POLICY MANAGEMENT: AN ARCHITECTURE AND APPROACH

Miriam J. Maullo & Seraphin B. Calo

IBM Thomas J. Watson Research Center, Hawthorne, New York

ABSTRACT

This paper deals with the role of policies in the management of large complex systems. An approach is suggested for the transformation of policy statements into executable process decision functions; and, an architecture is outlined to help organize the elements of policy in a manageable way. To illustrate the concepts, an example is given from information processing that conveys more concretely the manner in which this architecture can be applied to policy management. Tools are introduced to deploy a Policy Management System.

1. INTRODUCTION

As systems become increasingly complex, the development of methodologies for their effective management becomes more and more critical. One important aspect of this general systems management problem is that of being able to raise the level of abstraction at which interactions can occur, so that the people responsible for managing the system do not become overburdened or overwhelmed by excessive detail. This requirement is commonly described as the ability to deal with systems in terms of policies rather than explicit controls. It should thus be possible, for instance, to specify service goals and have the system itself determine the operations needed to achieve them; or, to specify priorities and preferences for alternative services, and have the system itself allocate resources accordingly.

Such mechanisms are clearly desirable, and can be specified and effected for specific services in specific systems. In order to be more generally useful, however, the basic concepts need to be better understood. In particular, more formal answers are needed for such questions as: what is meant by **policy**; how can it be captured, and specified to the system; and, how can it be automatically interpreted and executed.

In the first section, we present some general definitions and considerations that help put the concept of **policy** in context with respect to the management of information processing systems.

In the second section we discuss what people typically mean when they use the term policy, and assert that it denotes considerations at a number of different conceptual levels. We describe six such levels in terms of their specificity, methods of communication, and amenability to automation. Our basic interest is in policies as they relate to information processing systems, and some of the descriptions will reflect that fact. Indeed, at the end of the second section, explicit examples are given relating the definitional levels proposed for policies that might be applied in the management of such systems.

2. GENERAL CONSIDERATIONS

Our motivation for investigating the specification and implementation of policies is the desire to develop information mechanisms that can make systems management easier and more effective. **Systems Management** deals with the set of processes that are applied to the administration, coordination, and control of information processing systems within an enterprise, according to a given

0-1886-3820-6/94 $03.00

set of policies. It is thus concerned with the achievement of goals within policy constraints. A **process** thus becomes that set of coordinated actions that are designed to achieve those goals. The processes typically deal with such issues as:

- the allocation of resources,
- the resolution of priorities, and
- the scheduling of events.

Because of the flexibility of computing systems and the uniqueness of each organizational environment and its workload characteristics, these processes must all adapt, customize and refine the generic capabilities of the available computer resources to meet their own functional needs. This can usually be done in a number of different ways, and the set of specific choices made by a particular enterprise characterize its computing environment as profoundly as the capabilities of the underlying system. Thus in trying to come to terms with the management of complex systems, we are immediately faced with issues of **policy**.

Intuitively, it is clear that similar policies can be applied to different processes within a given system, or to similar or different processes within entirely different systems. Also, while good policies normally take into account such systems related considerations as security, integrity, and efficiency, they are typically more oriented toward organizational goals and considerations. This leads us to the conclusions that policies are distinct from the processes themselves, and are in some sense more generic and enterprise driven.

A **policy** then is a set of considerations designed to guide decisions on courses of action. Policies are used to:

- refine processes (e.g., thresholds),
- to resolve conflicts among concurrent processes (e.g., priorities), and
- to establish processes in time (e.g., schedules).

The policies are thus not the processes or the decisions themselves, but the information mechanisms used to make the decisions. **Policy Management** deals with the establishment, communication, maintenance and execution of enterprise information processing policies. As such it spans a wide spectrum of organizational activity, from the high level goals conceived by human intelligence, to the sets of management tasks executed by computer automation.

Computer system administrators and operations managers, do not seem to have yet adopted the practice of explicitly organizing and systematically implementing policy in software. Part of the problem is that a formalism have not been developed for supporting such activities. Policy management exists, but it is typically carried out in ad hoc ways that include collections of command lists, menu panels, system exits, locally implemented services, etc.. What is lacking is a standard methodology for understanding the implications of policy and for mapping these directly into software.

