J Netw Syst Manage (2007) 15:447–480
DOI 10.1007/s10922-007-9083-8

Policy-based Management: A Historical Perspective

Raouf Boutaba · Issam Aib

Published online: 3 November 2007

Abstract This paper traces the history of policy-based management and how it evolved from the first security models dating back to the late 1960's until today's more elaborate frameworks, languages, and policy-based management tools. The focus will be on providing a synthesized chronicle of the evolution of ideas and research trends rather than on surveying the various specification formalisms, frameworks, and application domains of policy-based management.

Keywords Policy-based management · Policy-based networking · Policy history

1 Introduction

Policy-Based Management (PBM) is a management paradigm that separates the rules governing the behavior of a system from its functionality. It promises to reduce maintenance costs of information and communication systems while improving flexibility and runtime adaptability. It is today present at the heart of a multitude of management architectures and paradigms including SLA-driven, Business-driven, autonomous, adaptive, and self-* management.

This paper traces the history of policy-based management and how it evolved from the first security models dating back to the late 1960's until today's more elaborate frameworks, languages, and policy-based management tools. The focus will be on providing a synthesized chronicle of the evolution of ideas and research trends rather than on surveying the various specification formalisms, frameworks, and application domains of policy-based management.

R. Boutaba (✉) · I. Aib
David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, Canada
e-mail: rboutaba@uwaterloo.ca

Although the focus is on the chronology of events, we have nevertheless classified the works based on different functional areas of policy in order to best highlight these areas and to bring a consolidated account of the research efforts that were conducted over the years.

Section II presents the earliest works on security policy, dated as far back as 1966. Section III begins with the mid 1980's where policy started to be identified as a management paradigm within the network and distributed systems management community. Section IV relates the activities that emerged in the late 1980's and early 1990's in the use of policy for inter-networking and routing. Section V considers the evolution of policy specification in terms of languages as well as information models. Sections VI and VII trace the recent shift in interests of the community from policy specification to policy refinement techniques where the refinement and analysis problem was first identified in the 1990's. Section VIII-B relates the recent growing interests of the information systems management community in the optimization of the business value [1] and using policy to design elaborate business-driven, autonomous, and self-managed frameworks. Finally, we conclude with a summary of the collective achievements of the research community some directions for the future.

2 Security First

The early works on policy focused on emphasizing security considerations first constrained to single time-sharing mainframe computers of the 1960's, expanded later to individual enterprise boundaries, and then evolved to target multi-network environments. The years from 1972 to 1975 marked a burst of security modeling activities which followed the security concerns raised by the widespread success of the time-sharing technology, the continuous decreasing cost and size of computers, as well as the spectacular success of computer-security experts such as the "tiger teams" in easily attacking and taking over systems [2].

A *security policy* defines the (high-level) rules according to which access control must be regulated. Access control is the process of mediating every request to resources and data maintained by a system and determining whether the request should be granted or denied. The access control decision is enforced by a mechanism implementing regulations established by a security policy. Different access-control policies can be applied, corresponding to different criteria for defining what should and should not be allowed, and to some extent define the different means of security assurances [3].

A security model implements security policy by providing a formal representation of the policy and its functions. The formalization allows the proof of properties on the security provided by the access control system being designed. The security mechanisms further define the low-level (software and hardware) functions that implement the controls imposed by the policy and formally stated in the model [3].

2.1 Access-control Matrices (1966, 1969)

In 1969, Butler W. Lampson [4], [5] introduced the abstract concepts of protection domains and access-control matrices for shared computer systems. In his model, the essential property of a domain is that it has potentially different access rights than other domains. Objects are shared between domains and are the things of the system that require protection. Typical objects include processes, files, memory segments, terminals, and domains. Access control is assured through an access control matrix. Each (domain, object) entry in the matrix contains a list of access attributes which define the access rights of that domain to the objects. Attributes can be of different forms, such as read, write, owner, call, control, etc.

Lampson model, also known as the Access Control List (ACL), views the access control matrix in a column-wise fashion. An alternative approach would be to view them in a row-wise manner. This is the approach taken by the capability-based access control model. This model partition the access-control matrix by subject rather than by object. It was first introduced in 1966 by J.B. Dennis [6] and later elaborated by R.S. Fabry [7] in 1974.

2.2 Bell-LaPadula Model (1973)

The Bell-LaPadula Model [8] is considered to be the earliest formal model for data confidentiality through policies. A *confidentiality policy* prevents the unauthorized disclosure of information. The model was developed by David Elliott Bell and Len LaPadula in 1973 to formalize the United States department of defense multi-level security policy. It is a formal state transition model of computer security policy that describes a set of access-control rules (policies) which use security labels on objects and clearances for subjects. Security labels range from the most sensitive, e.g., "Top Secret", down to the least sensitive, e.g., "Unclassified" or "Public".

