6

# Towards policy driven systems management

*Phillip Putter*[1], *Judy Bishop*[2], *Jan Roos*[2]
*pputter, jbishop, jroos@cs.up.ac.za*

**Abstract**
As the size and complexity of information systems increase organisations need to rely on integrated management systems to an even greater extent to ensure that users' requirements of information systems are met. *Management policies* have been identified as a mechanism by which changing user requirements can be captured and introduced into management systems in order to modify the behaviour of managed systems. This paper refers to systems which allow policies to be stated explicitly with the purpose of modification of management system structure and behaviour as *policy driven* management systems. This paper sets the scene and shows the way towards policy driven systems management.

**Keywords**
Management policy, systems management, meta-objects

## 1. POLICY DRIVEN SYSTEMS MANAGEMENT

### 1.1. Systems management

Systems management can be defined as the activities required to ensure that information systems function in accordance with user requirements and objectives [MOF94]. Figure 1 shows a common way in which management activity is frequently viewed [OSI91]. The Figure shows a manager interacting with a managed object (MO) via a management interface. MOs represent abstractions of managed resources. Managed resources have functional responsibilities which they fulfil through interactions via

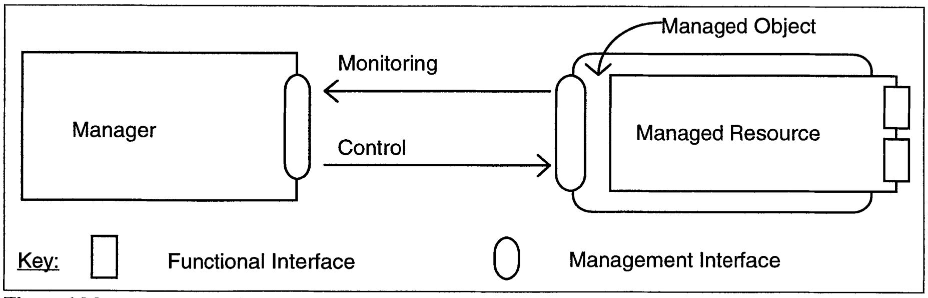

**Figure 1** Management Activity

a functional interface.

Managers are regarded as objects with management responsibilities, and may themselves be the subject of higher level management. Managers manage MOs by monitoring their behaviour, making

[1]Emerging Technologies Group, Momentum Life, P.O. Box 7400, Hennopsmeer 0046, SOUTH AFRICA
[2]Computer Science Department, University of Pretoria, Pretoria 0002, SOUTH AFRICA

management decisions based on the monitored behaviour, and modifying MOs' behaviour through management operations [MOF94].

Systems management concerns itself with activities which attempt to ensure that the functional service levels required by the systems' users are met. Systems management does not concern itself directly with the functional activities of managed systems, and could therefore be seen as a *meta* activity. This paper argues that the meta nature of systems management can be exploited explicitly achieve systems that are easier to extend and modify, something which is discussed in more detail in Section 3.

## 1.2. Complexities of large scale systems management

The nature of the management activities shown in Figure 1 is, of course, much simplified. Large scale management systems can consist of large numbers of managed- and managing objects. Very large management systems introduce a number of problems [SLO94]:

- Large scale management systems often cross organisational boundaries, making it difficult to manage such systems from a central point.
- Large management systems usually require multiple managers, and more often than not, hierarchies of managers, introducing problems with the delegation of authority and responsibility.
- Managed objects could fall in the responsibility domain of more than one manager, introducing the possibility of conflicting management requirements of different managers being brought to bear on MOs.
- The large numbers of MOs make it impossible to manage MOs individually, introducing requirements for grouping mechanisms.
- As the size and complexity of management systems increase it becomes more important to automate as much as possible of the management process to assist human managers with systems management.

Successful management of large systems requires mechanisms which simplify the management process to [MAS93]:

- Raise the level of abstraction at which interactions occur, allowing managers to interact with groups of MOs instead of individual MOs.
- Deal with systems in terms of management policies instead of controls, allowing users of information systems to specify required service levels instead of specifying how these required services levels can be achieved.
- Automate the process by which management policy is captured and transformed to control operations required to achieve policy goals.

