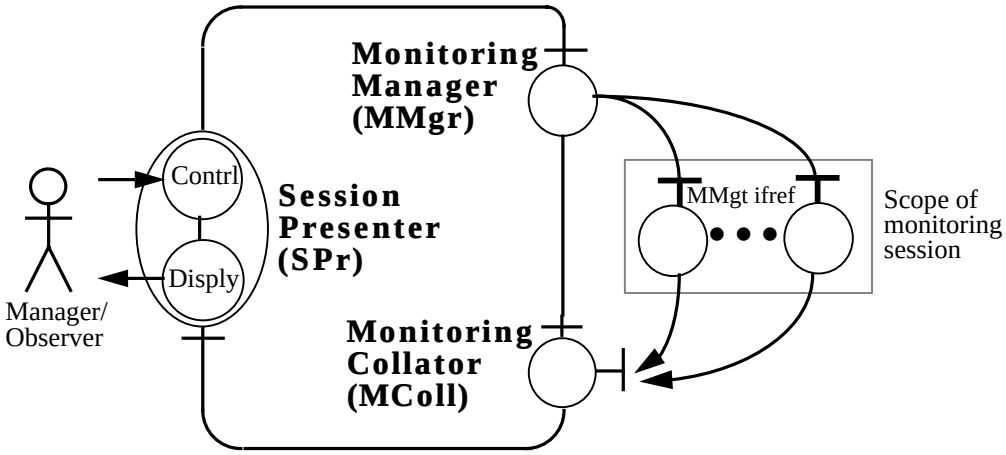
The components such as the SPr, MMgr and the MColl, as well as the objects under observation can be distributed throughout the system. In such cases, a hierarchy of local management and collation structures will be linked together to cater for objects distributed over long distances. Although not dealt with explicitly, the following discussion does not preclude this option.

8.4 Monitoring session management facilities

The monitoring view, management view and the control facility mentioned in §8.2 are provided by the following components (Figure 8.2):

A **Monitoring manager** (MMgr) provides a monitoring session with the management view and with the necessary functions for managing the objects which are within the scope of monitoring. Thus, the MMgr has to keep track of the objects within the scope of monitoring, and to take the necessary actions when the scope of monitoring requires modification.

Figure 8.2: The configuration of a monitoring session



A Monitoring Collator (MColl) is responsible for collecting the information from the monitored objects or from other monitor-collators, necessary to provide the monitoring view. When monitoring more than a single object, more than one monitor-collator may be necessary. The MColl may store the information locally or forward it to another MColl, a file, or a display agent such as a terminal or Session Presenter.

The Session Presenter (SPr) provides the front end to the MMgr and the MColl of a monitoring session. It presents the monitoring information to the observer and allows the observer to control the progress of the monitoring session. It allows the integration of management and collation display and control functions. In its simplest form this could be a terminal with textual output, although graphical tools are more likely to be appropriate in a distributed environment.

8.5 Managing a monitoring session

8.5.1 The phases of a monitoring session

There are four phases in the life of a monitoring session:

1. the setup of the monitoring session components
2. the setup of the initial scope of monitoring
3. managing an ongoing monitoring session
4. closing down a monitoring session.

The rest of this paper deals with the management of an on-going session and in particular with the problem of extending the scope of monitoring dynamically.