The Bell-LaPadula model is built on the concept of a state machine with a set of allowable states in a system. The transition from one state to another state is defined by transition functions. A system state is defined to be"secure" if the only permitted access modes of subjects to objects are in accordance with a security policy. With Bell-LaPadula, users can create content only at or above their own security level (e.g. "Secret researchers" can create "Secret" or "Top-Secret" files but may not create "Public files") reflecting the *no write-down* policy. Conversely, users can view content only at or below their own security level (Secret researchers can view Public or Secret files, but may not view Top-Secret files) reflecting the *no read-up* policy.

2.3 Integrity Policies (1977)

In conjunction with confidentiality, an *integrity policy* describes how the validity of the data items in the system should be maintained from one state of the system to another and specifies the capabilities of various principals in the system. The first model for integrity policy was proposed in 1977 by Biba [9].

2.4 D. Clark and D. Wilson Model (1987)

Ten years later, D. Clark and D. Wilson's work on "A Comparison of Commercial and Military Security Policy" [10] stimulated further consideration of novel security policies. Clark's primary contribution to security policies was in introducing access triples (user, program, file), where previous work had used duples (user, file) or (subject, object). The following year witnessed an explosion of Clark-Wilson response papers. In 1989, D. Brewer and M Nash published "The ChineseWall Security Policy" [11], abstracting British financial regulations. Its contribution was a user's free-will choice limiting future actions in a non-discretionary way [12]. The multiplicity of these policies, however, was more apparent than real, as is explained in [12].

The henceforth mentioned security models are now classified as part of *Mandatory access control (MAC) policies*. These policies enforce access control on the basis of regulations mandated by a central authority.

2.5 D. Estrin's Access-control of IONs (1985)

The need of policy in network management has first been addressed by Deborah Estrin in 1985 in her access control solution for Inter-Organization Networks (ION) [13], [14]. An ION is formed when two or more organizations interconnect their internal computer networks. She demonstrates that traditional network design criteria of connectivity and transparency need to be complemented by a primary high-level requirement which is that of access-control.

In an ION the goal is not to prohibit access by outsiders as some outside access is explicitly desired. However, there is a need to control invocation as well as information flow, as the potential damage of unauthorized access to strictly internal resources can be high. The obvious solutions of (1) physically isolating externally-accessible systems from internally-accessible systems, and (2) increased access-control on all internal systems are not acceptable. This is because IONs evolve in number and extent and may overlap; Thus, they require the implementation of logical rather than physical network boundaries.

The paper was concerned with access control policies such as (i) Only MIT computer science researchers can access the ARPANET, (ii) ABC users have access to the MIT educational software host, (iii) XYZ users have access to the MIT design simulation program, and (iv) Both ABC and XYZ users have access to MIT electronic mail.

There are implications of implementing such policies on the design of the ION gateways, the logical grouping of users into category sets (similar to domains), as well as the equipment of ION facilities (software distribution servers, electronic mail server, etc.) with discretionary and non-discretionary access-control. It is very difficult to identify in the general case the category set of a packet based on its header. The IP header for example provides information useful to the routing of a packet to its destination. This routing information pertains to the physical location of the destination. When networks cross organizations as well as geographic

boundaries, logical information is desired in addition to topological information. That is, policy control mechanisms need to know the organization domain to which a message is being sent and from whence it came, in addition to the physical locations.

Policy can be enforced at the packet-level at the ION gateway. However, early gateways and the packet-level protocols as they were defined, do not carry enough information in each packet header that the gateway needs to make policy decisions. Although higher-level gateways are able to implement more refined control based on the application type, such as using the Transport protocol number, it is not possible to make all kinds of policy decisions based on packet information solely.

As an alternative to packet-level interconnection, D. Estrin suggests two possible schemes. First, a centralized access-control policy server authority is used in order to map traffic information to category information and grant appropriate access privileges through a visa scheme. This scheme has an inherent imperative to change at least the checksum generation/calculation of packet-level protocols. The second alternative would be to implement higher-level controls at the end hosts. This latter alternative allows the control based on the semantic information of a message such as the size of a purchase order, or the name of a requested file.

2.6 Role-based Access Control Policies (1992)

Role-based Access Control (RBAC) is a security mechanism that has gained wide acceptance in the field because it can lower the cost and complexity of securing large networked and web-based systems [15]. RBAC was first introduced in 1992 by David Ferraiolo and Rick Kuhn [16]. Although RBAC is not directly concerned with policy specification, it has been accepted as a security model which permits the specification and enforcement of organizational access control policies [17].

Role-based policies regulate the access of users to the information on the basis of the organizational activities and responsibility that users have in a system [3].

Instead of specifying all the accesses each user is allowed to execute, access authorizations on objects are specified for roles. RBAC has a number of advantages. It allows better authorization management by separating user assignment to roles and the assignment of authorizations to roles. Role hierarchies are supported to reflect natural administrative hierarchies and can be exploited in authorization propagation and delegation. To minimize the danger of powerful roles, users can sign on with the least privilege required for a particular activity. Finally, RBAC allow better static/dynamic constraints enforcement, such as a cardinality constraint to restrict the number of roles allowed to exercise a given privilege [3].

