DISTRIBUTED SYSTEMS MANAGEMENT AS A GROUP ACTIVITY

Graham Dean, Tom Rodden, Ian Sommerville and David Hutchison

Department of Computing

Lancaster University

Lancaster

LA1 4YR

UK

Email: graham@comp.lancs.ac.uk

It is important to consider systems management as part of a whole organisational management strategy and, as such, to be aware of the impact of people on management. Within a flexible management framework we bring people within the terms of reference associated with systems management.

INTRODUCTION

Traditionally, network management has concentrated on the lower and physical levels of networked systems. For example, most commercial management packages provide low level information giving statistics on network load and traffic, workstation load and event reporting. This information can then be used by a skilled network administrator to aid decision making in the formation of management policies, e.g. the number of users allowed on the system at any one time. Information of this form has proved invaluable to managers in the past. However, as networked computing systems have become common, a need for higher level system information has become apparent. Details regarding hardware and software configurations, roles taken on by people involved in the system and specific policies implemented become important to effective management of the system. This is presently done in a rather ad hoc manner, with little or no resources being allocated to the configuring and planning of networked systems. The **I**ntegrated **E**nterprise **M**anagement **F**ramework (**IEMF**) proposed in this position paper addresses the issues involved in supporting the management of large scale (enterprise wide) networked computer systems.

As in all forms of management, a simplistic view of systems management would suggest that it was not a difficult task. If well defined procedures were followed, effective management policies defined and fully implemented and all relevant information recorded then management becomes a simple matter of routine. Unfortunately, this does not match the reality found within most organisations and leads us to believe that a comprehensive and flexible management framework is needed. We stress the word *flexible* because we need to accommodate many different working practices and note that a lot of systems management is actually *management by exception*.

This paper briefly presents a framework to support systems management developed from the observation of actual management within our local environment. A range of different activities were found to be important in effective systems management, these are briefly reviewed in the following section. The model of management underlying the framework and the framework itself are described in section 3. Finally we consider how existing work on network and systems management relate to the framework and supporting system developed to date.

SYSTEM MANAGEMENT ACTIVITIES

The success of low cost workstations has led to an increased prominence of computer networks within most organisations. This has transformed the way in which computer systems are used and the forms of work supported by these systems. Equally, systems management has been transformed from an isolated activity focused on a remote central machine to a crucial factor of users' everyday work. As the work of users and more recently of groups has become centred on computer systems so the effective management of these systems has become intertwined with the practises of everyday work. The net result is that an ad hoc approach to systems management is no longer tenable within modern organisations [Dean et al 1992].

This is compounded by the dramatic growth of modern networked systems. Sites have rapidly grown from small scale endeavours, where a single manager was responsible for management, into large scale enterprises requiring multiple managers. Any effective management framework must be aware of the large number of legacy systems that exist and must be catered for. There may well be strong dependency relationships between the various portions of an enterprise and to enable managers to effectively carry out their responsibilities we need to provide support for managers to cooperate together and coordinate their activities.

Understanding Management Activities

To illustrate the problems that can occur when adequate support for management is not provided we have undertaken a detailed study of systems management within the domain we are most familiar, a University Computing Department. We also believe the network studied is sufficiently illustrative of networks within most organisations that it provides a set of results which are generally applicable. The department network studied has grown from a small ethernet with a handful of specialised workstations to 5 connected networks supporting over one hundred different machines. The network is also connected to a campus network serving a number of different departments and ultimately to the outside world via internet. The fact that management problems are evident within this relatively small environment serves as an illustration of the multitude of problems that can occur in larger organisations.

Management of the distributed system constituted by the network evolved over time in an ad-hoc manner, with procedures and policies developed as necessary. A consequence of this approach to distributed system management is that a number of crucial problems occur:

- There is a lack of recorded information detailing the changes that have been made to the system.
- Different personnel are assigned to the same task with no effective means of coordinating their activities and actively supporting the cooperation between them.
- No provision for the support of policies is provided. At the macro level many organisations are only too aware of the problems of not developing and adhering to network policies. Unfortunately, the same problems occur at the micro level but with little support available to facilitate the development and implementation of these policies.

