**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**

# Visualisation of Distributed Systems

**Yigal Hoffner**

**Abstract**

This paper presents a model of the visualization process and discusses the problems associated with the process of visualization of distributed systems. The problems can be divided into those concerned with incomplete information, and those concerned with unexpected information arriving from the monitored system. Solutions to these problems are introduced and are used to obtain guidelines for the designer of distributed systems and for the designer of a visualization tool.

APM.1019.01 **Approved** 25th October 1994

Technical Report

**Distribution:**
**Supersedes**:
**Superseded by**:

# Visualisation of Distributed Systems


# Visualisation of Distributed Systems

Yigal Hoffner

APM.1019.01

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 explains the problems associated with the visulaization of distributed systems and the solutions to these problems. In addition, the requirements which these solutions set for the monitoring infrastrcutre in such systems and for visualization tools, are introduced. Thus, the paper is aimed at the designers of distributed systems and the designers of visualization tools.

## 1.2 Context

This document should be read in conjunction with [AR.010 93] and [AR.010 93]. The documents are related to each other in the following fashion:

- this document explains the problems with visualizing distributed systems and discusses the requirements such a process poses to the monitoring infrastructure
- TR.39 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.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.

## 1.3 Overview

The inherent complexity of distributed systems poses many problems to the different groups of people such as designers, implementors, administrators and users, involved with these systems. Tools which can visualize the events which take place inside distributed systems can help in dealing with this complexity [ZERNIK 91], [ZERNIK 92], [JOYCE 87] and [MCDOWELL 89].

A visualization tool constructs a model of the system under observation. A monitoring infrastructure is necessary in the observed system in order to obtain the appropriate information from it for the visualization tool.In a distributed system there are several problems with such an arrangement due to the effects of separation, concurrency, heterogeneity, federation and evolution [HOFFNER 93], [AR.010 93].

This document presents a model of the visualization process. A model of language processing is used to classify and explain the issues associated with the visualization process. The problems associated with visualization of distributed systems are divided into those concerned with:

- incomplete information

- unexpected information arriving from the monitored system.

Solutions to these problems can be divided into those solutions which:

- are based on prior agreement between the different systems being monitored
- involve the processing of the information which arrives from the monitored system.

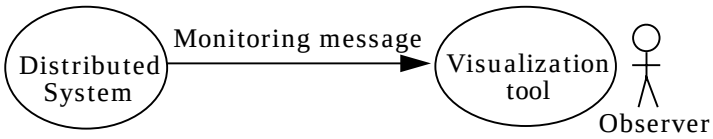
Another way of looking at these approaches is as preventive solutions which are based on prior agreement, and solutions which deal with cases where agreement cannot be reached.

# 2 The visualization process

## 2.1 A general model of the visualization process

The visualization tool aims to present the observer with a model of the system under observation, based on the information derived from it (Figure 2.1).

**Figure 2.1: A general model of the visualization process**

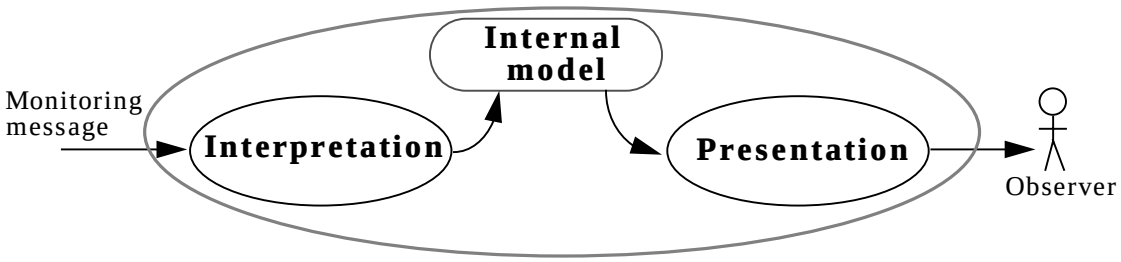



Due to the indirect nature of observations in computer systems, it is necessary to base the visualization process on information obtained from the monitoring infrastructure of the system in the form of monitoring messages. This design of the monitoring infrastructure is addressed in [AR.010 93].

The process which takes place within the visualization tool consists of two stages (Figure 2.2):

- the **interpretation** of the monitoring messages arriving from the observed system and the creation and updating of the internal model. The internal model is an image of the observed system and the events which take place in it
- the **presentation** of the internal model in a form amenable to interpretation by the observer.

**Figure 2.2: The visualization tool**




Forobject example, the visualization of the interactions between objects in a distributed system requires the internal model to represent objects which are known to exist and the bindings between them. One possible representation of such information in the internal model is in the form of a graph where nodes represent objects, and arcs between them represent bindings between objects. Interactions between objects can be interpreted and checked in the context of the graph to determine whether the source and destination of the message exist.

## 2.2 Requirements from the internal model of the visualization tool

There are two requirements from the internal model:

- integrity requirement which consists of two related issues:
  - — coherency requirement: describes the relation between the model and the system it is depicting
  - — consistency requirement: avoiding conflicting or contradictory information within the internal model. For example, that any referenced data item must actually exist in the model
- it has to be in a form which can be made presentable to an observer.