In 2000, the National Institute of Standards and Technology (NIST) proposed a unified model for RBAC, based on the Ferraiolo-Kuhn (1992) model, in the framework developed by Sandhu et al (1996). The model was further refined within the RBAC community and has been adopted by the American National Standards Institute, International Committee for Information Technology Standards (ANSI/INCITS) as ANSI INCITS 359-2004 [18]. Some policy specification languages have also adopted the RBAC security model such as Ponder [19] and XACML [20].

3 Policy as a Management Paradigm

With the continuing efforts in the use of policy for security and access control, the recognition of the need for and the standardization of the Internet domain name service (DNS)) [21], [22], the appearance of domain-based systems for access control in commercial distributed systems [23], as well as the demonstrated need for using policies and domains for IONs [13] made it desirable to take a more abstract approach to the use of domains and management policy.

In this perspective, the recognition of policy as a management paradigm can be traced back to 1988 with the PhD works on domains for policy-based management of D.C. Robinson [24]–[26], J.D. Moffet [27]–[29], and K. Twidle [30], [31] under the supervision of M. Sloman.

D.C. Robinson and M. Sloman [25] view Management policy to be the sets of rules for achieving the scalable management of a distributed computing system, as well as achieving the sub-objectives of optimizing resource usage, cost, revenue, and performance. The requirements of scalability, flexibility, uniformity, and verifiability are believed to be achievable through policies and management domains.

A domain is viewed as a set of resources to which the same policy applies. The domain also has an associated manager who can alter its associated management policies. Hence, domains represent the sphere of influence of a manager.

Domains provide a powerful, uniform, yet flexible, means of managing large distributed computing systems. They can be applied in the management functions of Fault, Configuration, Accounting, Performance, and Security. The research focus of Sloman's team was more targeted to security concerns.

Sloman's domain concept provides additional structuring over Estrin's view of domains in two ways. First, domains are object oriented, hence can have types and instances. In addition, they can have disjoint, sub-domain, and overlapping relationships as well as reflexivity and transitivity properties.

On the other hand, Estrin's domains provide an efficient solution that enforces access control first at the ION gateways then at the user process level when higher-level knowledge is required. Domains as conceived by Sloman may in turn downgrade performance due to their high-level structuring. They require encryption keys to be associated to each security domain, with all communication between participating processes encrypted using that key. Hardware encryption may then be required to achieve acceptable performance [25]. An example implementation of domains in structuring access control policy based on capabilities is described in [24] and one based on access control lists is described in [30].

Similarly focusing on the technology of management (as opposed to the management of a given technology), Boutaba [32] defines a goal-oriented recursive structuring of management domains (figure 2(a)). Domains can be organized into hierarchical and/or cooperative structures. Each domain is composed of a manager and a set of managed resources. A manager at a given level receives management goals from its superior manager, decides on the policy to follow in order to achieve the goal(s), transforms policies into management plans, then executes the plans by sending appropriate sub-goals to its subordinate managers or control actions to the

managed resources in the case of a last level manager (figure 2(b)). The construction of domains may be realized according to different criteria such as location (physical boundaries), policy (same policy applies to the domain resources), organization, or ownership (for security and access control).

Domains as defined by Boutaba, although restrictive in the way they are internally structured, facilitate management automation by means of recursive policy-driven refinement and delegation of management goals throughout the management hierarchy.

4 Networking Policies

As the Internet grew, the need for policy to enforce network quality of service and to select routes in order to restrict the use of network resources to certain classes of customers became a necessity.

4.1 Clark Policy Routing (1989)

In his 1989 request for comment, D. Clark [33] underlines the lack of existing network protocols dealing with routing policies. A routing policy is concerned with finding the appropriate network route from source to destination, for each customer traffic, that satisfies the requirements on authorization, quality of service, and cost. He identified the Exterior Gateway Protocol (EGP) [34] to be the closest existing mechanism which can serve as a starting ground for implementing policy routing.

In his proposal, a packet contains a Policy Route (PR), which is verified by each Policy Gateway along the way. Scalability is achieved through the use of *Administrative Regions* (ARs), which may include networks, links, routers, and gateways.

Each AR holds a list of policy terms that determine its routing policy regarding traffics coming from other ARs. The policy term structure is described in section V-A.

Clark argues that PR creation cannot be fully automated by the system, but rather will in general require human intervention. The approach to PR creation is thus a joint one, in which the system provides support to the policy administrators. Most commonly, the desired PR will be selected from among those available by first finding all valid PRs, and then selecting one that meets the requirements of the user and has the lowest cost. These two stages are called synthesis and selection. To synthesize a PR across a sequence of ARs, one must find a Policy Term in each AR that would permit such a PR. The Policy Terms in each adjacent AR must be compatible in their billing conditions and other particularities.