Having defined to some extent what is meant by policy, the next step in trying to understand how Policy Management could be better supported by processing systems is to attempt to isolate the elements that make up policy. Policies are fundamentally geared to addressing practical matters in an appropriate, consistent and efficient way. Organizationally, this involves various levels of activities, some or many of which may not be associated with data processing at all.

Elements of Strategic Management

The elements of strategic management are defined within the discipline of Business Administration (Jauch, 88) as:

1. analysis and diagnosis,
2. choice, and
3. implementation.

The element of **analysis and diagnosis** includes the impact of threats and opportunities and an examination of the strengths and weaknesses of the enterprise. **Choices** are considerations of the various alternatives, and are made to assure that the appropriate strategy is chosen. In **implementation**, plans, policies, resources, structure and administrative style are made to match the strategy adopted by the enterprise. And, processes are implemented to meet guided and established objectives.

Because policies are a component of the element of implementation, they are hard to distinguish from the process component.

More specifically the elements of **implementation** are:

- communicate measurable objectives,
- determine key managerial tasks,
- assign tasks to various parts of the organization,
- state policies as guides for action, and
- clarify goals at various managerial levels.

Implementation thus become a function of resources, policies and processes. Figure 1 illustrates this view.

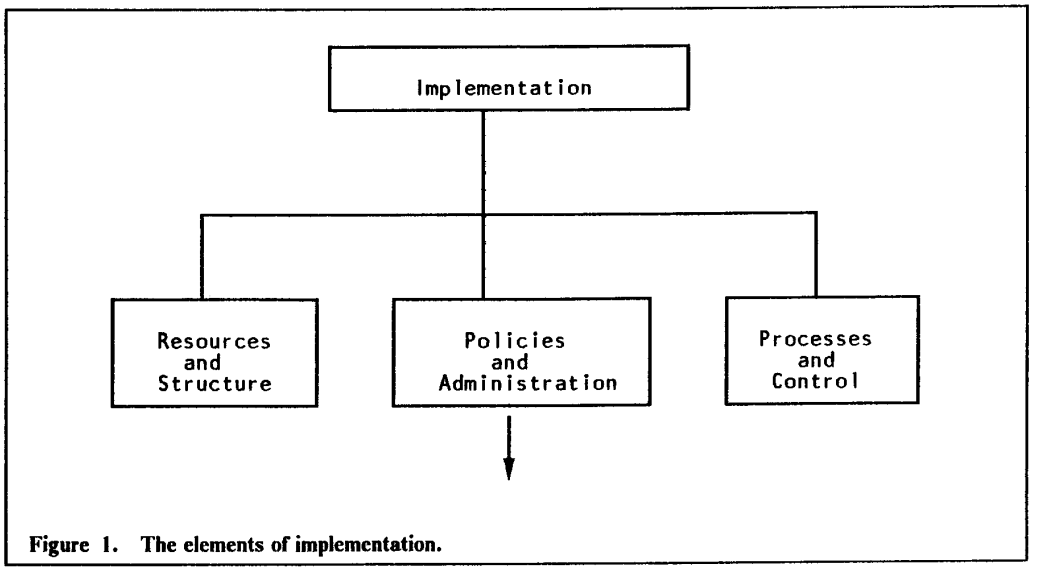

Figure 1. The elements of implementation.

3. THE POLICY HIERARCHY

The general term "policy" is used to describe a number of different levels of considerations, and it is necessary in attempting to deal with policy issues in a concrete manner to make appropriate distinctions among these different kinds of "policies".

In the following, a characterization of policy considerations in terms of several distinct levels is suggested. These are distinguished by the degree of precision which they represent, the manners in which they are communicated, and the extent to which they can be accommodated by automation. Conceptually, the levels can be considered to be traversed in an hierarchical way, from broad based directional policy pronouncement, through more focussed functional policy formulation, to specific procedural policy execution. This path can also be described in terms of the development from words to specifications to code.

Societal Policy (Principles): The most broadly based and difficult to characterize class of policies are those that apply to human interactions in general, and in essence prescribe modes of conduct. While in some sense the most basic, they are also the most subjective. This level of principles is included here because societal changes tend to affect directional changes that begin to dictate the decision making processes. (E.g., quality as a primary criterion for success.)

Directional Policy (Goals): At this level policy remains subjective and its ramifications remain unclear. These policies are designed with the intend to actually dictate the direction in which processes will be routed. Examples are organizational and corporate goals. This type of policy must be broken down in order to be effected, and may be propagated to many areas within an organization. Clearly this is spoken, often written policy, and is not expressed in any structured manner that would be amenable to computer implementation. (E.g., quality goals.)

