Domains: A New Approach to Distributed System Management

D.C. Robinson

Digital Equipment Co.
DEC Park II
Imperial Way
Reading, UK

M.S. Sloman

Dept. of Computing
Imperial College
180 Queens Gate
London, UK

ABSTRACT

Distributed computing systems (dcs) are increasingly becoming an integral and strategic part of organisations. Research in dcs has concentrated on languages and operating systems which address the problems of implementing distributed systems, but has not considered the problems of long-term management of distributed systems. This paper presents a new approach to distributed system management. The central concept is the Domain Model, which allows administrators to specify and structure management policies. As such, it provides a uniform interface across different functions of management while being able to scale and handle multiple organisation systems.

1. INTRODUCTION

Today, modern computer systems are not just single machines in the data processing department. The emergence of high speed Local and Wide Area Networks permits the construction of computer systems which are distributed throughout an organisation and may even span organisation boundaries.

Distributed Computer Systems (dcs) are increasingly becoming an integral and strategic part of organisations [Baker 86]. They are part of the overall organisation structure providing services used by the rest of the organisation. Therefore, dcs need effective and timely management to ensure they meet the aims of the organisation.

There are a number of distributed programming languages [Andrews 86, Black 87, Kramer 87] and distributed operating systems [Cheriton 84, Mullender 86] which address the problems of separate address spaces, parallel execution, heterogeneity, partial failures and explicit communications. The common theme in these approaches is that they provide *objects* as units of encapsulation, typing and distribution. On the whole, these languages and operating systems only address the problems of implementing distributed systems and do not consider the long-term management of large distributed systems.

The overall objective of management is to maintain the service required by users and to allow the system to evolve to include new functionality. Sub-objectives include:

optimise use of resources,
minimise cost (to users and resources),
maximise revenue(to resource providers),
optimise performance of resources, etc.

Management policy defines the set of rules for achieving these objectives. For example, access control policy is the set of rules defining the resources a user can access and fault management policy defines where a fault should be reported and any recovery action. The policy is defined by the system *administrators*.

We believe that it is necessary to separate policy from mechanism (the implementation of a policy). This allows multiple implementations of the same policy. For example, access control policy can be implemented by *access control lists* or *capabilities* [Lampson 74]. By concentrating on policy, we can provide a uniform approach to the many different functions of management (see below).

The techniques for implementing distributed systems are comparatively well understood, but far less research has been devoted to techniques for managing large and multi-organisational distributed systems. Thus, we developed the *Domain Model* described in this paper to cope with the complexity of managing large distributed systems and to allow administrators to structure management policies.

2. MANAGEMENT OF RESOURCES

As mentioned above, the inherent problems of distribution can be solved by the object model. We argue in [Robinson 88], that there is a close match between the properties of the object model and those of a dcs. Therefore, we model all resources in a dcs as objects. In this section, we describe our object-oriented view of dcs resources.

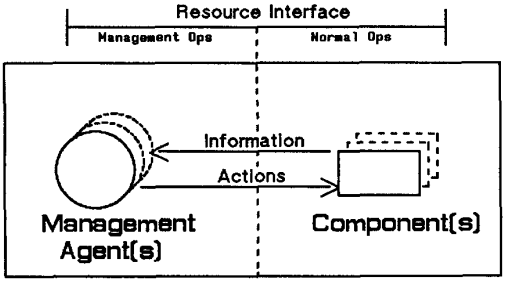

Figure 1. Management Agents and Components

A resource may encapsulate both components and management Agents as shown in figure 1. Components implement the functionality of the resource, while management agents enforce the management policy as applied to the resource. Management agents may be human or automated, and if automated they may either be integrated with the components or separate objects. All resources have an interface which has both normal and management operations. The normal operations invoke the resource's functionality, while the management operations permit administrators to set policy, obtain management information and perform management actions on the resource. For example, a management operation can change the recovery policy of the resource. Management operations are normally only used by administrators.

In addition to the resource interface, there can be internal interfaces. These are between the management agents and the components as well as between components themselves. These internal interfaces are not visible to clients of the resource.