A host wishing to send packets outside the local AR must first obtain a PR to put the packets into. In the normal case, it would do so by directing a request to the local Policy Server, supplying the desired destination and other negotiable conditions. (For example, the Type of Service (TOS) is negotiable, the current time is not.) The Server, based on this input, must select the most appropriate PR and return it.

4.2 BGP (1989), IDRP (1994) and IDPR (1993)

Building on the experience gained in EGP, the Border Gateway Protocol (BGP), first published in 1989 [35], is the current core routing protocol of the Internet. The primary function of a BGP speaking system is to exchange network reachability information with other BGP-based systems. This includes the list of Autonomous Systems (ASs) to traverse. At the AS level, some policy decisions may be enforced.

Policy enforcement is not part of the protocol itself, but instead is manually configured at each BGP router. The enforcement of the policies is accomplished in two ways. First, by specifying the procedure used by the AS router itself to select the appropriate paths, and second by controlling the redistribution of routing information to neighboring ASs. Policy decisions can be based on political or competition constraints (such as to avoid a non trusted AS) or performance aspects in order to choose the appropriate AS path.

The Inter-Domain Routing Protocol (IDRP) was defined in 1994 in ISO 10747 [36] for OSI defined network environments. BGP, IDRP allows manually enforceable policy-based routing.

In 1993, Steenstrup introduced an architecture for inter-domain policy routing (IDPR) [37]. Unlike BGP and IDRP, the IDPR architecture supports link state routing information distribution and route generation in conjunction with source specified message forwarding. IDPR defines *policy routing* as the route generation and message forwarding procedures for producing and using routes that simultaneously satisfy user service requirements and respect transit domain service restrictions.

With policy routing, each domain administrator sets *transit policies* that dictate how and by whom the resources within its domain should be used. Each domain administrator also sets *source policies* for traffic originating within its domain. Unlike transit policies, source policies are usually private. They specify provided and requested services based on: domain access restrictions, network path quality, and cost.

Route generation using IDRP is inherently complex and involves a combination of service constraints. For example, “finding a route delay of no more than S seconds and a cost no greater than C”. Most of these multi-constraint routing problems are NP-Complete [38]. To cope with this complexity, IDRP supports the ability to group Administrative Domains (ADs) into superdomains, hence forcing the existence of a domain hierarchy.

4.3 IETF/DMTF Policy Framework (1999)

The IETF [39] and the DMTF [40] have shown their interest in network policies and put considerable effort for creating a standard policy framework and related protocols. RFC 3198 [41] defines a network policy as a set of rules to administer, manage, and control access to network resources. Policies are defined by an administrator and specify how the devices in the network should deal with different types of traffic. They also provide a way of consistently managing multiple devices comprised of complex technologies.

The IETF Policy Framework (POLICY) Working Group [42] has developed a policy management architecture for policy-based networking, the basic components of which are depicted in figure 1. The *Policy management tool* is a human friendly user interface for specifying, editing, and administering policies that are to be enforced in the network. The *policy repository* is used to store the policies generated by the management tool. It can be a LDAP server or a Directory Enabled Network (DEN) device. The entities (such as routers, firewalls, and hosts) that can take actions to enforce the policies are known as the Policy Enforcement Points (PEPs). The *Policy Decision Point* (PDP) is a resource manager or policy server that is responsible for handling policy events, retrieving the policies from the repository, interpreting them and deciding on which set of configuration policies ought to be enforced by the PEPs. The PDP enforces the policies based on the "*if condition then action*" or "*at time do action*" rule sets it has received from the PDP. A *local policy decision point* (LPDP) is a scaled-down PDP that may exist within a network node and is used in cases when a policy server is not available. Basic policy decisions can be programmed into this component.

Subsequent to the IETF/DMTF adoption of policy-based networking, several works proposed similar or more elaborate designs of policy architectures particularly in the area of quality-of-service management in Differentiated Services (DiffServ) networks. Some of the early works include [43]–[46].

4.4 The Common Open Policy Service (2000)

In January 2000, the Common Open Policy Service (COPS) protocol [47] has been published by the IETF Resource Allocation Protocol (RAP) working group in RFC 2753. COPS is aimed for the general administration, configuration, and enforcement of policies in IP networks. It is a simple query and response protocol for the exchange of policy information between a policy server (PDP) and its clients (PEPs). It is assumed that at least one policy server exists in each controlled administrative domain. The basic model of interaction between a policy server and its clients is compatible with the framework for policy-based admission control defined in RFC 2748 [48].