8.6 Setting up the monitoring session components

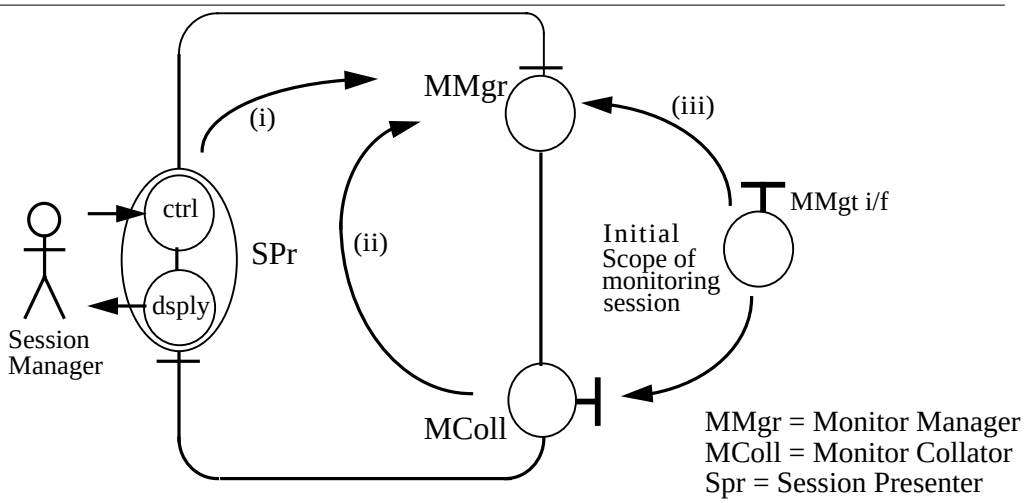
This involves setting up the SPr, MMgr and MColl and will largely depend on the way the components are designed and implemented and on the run-time environment.

8.7 Setting up of the initial scope of monitoring

In order to create the link between the MMgr, MColl and the initial set of objects which the observer wishes to monitor it is necessary to supply the MMgr with sufficient information about the object for it to be able to obtain its management interface. This can be done in several ways:

- the MMgr is given the required information about the object from:
  - — the session manager (i) can notify the MMgr either using information derived from the SPr which implies that the object is already interacting with the MColl, or from having information about the object from having started it, for example
  - — the SPr using information from monitoring messages (i)
  - — the MColl using information from monitoring messages (ii): this requires the object to be able to obtain a reference to the MColl interface
- directly from the object (iii): this requires
  - — the MMgr to have an interface on which the object can notify it of its wish to participate in the monitoring
  - — the reference to this interface can be obtained by the object.

Figure 8.3: Setting up the initial scope of monitoring



8.8 Managing an ongoing monitoring session

Managing an ongoing monitoring session includes the control of the monitoring facilities of the objects within the scope of monitoring as well as changing the scope of monitoring as the need arises.

Objects within the scope of monitoring may interact with objects outside it. In some cases it may be desirable to bring objects outside the scope of monitoring into it. This can be done in two ways:

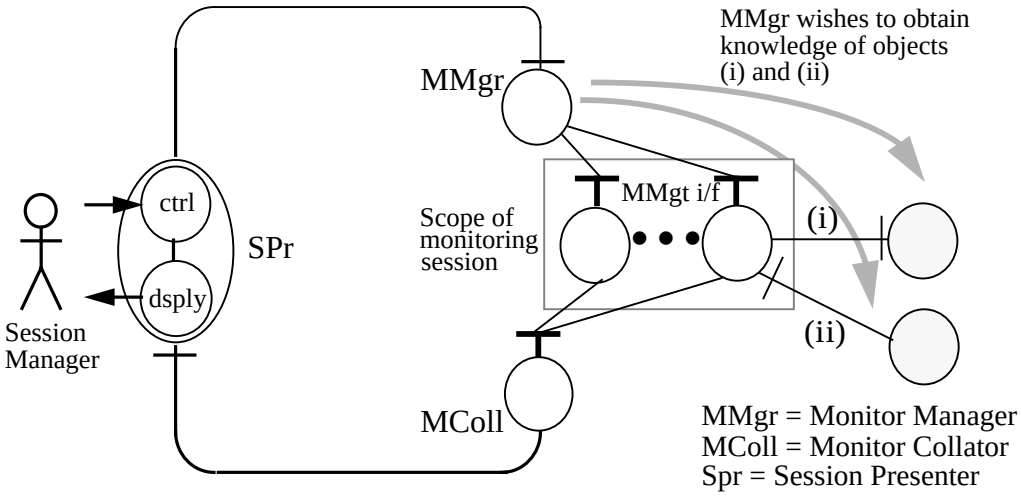
- the object inside the scope of monitoring requesting the objects outside it directly without the intervention of the MMgr. However, this basically requires that every service interface in the system has a subset of the management interface discussed in §7.5 and is therefore considered too cumbersome for implementation
- by giving the MMgr sufficient information about the objects outside the scope of monitoring to enable it to obtain access to the management interface of these objects.