A range of constituent activities are involved in distributed systems management, these activities involve a variety of different interactions. In particular, the activities surrounding three different types of interaction can be highlighted as important to distributed systems management.

Interaction between the administrator and the system : A major issue in systems administration surrounds the interaction between the distributed system and the system manager responsible for this system. Obviously, this interaction is the principle means by which a distributed systems is controlled. However, this interaction is seldom recorded for future use by systems managers and rationale for particular actions are subsequently lost.

Consider for example, the setting of filestore quotas within the research department studied. While quotas were applied within the department an initial inspection suggested no pattern to the levels set for particular individuals. However, discussion with the systems administrator provided sufficient rationale for most of the levels set. The problem in this case was not the rationale applied by the administrators but the fact that the reasons for these decisions were not recorded and available within the system.

Interaction between user and system administrator: The ubiquitous use of computer systems to support work is such that the activities of users are becoming increasingly important to the effective management of distributed systems. In particular understanding and supporting the interaction between users and system administrators is crucial to systems management. Consider for example the problems of version management generated by the particular needs of users.

Within the department studied many different software applications were available and many different versions of these applications existed. This effect was largely due to the working practices of a research department where individual members of staff had autonomous control of the procurement, installation and maintenance of software. In this case each user is, to a limited extent, a system manager. In some cases it is essential to allow multiple versions of software to co-exist. This is the case when the department needed to provide demonstrations of systems developed using software no longer supported (this is true in the case of Ph.D. students who need to demonstrate software to external examiners, in some cases a considerable time after they have left the department).

Interaction between different system administrators: A final issue is communication between system administrators. The department studied currently employs two system administrators who work together to provide a valuable management service. Sometimes their management activities are completely independent and both administrators can carry on with their own task without unduly worrying about the other's. In most cases however, the opposite is true. There is either a strong dependency relationship which exists between the two activities (e.g. a temporal relationship, one activity must be completed before the other is started) or, more commonly, a dependency relationship which exists as a side-effect of one of the activities (e.g. installing an updated object-code library may create a need for applications which use the library to be re-compiled).

An interesting problem which arose within the department illustrates the problem associated with this form of interaction. Generally, users of the computing system have their mail delivered to a central machine which can be accessed via a large number of terminals, workstations etc. In some cases people choose to have

their mail accounts on local workstations. In order to facilitate this arrangement one of the system administrators has a policy of running a program for distributing a master mail configuration file across machines. This was done with the intention of keeping the mailing system configuration files consistent across the network. By using tool support to automatically distribute the configuration files the administrator was relieved of a lot of tedious work.

Unfortunately, the other administrator was unaware of this policy and directly implemented changes to local configuration files when asked by a user to move his mail account. This of course worked fine for a while until other changes were made to the mailing system, via the master configuration file, and distributed to workstations. This had the net effect of over writing all the previous changes to the configuration files. This simple example shows why an exclusively technical approach to systems management will fail because not enough attention is given to the current working practices of management staff. The framework we have developed builds upon the results of our studies and incorporates two notions central to distributed systems management, the *recording of information* and the *support of the interaction* needed to effect systems management.

A MODEL FOR DISTRIBUTED SYSTEMS MANAGEMENT

In order to work together and coordinate their activities managers need to be provided with a supporting management framework. Fortunately the system been managed provides an ideal focus for much of the activities undertaken. In fact, a simple model of a number of systems managers cooperatively interacting with a shared representation of the system becomes a persuasive model of management. Two important concepts are embodied within this model of work:-

- An explicit representation of system management information.
- A effective set of mechanisms set around the shared system model to coordinate the various interactions central to distributed systems management.