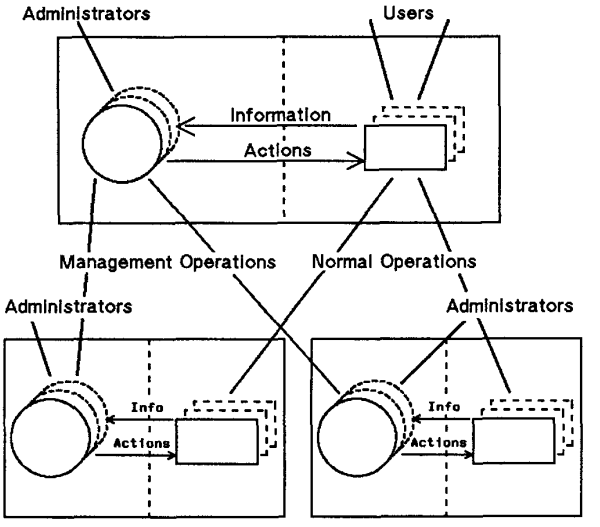

Figure 2. Usage of Other Resources

A resource may use other resources to provide its functionality. The resource's components can only use the normal operations of the other resources. However, its management agents can use both sets of operations. In addition, the administrators of the other resources will use the management operations.

2.1 Management Functions

In this section, we will briefly describe the main functions relating to the management of a distributed system. The ISO Open System Interconnection (OSI) management framework [ISO 2058] also defines a similar set of functions.

2.1.1 Configuration Management

A distributed computing system is composed of many different types of resources, both hardware and software. There may be several instances of the same resource type. The role of configuration management is to select suitable resource types and install the required number of instances to meet the organisation's aims. For software resources, this involves the allocation of resource instances to processors. Configuration management also includes defining the interconnection between resource instances. That is, defining which resources can interact. This is sometimes known as **binding**. See [Twidle 88] for an example of the application of domains to configuration management.

Configuration management also includes controlling the state of resources. Administrators can stop and start resources. This is necessary for maintenance and

reconfiguration. This may be a planned upgrade or to cope with failures. Some resources automatically reconfigure themselves. For example, adaptive routing in communication systems [McQuillan 74].

The allocation and registration of names for components is also considered part of configuration management.

2.1.2 Fault Management

Fault management is the set of facilities which enable the detection, isolation and correction of abnormal operation of the dcs. Faults may develop over time or be inherent in the design of components. This will result in an error when the component is exercised. A failure of resource occurs when it no longer performs according to its specification [Laprie 85]. The aim of fault management is to minimise the occurrence of resources failures and to recover from failures when they have occurred.

Error detection could involve monitoring the the frequency of errors to give early warning of a resource failure. The next step is to locate the faulty component. This may require analysis of histories of errors and performance logs. Also, diagnostic tests can be used to exercise suspected faulty components. When the fault is located, fault management will instigate recovery action. This may involve reconfiguration to isolate the faulty component(s).

2.1.3 Security Management

The objective of security management is to maintain the confidentiality and integrity of the information within the computer system. Information should only be revealed to and modified by authorised users and only be modified in an authorised way. An essential aspect of security management is access control. That is, controlling who can access a resource. This requires authentication of the source of a request. Other aspects of security management include management of encryption keys and logging of abnormal events.

2.1.4 Accounting Management

Accounting management enables resources to charge for the usage of the service(s) they provide. This requires the monitoring and recording of usage of a resource by individual users. Authentication is necessary to reliably identify the users. Accounting management is also concerned with determining the tariffs for a resource including the criteria for charging (per invocation, duration of usage, time of day, quality of access, etc.).

2.1.5 Performance Management

Performance management is the set of facilities needed to evaluate the behaviour of resources. This requires monitoring the performance of the resources to gather statistical data, and to maintain and examine logs of system state histories. This is used to detect bottlenecks in the dcs and to plan for future modifications. The result of performance management can be instructions to re-configure the dcs. Performance management is also used to help detect faulty components.