## 2.3 The visualization process and language processing

In order to gain better understanding of visualization problems it is helpful to view the monitoring messages, sent by the monitoring infrastructure of the distributed system, as sentences expressed in a language of interaction. As far as the visualization process is concerned each monitoring message describes a monitoring event which occurred in the system, and constitutes a sentence which is used to update the internal model of the visualization tool. The problems associated with distribution can then be re-defined and classified in terms of the language analysis model.

From experience with compilers, interpreters and translators, it has been found that language models can help in designing tools which have to analyse and interpret messages [WINOGRAD 72]. The DEMON visualization tool [HORNE 90], for example, uses such a language model.

### 2.3.1 The Visualization Process and the Language Model

The language analysis model [GREENE 86] divides language processing into lexical, syntactic, semantic and discourse analysis stages. The language model can be used to classify the problems of the visualization process as shown in Figure 2.3.

#### *2.3.1.1 Lexical Analysis*

The lexical analysis stage identifies the constituent parts or lexemes of the monitoring message. This requires either:

- full list of all possible lexemes
- agreement on lexeme separator(s).
- lexical analysis raises the issue of different **representations** or **encoding** of lexemes. Different implementations of monitoring may use different encoding and some form of translation may be necessary to overcome this problem.

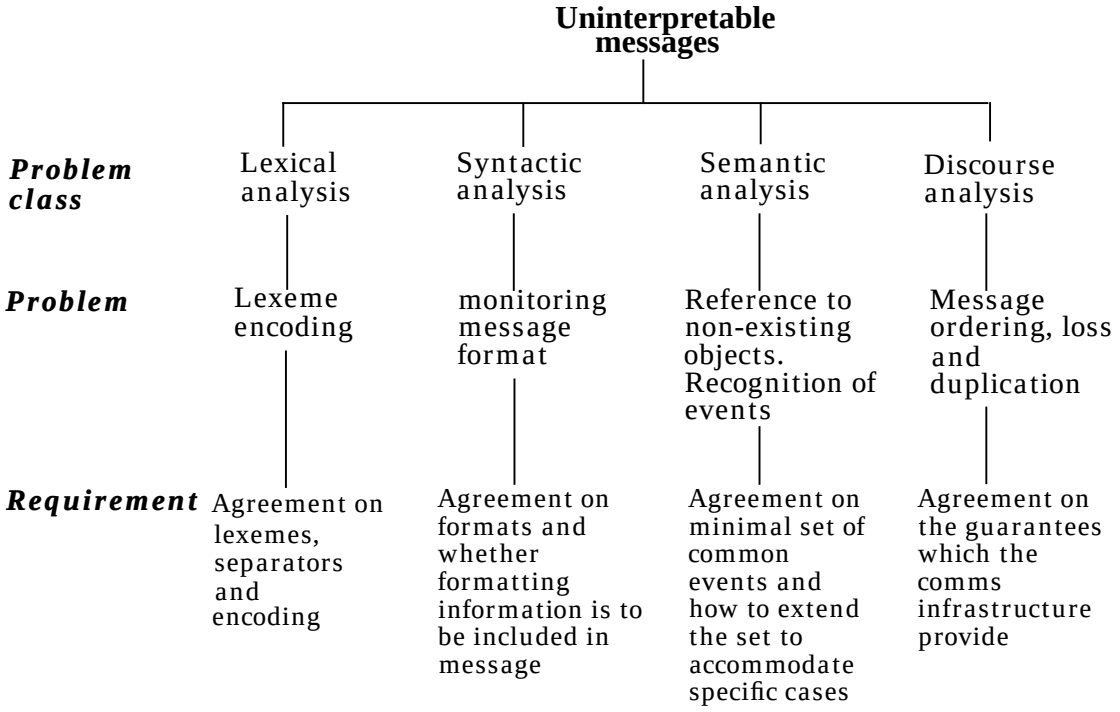
Since the lexemes in a monitoring message will be a combination of pre-determined reserved words as well as names denoting things such as object identifiers, it is necessary to agree on lexeme separator(s).

#### *2.3.1.2 Syntactic Analysis*

Syntactic analysis determines the structure of the monitoring message and the type of some of its lexemes from their position in the monitoring message.

There are two ways by which syntactic analysis can be carried out:

**Figure 2.3: Using the Language model to classify the problems the visualization of heterogeneous distributed systems**




- identify the monitoring event type from monitoring message structure
- identify sentence structure from monitoring event type.

The latter is simpler but requires the monitoring message to carry the monitoring event type information and also have agreement on the set of possible events. This raises the issues of agreement on event types and their representation. Also, syntactic analysis raises the issue of formatting information and whether to include monitoring message format information in each monitoring message.

#### *2.3.1.3 Semantic Analysis*

The semantic analysis stage uses the incoming sentence to construct and update the internal model of the visualization tool:

- entities which are referred to in the monitoring message must either already exist in the internal model or be created as a result. Entities may be objects in a distributed system which are known to exist
- operations on entities must not contradict what can be done to them, i.e. they must be valid in the given context. Permitted operations on these objects are: destroy object, bind to another object, invoke another object and receive reply from another object
- any attributes of entities must fit the type of entities.

Semantic analysis raises the issue of referral to entities which do not exist in the model and thus the problem of the **coherency** of the internal model. This is tied to the problem of incompleteness as well as the problem of the consistency of the model.