The shared system model provides system managers with an abstraction of system resources with which they can interact. The system model in addition to including resources such as printers, workstations, etc., includes information about organisational issues such as responsibility, expertise etc. In fact this model can be thought of as a representation of appropriate portions of the *enterprise* within which the system is set.

As part of the IEMF we have been developing a notation for describing system resources and the underlying concepts that we feel are necessary for the support of system management activities. **EMANUEL** (**E**nterprise **Man**agement and Q**ue**ry **L**anguage) is a language which draws heavily on the traditions of module interconnection languages, of which SySL [Sommerville & Thomson 1989] exerts the greatest influence. Module interconnection languages were originally intended to support the modelling of large, complex software systems in a formal notation. This notation would then be used to drive the system build process. We require a notation for describing large, complex networked systems, resources in the network and the various relationships that exist. We then want to use that information to drive management activities and management applications. Module interconnection languages have thus proved to be an excellent starting point.

The information that we require to capture in the information model is categorised into three areas. We wish to model firstly resources (such as workstations, printers, disk drives etc.), secondly relationships between resources (e.g. dependency relationships) and finally, information about the enterprise. The majority of information models used by management systems do not address this final category. In order to promote a consistent view of system information, it was important to us that we used the same notation for all three levels.

Concepts that were discussed when talking to system managers were *responsibility*, *expertise* and *interest*. It will be noted that these concepts inherently refer to human users of the systems. We explicitly model these concepts in our information model as relationships between users of the system and system resources. Indeed, relationships that exist between various entities are considered so vital that we introduce two specific types of relations, *descriptive* relations and *prescriptive* relations.

Descriptive relations are used purely as modelling facilities, which effectively capture meta-information about an enterprise, and are used most frequently when browsing the system model through the graphical editor. Prescriptive relations prescribe the system as we want it. That is, if we wish to change the physical system in some way, we define prescriptive relations in the information model corresponding to these changes and then submit the model to management applications.

We also use the same notation for generating simple queries over the model to aid the generation of personal views of the system Without this facility, large complex system descriptions are hard to comprehend, users can neither gain an overall picture of the system nor isolate the specific details they are interested in.

As well as capturing information using the formal notation EMANUEL, we have discovered a need for informal information about system entities. Often system managers merely wanted to record some informal comment about a resource (e.g. possible bug-fix due out sometime later this year) or as a means of, very

informally, coordinating their activities. We responded to this need by providing an annotation mechanism allowing annotations to be attached to system entities. At present, these are purely free form text with no attached semantics. It may be fruitful to explore the notion of *semi-structured* annotations which we can process in some way. Another area we intend to explore is the possible migration of information from annotations to the formal information model.

THE IMANUEL NOTATION

Rather than go into the details of the notation we use some simple examples to express the concepts found in system management which we are interested in modelling. The underlying philosophy of EMANUEL is to be simple, yet powerful to be capable of expressing the relationships needed for successful system management. Entities are described at two levels, at the first level structural details are expressed (though, through the use of *refinement* constraints can be attached; this is useful when modelling organisational policy, see figure 3), at the second level these templates are filled in with the necessary details to specify particular components.

```
structure MACHINE is
    Name,
    Supplier,
    {Monitor}+,
    Keyboard,
    Processor,
    [Disk],
    Memory,
    Operating_System,
    Network_Interface
end structure
```

```
structure NETWORK is
    Name,
    Topology,
    Speed,
    Medium
end structure

relation CONNECTED_TO is
    domain (NETWORK, MACHINE),
    range NETWORK
end relation
```