Organizational Policy (Practices): At the next level policy still remains predominantly subjective, but is generally better understood. This type of policy deals with interpretations of corporate goals or directives in terms of their organizational impact. Plans are developed, approaches are formulated, and contractual agreements are often engaged at this level. This is policy put into words, as is directional policy, but more formally in terms of descriptive documents. A broad set of goals begins to be associated with the terms of organizational policy. (E.g., quality programs.)

Functional Policy (Targets): It is at this level that the process of mapping strictly spoken policy to more precise methods of characterization begins. The functions that must be accomplished to carry out organizational policies are identified, and the informal descriptions are broken down into fragmented considerations that are more specific in nature. These considerations are formulated in terms of the functional areas of importance, e.g., integrity requirements, configuration specifications, or workload targets. (E.g., quality measures.)

Process Policy (Guidelines): The decoding of policy is completed at this level by the identification of the sets of processes that must be sustained, and the manners in which they interact. Fragmented expression of policy are associated with some structured language, such as pseudocode, macros, library organization definitions, database schemas, etc. Some encoding for computer support may actually take place at this level. (E.g., quality tracking.)

Procedural Policy (Rules): At this level the statements of policy no longer exist, having been completely replaced with encoded procedures that are executable. They are written in languages that computers understand. Some examples are: command lists, program exits, supervisor calls, etc. (E.g., manufacturing support software.)

4. THE TRANSFORMATION OF POLICY

A single directional policy could generate many sub-policies at the various levels in the hierarchy, because it is being transformed and proliferated as it travel down the hierarchy into other kinds of policies, as depicted in -- Figure id 'smpb' unknown --.