#### *2.3.1.4 Discourse Analysis*

Whilst lexical, syntactic and semantic analysis deal with a monitoring message as the unit of data, discourse analysis deals with the relation between sequences of monitoring messages. The interpretation of sequences of monitoring messages is the task of the visualization tool.There are several problems with discourse analysis: monitoring messages ordering, loss of messages and duplication of messages. These topics will be discussed in the rest of this document.

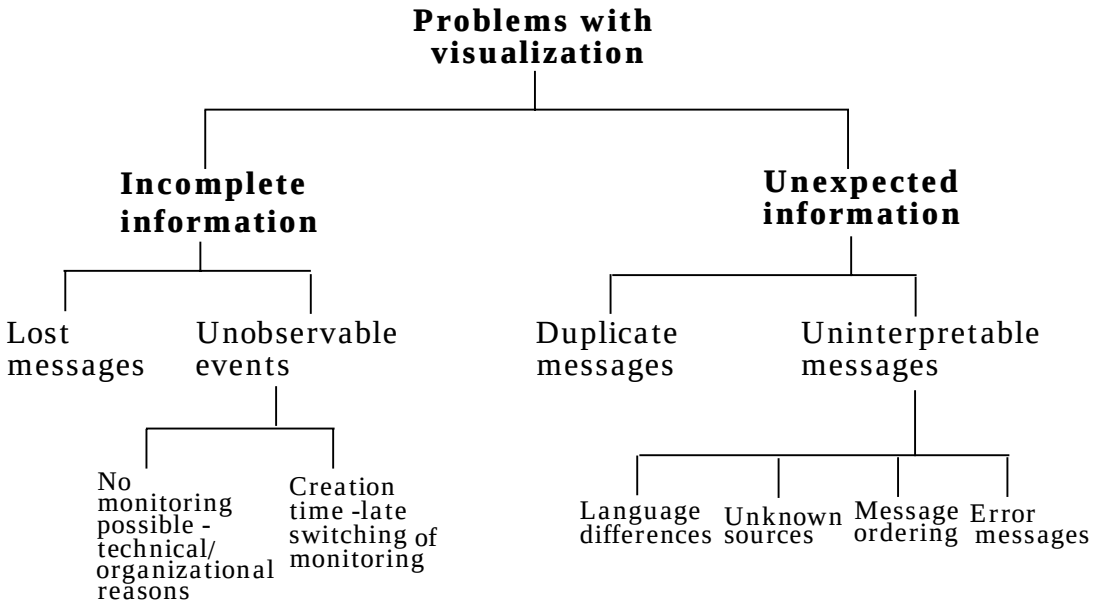
# 3 Problems with the visualization process

## 3.1 Visualization and distribution

The visualization process is affected by certain aspects of distributed systems: physical distribution, concurrency, heterogeneity, federation and evolution [AR.010 93] and [AR.010 93]. The problems which arise from distribution can be divided into two major categories (Figure 3.1):

- incomplete information problems
- unexpected information problems.

**Figure 3.1: Classification of problems with the visualization process in distributed systems**




## 3.2 Incomplete information problems

One of the problems of using monitoring information for the visualization process is that there are cases where information about events of interest cannot be obtained. The lack of information may make subsequent monitoring messages uninterpretable by the visualization tool. We distigusih between two general cases:

- **Monitoring information loss**: Physical distribution introduces levels of indirection between the observer and the observed system. Networks may cause monitoring messages to be delayed for long periods whilst partial failures may cause monitoring messages to be lost or delayed indefinitely.

- **Unobservable events**: Some events in a system cannot be observed due to technical or organizational reasons which prevent either the installation or the activation of the necessary monitoring facilities.

## 3.3 Unexpected information problems

A problem with the visualization process arises when a message arrives at the visualization tool which it cannot interpret in its current context [SAMPSON 92]. This may be the result of :

- **Duplicate messages**: Monitoring messages may be duplicated if the communication infrastructure used does not guarantee that every message arrives only once at the destination
- **Uninterpretable messages**:
  - *language differences* or *hetergoneity* problem: the set of monitoring messages from heterogeneous parts of the systems may vary, as may the format and the encoding of messages
  - *unknown message sources*: due to the nature of distributed systems it is possible to have monitoring activated in objects which the visualization tool does not know exist, resulting in unexpected messages
  - *message ordering*: the order in which messages arrive at the visualization tool may not reflect the order in which they were sent. They may therefore be processed in the wrong order causing the visulaization tool difficulties
  - *error messages*: multiple levels of indirection together with complex failure modes may cause error messages to arrive from unexpected parts of the system. Such messages will have to be dealt with by special facilities in the visualization tool.

## 3.4 Solutions to the visualization problems

There are two general types of solutions which can be applied to the problems outlined above: preventive solutions and solutions which require some form of post-monitoring processing.

### 3.4.1 Preventive solutions

Preventive solutions are based on prior agreement or use of special facilities to avoid the need for post-monitoring processing.

### 3.4.2 Post-monitoring processing solutions

Post-monitoring solutions can be divided into two categories:

- pre-visualization tool processing: in most cases, the solution to the problems of the visualization can be achieved by the appropriate configuration of the monitoring infrastructure or some form of processing applied to the monitoring messages, before they reach the visualization tool
- inside visualization tool processing: where unexpected events and incompleteness events programmed for the visualization tool itself.