2.1.6 Monitoring

All the above functions require monitoring of the system to determine current state; to determine abnormal events; to record usage of resources or to measure performance. Indiscriminate monitoring can generate large amounts of information, which must be filtered and distributed to the relevant administrators and management agents.

2.2 Discussion

The above management functions are interdependent. Fault and performance management require the use of configuration management for reconfiguration. All management operations must be subject to access control to prevent unauthorised administrators performing management operations.

A common management framework is needed to permit the specification of standards for performing the above management functions. This has also been recognised by the OSI Management Framework [ISO 2058], but it only addresses communication systems.

There are a number of problems related to the management of distributed systems which must be addressed by a management framework.

a. Size: we are concerned with large systems in which the number of resources is potentially unbounded. Therefore it is necessary to partition resources into manageable groups. This partitioning could be according to function to reflect the functions identified above or according to location e.g. the computers connected to a single network.

b. Multi-Organisational: a dcs typically spans multiple organisations. Even where a dcs is entirely within an organisation, it is likely to span administrative boundaries within the organisation. It should be possible to reflect this organisational structure within the management of a distributed system

c. Multi-Vendor: a dcs can contain components from many vendors. These components are likely to have

different interfaces and approaches to management even where they are providing the same type of service. A common approach and standards for management are needed to cope with these differences.

d. Long Lived: DCS are typically long lived systems. They evolve to meet new user requirements by adding and removing resources. It is impractical for the entire dcs to be shut down while modifications are made. Instead, changes are made while the system is operational with only those components directly affected being unavailable for the duration of the change.

3. THE DOMAIN MODEL

Our approach to dcs management is to concentrate on structuring management policies. This is provided by our Domain Model. It allows administrators to define the aims of the dcs and monitor its behaviour.

3.1 Requirements of dcs management

Before introducing the Domain Model, we will outline the requirements for managing distributed computing systems.

3.1.1 Scalable

In principle, there is no limit to the size of a dcs as new resources can easily be added and separate networks linked by gateways. The management system must scale to match the size of the dcs. Also, the size of the dcs means that it may be partitioned and managed by several administrators. These administrators may belong to different organisations. The area of responsibility of each administrator needs to be clearly defined.

3.1.2 Flexible

The management system should not unduly constrain the specification of policies. The only constraints should be those imposed by the aims of the organisation(s).

Flexibility is also required for multiple organisations where there is no single authority or source of policy. It should be possible to specify relationships between the authorities.

DCS are typically long lived systems. Therefore, the management system must accommodate evolution of the dcs to include new resources and changing user requirements.

3.1.3 Uniform

A uniform management interface for administrators will aid managing a dcs. Especially if it is uniform across different functions of management as well as different vendors.

3.1.4 Verifiable

Administrators need to know the effects of a policy. Therefore, the management system should be amenable to verification. For example, checking who can access a given resource. Tools to help with the analysis, and rules to give predictable configurations are necessary.

3.2 The Domain

The key to meeting the requirements is **modularity**. While the number of resources in a dcs is unbounded, the number that any individual administrator manages is always finite. Therefore, we partition the management policies into **Domains**.

A Domain is a set of resources which share some common attribute. In particular, it is the set of resources to which the same management policy applies. For example, the set of resources accessible by a given user or the workstations connected to the same network. The domain also has an associated manager, who can alter the management policies.

The **manager** can be a distributed set of software processes which cooperate to manage a set of resources, a human administrator managing a group of people in a department, a number of programmers responsible for maintaining a distributed software service, or a user who manages the files he owns in the distributed computer system. The manager can thus be either a single entity or a set of entities. In the latter case, the managers cooperate to specify the policy.

An entry in a domain implementation would typically be a group of attributes and a reference to (e.g. a name for) a real world resource. For example, entries in domains specifying access control policy could be capabilities. Each capability is a reference to a resource and the set of operations which can be invoked on that resource [Robinson 88].