8.8.1 Extending the scope of Monitoring

If we adopt the latter solution then the problem of dynamically extending the scope of monitoring requires the MMgr to be able to do two things:

- to find out about any client or server objects outside the scope of monitoring with which objects within the scope of monitoring are interacting with (Figure 8.4)
- to gain access to the management facilities of those objects, having found out about them.

Figure 8.4: Finding out about objects outside the scope of monitoring



There are two cases: that of the server and that of the client outside the scope of monitoring.

8.8.1.1 Obtaining access to a server

In the case of a server being used by an object inside the scope of monitoring (i), it is possible to provide the MMgr with the interface reference of the service. The MMgr can then invoke the getManagementInterface[] operation on the server and get a reference to its management interface.

8.8.1.2 Obtaining access to a client

The case of a client outside the scope of monitoring (ii) is more complicated in that there is currently no way in which an object can find out sufficient information about the clients using it, to enable gaining access to the client's management interface.

In fact, even if a server can find out who its clients are, this does not guarantee its ability to gain access to any of its interfaces. This is because the client may not have exported any interfaces to the Trader or to any other repository of such information. There is no way in the current engineering model to enable a client to gain access to its server.

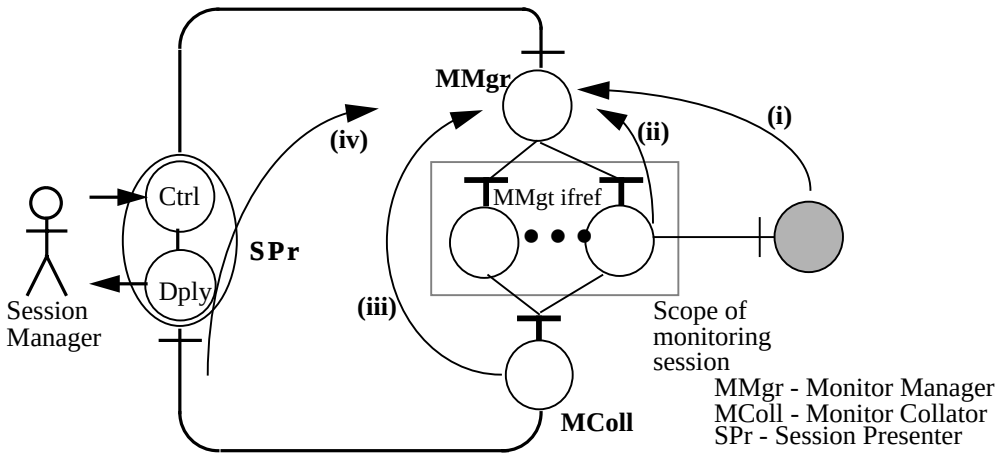
The problem of getting sufficient information for the MMgr is involved with a more general problem exposed in [TR.39 93], and concerns the way in which a server object gains information about its clients.

8.8.2 Notifying the MMgr

A need to notify the MMgr of client/server objects outside scope of monitoring has been identified. The information concerning clients and servers can be transferred to the MMgr in a number of ways (Figure 8.5):

1. from the objects outside the scope of monitoring directly to the MMgr
2. from the objects which are within the scope of monitoring to the MMgr
3. from the MColl directly to the MMgr
4. from the MColl to the MMgr through the monitoring information channel through the SPr by user intervention.

Figure 8.5: Possible solutions to notifying the MMgr of objects outside the scope of monitoring



9 Monitoring across boundaries

9.1 Introduction

This chapter points out the areas in which agreement is necessary for the successful implementation of monitoring across heterogeneous federated distributed systems.