The approach to dealing with these problems is describede under the following topics:

- heterogeneity
- ordering
- un-observability
- visualization tool
- presentation.

# 4 Monitoring and heterogeneity

## 4.1 Introduction

When monitoring a distributed systems it will be necessary to monitor activities across domain boundaries. By a boundary we refer to any point where differences between parts of the system have to be dealt with explicitly, either by prior agreement or by providing the necessary mechanisms to overcome the differences.

Different monitoring domains may provide monitoring facilities which are based on different models of computation and engineering, and are likely to provide a different set of events to reflect the different decisions taken by teh designers and implementers of the system. In addition, the format and the encoding of messages may vary from one system to another.

## 4.2 Preventive solutions to the heterogeneity problem

Preventive solutions are based on prior agreement or use of special facilities, and concern issues such as:

- minimal set of common monitoring events and classes of monitoring events, e.g. interaction level events
- information carried in the monitoring message
- monitoring message formatting issues
- monitoring message encoding
- how to get information concerning unobservable events by different means
- the information necessary to facilitate causal ordering of events.

There are additional requirements on the monitoring and management infrastructure to ensure that:

- the appropriate monitoring facilities are activated where and when necessary
- the necessary collation facilities can be set up to collect the monitoring information and direct it to its destination.

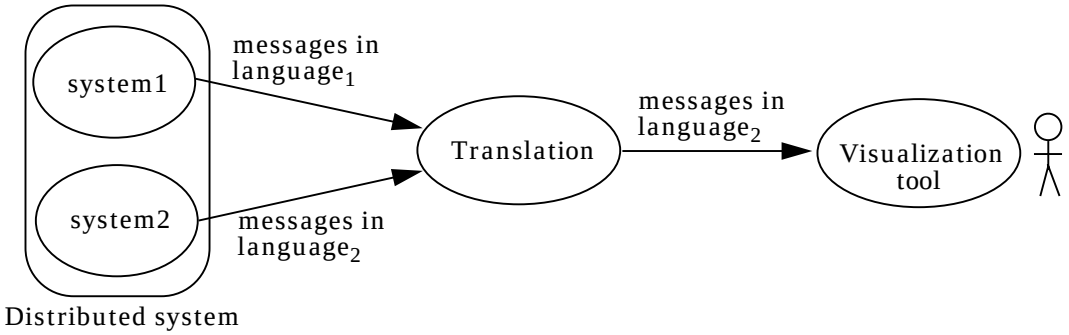
## 4.3 Post monitoring solutions to the heterogeneity problem

### 4.3.1 Translation

In case of different implementations of monitoring which result in format or encoding differences, it may be possible to translate the monitoring messages to an acceptable representation (Figure 4.1). At the very least, agreement on

where information concerning the definition of a minimal set of events can be obtained.

**Figure 4.1: The translation solution**



# 5 Monitoring and unobservability of events

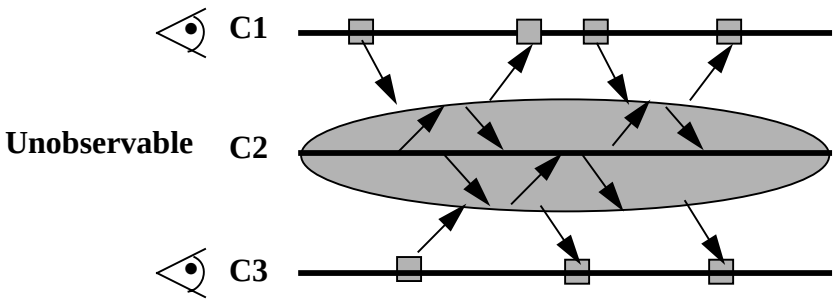
## 5.1 Introduction

There are situations in a distributed system where it is not possible to obtain information from an object directly. This can happen for several reasons:

- the client-server model allows the separate design and construction of objects. This may result in the absence of monitoring facilities or the absence of dynamic management of monitoring in objects. This means that it may not always be possible to stop an existing application, re-write, recompile and re-run it
- when monitoring across federation boundaries it is possible to encounter conflicting monitoring policies which do not allow certain objects to be monitored. For example, some management domains may disallow the activation of monitoring at certain times for reasons of performance, or by certain observers for reasons of security. In general, in federated systems it will not be possible to force system designers and implementers to adhere to common notions of monitoring and management
- communication and partial failures can be the cause of incomplete observability
- the nature of distributed systems will manifest itself in objects being created, migrated, passivated and destroyed dynamically. The scope of monitoring in such systems may be extended dynamically. In such cases it is possible to have monitoring activated in objects which the visualization tool does not know exist, resulting in unexpected messages. In cases where a monitoring session encompasses both short and long lived objects, it is possible for the monitoring session to have missed certain events, making it difficult ti interpret subsequent events.

What is necessary in these cases in order to fully reconstruct the flow of events is additional information about the events in the unobservable objects (Figure 5.1).

**Figure 5.1: Incomplete observability**



## 5.2 Preventive solutions to the incomplete observability case

In general, if monitoring cannot be activated in an object but we can intercept the messages coming in and going out of the object, then this information may be sufficient to deduce the order of events in the object or at least draw partial ordering between the interacting objects. There are several ways of monitoring the object's incoming and outgoing messages: network monitoring, protocols containing causal history information, proxy and witness objects.