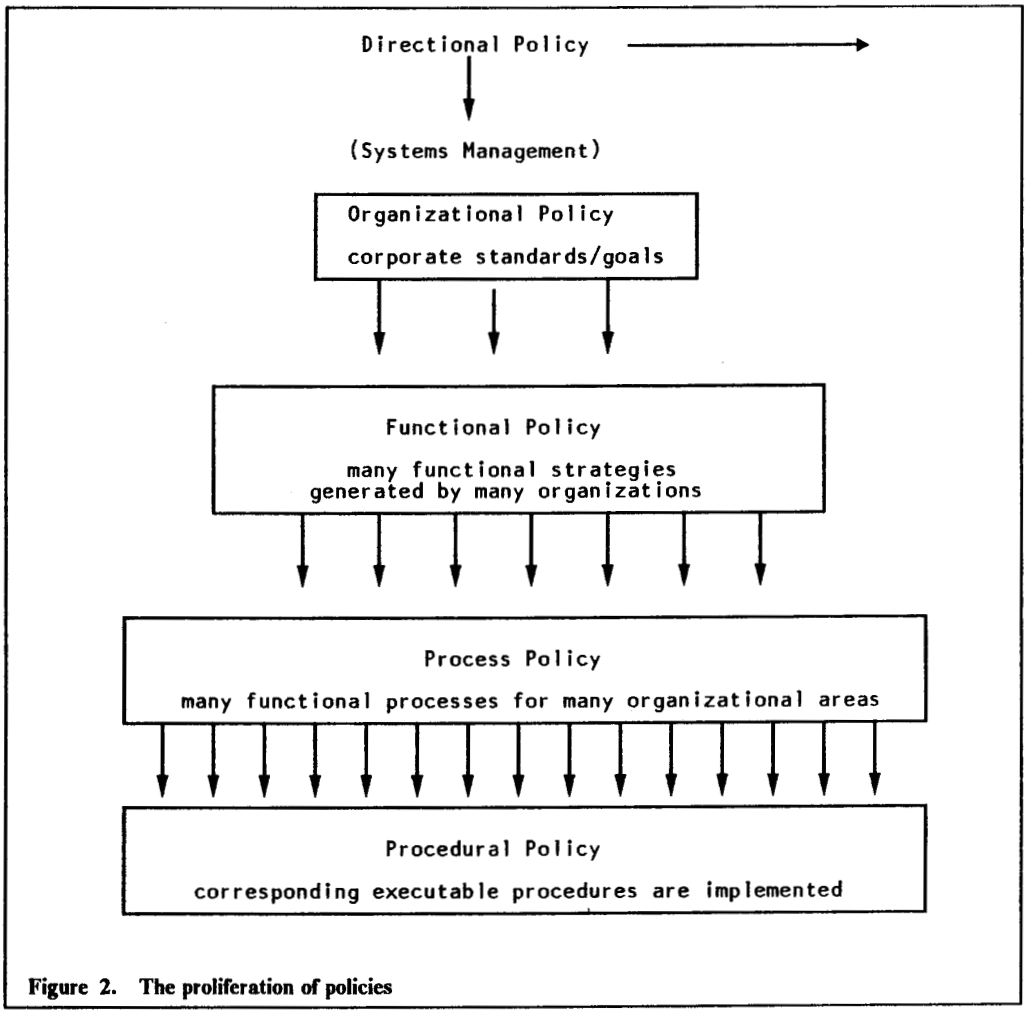

Figure 2. The proliferation of policies

Information Processing Examples

In order to make some of the distinctions more concrete, included below are some examples of policies at different levels as they might be applied to an information processing system.

Customer Satisfaction Example

Directional Policy (Goal):
Achieve 100% Customer Satisfaction

Organizational Policy #1 (Practice):
Meet or exceed all service measurements

Functional Policy #1.1 (Target):
Batch Jobs will complete print by 07:00 am

Process Policy #1.1.1 (Guideline):
At 07:30 am Time Sharing responsiveness
is more important than batch throughput

Functional Policy #1.2 (Target):
No outages to IMS and CICS during first shift

Process Policy #1.2.1 (Guideline):
Activate Cross-system recovery resources
at 06:45 am

Cost Control Example

Directional Policy (Goal):
Effectively Control Cost/Performance

Organizational Policy #2 (Practice):
Make computer resources available just-in-time

Functional Policy #2.1 (Target):
DASD installed where needed, as needed

Process Policy #2.1.1 (Guideline):
Maintenance of MVS "institutional" systems has priority over minidisk growth

Procedural Policy #2.1.1.1 (Rule):
Implementation of PR/SM definitions to favor certain Logical Partitions

Process Policy #2.1.2 (Guideline):
1st shift computing requirements for VM development have priority over VM admin.

Procedural Policy #2.1.2.1 (Rule):
Implementation of PR/SM definitions to favor certain Logical Partitions

Functional Policy #2.2 (Target):
PWS technology deployed where needed

Process Policy #2.2.1 (Guideline):
RISC and PS/2 technology allocated to development based on schedule and revenue impacts

Process Policy #2.2.2 (Guideline):
Favor CICS/2 and EASEL Runtime over ED/VM

5. POLICY MANAGEMENT

Given the distinctions that have been made concerning the different levels of policy, and the characteristics of the policy information at each level, we now explore the application of computer technology to help support the management of policies within an enterprise. This requires an understanding of machine limitations at each level of the policy making hierarchy, i.e., what can a machine do and at what level. Also, what can't machines do for Policy Management, given today's technology. In some sense, Policy Management faces all the difficulties associated with natural language processing, in that policies are generated by and meant to be understood by people, yet more and more we would like them to be implemented in (or at least supported by) computer software.

Transformation and Representation

The proposed hierarchical organization seems not only useful for understanding the underlying evolution or transformation of policy, but it is also suggestive in terms of the design and architecture of an appropriate software representation. The process of moving from one level to another in the **Policy Hierarchy** can be viewed as a process of decoding pieces or **fragments** of policy. Moving from a higher level to a lower (closer to computer executable policy) is a function of varying degrees of understanding. Higher levels involve less specification, but more language. As the fragments of language are decoded, the policy statements are simplified, more compact - less language, more specific considerations, more structure is enforced.

It is important at which level you attempt to enter the hierarchy with computer support. While communications oriented mechanisms (e.g., word processing, electronic mail) may be useful at all levels, semantic support for considerations above the Process Policy level are not presently feasible. Even the transformation of Functional Policy into Process Policy also requires an understanding of the language in which the functional considerations are originally expressed. The level of understanding needed is thus that of human intelligence. While the transformation process cannot thus be expected to be done entirely by computer, some advantages in terms of systematization and consistency of expression can be accrued by providing computer support for the process. A grammar/lexicon-driven approach will be proposed to facilitate this stage of interpretation.