Fig. 1: Generic Structure and Relation Modelling

To illustrate the template facility of EMANUEL the figure above gives a simple description of two generic entities (MACHINE and NETWORK) and a relation definition CONNECTED_TO. The description of the two entities expresses the composition relationships that exist and any associated cardinality for individual slots. In the description of the MACHINE we see that it may have one or more monitors ({Monitor}+) and an optional disk ([Disk]). The CONNECTED_TO relation is defined as a relation which could exist either between a MACHINE and a NETWORK or, alternatively, as a relationship between two NETWORKs.

```
component dcl-sparx1 : MACHINE is
    Name => "dcl-sparx1",
    Supplier => "Sun Microsystems",
    Monitor => Sun_Colour,
    Keyboard => Sun_Keyboard,
    Processor => sparc,
    Disk,
    Memory => "16Meg",
    Operating_System => "Solaris 1.0",
    Network_Interface => Ethernet_Controller,
    relation CONNECTED_TO (backbone)
end component
```

```
component backbone : NETWORK is
    Name => "Lancaster Campus Backbone",
    Topology => "Ethernet Bus",
    Speed => "10 Mbits/sec",
    Medium => "Co-axial"
end component
```

Fig. 2: Component Relationships

Fig. 2 illustrates concrete realisations of the pre-defined templates. The relationship which exists between the realisations (**component**) and the templates (**structure**) is an "is-a" relationship; we can see that both dcl-sparx1 is-a machine and backbone is-a network. Values are associated to the various slots in the template definitions and relationships between components expressed. In this case it can be seen that the machine "dcl-sparx1" is connected_to the network "backbone". The existence of such a relationship could also be expressed within the "backbone" component (where the relationship would be read as HAS_CONNECTED); no constraints are imposed on the positioning of information, it is simply placed where the user feels the information best expresses the relationship between the two components.

structure USER_ACCOUNT **is**	**stucture** SE_ACCOUNT **:** USER_ACCOUNT **is**
Name,	Name,
User_ID,	User_ID,
Group_ID,	Group_ID => 68,
Quota,	Quota => "40Meg",
Filestore,	Filestore,
Mount_Partition	Mount_Partition => "/use/se"
end structure	**end structure**

Fig. 3: Specification of Simple Policy

As mentioned in the first section of this paper, the management of organisational policy is considered to be of vital importance. The recording (and to some extent, the implementation) of policy can be accomplished through the use of the EMANUEL modelling language. In the above figure we have a template USER_ACCOUNT which is refined by the template SE_ACCOUNT. The structure SE_ACCOUNT inherits all the slot names of the structure USER_ACCOUNT and adds some constraints on the values to be associated with the slot names. We can see that the use of SE_ACCOUNT expresses the policy "All members of the Software Engineering research group have a quota of 40 Megabytes and their home directory is to be mounted on the partition "/usr/se".

The use of EMANUEL to express policy in this way has two important consequences, firstly the recording of (sometimes very informal) policy is encouraged and made accessible to other people, secondly management applications (such as programs to create user accounts)have access to this information via an API (Application Programming Interface).

structure USER **is**	**component** graham **:** USER **is**
Name,	Name => "Graham Dean",
Login,	Login => "graham",
Research_Group,	Research_Group => "Software_Engineering",
Office	Office => "B29",
end structure	**relation** HAS_RESPONSIBILITY (c++, Isode),
	relation HAS_EXPERTISE (Miranda),
	relation HAS_INTEREST (X-windows)
	end component

Fig. 4: User Modelling

One of the interesting consequences of freely available software is that to a large extent every user of a system is to some part a "system administrator", this is particularly true within a research establishment such as a University department. One effect of this is that management knowledge is distributed throughout an organisation with people assuming different roles depending on particular contexts. Using EMANUEL we have tried to capture some of relationships that exist between individual users of a system and the various resources found within.

• Responsibility

For every resource within a system it is important to be able to track responsibility for it. If this information is not present then a situation arises where users of a system do not know who to contact when

problems arise. A consequence of this is that time will be wasted both by the user, trying to contact the relevant person, and by other system administrators who will, inevitably, be caught up in the situation.

• Expertise

In some ways expertise can be considered similar to responsibility (although, obviously, the presence of responsibility in now way implies expertise!), perhaps the major difference is that expertise is *volunteered*. In this way users of a system are more likely to offer their skills if no specific responsibility is attached to it.