## 1.3. Pressures on organisations

Organisations experience tremendous economic, social and technological pressures to survive in changing business environments [SLO94]. Large portions of information systems are dedicated to meeting business needs that will change. Modifications to dedicated information systems are costly and time consuming [POO91]. The dynamic nature of business requirements places pressure on shorter delivery times to exploit business opportunities as and when they present themselves.

Changing business requirements combined with ever increasing complexity and scale of mission critical information systems place additional pressures on systems designers and developers. Successful information systems need to be delivered rapidly and in such a way that it can adapt to changing business requirements with ease.

Policy driven management systems, by their very nature, need to be able to adapt to policy changes. Later sections of this paper will focus in detail on the mechanisms required to achieve policy driven management systems. Many of these mechanisms should also make a positive contribution to fulfilling organisational requirements for modifiable and extendible information systems.

## 2. MANAGEMENT POLICY

### 2.1. Introduction

Management policy can be defined as the relationship between a set of managers and a set of managed objects which obligates and authorises managers to perform management activities on managed objects. Management policy serves as guidelines which influences the decision making process in the light of given constraints [NGU92, SLO94].

This paper is based on the point of view that all entities in managed and managing systems should be modelled and implemented as objects. These include:

- managers,
- managed objects,
- policies,
- functional systems and subsystems, and
- services provided by support environments or management platforms.

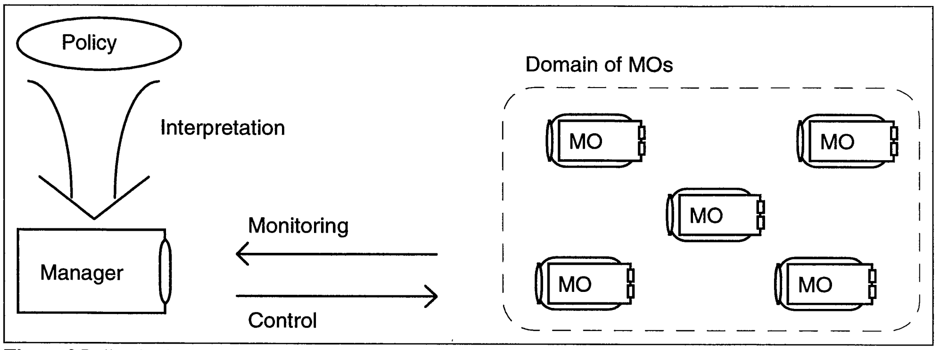

**Figure 2** Policy in the management process

Managed and managing systems are viewed as groups of objects which collaborate to fulfil a common goal [ROO93]. Management policy can be used as a mechanism to capture the goals and requirements of the end users of information systems. Captured user requirements and goals need to be transformed into management operations which can be used to influence the behaviour of managed systems to ensure that user requirements are met.

Policy statements transformed into management operations in such a way influence both the behaviour of the objects of which the managed and managing system consist and system structure [SLO94, MAS93]. Managers interpret policy and uses it to guide decision making in the management process, as shown in Figure 2.

### 2.2. Policy as a separate concern

Policy exists everywhere in organisations' management systems and can be seen in the functioning and operational rules of organisations. Policy statements are usually represented implicitly by burying them deep within the code of managed and managing systems, and are seldom articulated [POO91].

The close linkage between policy and the systems introduces a number of problems [MOF94, MAS93, NGU92]:

- It makes it difficult to capture, store, query and modify policies. This causes policy to be managed in various ad hoc ways.
- Managers cannot interpret policy in a consistent way, making it difficult to implement re-usable managers.
- It leads to inconsistencies, confusion and conflict.
- It makes it very difficult to modify policy dynamically and to forecast the effect of policy changes.
- It makes it difficult and time consuming to modify policies because changes need to be represented in terms of changes to implemented systems.