Domains specify the sphere of influence of a manager. Management policy is altered by manipulating domains that specify the policy i.e. changing the entries in a domain. Therefore by controlling the domains that administrators can manipulate and the way they can manipulate the domains, it is possible to control the influence an administrator has on the management policy.

Domains are used to structure management policy. This is achieved by defining a relationship between domains as is discussed in section 3.4.

3.3 Types of Domain

Domains are used to group resources to which the same management policy applies. However, policies are typed according to the function of management they specify. For example, access control policy, configuration policy, fault policy, etc. Policies of different types are not grouped into the same domain even when they refer to the same resource.

Domains are typed by the type of policy they contain. This gives a functional partitioning of the overall management with a different type of domain for each management function. In section 4, we give some examples of different domain types.

A domain is also an object in that it is an instance of a type (class). There is a generic domain type which provides the basic operations to manipulate domains (*add, copy, remove, list* - complete list in [Robinson 88]). This generic domain type has a sub-type for each different domain type. This is necessary because of the type and structure of entries is dependent on the policy type.

3.4 Relationships between Domains

An individual domain specifies a common management policy for a set of resources. Structuring of management policy is achieved by defining relations between domains. There are four relations; disjoint, sub-domain, touching and overlap. A relation can only be defined between domains of the same type e.g., two access control domains.

3.4.1 Disjoint

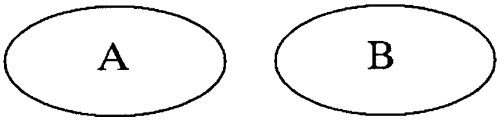

Figure 3. Disjoint Domains

There is no commonality of resources or policy between the domains and so there can be no interaction between the domains. For example, let domains A and B represent the resources managed by administrator A and B respectively. Then these two administrators manage disjoint sets of resources.

3.4.2 Sub-Domain

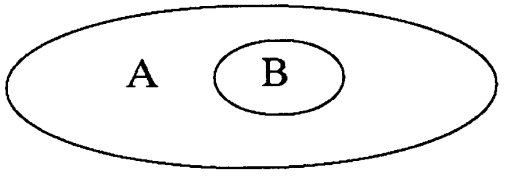

Figure 4. Sub-Domains

This represents the case where a common management policy applies to all the resources in domain A. Domain B is a subset of the resources in domain A, to which a refinement of A's management policy apply. For example, let domains A and B be the access control domains for users A and B respectively. Then user A can access all the resources in domain A (including domain B) while user B is restricted to the resources in domains B. The sub-domain relation is suitable for structuring large numbers of resources. It is analogous to organising files into sub-directories.

3.4.3 Touching

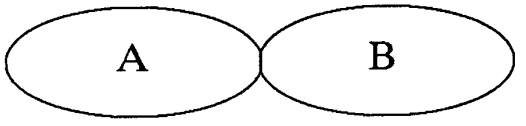

Figure 5. Touching Domains

Touching domains represent domains of the same type which have a mapping between their management policies to permit interaction. For example, consider A & B in figure 5 to be two lans interconnected by a gateway. The gateway policy limits which nodes can exchange messages with nodes on the other network.

A mechanism is needed to permit domains which were previously disjoint to become touching. This permits interaction to be developed between independent organisations.

3.4.4 Overlap

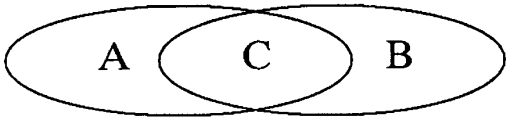

Figure 6. Overlapping Domains

When two domains overlap, it means that the two sets of resources are not disioint. Rather, there is a subset of

the resources of each domain that are common. The resources in the overlap have management policies in common with resources in both domains. For example, let domains A and B be the set of resources accessible by users A and B respectively. The resources in C are accessible by both users. Therefore, administrators can make resources available to both users by adding a suitable access right to the overlap of the users access control domains.