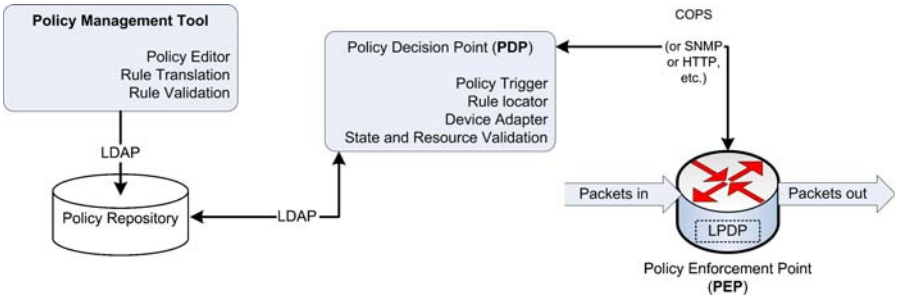

Fig. 1 The IETF policy framework

COPS supports two common models for policy control: outsourcing and provisioning. The Outsourcing model addresses the kind of events at the PEP that require an instantaneous policy decision (authorization) in which the PEP delegates responsibility to an external policy server (PDP) to make decisions on its behalf. In the Provisioning model, the PDP may proactively provision the PEP reacting to external events (such as user input), PEP events, and any combination thereof. RFC 3084 [49] defines COPS-PR, which is the COPS client type for the provisioning model. In addition, RFC 3317 [50] specifies a set of policy rule classes for configuring QoS Policy for Differentiated Services and are intended for use by the COPS-PR QoS client type.

In COPS-PR, each client has to maintain a special database, the Policy Information Base (PIB) [51], where all the received configuration data are stored. The PIB is a structure similar to a management information base (MIB), and can be described as a conceptual tree namespace, where the branches represent structures of data, or provisioning classes (PRCs), and the leaves represent instances of these classes, called provisioning instances (PRIs).

Note that the IETF policy working group has not been active since 2004. The COPS protocol continues nevertheless to be employed in the industry and by other standardization bodies such 3GPP for mobile and IP Multimedia Subsystem (IMS) applications [52].

5 Policy Specification Languages

This section presents a selection of contributions made in the investigation of the appropriate way to represent policies.

5.1 Clark's Policy Term (1989)

The Clark's policy term [33] constitutes one of the earliest works on network policy specification. Clark proposed a template to represent network policies called *policy term* (listing 1). Scalability is achieved through the use of *Administrative Regions* (ARs), which may include networks, links, routers, and gateways. A term is made up of four fields: source, destination, user class, and other global conditions. The source field is made up of the source host IP address, the source AR, and the entry AR. The destination field is made up of the destination IP address, the destination AR, and the exit AR. Wild cards along with user classes allow for better flexibility and scalability [38].

Listing 1 A Clark Policy Term

$((\texttt{132.227.60.13}, 1, -), (\texttt{194.2.232.170}, 3, -), \texttt{University}, \texttt{Unauthenticated UCI})$

Traffic flow marked of class "University" can pass without authentication between the host with IP address 132.227.60.13 in AR1 and the host with IP address 194.2.232.170 in AR3.

The notation used by Clark was a good starting point for abstract network policy representation. However, it lacks the ability to represent explicit paths formed by a sequence of ARs as part of the terms [38]. Also, the language syntax is not natural for the policy manager since it requires the memorization of the semantics of each parameter position in a term.

5.2 Traffic Flow Policy Languages (1998–2001)

These are relatively low-level policy languages that make use of pattern matching in the selection of the network traffic on which a policy can be applied. Their efficiency in policy-based management is limited since they do not provide a view of how the overall policy managed network system will work. A policy is simply an abstract device-level representation of the functionality of a node. PAX-PDL [53], SRL [54], and PPL [38] are some examples of such languages. *Pattern* is the basic concept of these languages. Simple patterns can be combined to form more complex patterns. SRL adds to pattern recognition the ability to act on the identified packet/flow by tracking its statistics. The Path-based Policy Language PPL [38] was intended to bring a satisfactory solution to the integration of the IntServ and DiffServ IETF models by having a policy extended to act on the full data path of a flow. This avoids many NP-complete calculations that try to establish a coherent path based on policies spread over different nodes. Paths are analogous to static routes when they are completely instantiated. Pre-computing paths, on the other hand, may also incur excessive overhead if not designed appropriately.

Listing 2 Priority management in PPL

Assign high priority to the network managers data traffic during working hours.

Policy $<$ net_manager $>$ {*} {traffic_class $=$ {data}} {*} {time$\geq$0800, time$\leq$1600 : priority : $=$ 10}

The use of userID in the grammar specification is not a needed feature and unnecessarily complicates the grammar. As also noticed by Nicodemos [55], it would better be specified as part of some meta-information that could be used for the analysis of policies. Current traffic flow policy languages lack mechanisms for grouping policies and do not tend to take advantage of Object Oriented concepts.

Higher-Level policy languages have also been proposed, starting with the elementary Clark Policy Terms [38]. A policy definition is generally viewed as an "**IF** *Condition* **Then** *Action*" statement or further refined to an "**On** *Event* **Then IF** *Condition* **Then** *Action*".