The final transition to Procedural Policy may be completely suitable for that level of understanding that is available artificially to a computer systems, that of understanding the goals and functions of processes. Thus, as the formulation of policy progresses, the decoding into concrete specifications is succeeded by an encoding into machine interpretable modules. Thus while the original goal is to create explicit representations of the policy considerations, the later stages of the process are concerned with the encoding of these representations so that they can be integrated into the operation of the system.

In order for such a policy management process to be possible, there must exist a common representative model of the system and its control structures in order to maintain context, coherence, and integrity. It will be suggested that this model should reside in an object-oriented information facility that maintains a real-time representation of the system being managed and the status of its constituent components.

In summary then, the above suggests that the transformation from spoken or written statements of desired policies to computer executable modules would be founded upon:

- real time representation of the structure of the system model,

- policy statement representation,
- control process representation, and
- structural integration.

Each would need to be done explicitly and systematically within the framework of an appropriate overall policy management system.

6. AN ARCHITECTURE FOR POLICY MANAGEMENT SYSTEMS

An architecture for Policy Management is suggested based upon the indicated classification of the various levels in the policy making hierarchy, and an approach for the transformation of policy from the natural form in which it is most often expressed to its computer implementation counterpart. Tools are then introduced and outlined that would aid in this transformation, by a process of decoding policy statements and encoding policy considerations. The challenge is to maintain explicit representation at each step of the process, thus creating implementable policy considerations that support the original intent.

A **Policy Management System** must support both the transformation and the representation of policy as described above. Minimally it should consist of three specific components:

1. a format for representing policy statements, or more specifically fragments of policy;
2. a form of representation for the objects and processes upon which the policy structure is formulated; and
3. a vehicle for explicitly capturing and administering the effect of the policy structure on the processes acting upon the system objects.

The goal is to create processes that are generated as a consequence of explicit representation of both the objects and the considerations that bear upon those objects. In this approach the objects themselves are the origin and cause of system activities, rather than being viewed as arguments or by-products (input/output), and bear direct relationship to the executable applications that effect policy. These relationships forming the basis for a Policy Management Architecture are illustrated in Figure 3 on page 10.