Overlapping domains are controversial in that they effectively mean that a resource is managed by two or more managers. It may be a policy decision to prevent this, for example, by creating a new domain with a single manager to contain the common objects. This approach is taken in many organisations. In many cases this may be impractical and shared management may be the best solution. An example is a datalink connecting gateway halves in two packet switched networks. The datalink is managed by both network management centres. In the case of shared management, protocols are needed between managers for concurrency control and to agree on particular management actions, e.g. when to take a resource out of service.

4. EXAMPLE DOMAINS

To demonstrate the use of the Domain Model for management of a dcs, consider a collaboration between a university department (A) and industrial research laboratory(B). Both A and B have local area networks with with various workstations and servers attached. A has a single lan and B has two, interconnected by a gateway. Another gateway interconnects the two organisation via a high speed link. There are four workstations (WS1-WS4) and two file servers (FS1 & 2) at A, with three workstations (WS5-WS7) and two file servers (FS3 & 4) at B. Also, there are two miscellaneous servers S1 and S2 at B. The topology of the network is shown in figure 7. Some resources are seconded to the collaboration and can be accessed from either organisation.

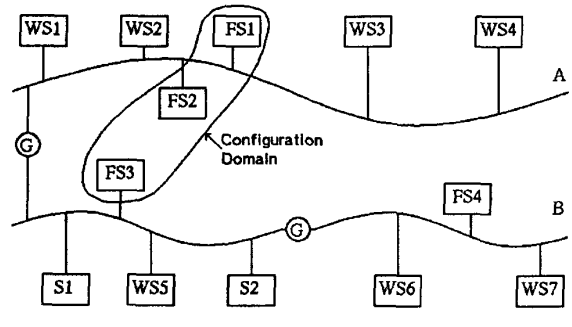

Figure 7. Network Topology and a Configuration Domain

We will use this example to illustrate the application of domains to **configuration, fault, security,** and **monitoring** management.

4.1 Configuration Domains

A and B are collaborating on development of a reliable file server. There is a project manager on site B who is responsible for allocating the relevant software components to the three different fileservers FS1, FS2 and FS3 and performing the necessary bindings etc. so that they appear as a single reliable file server to users. This is an example of a **configuration domain** which corresponds to a single logical resource built from multiple physical components (figure 7). The internal management agents implement the replication policy set by the project manager.

4.2 Fault Domains

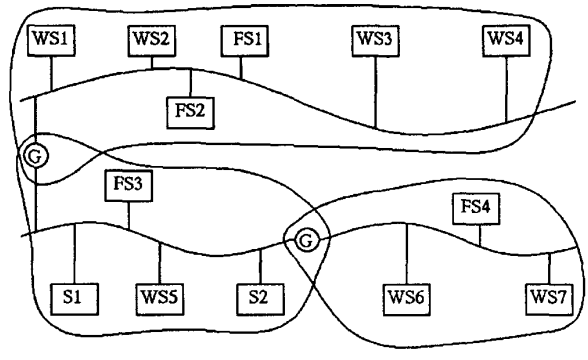

Figure 8. Fault Domains

There are two different types of **fault management domains** in this environment. The first corresponds to the reliable file server. This contains the fault management agents which detect server failures and modify the replication strategy accordingly and take recovery action to restore consistency when components recover.

The second corresponds to the physical topology. An operator is associated with each lan whose responsibility is to handle fault reports, perform diagnostics, take dumps and reboot after machine crashes. The gateways are shared resources in the overlap domains shown in figure 8. There is an agreed protocol between the operators to prevent them performing inconsistent operations on the gateways.