5.3 Policy Description Language PDL (1999)

PDL [56] is a declarative policy definition language for information systems management developed by Bell Labs. It defines a policy as a *collection of general principles specifying the desired behavior of a system*. Policies are formulated using the Event-Condition-Action (ECA) rule paradigm of active databases and are supported by a rich event sub-language for un-interpreted concurrent actions.

A PDL policy is a finite set of well-typed policy rules of the form "*event* ***causes*** *action* ***if*** *condition*". Policy rules map series of events into sets of actions. The language has clearly defined semantics and has been deliberately limited to a policy specification that is succinct and can be efficiently implemented. According to [57], it has been used in for the policy-based management of the SARAS soft-switch at Bell labs.

Listing 3 Sample PDL policies [58]

```
// policy for product order processing
defectiveProduct causes stop
orderReceived causes mailProduct
orderReceived causes chargeCreditCard
/* policy for Soft-switch overload control in the presence of an excessive number of signalling network
   time outs */.
(normalmode, group(callmade|timeout)) triggers overloadmode if (Count(timeout)/
   Count(callmade)>ThOvld)
```

5.4 The Ponder Framework (2000)

Ponder is a declarative, object-oriented language for specifying security policies with role-based access control as well as general purpose management policies. It has been published by Damianou et al. in [59] and a year later in the 2001 first IEEE workshop on policies [19]. However, the notation used draws upon the early works in [60], [61] and the more consistent specification of Mariott [62].

Ponder views policy as a *rule that defines a choice in the behavior of a system*. This behavior is intended to reflect objectives of the system managers. At a more formal level, a Ponder policy defines a relationship between objects (*Subjects* and *Targets*) of a managed system. It supports four *basic policy* types: *authorizations, obligations, refrains*, and *delegations* and three *composite policy* types: *roles*, *relationships*, and *management structures* that are used to compose policies. It also has a number of supporting abstractions that are used to define policies: *domains* for hierarchically grouping managed objects, *events* for triggering obligation policies, and *constraints* for controlling the enforcement of policies at runtime. All constructs can be specified as single instances or as types that can be instantiated as many times as needed. Types in ponder can be parameterized which adds to the generality of the language.

Listing 4 Application-level video conference authorization [55]

Members of Agroup plus Bgroup can set up a video conference with USA staff except the New York group. The constraint of the policy is composite. The time-based constraint limits the policy to apply between 4:00 pm and 6:00 pm and the action constraint specifies that the policy is valid only if the priority parameter (the 2nd parameter of the action) is greater than 2.

```
inst auth+                          VideoConf1 {
   subject                         /Agroup + /Bgroup;
    target                        USAStaff − NYgroup;
    action                      VideoConf(BW, Priority);
     when     Time.between(‘‘1600’’, ‘‘1800’’) and (Priority > 2);
}
```

Obligation policies are similar to PDL policies and allow the specification of management policies. An obligation states the actions that must be performed by manager objects within the system when specific events occur and a set of conditions are met. That is, obligations are event triggered and they follow the ECA paradigm. Ponder supports a rich event composition mechanism similar to that of PDL (section V-C). Ponder obligations also support exception management in a way similar to object-oriented programming.

Roles are a key concept in Ponder. They provide a semantic grouping of policies having a common subject, such as a DiffServ edge router. Ponder roles are inspired by the RBAC model but do not support inheritance of privileges within a Role Instance Hierarchy - which actually causes problems within RBAC.

Within the scope of a composite policy structure or a domain sub tree, there is a need for *Meta policies* [63]–[65], which are policies regarding permitted policy types and authorized concurrent action sequencing.

It is worth mentioning that several languages for specifying security as well as management policy preceded Ponder. Examples include the language proposed in [66], the one proposed in [67] for the specification of ECA rules for network management databases, the ASL language [68] for formal logic-based access-control policies supporting RBAC, the language presented in [69] for security policies based on permissions and obligations using deontic logic, and the LasSCO [70] language for graphically specifying access-control policies on objects using domains. Also worth mentioning is the Policy Definition Language (different from the PDL language described in section V-C) defined in 1996 by Koch et al. [71] and which provides a means to specify management (obligation) and security (authorization) policies supporting subjects and targets as domain expressions, events specified using a simple Event Definition Language (EDL), constraints, and actions. The syntax of Koch PDL is close to that of Ponder, although Ponder goes farther in terms of formalizing policy specification and implementation. Koch presents a framework for policy definition and refinement that features three levels of abstraction: requirements, goals, and operational policies. The language is defined using BNF and provides ancestor and descendant links in the definition of a policy in order to link the policy refinement hierarchy. The framework which is

implemented in [72] features also the use of monitoring agents and a tool for the graphical representation of chained policies.