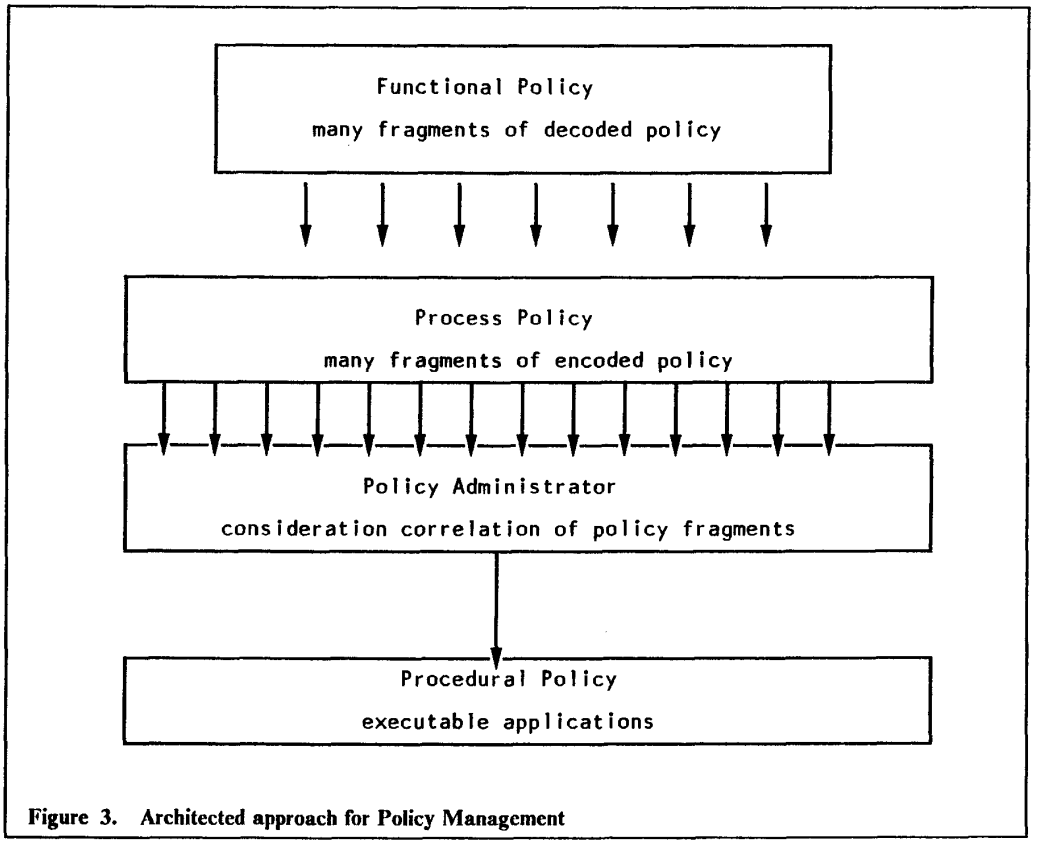

Figure 3. Architected approach for Policy Management

7. NECESSARY TOOLS

The components of a Policy Management system based upon the above description would then be similar to the ones described below.

Functional Specifications Language (FSL)

The considerations and attribute (parameter) definitions constituting policy statements must be elicited in a structured manner. This could be accomplished simply by constraining policy administrators to function statements from a standardized user interface for systems management activities. Such statements could be input to a workstation-based system console via a language component that could be graphic in nature.

The difficulty with a strictly taxonomical approach, is that it lacks the flexibility needed in a dynamic management framework. As the system changes and resources are added, deleted, and changed, the available primitives must also change in such a way that desired policy statements can be entered unambiguously. This would require the real time evolution of the management vocabulary.

It is therefore suggested that the lexicon be dynamically established by the system model. The set of objects associated with the Object Oriented Information Manager would effectively become part of the form of expression for functional policy. Constructs would be provided so that the objects are self-defining with respect to appropriate management activities. Creating a lexicon would involve the creation of a **concept set** of considerations.

Each concept set contains lists of attributes of the **considered objects**. These can be **Consideration Attributes** or **Expression Attributes**. Consideration Attributes relate to the ways in which the objects are of concern and loosely refer to actions that can be taken against that object; e.g., start, purge, cancel, etc. Expression attributes express the form of manipulation. These can be **Computable Expression Attributes** e.g., minutes, lines, etc; or they can be **Non-computable Expression Attributes**, e.g., up, down, waiting, etc. These in general correspond to some expression of the state of the object.

To guide the lexical use of the objects a grammar is needed. As objects are being described in terms of attributes and considerations, creating policy statements, the system parses the fragments and looks ahead in the grammar, presenting only those options that make sense. It is expected that only a small, expressive, carefully designed, core grammar is required to capture policy fragments succinctly.

The goal is to provide a mechanism for expressing policy that starts at a level higher than that of directly encoding procedures, and, that could be used to generate those procedures. The level of interaction should correspond to the level of decoding written or spoken policy.

Object Oriented Information Manager (OOIM)

For the explicit representation of the information structures needed, an object oriented (OO) approach seems the most effective (Finkel, 91). It is a natural mechanism for describing systems management elements (CPU's, DASD, controllers, printers, operating systems, access methods, application programs, network links, channel connections, etc.), and it has modularity and structuring characteristics that can be of great value in dealing with complex systems. Object oriented approaches provide a a concise representation of the concept of **genericness** with (normalized) message invocations of object-specific methods. This is very important in attempting to simplify the software interactions needed to control many different system components.

Systems management requires dynamic definition. Addition and deletion of instances of objects need to be supported without requiring the redefinition of the entire software system. Object oriented designs are quite useful in this regard; and, further, encourage a supplier/consumer model, allowing disparate software from diverse sources to work together. They also provide a natural expression of concurrency.

It is therefore suggested that **from a systems management point of view**, the enterprise system should be explicitly modelled in an active, object oriented representative model that can keep track of status information as it evolves. This allows all the management software to deal with a common, well defined, higher level representation of system activity. Such form of representation would capture the set of items associated with the terms of organizational policy. The method programs associated with objects in the active model can then deal with the specific details of interactions with the system components that they represent.

Methods Specification Language (MSL)

Having explicitly described the objects in the Object Oriented Information Base, the task then becomes to explicitly describe the active processes (sets of actions against those objects). A language is needed that specifies process flows and takes the considerations and attributes characterizing

policy statements and maps them into internal representations that can be referenced and utilized by the enterprise system software (Masullo, 90).