9.1.1 Framework for standardization

The integration and use of monitoring facilities in a distributed system can be viewed at three different levels:

- *application specific and general distributed system monitoring*: between the application program and the distributed infrastructure
- *single domain*: between the application program and services in the same domain
- *multiple domains*: between the services and clients residing on different domains

To enable monitoring in a distributed system requires agreement in several areas:

- what set of basic monitoring facilities should be available in each object
- what set of basic management facilities are necessary in order to manage monitoring, e.g. how are monitoring facilities queried and activated
- the set of procedures and mechanisms necessary in order to gain access to the monitoring facilities
- how is monitoring information to be collected
- and finally, how is the information to be presented and interpreted:
  - — what models can be derived using the monitoring information
  - — which graphical/textual representation

9.1.2 Management domains and boundaries

A **management domain** consists of a set of objects which have some attribute in common or which are managed by the same management agents [SLOMAN 89b].

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.

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.

Extending the scope of monitoring may take place across federation boundaries, which may have conflicting policies concerning monitoring. In such cases, 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 conflict, the client may request access to a management agent of the server.

9.2 Integrating application and distributed infrastructure monitoring

Distributed platform monitoring is by nature not application specific, that is, it provides information on types of events which are common to all applications using the platform. By contrast, application monitoring will be incorporated by the application programmer in order to provide information specific to the application. There are cases when it will not be sufficient to provide either information separately, but where application specific and independent monitoring should be combined, for debugging purposes, for example (Figure 9.1).

Figure 9.1: The integration of application specific and distributed infrastructure monitoring

In addition, it can be useful to be able to use the facilities provided by the platform, such as the collation set up, rather than re-design and implement them with each application. If this is to be facilitated, then agreement on how distributed platform can be extended to incorporate application specific monitoring must be reached.

In order to be able to integrate application specific with distributed infrastructure monitoring, agreement on the following issues is necessary:

- interfaces (Figure 9.2):
  - — distributed-infrastructure interface which the application designer can use (Figure 9.2). This interface should be offer the same operations as the external monitoring management interface
  - — application management interfaces to allow application-specific management to be carried out through the infrastructure facilities. This requires additional operations in the external monitoring management interface for querying and activating the application specific events
- event message format and adherence to rules on extending existing set of events

9.3 Integrating monitoring in a single domain

In a distributed system, an application may use services available in the same domain (Figure 9.3).

Figure 9.2: Interfaces for application specific and distributed infrastructure monitoring integration

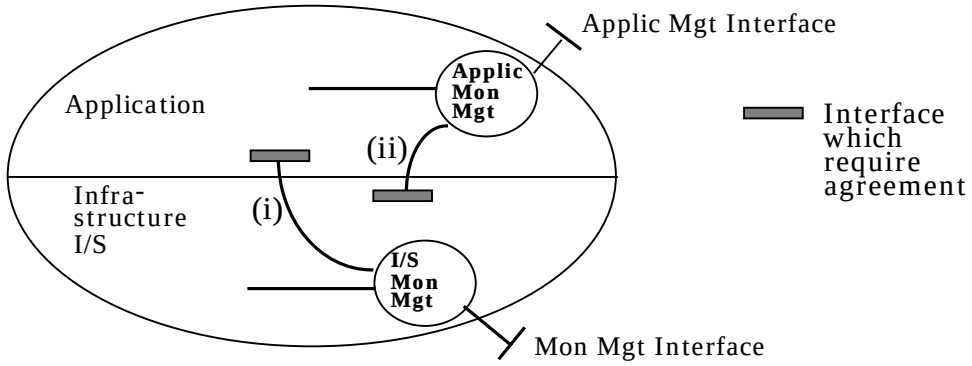
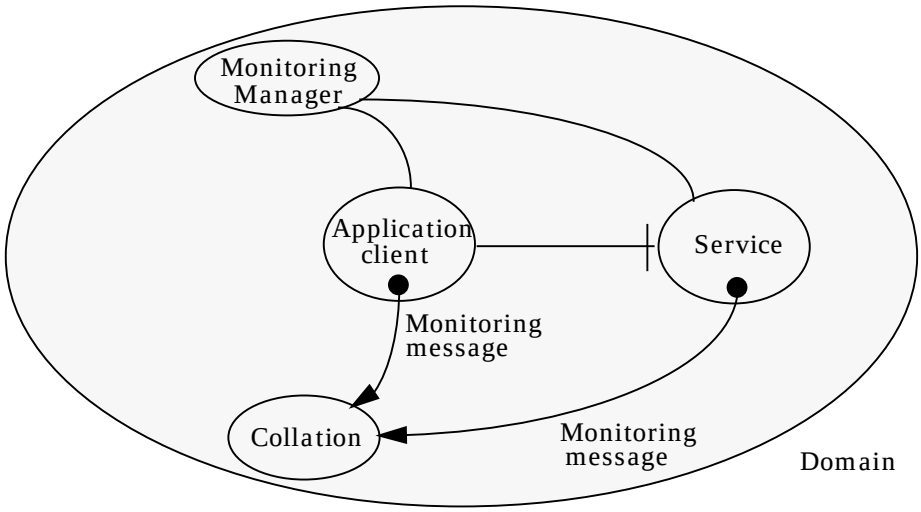