Research shows that it is becoming well accepted that policy should be modelled and implemented as a separate concern [MOF94, NGU93, MCB91]. Modelling policies as a separate concern:

- recognise them as deliberate,
- forces them to be well defined,
- makes it easier to verify their correctness, and
- makes it easier to manage policies.

A separation between managers and the policies which influence their behaviour allows managers to be re-used in different contexts and permits management policies to be modified and interpreted by managers [MOF94].

## 2.3. Policy management

Management systems where policy is regarded as a separate concern require policy management services to succeed. Policy management services should provide mechanisms to [SLO94, NGU93]:

- Create, modify and delete policies.
- Represent and interpret policies.
- Store and retrieve policies.
- Negotiate the resolution of conflicting policies.
- Communicate new and modified policies to affected managers.

A consistent model for the representation of policies is required to effectively manage policies [NGU92, MOF94]. The policy model should capture policy on as high a level of abstraction as possible and should allow high level policies to be transformed to into concrete plans and procedures to achieve the required goals [MAS93].

The size and complexity of large management systems require automation of various aspects of systems management [MOF94, SLO94]. One of the main goals of policy driven management systems is to automate as much as possible of the management process. Higher level policies are usually more abstract, and require specific attention to ensure that sufficient information is gathered to allow them to be transformed to the management operations required to fulfil them.

The policy model should provide mechanisms to refine abstract policies to detailed operational plans and should be automated as far as possible [MCB91, SLO94]. Automated processes should exploit human expertise on all levels of transformation. Some of the aspects which might be automated include:

- Capturing policy statements from end users and bridging the gap between policy and the operations required to support them.
- The detection of problems in captured policies, e.g. insufficient detail to allow policies to be transformed to plans and procedures.
- Diagnosis of policies to detect which managers and MOs are affected by them, and the distribution of policies to affected managers.

- The detection and resolution of policy conflicts.
- The transformation of policy statements into management operations.

Representation of the relationships between policies is required to allow human managers to determine that stated policies have been satisfied [MOF94]. Policy relationships can be represented as networks of policy nodes, and should form part of the policy model. Relationships between policies should provide a controllable linkage between policies and plans and procedures [MCB91, NGU92].

## 2.4. Policies and domains

Because large management systems consist of large numbers of managers and MOs, policy cannot be applied to individual MOs. Instead, a structuring mechanism can be used to simplify large management systems by grouping MOs to which a common policy applies. Domains form a framework for partitioning of large systems and simplify large systems by acting as a naming construct for objects. Domains can also be used for the specification of viewpoints with specific focus on systems [SLO94].

Domains are objects which can be used to group other objects together. The reason for grouping the objects together is unimportant. The remainder of this paper distinguishes between domains grouping objects for the application of a common management policy (referred to as *management domains*) and domains grouping objects to structure functional systems (referred to as *functional domains*).

## 2.5. Relationships between policies

The relationship between policies closely resemble the organisational structure. Most organisations are structured in a hierarchical fashion in which authority is delegated downward, leading to the definition of hierarchies of policies. Policy hierarchies are characterised by partitioned targets, refined goals and objectives, and delegated responsibilities. A single high level policy may lead to many lower level policies [MOF94, MAS93].

In the case where an organisation follow a management style which is less hierarchical in nature, overlaps in management responsibility are more likely to occur. As soon as multiple managers manage the same MOs or a manager fulfils more than one management responsibility the possibility of policy conflicts exist.

Different approaches can be taken to manage conflicting policies: conflict can be avoided or resolved when it occurs. Some authors propose a combination of resolution and avoidance of policy conflicts for optimal results. Another approach might be to allow a degree of policy conflict. A detailed discussion of the detection and resolution of policy conflict is beyond the scope of this paper. Interested readers are referred to [MOF94].

## 3. IN PURSUIT OF POLICY DRIVEN MANAGEMENT SYSTEMS

## 3.1. Introduction

This paper argues that policy driven systems management requires a new way of looking at existing systems management solutions. Without the separate consideration of a number of concerns, the implementation of policy driven systems management would be extremely difficult, if not impossible. Aspects requiring separate consideration include:

- management policy,
- the use of objects as building blocks for the construction of management systems,
- object grouping mechanisms, and
- the system's structure and behaviour, also referred to as its self representation.