• Interest

Expressing an interest in a resource indicates that you wish to be involved in tracking any new developments to that resource, for example a bug fixes or version update. Using this construct it is relatively simple for system administrators to inform any users who may have some form of dependency relationship with system resources that changes are being made.

IEMF TOOLSET ARCHITECTURE

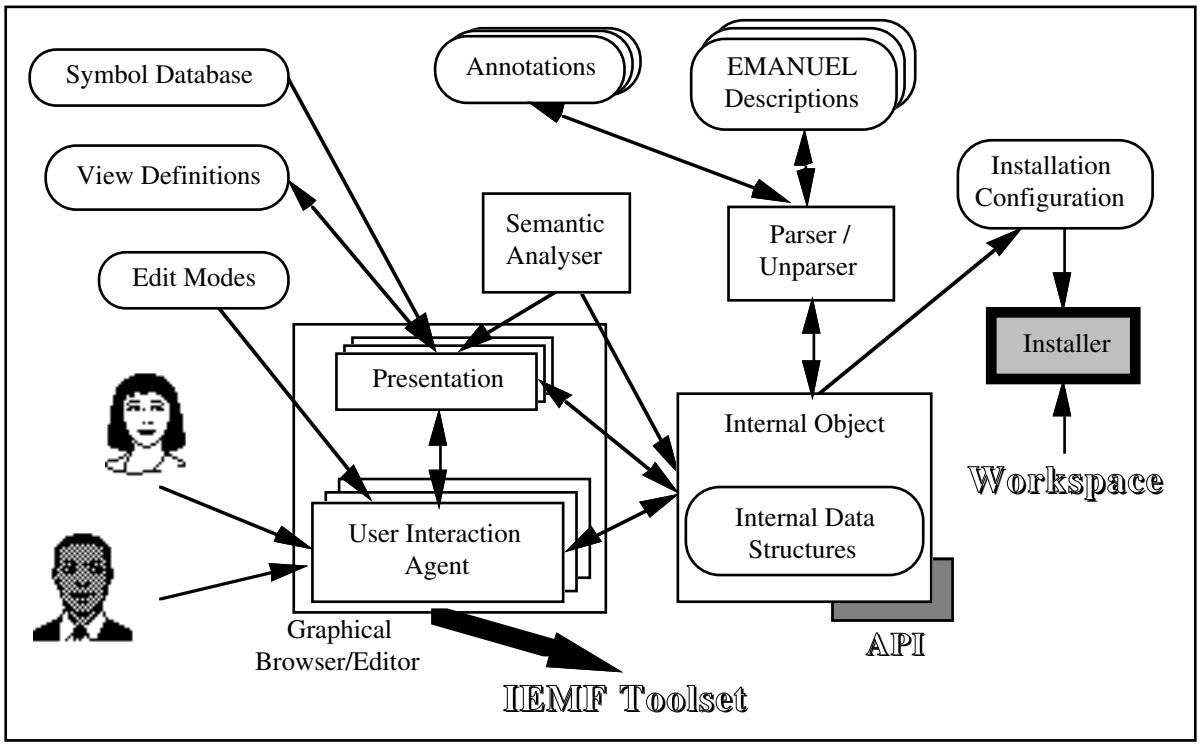

Fig. 5: Toolset Architecture

The IEMF toolset is made up of various functional components, each of which is described below:

• Graphical Browser/Editor

Central to the philosophy of the management framework we propose is the accessibility of relevant management information. As well as the information being readily accessible it should also be presented in a easily understandable manner.

• User Interaction Agent

The UIA (User Interaction Agent) performs all actions on the internal data structures on behalf of the user. The use of UIAs allows the coordination of operations by multiple users and enables different presentations of the system description to be kept consistent.

• Presentation

The presentation component is responsible for all the graphical output of the underlying EMANUEL description. One of the major problems found in any management system is the immense amount of information that is available, this can result in users suffering from information overload. A

mechanism for filtering this information is provided by the presentation component in the form of view definitions. This allows users to specify the type of information they are interested in and enables them to focus on the relevant, important information (e.g. one such view would be the physical network, specified by choosing MACHINE entities and CONNECTED_TO relationships).