4.3 Security Domains

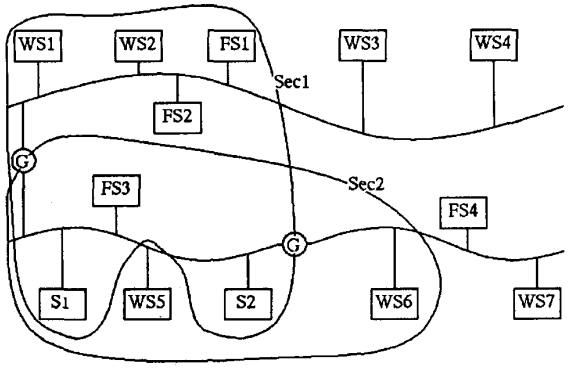

Figure 9. Security Domains

Only workstations WS1 & WS2 at A and WS5 & WS6 at B are involved in the collaboration. Therefore, access control is necessary to limit access to resources. The security policy is provided by **security domains**. These define the set of resources that can interact. In our example, we assume that there is no direct interaction between workstations in the different organisations Therefore, WS1 will be in a different security domain from WS5. However, both WS1 and WS5 can access all the servers. This is achieved by overlapping security domains (Sec1 and Sec2).

These security domains are similar to Estrin's Inter-Organisational Networks [Estrin 87] in that they define inter-organisational access control policy. However Estrin's IONs are be implemented by category sets that are checked at a gateway. This means that IONs can only correspond to physical networks.

Our security domains can be implemented both across networks and on a single network [Robinson 88]. An encryption key is associated with each security domain. All communication between workstations in a domain is encrypted with this key. Only if the destination workstation is in the same domain, will it be able to decrypt the message. Where a workstation is in more than one domain, it will have multiple keys. Encryption has the advantage of preventing eavesdropping. However, it does require hardware encryption to achieve reasonable performance.

Domains can also be used for finer grained access control. It can also provide the basis of a user's interface to the dcs by defining the resources he can access. An example of an implementation of domains for structuring access control policy based on capabilities is described in [Robinson 88] and one based on access control lists is described in [Twidle 88].

4.4 Monitoring Domains

Assume the **monitoring domains** correspond to the fault domains shown in figure 8. As mentioned in section 2.2, all the management functions need access to monitored information. A solution is to provide a generalised monitoring service with a monitoring manager (possibly replicated) to which clients can provide a profile of information they require and whom they wish this to be provided - periodically, on request etc. The monitoring manager instructs monitoring agents on each component to collect and transmit the relevant information according to the monitoring policy that covers the workstations and servers in the collaboration.

The monitoring domains should be considered to overlap in that gateways report monitored information to two monitoring centres. In addition the domains correspond to touching domains in that the monitoring centres have to provide information on their components to other domains, as some of the other types of domains which require monitored information cross network boundaries. The mapping between domains defines the interaction. The monitoring policy would specify whether configuration and fault management agents can obtain monitored information directly from monitoring centres in other domains or whether they need to get the information from their local monitoring centre.

4.5 Inter-relation between Domain Types

It is important to consider the inter-relation between domains of a different type. One form of inter-relationship is that a management service of one domain type is used by the management agents of another domain type, as discussed above. Another aspect of this inter-relationship is that a management action in one domain can have implications in another. For example if the maintenance operator decides to perform diagnostic tests on a server, it will not be available to a configuration manager for installing software. Similarly, all components on which the configuration manager should perform configuration operations must be in his access control domain. Thus taking our example (figure 9) we can not add FS4 to the reliable file server domain because this would violate the access control policy.

The Domain Model enables these inter-relationships and possible conflicts to be to be identified. The result is consistency of management policies. This is particularly important where there are several administrators managing the dcs.

For example, it is not reasonable to specify an security domain that includes resources on unconnected networks. This is identified by a mismatch between the configuration and security domains. An administrator should be warned of

this error.

In the same way, there is an inter-relation between fault domains and configuration & security domains. There is a conflict in management policy if a fault domain includes servers that are on unconnected networks or where the access control policy will be broken.

Other examples of domains can be found in [Robinson 88].

5. *RELATED WORK*

The majority of work in the literature relates to mechanisms for management. Either the policy is pre-defined (e.g. access policy in military systems) or there is no guidance of how it should be specified and structured. However, we have identified three general approaches in the literature on distributed system management: centralised (e.g. LOCUS [Popek 85]), autonomous (e.g. Butlers [Dannenberg 82]), and hierarchical (e.g. INTERNET Domain Naming [Mockapetris 83]). We will briefly discuss these examples and indicate the shortcomings in each approach.