5.5 IETF Policy Core Information Model (Feb 2001)

The IETF policy working group [42] defined a set of information models that capture all aspects of a managed environment. At the top of the hierarchy is the Core Information Model (CIM) [73], which is composed of a small set of classes and associations that establish a conceptual framework for the schema of the rest of the managed environment. Second, comes the Policy Core Information model (PCIM [74], later extended to PCIMe [75]) which defines a set of classes and relationships that provide an extensible means for defining policy control of managed objects.

PCIM enables administrators to represent policy in a vendor-independent and device independent way. Thus, service-level abstractions can be supported at a higher level, and be translated at a lower level to device-specific configuration parameters, across an aggregate of heterogeneous managed entities.

In addition to the ability to structure policy rules into groups, PCIM introduces the notion of *policy Role* and *component Role*. A role is a type of property that is used to select one or more policies for a set of entities and/or components from among a much larger set of available policies. Components are assigned specific roles and this helps the management system or the system administrator in selecting the appropriate policies that apply to these roles. If ever it is desired to assign a different functionality to that same element, a simple role exchange is done and the appropriate set of policies is applied accordingly.

Listing 5 Level of abstraction of a QPIM policy

High-level policy:

In the human resources department, applications should receive better quality of service than simple web applications unless it is an executive's web application.

QPIM policy (a possible formulation):

```
If packet IP@(source ∨ destination) ∈ human resources department then
  if packet source/destination IP@ ∈ execSetthen
    packet.priority ← High
  else if packet.protocol = = HTTP then
    packet.priority ← Low
  else
    packet.priority ← Default
  end If
end if
```

The IETF Policy QoS Information Model (QPIM), standardized in [76], builds on PCIM and deals with QoS enforcement for differentiated and integrated services

using policy. High-level business needs, available network topology, and the desired network QoS methodology such as DiffServ [77] or IntServ [78] drive the process of QoS policy definition in QPIM. Listing 5 illustrates the abstraction level of policies defined using QPIM.

Note that PCIM(e) and QPIM have a high degree of articulation in policy definition. In fact, nearly every single component of a policy is made up to be a full class. If the policy is say of the form $(C1 \wedge C2 \wedge C3) \vee (C4 \wedge C5) \Rightarrow A1, A2, A3, A4$ then this needs to be modeled in at least ten classes! Five for the conditions, four for actions, and one for the policy itself let alone objects resulting from the instantiation of the different relationship classes.

This high-level of verbosity in the CIM model makes the writing of policies less intuitive. In order to simplify the usage of the CIM policy models, the CIM Simplified Policy Language (CIM-SPL) was created by Jorge Lobo within the DMTF Policy Working Group in january 2007 [79] and is a proposed standard. The objective of CIM-SPL is to provide a means for specifying *if-condition-then-action* style policy rules to manage computing resources using constructs defined by CIM. The design of CIM-SPL is inspired by existing policy languages and models including policy definition language (PDL) from Bell Laboratories [56], the Ponder policy language from Imperial College [19], and the Autonomic Computing Policy Language (ACPL) [80] from IBM Research. One of the main design points of CIM-SPL is to provide a policy language compatible with the CIM Policy Model and the CIM Schema. Ongoing efforts are considering the implementation of CIM-SPL over C++ and Java. Listing 6 shows a CIM-SPL specification of a policy that is invoked when a file system is 85 percent full. The policy expands the storage pool by 25 percent.

Listing 6 A CIM-SPL policy

```
  Import CIM_X_XX_XXXX :: CIM_LocalFileSystem;
  Strategy Execute_All_Applicable;
Policy {
  Declaration {
    computer_system = collect(Self, CIM_HostedFileSystem, PartComponent,
      GroupComponent, null, true)[0];
    storage_config_service = collect(computer_system, CIM_HostedService,
      Antecedent, Dependent, CIM_StorageConfigurationService, true)[0];
    logical_disk = collect(Self, CIM_ResidesOnExtent, Dependent, Antecedent,
      null, true)[0];
    storage_pool = collect(logical_disk, CIM_AllocatedFromStoragePool,
      Dependent, Antecedent, null, true)[0];
    fs_goal = collect(Self, CIM_ElementSettingData, ManagedElement, SettingData,
CIM_FileSystemSetting true)[0];   }
```

Listing 6 continued

```
Condition { (AvailableSpace / FileSystemSize) < 0.25 }
Decision {
  storage_conf_service.CreateOrModifyElementFromStoragePool(''LogicalDisk'',
  /* ElementType Volume */
  8, job, fs_goal,
  1.25 * FileSystemSize, storage_pool, logical_disk)
  | DoSomethingOnFailure() }
} : 1 /* policy priority */;
```

5.6 OASIS eXtensible Access Control Markup Language (XACML) (2003)

In April 2001, the OASIS XML interoperability consortium started to work on XACML for the purpose of standardizing security access control using XML. Two years later, the first XACML standard was issued [81] and included a declarative XML-based access control policy language and a request/response language. Policies in XACML are specified using subjects, Targets, Resources, and Actions. The *Target* is the constraint to be met by the *Subject* and *Resource* for the policy *Action* to be applicable. XACML policies can be nested into policy sets. An XACML Policy represents a single access control policy and is expressed through a set of rules.