### 5.2.1 Network monitoring

One possible solution is to monitor the network which carries messages between the monitorable objects and the non-monitorable object. This, however, is a low level solution which is made difficult by the complexity of communication infrastructures.

### 5.2.2 Appending causal history to the exchanged messages

An alternative solution to the problem requires that the messages to and from the unobservable object must carry sufficient history of previously exchanged messages. Such information will allow the reconstruction of the sequence of events in the unmonitorable object from indirect observations. There exist a number of protocols which carry such information and which can be exploited for monitoring purposes.

#### *5.2.2.1 The CBCAST and Psync protocol*

Another solution consists of the CBCAST protocol [BIRMAN 87]. This protocol adds dependeny information to a message before sending it to its destination. The dependency information consists of a history of previously received messages which are potentially causally related to the message being sent.

It is possible to use communication protocols such as Psync [PETERSON 89], in which the partial ordering of previously received messages is encoded explicitly with each message.

PSync appears to provide a highly efficient form of CBCAST, as it forwards compact summaries of causally related events without the overheads of the message diffusion technique of CBCAST.

### 5.2.3 The proxy concept

In some cases it is possible to insert a proxy object between two interacting objects in order to intercept the messages exchanged between them. The proxy can generate the necessary monitoring messages. Inserting a proxy between two other objects can be done by using the Trader [AR.005 93], for example.

### 5.2.4 The "witness" concept

Another way of solving the incomplete observability problem is to use the facilities provided by groups [AR.002 93]. The external monitor can be made a witness (ie. a silent and passive) member of a group consisting of itself and the unobservable object of interest. Copies of all the messages to and from the unobservable object will be sent to the witness member. The witness member is incapable of taking over the other member's task and does not participate in the interaction between the unmonitorable object and other objects.

## 5.3 Post monitoring solution: using inference and management information

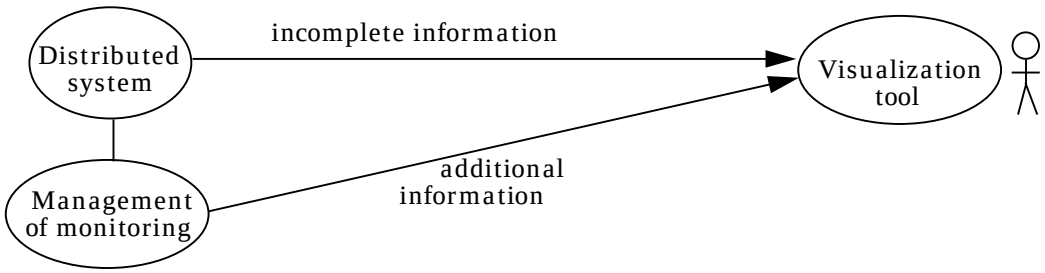
There are two ways by which additional information, necessary for consistent and coherent interpretation of monitoring messages, can be obtained:

- provide inference capability to detect and provide the visualization tool with the missing information (Figure 5.2).

**Figure 5.2: Incompleteness solution using inference to deduce missing information**


- provide a link between the management of monitoring and the visualization tool (Figure 5.3) This approach can help in cases where the scope of monitoring may be extended dynamically during a monitoring session. In such cases it is possible that the management of the monitoring session will have the necessary information about the objects in the scope of monitoring. This may provide the visualization tool with information on objects whose creation has not been announced by a monitoring message, for example (Figure 5.3).

**Figure 5.3: Incompleteness solution using information available to the management of monitoring**




The latter solution places a requirement on the way in which the management of monitoring is organized. It identifies the need to link together the management and the information which it has on what is being monitored, and the visualization tool.

# 6 Monitoring and ordering

## 6.1 Introduction

The problem of ordering stems from the relation between the events being observed, their related monitoring messages and their collection. When monitoring a distributed system it is necessary to collect monitoring messages from different parts of the system. Given the nature of communication in such systems, it is not usually possible to assume that the order in which the monitoring messages are collected will reflect the order in which they were sent. Furthermore, the relationship between the monitored event and the respective monitoring message cannot be assumed to be straight forward, unless certain precautions are taken. For example, making the monitored event and the generation of the resulting monitoring message indivisible.

In order to present the observer with a coherent and consistent picture of the monitored events it is necessary to re-order the monitoring message so that the new order will reflect the sequence of events which took place in the system. In the absence of a global clock or of sufficently accurate set of synchronized clocks, it remains possible to order the message according to their causal relationship.

## 6.2 Preventive solution to the ordering problem

The ordering problem can be prevented by using one of the following methods [AR.010 93] [RAYNAL 90]:

- use of physical global clock
- by using communication infrastructures which guarantee that the order of message arrival reflects the order of sending them. This solution introduces large overheads on message communication and is therefore generally unacceptable.

### 6.2.1 Physical and synchronized clocks solution

One solution to the problem of reconstruction is to include a physical global clock which allows the total ordering of events in the system. Total ordering means that any two events, including causally unrelated events can be ordered. The problem with this solution is that the resulting system cannot be considered distributed anymore [RAYNAL 90].