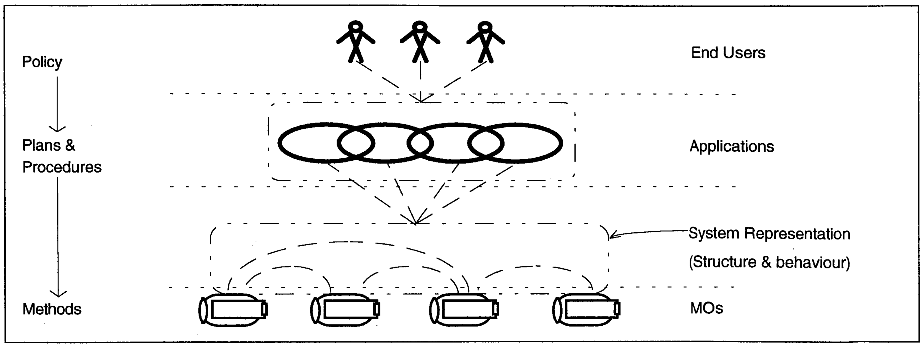

**Figure 3** Management Systems

Figure 3 gives a graphical representation of a management system in which the above mentioned concerns have been modelled separately. The following sub-sections discuss the modelling and implementation requirements of these separate concerns in more detail.

## 3.2. Modelling requirements

### *Objects*

Policy driven management systems fit elegantly into the object-oriented paradigm [ROO93, BEK93]. All policy driven system management entities should be modelled as objects. Objects encapsulate state and behaviour and are defined in terms of the attributes visible at its boundaries and its behaviour.

Objects can represent abstractions of physical equipment, logical components, or collections of information. An object's behaviour is defined by the operations which may be applied to it and its reaction to environmental stimuli. An object's state can only be manipulated and queried via operations exported to the environment by its interface.

This paper argues that objects should be viewed as a set of attributes and a set of operations which, when combined, realise a specific abstraction of a real world entity. Different abstractions of the same real world entity can be formed by combining different sets of attributes and operations. In this way all objects can realise different abstractions of themselves. For the purpose of this paper different abstractions of an object will be referred to as *viewpoints* on the object.

### *Viewpoints*

The ability of an object to represent different viewpoints of itself is regarded as an useful abstraction mechanism which can be used to model and implement policy driven management systems. Figure 1 shows an example of a management viewpoint and a functional viewpoint of the same object. Viewpoints can also be used effectively to modify interfaces offered to particular objects in the environment: in this way it is possible to allow objects to add and remove operations to extend or restrict its interface.

This paper views all systems as groups of objects related to fulfil a common purpose. In this way management systems consist of one or more manager objects related to a number of managed objects with a common purpose of fulfilling the management requirements imposed by the policy to which the system conforms.

Functional systems, on the other hand, consist of a group of related objects with a common functional goal. Management systems can be seen as a different viewpoint of the functional system. The different viewpoints focus on specific aspects and allow users to abstract away from detail which is not necessary in its specific requirements. Some objects may form part of more than one grouping, e.g. a management grouping to apply a specific management policy and a functional grouping to achieve a specific functional goal.

Support for different management viewpoints are also required. Examples of these different viewpoints can be the different day-to-day operational management requirements and the requirements of strategic managers. A more detailed discussion of different management viewpoints can be found in [PUT93].

Domains are discussed as a grouping mechanism in Section 2.4. Although most authors agree that domains should be used only to apply management policy, this paper agrees with the argument that domains can also prove to be extremely valuable for the formation of functional groupings of objects [SLO94, MAS93].

Domains can be seen as different implementation viewpoints on the same objects: in one instance objects may be grouped for the application of management policy, in the other to achieve a common functional goal.

### *System representation*

All systems consisting of groups of related objects have a distinct *structure* and *behaviour*. A system's structure describes the way in which objects are related to fulfil the system's goal, i.e. in a hierarchical fashion or as a network of co-operating objects. A system's behaviour describes how the related objects interact with each other and with the environment. Behaviour can focus on either the functional- or the management behaviour of systems.