5.1 Centralised

The LOCUS distributed system extends UNIX to the distributed environment. It provides a uniform system to its users by giving transparent access to resources even across heterogeneous machines. However, a LOCUS system is still centrally managed as is a stand alone Unix system. For example, access control policy is still specified in **/etc/passwd**. This centralised approach does not scale. Furthermore, it does not cater for multiple organisation distributed systems.

5.2 Autonomous

Dannenberg's Butlers provide totally distributed management. The dcs consists of several workstations which are individually managed by their owners. If desired, the owners can share their resources with other users. In this case, the owner specifies a policy defining who can access a resource and from where the access is permitted. A butler runs on each workstation to implement this policy. Also, the butler can be used to find a suitable resource on another workstation. This autonomous approach scales but it does not result in an integrated system.

5.3 Hierarchical

The most frequently used management structure in a distributed system is a hierarchy. This is because it scales and defines a clear line of responsibility. For example, in the allocation of names. An authority can allocate names for organisations connected to a wide area network, perhaps divided between academic and commercial. Within each organisation, it allocates names for individual networks and resources connected to these networks. However, not all systems can be hierarchically structured. For example, if two organisations agree to collaborate, neither organisation is over the other. Nor is there a third supervisory organisation. Cooperating organisations are more closely represented by interacting hierarchies with arbitrary links between the hierarchies.

The Domain Model is flexible enough to support the three above approaches. In addition, it can allow a mixture of approaches in the same system. The important point is that domains are at the level of specifying and structuring management policies, not at the level of implementation. This simplifies an administrators task.

5.4 Other Examples

As already mentioned, Estrin's [Estrin 87] Inter-Organisational Networks (IONs) are an example of domains. They define a coarse grained access control policy which correlates with the physical network configuration. As with domains, IONs are allowed to overlap. Our approach is more flexible in that domains are logical relationships, we support sub-domains and can have multiple domains of the same type or different types on one network. Estrin does not consider the relationship between different types of management.

The ANSA **trading system** [ANSA 87] is an example of a configuration domain specifically for binding component interfaces. An export is registered in a **trading context**. When importing an interface, a trading scope is specified. Unless the export is found in this scope, the import fails. Domains can represent trading contexts and the relationship between contexts. In addition, domains are being applied to federated trading systems to define the degree of co-operation between separate trading systems.

When considering the subject of distributed systems, it is necessary to consider the work of ISO. At present, the OSI management work is still in its formative stages [ISO 2058] and is specific to communications. They have identified three points at which management is to be applied to communication systems:

- Protocol: management of a protocol internal mechanism for one of the seven layers (e.g. flow control window of a transport layer connection).

- Layer: management of all the resources associated with a particular layer (e.g. network layer routeing).
- System: management of all the seven layers.

There is also a growing realisation that networks may belong to different administrative *domains*. There are issues of trust and requirements for *firewalls* between domains. At present, ISO is only considering the simple case of neighbouring domains. Our work on domains shows that other relationships between administrative domains are possible and need to be considered.

Our work on domains has influenced ECMA with the domain concepts being applied to the ECMA security framework.

6. CONCLUSIONS AND FURTHER WORK

Domains provide the means for structuring the management of future large distributed systems. We have identified the main functions which are performed in the management of distributed systems and shown that these functions cannot be considered independently. There is a interaction between these functions. In our opinion, this is best accomplished by specifying management services corresponding to each of the functions.

The modularity inherent in the concept of domains permits the scaling necessary to manage large distributed systems. We have shown the need for subdomains and touching domains to provide the flexibility of structuring management to meet the requirements of inter-organisational large distributed systems. Domains specify an explicit boundary of management influence which can help in detecting conflicts between types of domains. This should lead to more consistent management policies.

We are currently working on a more formal specification of domains using Z [Moffett 88]. In addition we are investigating the applicability of object-oriented database concepts (particularly inheritance and aggregation) as an approach to implementing domains. The various types of domains could be specified as a specialisation (sub-type) of a generic type domain.