The concept of management roles (e.g. Printer Administration) is implemented via the view definition facility and the use of edit mode configuration files. The view definition highlights the particular information related to printer administration and the edit mode acts as a capability mechanism for user access. This approach is particularly useful when system administrators with specialist knowledge are unavailable and work has to be delegated to staff without such knowledge. Using these facilities allows experienced system administrators to build up a form of organisational memory which can be accessed by other people.

• Internal Object

The internal object is the internal representation of an EMANUEL description used by the graphical browser/editor. It provides a set of services via a well defined interface enabling the de-coupling of the internal data structures from other functional components within the architecture. This approach provides us with two advantages, it allows a simple implementation strategy for consistency between multiple views of a system and also provides a convenient means of providing an API interface.

• Semantic Analyser

The semantic analyser checks all editing operations on the internal data structures and both ensures the consistency of the description and the likely impact of changes being made. This allows system administrators to perform "what-if" scenarios and see the likely consequences, this is of major concern when dealing with a system which forms part of a much larger network.

• Parser

The parser is used to transform the EMANUEL description from its textual form into a form suitable for internal representation (making use of the services provided by the internal object). It is also possible to un-parse the internal representation to purely textual form. This facility was included to allow new tools in the IEMF toolset to work from a static EMANUEL description if so required. An annotation file also forms part of the parser's input, this allows users to add annotations to system entities in two ways. An annotation can be added to an entity directly through the graphical interface or via an email interface allowing interaction through a non-graphical terminal device.

• Installer

The installer is one example from a set of tools which will be implemented as part of the IEMF toolset. Installation of software is still a complex operation due to heterogeneous platforms and local configuration conventions. The installer's functionality will include the distribution of software from, as well as to, a distributed environment as well as a centralised one.

CURRENT STATUS

The project has completed the initial development of a modelling notation to represent systems management information. This notation has been used to construct a model of the system first studied to generate the system management requirements driving the project .

A language analyser has been developed for this notation which constructs a shared system model from a description expressed in the notation. An associated toolset is currently being constructed to support the activities of systems management. These tools are developed using C++ on Sun workstations and exploit the graphical capabilities of X-Windows . These tools include:-

- A language analyser which checks the model for inconsistencies and allows the impact of potential changes to be assessed in advance of them being made.
- A multi-user editor [Ellis et al 1988] which allows a number of systems administrators to generate different views of the system model appropriate to the tasks they are responsible for.
- A graphical browser which allows users to explore the information within the system and interact with the system administrator by attaching annotations to appropriate components of the systems model.
- A software system installer.

RELATED WORK

Studies of management activities have been performed by both Hodge [Hodge 1992] and Kolst [Kolst 1991]. Hodge's study took place in the framework of the TOBIAS project which is specifically looking at support for system administration. Kolst's motivation for writing his article was to illustrate how system management encompasses a wide range of tasks with which system managers must familiarise themselves.

Network management has been the focus of much work in the TCP-IP and ISO/OSI domains. SNMP [Case et al 1990], and its successor SNMP V2 (sometimes known as SMP) [Case et al 1992], is the dominant protocol in the TCP-IP world, whilst CMIP [OSI 1990] is its equivalent in the world of ISO/OSI. Both protocols make use of a Management Information Base (MIB) with associated guidelines for the Structure of Management Information (SMI). MIBs provide a limited form of abstraction in which to construct an information model.

Projects looking at systems management, as opposed to network management, include IBM's SystemView [IBM], DEC's Enterprise Management Architecture [Kauffels 1992] and AT&T's Unified Network Management Architecture (UNMA) [Kauffels 1992]. The philosophy of all these projects is to provide a common management framework for management application providers and users to work within. The Open Software Foundation are also active in the area of systems management with their Distributed Management Environment [OSF 1991].