Both the structure and behaviour of a management system are influenced by the management policy to which it conforms [PUT93]. Management policy serves as the baseline requirement of management systems, dictating which objects should be used to fulfil the requirements (the system structure), and how they should interact (the system behaviour).

Policy changes influence the behaviour of the managing system directly and the behaviour of managed systems indirectly: management policy guides a manager's decision making process and the manager's interaction with the managed system modifies the managed system's behaviour.

This paper argues that, as policy influences system structure and behaviour, both the structural and behavioural aspects of management systems need to be represented and implemented separately to create open ended systems that can become policy driven. The structural and behavioural aspects of systems can be referred to as the system's representation and can be implemented using meta-objects [BEK93], as discussed in the next sub-section.

### *Meta-objects*

While objects define computation, meta-objects describe and monitor this computation. A meta-object contains information about an object's structure (e.g. relationships with other objects) and about its behaviour (e.g. the way messages are handled, objects are printed, created and initialised) [MAE87].

Meta-objects can be attached to functional objects to enrich the base object's behaviour [BEK93]. It is possible, for instance, to add management functionality to a functional object to allow the functional object to act as a managed object, as shown in Figure 4. Figure 4 shows an object which represents a router that allows packets to be passed between LANs using different communication protocols. This fictitious component will be used to illustrate the use of meta-objects to enrich functional objects, and will also be used in the example in Section 4.

Figure 4 shows two user viewpoints and a mechanism viewpoint of the router object. The first user viewpoint represents the functional viewpoint of the router, the second user view represents the management viewpoint of the router. In order to keep the example simple the router is assumed to have very simple functionality: it inputs a communication protocol A data packet, converts it to communication protocol B, and outputs the converted packed. In order to provide this server the object has three functional operations: read, convert, write. The functional object has only two attributes which can be used as object handles to the packets read and written: *inPacket* and *outPacket*. Each of the functional operations need to be mapped to the real resources' API instructions to perform the relevant

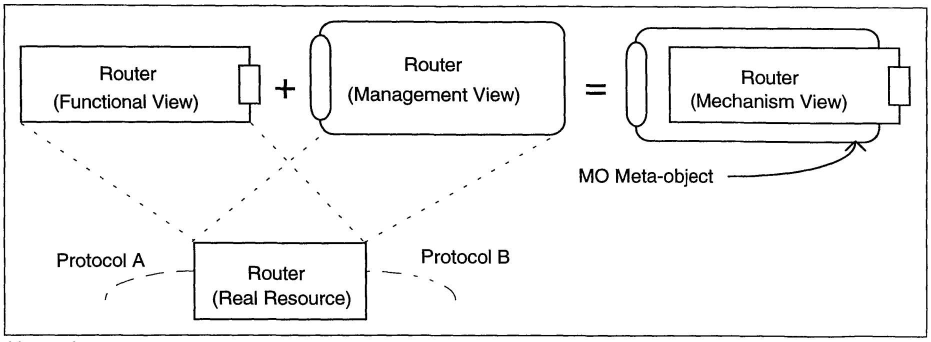

**Figure 4** Object enrichment through meta-object

operations.

The management view of the router allows a manager to query the router state, and to reset the *packetsConverted* attribute. In order to fulfil its management responsibilities the router managed object offers three operations: *get_throughput*, *reset_throughput* and *get_state*. The managed object has two attributes: *packetsConverted* and *state*. The packetsConverted attribute keeps track of the number of packets converted since initialisation or the last reset. Each of the managed object operations needs to be mapped to the real resources' API instructions to perform the relevant operation.