Since **Methods** capture the behavior of objects in the system model, this is a language for the **specification of methods**. The language breaks down the specification in terms of two basic classes of elements - one that identifies the considerations, attributes, and process descriptions; and, one that contains the actual executable procedures. Each consideration/procedure set is referred to as a **Fragment**.

All active fragments relevant to a particular process are **considered** concurrently by a piece of software called the **Administrator**. It allows many policy issues to be taken into account and resolves conflicting considerations by means of specially designed algorithms. The information flows that form the basis for possible control mechanism in Policy Management are represented in Figure 4.

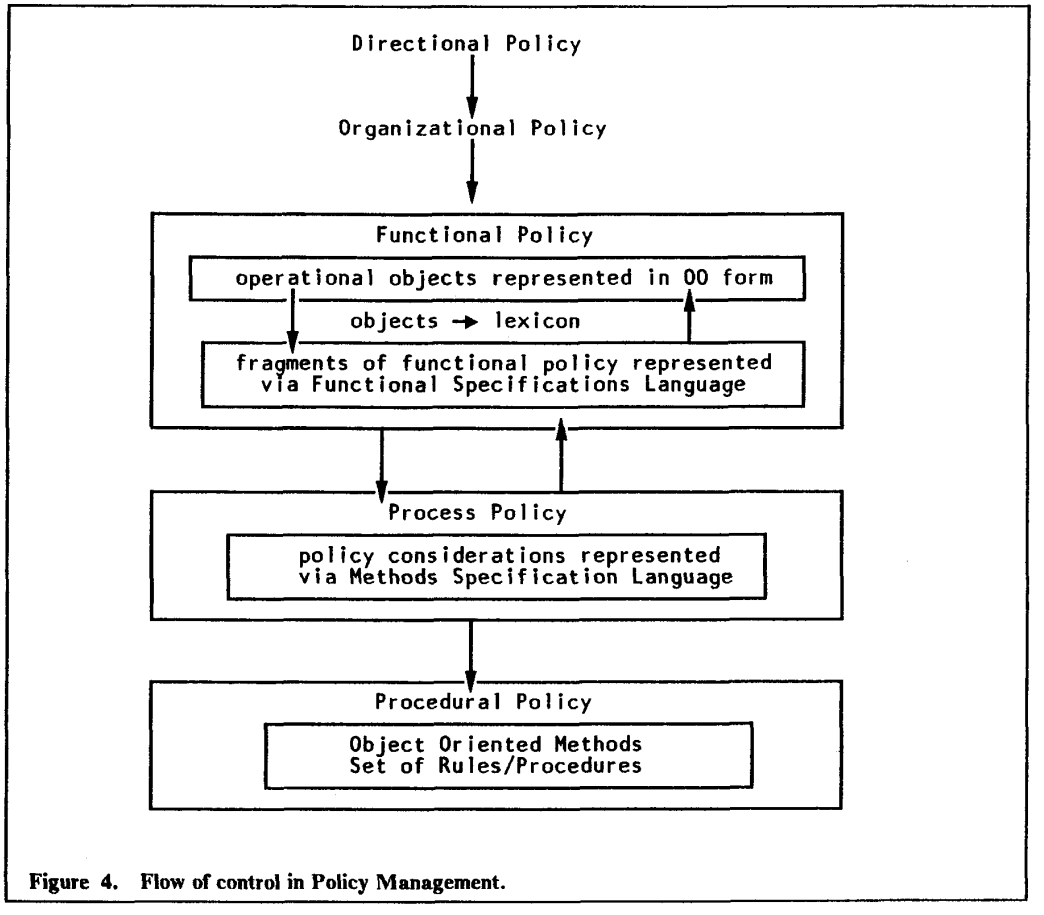

Figure 4. Flow of control in Policy Management.

8. A CONSISTENT POLICY MANAGEMENT APPROACH

The ideas presented above constitute an approach for providing Policy Management in complex systems that attempts to implement policies themselves at a more abstract level by gradual and

explicit representation of all of the information components involved. It suggests a consistent methodology for an area that has not been actively investigated, but is of vital importance to the manageability of complex systems.

The application of these techniques to Systems Management has been the underlying motivation in the definition of the necessary tools, but the tools themselves, and more to the point the ideas behind the tools are not limited to this particular application area. By adjusting the grammar and the lexicon employed, and specifying different consideration fragments, the types of tools described should be useful for completely different application domains, e.g., Robotics, instructional management and educational policy and in general business policy.