Work looking most closely at system administration information is that of Moira [Rosentein et al 1988], part of MIT's ongoing Project Athena, and that of Welch [Welch 1991]. Moira is directly concerned with giving support for system administration level activities through the use of an information model with which administration applications interact. Welch also proposes that this level (system administration level) of information should be represented as computer system schemas with which applications can interact. Finkel [Finkel & Calo 1992] makes the point that we require a well defined information model with which applications can interact. The RODM (Resource Object Data Manager) provides an object-oriented view of system resources with data and associated operations over that data encapsulated inside an object definition. NetMATE [Sengupta et al 1991] also takes the object-oriented approach in defining a model for network management, although in this case the information captured relates to low-level network resources.

CONCLUSIONS

In conclusion we feel that there is a need for support for communication and coordination between systems managers based around an explicit representation of the system being supported. Other management frameworks have focused on the technical issues of management support and have largely ignored some of the organisational issues. We see our work as complementing these technical solutions with support for systems managers to work together towards a common goal. The shared system model promotes the sharing of information throughout the enterprise whilst still allowing individual managers their own personal conceptual view of the system. Through the use of the modelling language EMANUEL and the supporting management toolset we provide the conceptual framework IEMF for system managers to work within. The issues that arise from this view of systems management would provide valuable input to the work of the ODP [ISO] (Open Distributed Processing) standards work, especially in the area of the enterprise viewpoint, one of the five viewpoints used in the work of the ODP.

ACKNOWLEDGEMENT

The work reported in this paper is being supported by the UK Science and Engineering Research Council's Specially Promoted Programme in Integrated Multi-Service Communication Networks under grant number GR/F 62674.

REFERENCES

Case, J.D. et al, "Introduction to the Simple Management Protocol (SMP) Framework", Internet-Draft, October, 1992.

Case, J.D. et al, "Simple Network Management Protocol", *Request for Comments 1015*, May 1990.

Dean, G. et al, "Cooperation and Configuration within Distributed Systems Management", *Proceedings of the International Workshop on Configurable Distributed Systems*, Imperial College, London, 1992.

Ellis, C.A., Gibbs, S.J. and Rein, G.L., "Design and Use of a Group Editor", *MCC Technical Report No. STP-263-88*, MCC, Austin, Texas, 1988.

Finkel, A.J. and Calo, S.B., "RODM: A control information base", *IBM Systems Journal*, Vol. 31, No. 2, 1992.

Hodge, D.R., “A Dependable Approach for Managing Very Large Distributed Systems”, *PhD Thesis (in prep)*, University of Newcastle upon Tyne, 1992.

IBM Corporation, “IBM SAA: SystemView concepts”, SC23-0578.

ISO/IEC JTC1/SC21 WD7 (N309-N315), Drafts of standard documents on Open Distributed Processing

Kauffels, F., “Network Management: Problems, Standards and Strategies”, Addison-Wesley, 1992.

Kolst, R., “What do System Administrators Really do?”, *Communications of the ACM*, Vol. 34, No. 12, December 1991.

Open Software Foundation, “Distributed Management Environment: Rationale”, October 1991.

OSI, “OSI - Systems Management Overview”, *DP 10040 (ISO/JTC1/SC21 N4865)*, June, 1990.

Rosentein, M.A. et al, “The Athena Service Management System”, *Usenix Conference Proceedings*, MIT Project Athena, Winter, 1988.

Sengupta, S. et al, “An Object-Oriented Model for Network Management”, *in Object-oriented databases with applications to CASE, networks and VLSI CAD*, Gupta, R. (Ed.), Prentice-Hall, 1991.

Sommerville, I. and Thomson, R., “An Approach to the Support of Software Evolution”, *The Computer Journal*, Vol. 32, No. 5, 1989.

Welch, B.B., “A Proposal for Computer System Schemas”, *Proceedings of I-WOOOS’91*, Palo Alto, California, October 1991.