Figure 9.3: Integrating application and service monitoring on a single domain



In order to be able to integrate monitoring in a single management domain it is necessary to agree on:

- facilities for finding out about the services being used
- how to gain access to their monitoring management facilities
- how to activate the monitoring facilities
- how to ensure that the monitoring messages are collated appropriately.

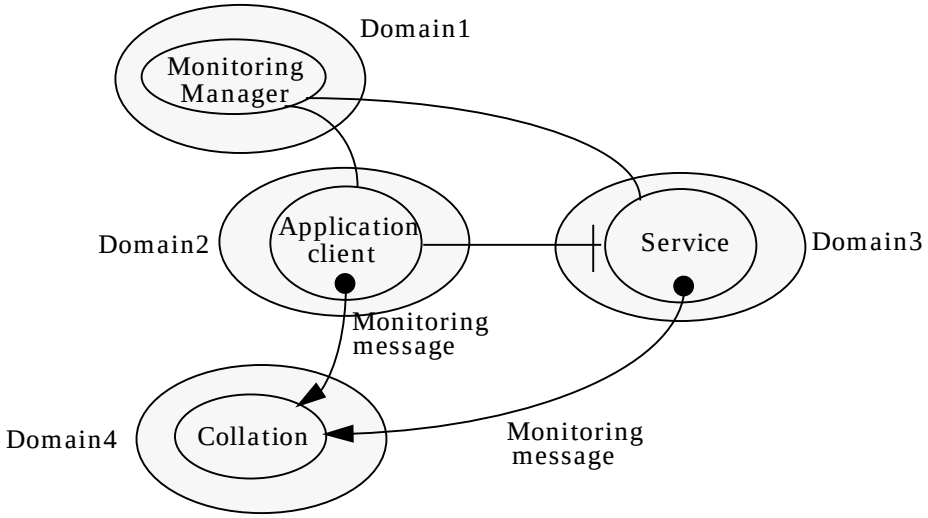
9.4 Integrating monitoring across several domains

In a multiple management domain environment, in addition to the requirements of the single domain case, it is necessary to overcome (Figure 9.4):

- different monitoring facilities and access to management facilities
- different data representations
- different management policy.

The diversity and complexity of management requirements is such that standards in this area are harder to define and apply than many OSI protocols

Figure 9.4: The integration of monitoring across node boundaries



[LABARRE 91] [ISO 10164-1] [ISO 10164-2]. This is because it is necessary to specify both standards for communication and, more significantly, standards for the management information which is to be communicated. An essential pre-requisite for the latter is a definitive architecture for management activity [DOMAINS 92].

The getManagementInterface[] operation can be used to request access to successive levels of an objects agents, until the appropriate level for policy clash resolution can be achieved. In order to facilitate such a dialogue agreement on common management interfaces and operations is required.