Another possible solution is to synchronize existing physical clocks. Two methods of synchronizing clocks are:

- probabilistic: these protocols provide a probabilistic guarantee of the accuracy of synchronization. The network time protocol (NTP) used in the ARPA network is an example of this type of synchronization
- bounded: these protocols provide a guarantee that the clocks will be synchronized to within $\pm\Delta t$.

There are, however, several problems with global physical and with synchronized clocks:

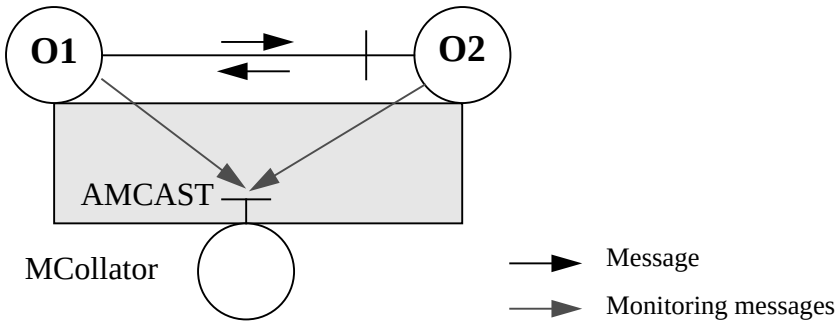
- they can only be obtained to a certain degree of accuracy which may not be sufficient for the required resolution
- increasing the accuracy of a global clock can be expensive.

### 6.2.2 Using communication infrastructure

Using a protocol which guarantees that the order in which the messages arrive at the destination is the same as the order in which they are sent from O1 and O2. Examples of protocols which can achieve this are [AR.002 93]

- atomic ordered casts (ACAST) protocol [BIRMAN 87] [CHANG 84]: guarantess that all messages are delivered in the same order everywhere, even if they are sent independently by different sources (Figure 6.1). (Such a protocol does not have to make use of the fact that there are causal relations between the messages exchanged between O1 and O2.).

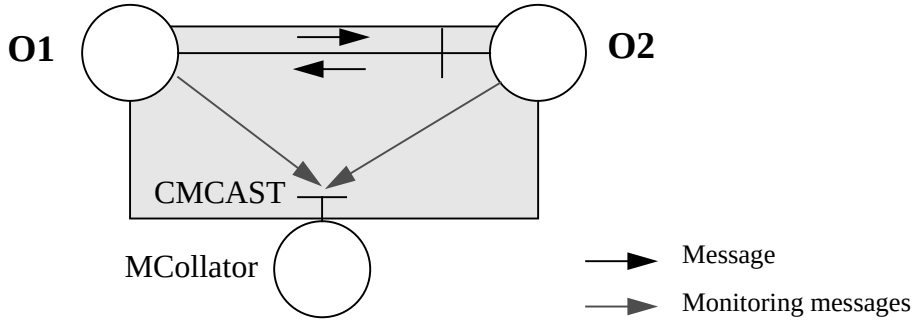
**Figure 6.1: Using an Atomic ordered multicast (AMCAST) protocol to to rder monitoring messages**




- causal multicast (CMCAST) protocol [BIRMAN 87]: this protocol maintains potential causality and also fifo ordering (i.e. CMCAST delivers messages in the order originated by its source) at all overlapping destinations for all potential cause and effect messages. This can be simplified since in most cases there will only be a single destination (Figure 6.2). There are several ways of implementing CMCAST, all of which equire the protocol to add some additional dependency information to a message m before sending it to its destination. This information consists of a history of previously received messages which are potentially causally related to the message being sent. Such messages are always absorbed first by the recipient before ordering the recipt of message m. If any messages in the history has already been received, then the destination simply ignores them. The causal relation can be obtained from the messages exchanged between O1 and O2.

There are problems with the protocol level approach. The strongest criticism is that the performance penalties, and hence the interference in the behaviour of the observed system, may be unacceptable. Furthermore, There may be situations where the required protocols cannot replace protocols already in use.

**Figure 6.2: Using a casual multicast (AMCAST) protocol to order monitoring messages**

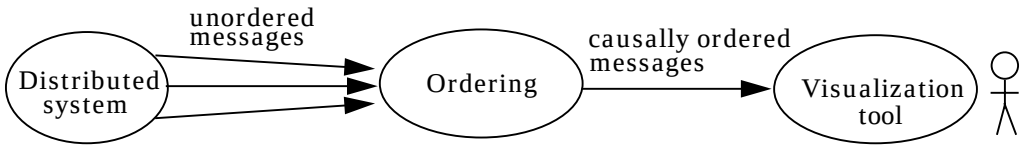



## 6.3 Post monitoring solution: using causal relations

In general, all the other solutions are based on the idea of potential causality introduced in [LAMPORT 78], and pioneered with communication protocols by [BIRMAN 87]]. The term potential causality derives from the principle that each cause has to precede its effect. The principle of such cause and effect can be applied to events in distributed systems, where the events are the sending of a message and the receipt of a message. Partial ordering specifies which events occurred before or after other causally related events. In cases where events are not causally related, however, all partial ordering can provide is an indication as to which events have occurred concurrently with other events, that is either before, after or simultaneously. In many cases, a simpler solution which provides partial ordering of events will be sufficient.