Although the concept and terminology of a domain has been around for some time, its importance to the management of distributed systems has not been sufficiently recognised. Considerable research is still needed in developing tools and techniques to support the management of the large distributed systems which will emerge in the 1990s!

7. ACKNOWLEDGEMENTS

The authors wish to thank their colleagues in the Special Interest Group on Distributed System Management (SIGDSM) for providing a forum for debating the ideas presented in this paper. In particular, Jonathan Moffett for his work in formalising the meaning of Domains. The work was supported by GEC-Marconi Research. However, the ideas expressed in the paper are those of the authors and do not necessarily represent the views of GEC-Marconi.

8. REFERENCES

[Andrews 86] G. Andrews and R. Olsson, "The Evolution of the SR programming language", Distributed Computing, 1, July 1986, pp. 133-149.

[ANSA 87] ed. A.J. Herbert and J. Monk, "ANSA Reference Manual Release 00.03," The ANSA Project, 24 Hills Road, Cambridge, UK.

[Baker 86] S.J. Baker, "Network Management - A Title of Responsibility", European Computer and Communications Conference, June 1986.

[Black 87] A. Black, N. Hutchison, E. Jul, H. Levy and L. Carter "Distribution and abstract types in Emerald", IEEE Transactions on Software Eng. SE-13(1), Jan. 1987, pp. 65-76.

[Cheriton 84] D.Cheriton "The V-Kernel a software base for distributed systems", IEEE Software, 1 (2), April 1984, pp. 19-43.

[Dannenberg 82] R.B. Dannenberg, "Resource Sharing in a Network of Personal Computers", PhD Thesis Department of Computer Science, Carnegie-Mellon University, 1982.

[Estrin 85] D. Estrin, "Inter-Organisational Networks: Stringing Wires Across Administrative Boundaries", Computer Networks and ISDN Systems 9(1985), pp281-295.

[Estrin 87] D. Estrin, "Controls for Inter-Organisational Networks", IEEE Transactions on Software Engineering vol. SE-13 no. 2 Feb 1987, pp249-261.

[ISO 2058] "Management Information Service Definition--Part 1: Overview", ISO/TC97/SC21 N2058, Nov 1987.

[Kramer 87] J. Kramer, J. Magee and M. Sloman "Constructing Distributed Systems in Conic", To be published in IEEE Transactions on Software Engineering,

[Lampson 74] B.W. Lampson, "Protection", ACM Operating System Reviews, vol.8 no.1, Jan 1974, pp18-24.

[Laprie 85] J.C. Laprie, "Dependable Computing and Fault Tolerance: Concepts and Terminology", Proceedings of 15th IEEE International Symposium on Fault Tolerant Computing, Michigan, June 1985, pp2-11.

[McQuillan 74] J. McQuillan, "Adaptive Routeing Algorithms for Distributed Computer Networks", PhD Thesis, Engineering and Applied Sciences, Harvard University, 1974.

[Mockapetris 83] P. Mockapetris, "Domain Names-Concepts and Facilities", Request for Comments 882, Network Information Centre, SRI International, 1983.

[Mullender 86] S.J. Mullender and A.S. Tanenbaum "The Design of a Capability Based Distributed Operating System", Computer Journal, Vol. 29 No.4, Aug 1986, pp. 289-299.

[Popek 85] ed. G. Popek and B. Walker, "The Locus Distributed Architecture", published by MIT Press, Cambridge MA. 1985.

[Moffett 88] J.D. Moffett and M.S. Sloman, "Management Domains", Imperial College Research Report DOC 88/6, June 1988.

[Quarterman 86] J.S. Quarterman and J.C. Hoskin, "Notable Computer Networks", Communications of the ACM vol. 29 no. 10, Oct 1986, pp932-971.

[Robinson 88] D.C. Robinson, "Domains : A Uniform Approach to Distributed System Management", PhD thesis, Department of Computing, Imperial College, London, March 1988.

[Twidle 88] K. Twidle and M.S.Sloman, "Domain Based Configuration and Name Management for Distributed Systems", These proceedings.