9.5 Monitoring facilities standardization issues

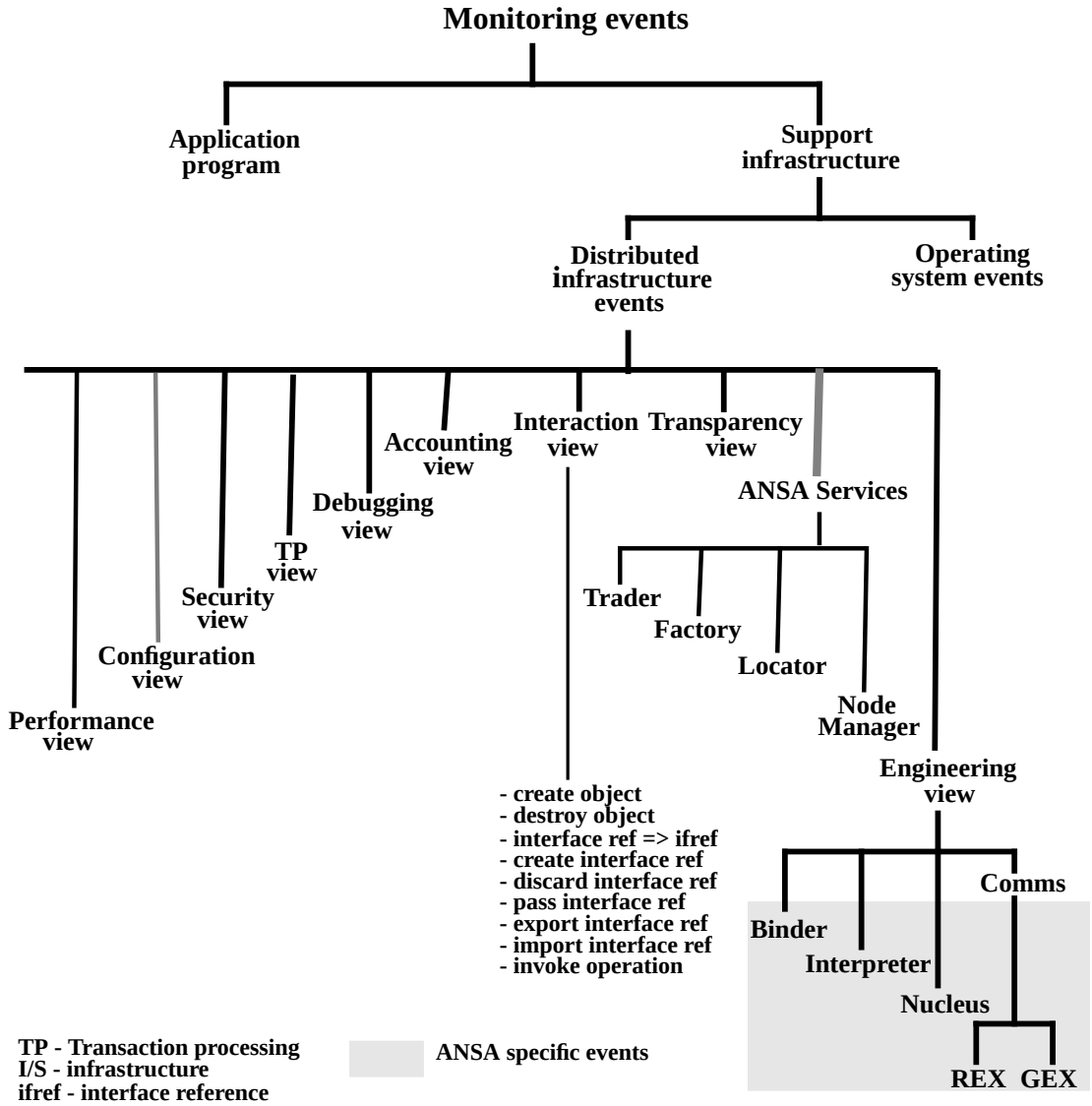
9.5.1 Basic set of events - a taxonomy

It is necessary to agree on a basic set of events which will be monitored and which are likely to be of interest to the different groups of people involved with the system. The basic set of monitoring events should reflect the common aspects of distributed application platforms.Thus a taxonomy of monitoring events in distributed systems is necessary. Such a taxonomy is given bellow.