The way in which the causal ordering solution is implemented is shown in (Figure 6.3). This solution requires sufficient information in the monitoring

**Figure 6.3: The ordering solution**




messages to enable relating the respective send and receive events to be correlated, thereby deriving the partial ordering of events.

In order to enable the ordering algorithm to relate the monitoring messages which depict the respective send and receive events, it is necessary to have the following information in each monitoring message which describes the interaction between objects:

- monitoring message source and destination
- event type - send/receive
- object identity of message source
- message identity: unique within the object
- local clock: either logical or real time

Causality based re-ordering methods consists of two main parts:

- the process within the observed system which generates the information necessary for subsequent re-ordering

- the re-ordering process itself whereby monitoring messages are ordered according to the causal relationships among the events they represent.

Four methods are discussed in [AR.010 93], [MATTERN 89], [SCHWARZ 92] and [FIDGE 91]:

- Simple method
- Lamport's Logical clock method
- Vector time method
- Causal history method.

Each method fits a different (set of assumptions about the) environment, has different advantages and disadvantages both in terms of the effect on the behaviour of the system, the ease of ordering, and the information which can be deduced from the monitoring messages.

From the point of view of distributed system designers it is important to decide which method is appropriate and incorporate it in the system. Agreement shuld be reached concerning the minimal method which would enable monitoring across federated system boundaries. This has imp0lications for dependability.

# 7 Monitoring and the visualization tool

## 7.1 Inrtoduction

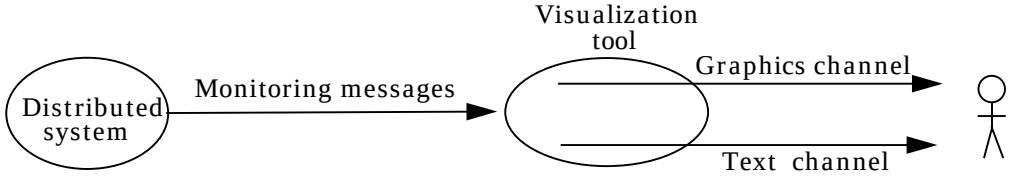
Where both the preventive and post monitoring solution do not solve the problem of incomplete or unexpected information, there will be a need to cater for the monitoring message in the visualization tool. For example, in the case of unexpected error messages arriving from unknown parts of the system.

## 7.2 Processing inside the visualization tool

A programmable visualization tool [HORNE 90] can be made to cater for unexpected events and error messages in a number of different ways:

- by doing minimal amount of interpretation of an unexpected or incomplete messages, possibly displaying it to the user in its original form, e.g. text output channel which can be integrated with the graphical output of the tool (Figure 7.1)
- by notifying the observer of a problem and any steps taken to resolve it. Within this option it is also possible to make any decisions, taken by the pre-processing mechanism or tool, explicit
- by allowing interactive user participation: allowing the observer to make decisions concerning unexpected events. Combined with the previous option of making decisions explicit, this can be used to provide a roll-back facility
- there may be a need for a direct channel of communication between the visualization tool and the manager of a monitoring session in order to facilitate the transfer of management information to the visualization tool.
- role-back facility: to cater for late arrival of messages and for cases where the decision taken by the tool proves to be erroneous at a later stage..

**Figure 7.1: Passing messages verbatim without attempting to interpret them graphically**



# 8 Monitoring and the presentation of distributed system behaviour

Note: This chapter requires further work and a major re-write!

## 8.1 Introduction

In both centralised and distributed systems there is a need to make the large amount of data produced during a monitoring session presentable to the user. The size and complexity of distributed systems makes this task harder. This is reflected in the complexity of the system models which observers wish to construct from the observed events. There is therefore a need for tools to help with presentation of monitoring information in an intelligible manner.

The purpose of monitoring dictates the models which the observer will need from the visualization tool. Thios in turn dictates what monitoring information is required from the system. Thus there is a need to identify the models of the system wqhixch are likely to be common across many distributed system platforms and specify the information required for their vissualization. This in turn would explain the designer of the monitoring infrastructure what is necessary in the ds.

## 8.2 Graphical presentation issues

### 8.2.1 Models and presentation techniques of distributed system behaviour

Mcdowell and Helmbold [MCDOWELL 89] list textual presentation, time-process diagrams, animation and multiple windows as the main techniques for displaying monitoring information.

The following are a number of examples of models of system behaviour which have been found to be useful:

- the interaction or the client-server model: this model attempts to construct a picture of the objects which exist in the observers sphere of interest, the bindings between the objects, and the interactions which take place through the bindings. By using a graphical display, this can be visualised as an animated sequence of events
- the interaction timing model: this model shows the interactions between two or more objects using time as a linear coordinate. This is a communication oriented view of interactions
- the object state model: this model shows the relevant state of an object and the effect of interactions with other objects on the state
- engineering model: this model shows the interactions between an object and its support environment.

Note: See [JOYCE 87] for algorithm versus state or activity versus object view. Also [MCDOWELL 89] for treatment of the use of graphics in debugging.

The observer may wish to switch between different models and also to combine them into more complex views of system behaviour. It is the purpose of monitoring which will dictate what model of the system the observer wishes to construct from the observed events.

## 8.3 Standards for graphical visualization

It is possible to visulaize distributed systems in different ways. For example, “blob and stick” diagrams are often used [ISO-439 91] to show interactions between objects. However, a hierarchical diagram showing ther lreations between objects in a system is also a valid representation.