The mechanism view of the router of Figure 4 shows how the two user views can be combined to implement a manageable router. The functional router object is enriched with a meta-object containing

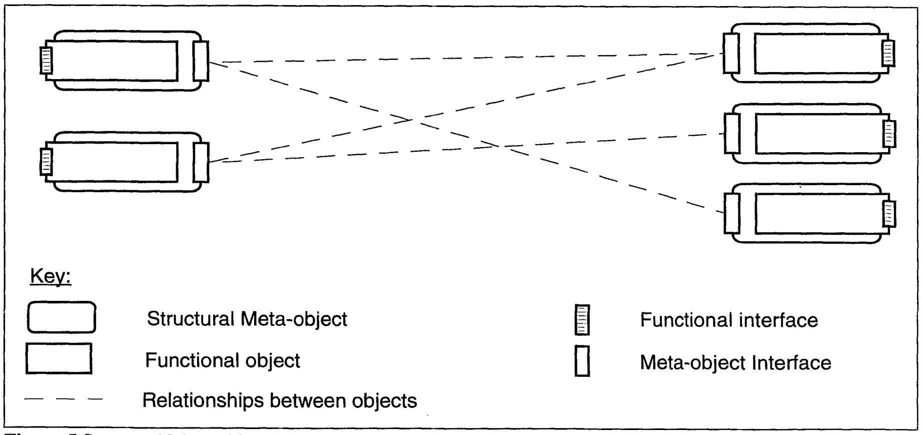

**Figure 5** Structural Meta-objects

the management attributes and operations.

In the same way that the functional router object can be enriched with management functionality meta-objects can also be used for the systems' structural representation. Figure 5 shows a number of functional objects related to fulfil a common purpose. Each object's relationships with other objects can be represented by structural meta-objects. The structural meta-objects encapsulate the behaviour related to system structure and are manipulated to modify and traverse these relationships. This clear separation between the functional objects and their structural aspects allows systems' structure to be changed without affecting functional behaviour.

Objects interact with their meta-objects by passing messages to them. All messages not understood by objects are passed to attached meta-objects. In this way levels of meta-objects can be built, referred to as a reflective tower. Examples of possible uses for meta-objects include [BEK93]:

- Explicit system representation (structure and behaviour).
- Enriching objects with managed object and / or managing object behaviour.
- The implementation of distribution transparencies.
- The implementation of exception handlers.

A detailed discussion on the mechanisms which can be used to attach meta-objects to objects and the use of meta-objects in the implementation of distributed management systems can be found in [BEK93].

### 3.3. Implementation requirements

Policy driven management systems require distributed object-oriented support environments. Apart from support for distribution of objects the support environment should provide:

- Policy services which allow policies to be captured, represented and managed.
- The representation of system structure and behaviour using meta-objects.
- Support for the implementation of domains.

A detailed discussion of the support environments for policy driven management systems is beyond the scope of this paper. Interested readers are referred to [PUT93].

The effective implementation of policy driven management systems require support for sufficient abstraction mechanisms to allow developers to manage the complexity of large management systems as well as the efficient implementation of higher level abstractions. Issues arising from the efficient implementation of policy driven management systems are beyond the scope of this paper, but are discussed in detail in [SLO94].

## 4. EXAMPLE

### 4.1. Introduction

The purpose of this section is to present an example of a policy driven management system in order to clarify the concepts presented by this paper. This example, although simple, will highlight the essence of policy driven management systems, as size limitations prohibit the inclusion of a detailed example.

### 4.2. Problem statement

Consider an organisation that has a large number of routers that form an essential part of its mission critical business processes. Because of the importance of these components a strategic management decision was made that a manager should assume the management responsibility over all protocol conversion services. A simple policy statement was formulated to guide the manager in his decision making: Ensure that all protocol conversion equipment remain operational at all times.

In order to allow the protocol conversion management system to be integrated with the organisation's larger systems management solutions the protocol conversion manager should report to a higher level manager.