- A taxonomy of monitoring events can help clarify the differences and commonalities between monitoring in different distributed application platforms. This can serve as a starting point for dealing the problems of monitoring across federated systems. A Monitoring event taxonomy scheme is presented in (Figure 9.5).

The taxonomy of monitoring events distinguishes between events which take place in the support infrastructure, and the application programs. This distinction arises from a realization that some of the activities in which monitoring is used may require the ability to combine application dependent monitoring with infrastructure monitoring (see §9.2). For example, when debugging a system.

Figure 9.5: Monitoring event taxonomy



With regard to the support infrastructure events, a further distinction can be made between the local operating system part and events relating to the distributed infrastructure.

The distributed infrastructure taxonomy is influenced by the object-oriented nature of the distributed systems we are dealing with, but also leaves the ability to classify events in an implementation oriented/dependent manner. Thus, the distributed infrastructure events are divided into engineering view events which encompass implementation details. An object-oriented view is provided which defines the events related to object creation, destruction, interaction, etc. Such a view should be independent of differences in the implementation issues, i.e. the engineering view.

In many cases, low-level events from the engineering view are probably sufficient for constructing higher-level views. However, the definition of high-level events can save much time in the interpretation and presentation phase since the necessity to recognize complex sequences of low-level events can be eliminated or at least reduced.

Furthermore, the divisions in the distributed infrastructure events provides the definition of high-level events relating to transaction processing (TP), security, audit and accounting views. A transparency view relates to the transparency mechanisms which are found in distributed systems [ANSA 90]. These may have to be monitored just like other parts of the system.

One possible area of commonality is that of the computational model which most of the existing platform support (DCE [OSF 91], OMG [OMG 91], ANSA [AR.001 93]). For example, agreement on interaction level events which facilitates the visualization of interacting objects will be useful.

9.5.2 Management of monitoring in objects

This area concerns the management facilities provided in each object as addressed in §7.3 and §7.5.

9.5.3 Access to management and scope of monitoring

This area concerns the problems of obtaining access to the management facilities of an object. It is addressed in §8.4.

9.5.4 The Monitoring manager and Monitoring collator

This area concerns the development of the interfaces between the components which manage a dynamic monitoring session with multiple objects. It is addressed in §8.4.

9.5.5 Presentation issues

Agreement on the visualization of distributed systems: for example, the “blob and stick” diagrams discussed in ODP [ISO-439 91]. See also [TR.41 93].

References

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

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

[AR.002 93]
AR.002: “A Model for Interface Groups”, APM,Architecture Projects Management Ltd., Poseidon House, Castle Park, Cambridge CB3 0RD, UK (1993).

[AR.003 93]
van der Linden, R., AR.003: “The ANSA Naming Model”, Architectural Report 003,APM,Architecture Projects Management Ltd., Poseidon House, Castle Park, Cambridge CB3 0RD, UK (1993).

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

[AR.007 93]
Herbert, A.J., AR.007: “Addressing and Interception”, Architectural Report 007, APM,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).

[ARM 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).

[DOMAINS 92]
"DOMAINS Standardization", Document D2f V1.0, Distributeed Open Management Architecture in Networked Systems, Esprit Project No 5165, (Sept 1992).

[HOFFNER 92]
Hoffner, Y. and Statham, A. "The Visualization of Distributed Systems", the Second International Conference on Computational Graphics and Visualization Techniques, Lisbon, Portugal, 14-17 Decmber 1992 (1992).

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

[ISO 10164-1]
ISO/IEC DIS 10164-1 Information Technology - Open System Interaconnection Part 1: Object Management Function, (Oct 1990).