## 8.4 Information for visualization and granularity of monitoring

### 8.4.1 Detail and granularity of monitoring

The monitoring infrastructure has to be able to provide sufficient information to allow distinguishing between system entities in a meaningful way. For example, it is possible to show interaction between objects without the detail of which interface the invocations are directed to, or it is possible to show which interface they are directed to. The latter case would require messages to contain information about the interface and the object it belongs to.

### 8.4.2 Inference

It is possible to use a message describing a send event to visualize the interaction between two objects, provided the id of the receiver is given in the message. This however, may be misleading since it implies the reception of the message by the receiver; this may not necesasrily be the case in a distributed system.

A different situation arise when the visualization tool corelates a message depicting a send event with its respective receive event and displays the interaction between the objects. The same thing applies to the reply to the invocation.

For each case, the number of messages required and the information in the messages isdifferent.

### 8.4.3 Graphical representation

Announcements: do not have a reply, therefore they can be displayed without awaiting the message depicting the receipt of the message

## 8.5 Time and ordering

See requirements in *RC.462: Distributed Infrastructure Requirements for Management and Monitoring.*

# 9 Conclusions

Wherever possible, preventive solutions to the problems of visualization of distributed systems are strongly recommended. Thus, agreement concerning monitoring and its exploitation is to be encouraged as much as possible. Where convergence onto a single standard cannot be reached, it may be possible to overcome the differences by using translation.

Monitoring infrastructure designers and implementors are encouraged to reach agreement on the set of minimal monitoring events necessary, what information is required in monitoring messages, formatting and encoding issues.

Agreement on monitoring management protocols may also be useful where additional information available to management facilities is necessary to make subsequent messages interpretable.

As far as the visualization tool is concerned, there is a requirement to the observer of special events or problems through the display of monitoring messages in textual form as well as graphical form. Making any monitoring message analysis decisions explicit may be useful, and user interaction is also to be encouraged.

In order to be able to derive a causal ordering of events it will be necessary to incorporate the appropriate information in monitoring messages.

A way of connecting together the management and the information which it has on what is being monitored, and the visualization tool, has been identified. This has to be addressed by the designers of the distributed system and the visualization tool.

Management facilities may be driven by the visualization tool itself, thus allowing monitored objects to be managed as well. This will provide graphical tools to do the different types of management for which monitoring is used.

# References


[AIM 90]
RM.097: ANSAware 3.0 Implementation Manual, APM Ltd., Poseidon House, Castle Park, Cambridge CB3 0RD, UK (February 1991).

[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.010 93]
Hoffner, Y. “AR.010: Monitoring in Distributed Systems”, Architecture Projects Management Ltd, Posiedon House, Castle Park, Cambridge CB3 0RD, UK (January 1993).

[BIRMAN 87]
Birman, K. "Exploiting Virtual Synchrony in Distributed Systems", ACM Operating Systems Review, 21(5), 123-138 (November 1987).

[BIRMAN 91]
Birman, K.P., "Maintaining Consistency in Distributed Systems", Cornell University, Dept. of Computer Science (November 1991).

[FIDGE 91]
Fidge, C. "Logical Time in Distributed Computing Systems", IEEE Computer, 24(8), 28-33 (August 1991).

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

[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 (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).

[HORNE 90]
Horne, G., Statham, A. et al “Demon: An Instrumentation System for CSA”, MARI Computer Systems Ltd., Old Town Hall, Gateshead, Tyne & Ware NE8 4HE, (March 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).

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

[LINDEN 92]
Linden, R. V. D. “AR.02.00 The ANSA Naming Model”, Architecture Projects Management Ltd, Poseidon House, Castle Park, Cambridge CB3 0RD, UK (1992)

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

[MATTERN 89]
Mattern, F. "Virtual time and global states of distributed systems". In Parallel and Distributed Algorithms, 215-226. Elsevier Science Publishers B.V. North-Holland, (1989).

[OSKIEWICZ 90]
Oskiewicz, E., Warne, J. & Olsen, M. "A Model for Interface Groups", TR.009, APM, Poseidon House, Castle Park, Cambridge CB3 0RD, UK (September 1990).

[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).

[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).

[TR.040 92]
Hoffner, Y. “RC.331: Re-ordering methods: Design and Implementation Issues”, Architecture Projects Management Ltd, Poseidon House, Castle Park, Cambridge CB3 0RD, UK (1992).

[TR.39 93]
Hoffner, Y. “TR.039: Management in Object-Based Federated Distributed Systems”, Architecture Projects Management Ltd, Posiedon House, Castle Park, Cambridge CB3 0RD, UK (January 1993).

[SAMPSON 92]
Sampson, P. “What to Do When Something Happens”, Document P/P/ DOC02.2/M14/1.0, MARI Computer Systems Ltd., Old Town Hall, Gateshead, Tyne & Ware NE8 4HE, (January 1992).

[SCHWARZ 92]
Schwarz, R. & Mattern, F. “Detecting Causal Relationship in Distributed Computations: In Search of the Holy Grail”, Department of Computer Science, University of Kaiserslautern, D-6750 Kaiserslautern, Germany (1992).

[WINOGRAD 72]
Winograd, T. “Understanding Natural Languages”, Academic Press (1972).

[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).