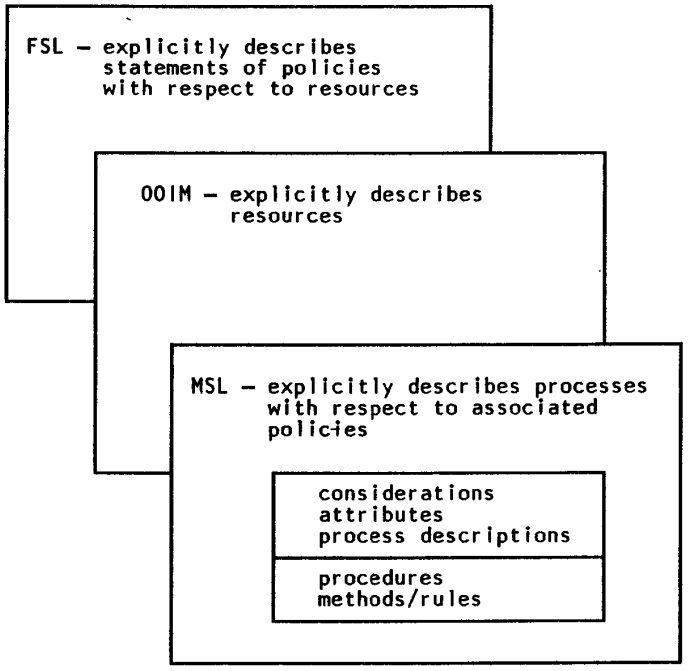

Figure 5. Consistency of approach in Policy Management.

9. SUMMARY AND CONCLUSIONS

It is clear that large, complex systems cannot be effectively managed unless facilities are explicitly provided to incorporate a base set of necessary functionality. Manageability needs to be considered as important a systems design requirement as performance and availability.

In studying systems management problems, we have come to the conclusion that explicit representation is needed in several key areas. Management processes that deal with **configuration, change, performance, problem, and operations** issues need to have accurate, timely information on

the status of system resources. An object oriented model management facility has been recommended for meeting this requirement.

Further, the management processes that define and control the operation of the enterprise system must also be effectively represented. As systems become larger, more interconnected, more comprehensive, and more critical to the real time functioning of the organization, simply keeping track of the active processes becomes difficult unless explicit structural approaches are maintained. (Consider walking in to an active computer center and being asked to determine how it is functioning, or even what is going on.) Given the object oriented facilities suggested for representation, the process flows are expressed in method programs, and so a methods specification language has been proposed for making it easier to define, identify, and document process descriptions.

In defining processes, it is immediately clear that some mechanism must be provided to allow the specification and application of those organizational policies that determine how processes should behave and interact. The methods specification language must thus be extended to capture policies in consideration fragments, thus allowing the conditional execution of alternative actions based on context.

While this allows Procedural and Process Policies to be incorporated in a formal, structured, and identifiable way, it does not capture the higher level policy issues that led to the implemented considerations. A Functional Specifications Language is thus suggested for the description of the policy considerations in a manner that is more convenient and understandable.

This more direct support methodology for the administration of policies should not only make it easier for them to be specified, and maintained, but also result in more easily altered policies and more tractable procedures. The benefits of this approach would include the ability to more systematically study the effect of policy over time, focusing on the needs of the enterprise and increase opportunities for automation. The dynamic nature of systems and systems management makes change a critical requirement, and the need to be able to ascertain the consequences of changing policies is one of the most challenging of systems problems.

References

[Bridges 77] Bridges,F.J., Olm,K.W., Barnhill,A. *Management Decisions and Organizational Policy* Allyn and Bacon, INC. (1977)

[Finkel 91] Finkel,A., Calo,S., Klein,D. *Model Management and the Automation of Computer Systems Operations* IBM Research Report #16932 (1991)

[Jauch 88] Jauch,L.R., Glueck,W.F. *Business Policy and Strategic Management* Allyn and Bacon, INC. (1977)

[Masullo 90] Masullo,M.J., Mozes,E. *A Methods Specification Language for Object Oriented Databases* IBM Research Report #16360 (1990)

[Masullo 92] Masullo,M.J. *A Functional Specifications Language (FSL) for Object Oriented Databases* unpublished IBM Research Report