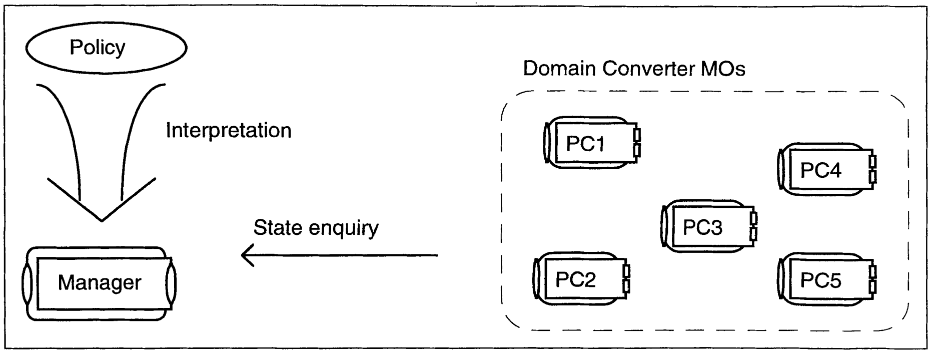

**Figure 6** Router Management System

## 4.3. The system

The structure of the management systems required to fulfil the management policy requirements is clear from the policy statement. At least one manager object (the policy subject) should be associated with managed objects representing all the routers in the organisation (the policy targets).

The goal of the management system is to ensure that all routers remain operational at all times. The goal's requirements dictate the required behaviour of the management system: all routers should be polled at regular time intervals to determine their current operational state. If any routers are found to be in a state other than OK a trouble ticket is generated which is handled by the organisation's systems management solution.

Figure 6 gives a graphical representation of the management system required to fulfil the stated management policy. The figure shows the manager object interacting with a domain of managed objects representing router objects. The manager object and the functional router objects present managed object interfaces via managed object meta-objects which have been attached to it.

Because of the separation between the policy to which the management system conforms and the manager interpreting the policy it is possible to change the management policy, to for instance, relax the restriction that policy routers need to be operational all the time to a availability requirement of 90%. Such a relaxation could, for instance, lead to the modification of the manager's behaviour in that the manager might increase the time between state queries directed at router managed objects, with the required effect on the management system.

## 4.4. Implementation

Experiments have shown that it is possible to attach meta-objects to Smalltalk objects by capturing the Smalltalk message passing behaviour. It is possible to attach meta-objects to any Smalltalk objects in this way. Meta-objects are Smalltalk objects that encapsulate state and implement behaviour in terms of operations[3]. Communication between objects and their meta-objects take place by passing messages between the object and the attached meta-object. Any messages not understood by the functional object are automatically passed on to attached meta-objects via extensions to the Smalltalk messaging infrastructure. A detailed discussion about capturing the Smalltalk message passing behaviour is beyond the scope of this document, interested readers are referred to [BEK93].

[3]Object operations are called *methods* in Smalltalk. The term operation will be used for the remainder of this section to avoid unnecessary confusion between the use of the terms *method* and *operation.*

In this way it is possible, for instance, to implement a router MO with the attributes *packetsConverted* and *state* and the operations *get_throughput*, *reset_throughput* and *get_state*, which can be attached to a router functional object. Attaching the MO meta-object to the router functional object the enriches the functional object's behaviour, as it now responds to the MO operations as well as the functional object operations *read*, *convert* and *write*. The passing of control between the object and the attached meta-object is handled transparently. To the object invoking the operation it seems as if the operation was performed by the functional object. Invoking any of the MO operations on a router object with an attached MO meta-object will cause the MO operation to be performed by the meta-object and the result of the operation to be.

The experimentation also focused on the explicit representation of the management system structure by constructing management systems from collections of functional objects. It was shows that it is possible to separate the relationships between objects from the objects themselves. No attempt was made to modify these relationships dynamically as result of policy modifications, relationships were identified and explicitly constructed in the experimentation.

The results of the experimentation, showed that:

- that it is possible to modify object behaviour by attaching meta-objects.
- the separation between the objects which make up management systems and the relationships between these objects can be achieved.

The authors feel that more research is required to determine the extent to which the modification of behaviour and structure can be automated.

## 5. CONCLUSIONS

This paper presented an approach to policy driven systems management. Research covered by this paper attempts to exploit leading edge object-oriented principles in management systems. The research offers a fresh look at the managed and managing systems involved in systems management.

The pursuit of policy driven systems addresses a number of concepts which should have a positive effect on the extendibility of functional systems. These include:

- The separate modelling, implementation and management of policies.
- Separation of structural and behavioural concerns of object-oriented systems.
- Using meta-objects to enrich object behaviour and modify object interfaces.
- Exploiting the implementation of viewpoints on objects to realise more than one abstraction of an object.