[ISO 10164-2]
ISO/IEC DIS 10164-2 Information Technology - Open System Interaconnection Part 2: State Management Function, (Oct 1990).

[ISO 10164-4]
ISO/IEC DIS 10164-4 Information Technology - Open System Interaconnection Part 4: Alarm Reporting Function, (Oct 1990).

[ISO 10164-5]
ISO/IEC DIS 10164-5 Information Technology - Open System Interaconnection Part 5: Event Report Management Function, (Oct 1990).

[ISO 10164-6]
ISO/IEC DIS 10164-6 Information Technology - Open System Interaconnection Part 6: Log Control Function, (Oct 1990).

[ISO-439 91]
ISO-439, Diagramatic Conventions for the RM-ODP, Working Document, ISO/IEC JTC1/SC21/WG7 (November 1991).

[JOYCE 87]
Joyce, J. Lomow, G., Slind, K. & Unger, B. "Monitoring Distributed Systems", ACM Transactions on Computer Systems, 5(2), 121-150 (May 1987).

[LABARRE 91]
LaBarre, L. "Management by Exception: OSI Event Generation, Reporting and Logging", The MITRE Corporation, 2nd IFIP Symposium on Integrated Network Management, Washington, USA (April 1991).

[LAMPORT 78]
Lamport, J. L. "Time, Clocks, and the Ordering of Events in a Distributed System", Comm. ACM, 21(7), 558-565 (1978).

[MCDOWELL 89]
McDowell C.E. & Helmbold, D.P., "Debugging Concurrent Programs", ACM Computing Surveys, 21(4), 593-622 (December 1989).

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

[RAYNAL 90]
Raynal, M. "Order notions and atomic multicast in distributed systems: A short survey", Technical Report 1197, Institut de Recherche en Informatique et Systemes Aleatoires. Rennes (France), March 1990.

[SAMANI 92]
Samani, M. M. & Sloman, M. “Monitoring Distributed Systems (A Survey)”, Imperial College Research Report No. DOC92/23, Imperial College of Science Technology and Medicine, (Sept 92).

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

[SLOMAN 89b]
Sloman, M. S. & Moffett, J. D. “Domain Management for Distributed Systems”, In Integrated Network Management I, Eds. Meandzija, B. & Westcott, J. North Holland, (1989).

[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).

[X.903 92]
Recommendation X.903: Basic Reference Model of Open Distributed Processing Part 3: Prescriptive Model, ISO 10746-3, (November 1992)

[MCDOWELL 89]
McDowell C.E. & Helmbold, D.P., "Debugging Concurrent Programs", ACM Computing Surveys, 21(4), 593-622 (December 1989).

[OSF 91]
“Introduction to OSF DCE”,Revision 1.0, Open Software Foundation, (December 1991)

[PETERSON 89]
Peterson, L. L., Bucholz, N. C. & Schlichtig, R. D. "Preserving and Using Context Information in Interprocess Communications", ACM Transactions on Computer Systems, 7(3), 217-246 (August 89).

[TR.39 93]
Hoffner, Y. “TR.39.01: Management in Object-Based Federated Distributed Systems”, Architecture Projects Management Ltd, Posiedon House, Castle Park, Cambridge CB3 0RD, UK (January 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).

[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).

[WINTERBOTHAM 87]
Chapter 14: Management", in Herbert, A. J. & Monk, J. M. ANSA Reference Manual, Release 00.03, ANSA, APM Ltd., Poseidon House, Castle Park, Cambridge CB3 0RD, UK (1987)

[ZERNIK 91]
Zernik, D & Rudolph, L. “Animating Work and Time for Debugging Parallel Programs Foundation and Experience”, Proceedings of Conference on Parallel and Distributed Debugging, ACM, (1991).

[ZERNIK 92]
Zernik, D., Snir, M. & Malki, D. “Using Visualization Tools to Understand Concurrency”, IEEE Software, 9(3), (May 1992).