The exploitation of object-oriented principles to the degree presented by this paper has not yet found wide acceptance in the area of systems management. This paper argues that, without these principles, it will become increasingly difficult for management systems to provide the users of information systems with the assurance that end-user requirements have been met, and to adapt to changes in user requirements.

Although experimentation with the concepts presented are still in its early stages, current prototyping results are very encouraging. No claims are made that the work presented solves all the complexities faced in the area of policy driven management systems, but it is argued that the research points the way to some solutions of existing management problems.

## 6. ACKNOWLEDGEMENT

All glory to the Lord our God, originator of all knowledge, without whom none of the work presented here would have been possible.

## 7. REFERENCES

| | |
|---|---|
| BEK93 | Reflective Architectures: Requirements for Future Distributed Environments. Bekker C, Putter P. Proceedings of the Fourth Workshop on Future Trends of Distributed Computing Systems, Lisbon Portugal (1993). |
| MAE87 | Concepts and Experiments in Computational Reflection. Maes P. Proceedings of Object-Oriented Programming: Systems, Languages, and Applications, ACM SIGPLAN Notices, Vol. 22, Nr 12 (December 1987). |
| MAS93 | Policy Management: An Architecture and Approach. Masullo MJ, Calo SB. Proceedings of the IEEE First International Workshop on Systems Management Los Angeles (1993). |
| MCB91 | A Rule Language to Capture and Model Business Policy Specifications. McBrian P, Niezette M, Pantaziz D, Selviet AH, Sundin U, Theodoulis B, Tziallas G, Wohed R. Proceedings of the Third International Conference on CAiSE, Norway (1991). |
| MOF94 | Specification of Management Policy and Discretionary Access Control. Moffett JD. Chapter 28 of Network and Distributed Systems Management. Sloman MS, Kappel K. Addison Wesley (1994). |
| NGU93 | Incorporating Business Management Policy into Information Technology. Nguyen TN. Proceedings of the Second International Symposium on Network Management, Halfway House, South Africa (1993). |
| OSI91 | Open Systems Interconnection: Management Overview. IS10040 (1991) |
| POO91 | Representing Business Policies in the Jackson System Development Method. Poo CD. The Computer Journal Vol 34 no 2 (1991). |
| PUT93 | A general building block for distributed systems management. Putter P. Masters Dissertation, University of Pretoria (1993). |
| ROO93 | Modeling Management Policy using Enriched Managed Objects. Roos J, Putter P, Bekker C. Proceedings of the Third IFIP/IEEE International Symposium on Integrated Network Management, San Francisco (1993). |
| SLO94 | Domains: A Framework for Structuring Management Policy. Sloman MS, Twidle K. Chapter 17 of Network and Distributed Systems Management. Sloman MS, Kappel K. Addison Wesley (1994) |
| YEM91 | Network Management by Delegation. Yemini Y, Goldsmidt G, Yemini S. Proceedings of the Second IFIP Symposium on Integrated Network Management, Washington USA (1991) |

## ABOUT THE AUTHORS

**Phillip Putter** is responsible for the development of networks, distributed systems infrastructures and systems management at Momentum Life, South Africa's fourth largest life assurer. Phillip completed an MSc at the University of Pretoria in 1993, and is currently busy with a PhD with research into policy driven systems management.

**Judy Bishop** is professor of computer science at the University of Pretoria, where she specialises in distributed systems and programming languages. Judy obtained her PhD in computer science from the University of Southampton in 1977. Judy has been a member of IFIP's Working Group 2.4 (System Programming Languages) since 1987, has served on several international programme committees, and is the author of seven books on programming.

**Jan Roos** teaches Computer Networking at the Department of Computer Science of the University of Pretoria. Jan obtained an MSc in Computer Science from the University of South Africa, and has extensive research and development experience in the areas network design and network management. In 1988 Jan spent a year in Germany researching high speed networking and network management.