The request/response language helps to form a query to ask whether or not a given action should be allowed, and interpret the result. XACML enables the use of arbitrary attributes in policies, role-based access control [20], security labels, time/date-based policies, indexable policies, 'deny' policies, and dynamic policies. As of 2007, version 3.0 is in preparation and will add generic attribute categories for the evaluation context and policy delegation profile [82].

Similar to the IETF policy framework [39], XACML assumes an architecture featuring a PDP, PEPs, and a policy repository. The request to access to a resource needs to be made to the PEP that protects that resource (like a filesystem or a web server). The PEP then formulates and sends the request to the PDP, which will trigger the appropriate policy to get the answer about whether access should be granted. The response includes one of the values: Permit, Deny, Indeterminate or Not Applicable.

5.7 Policy Management for Autonomic Computing (PMAC) (2005)

PMAC [83] is a generic middleware platform developed in 2005 by IBM to provide software components that can be embedded in software applications to take input from a policy-based management system.

The PMAC platform supports the system model adopted by the IBM Autonomic Computing (AC) architecture, which defines a framework for self-managing information systems [84]. The AC architecture presents two key abstractions: (1) An autonomic manager (AM), which Monitors computing resources, Analyzes the status of the resources, Plans action for the resources, and Executes the planned actions (known as the MAPE functions); and (2) A managed resource (MR), which is a computing system controlled and managed by the AM.

Listing 7 Boolean expression in ACPL Vs. SPL [83]

Boolean expression in ACPL

```
<And>
  <Not> <Equal> <PropertySensor propertyName = ''NumberOfPorts''/>
  IntConstant>
      <Value>16</Value>
    </IntConstant> </Equal> </Not>
  <Equal>
    <PropertySensor propertyName = ''VendorId''/> <IntConstant>
      <Value>5</Value>
    </IntConstant> </PropertySensor>
   </Equal>
  <Equal>
    <PropertySensor propertyName = ''Type''/> <StringConstant>
      <Value>Core Switch</Value>
    </StringConstant> </PropertySensor>
   </Equal>
</And>
```

The same expression in SPL

```
(Sensor(NumberOfPorts) ! = 16) and (Sensor(VendorId) = 5) and
(Sensor(Type) = ''Core Switch'')
```

PMAC is implemented in JAVA and provides two different policy languages: ACPL (Autonomic Computing Policy Language) [80] which is based on XML, hence verbose, and SPL which is concise and human friendly.

The AM in PMAC exposes a set of Java application programming interfaces (APIs) that are useful when the AM and MR are running in the same Java virtual machine. Alternatively, the AM can be run inside an application server and be offered as a stateless session Enterprise Java Bean (EJB) or as a Web service to remotely located managed resources. Thus, a managed resource can request policy guidance to the AM through RMI (in the EJB case) or SOAP (in the Web service case) protocols.

At the core of ACPL is a rich expression language, the Autonomic Computing Expression Language (ACEL), that facilitates writing policy rules [85]. ACEL has been designed so that it can express most common policy conditions while closely

following standard XML conventions. It is a strongly typed language that can be parsed and type checked almost entirely by XML parsers, thereby making it attractive to applications that can consume XML format (e.g. Web services policies).

A condition in a PMAC policy can be any ACEL expression of type Boolean with variables corresponding to sensor names. The value of the variable is the same for all occurrences of the variable in the AM policies. Variables in ACEL are represented by an XML element type called PropertySensor with an attribute Property- Name identifying the name of the variable.

6 Policy Refinement

Although there has been considerable effort in industry and academia, since the 1990's, regarding the specification and implementation of policies and recently Service Level Agreements (SLAs), little focus has been given to the refinement of high-level goals into low-level policies. The investigation of policy refinement and analysis has gained interest only recently.

Policy refinement is meant to derive low-level management policies from high-level requirements. Policy analysis always goes hand-in-hand with policy refinement and relates to analyzing a set of policies for consistency and correctness vis-a-vis of the high-level policy.

Listing 8 summarizes an example adapted from [86] and shows a simple manual refinement case.

Listing 8 A Manual policy derivation example

1. **Requirement**:

   provide an IP telephone to a user who logs in from a location outside their home office

2. The requirement looks like:

   provide *service* ***to*** *user* ***on*** *event* ***where*** *condition*

where (service = IP telephone) ∧ (event = user logs in) ∧ (condition = from a location outside their home office)

3. The policy to consider is then an obligation policy (section V-D) of the form

   on event **do** action **where** condition

4. Subsequent steps will then require the translation of the expressions in a, b, and c into their system-level equivalents.

6.1 Goal-driven Network Management (1992)

In 1992 Boutaba [32, 87] introduced an architecture for the integrated management of networked systems. The management architecture is based on (1) the use of domains as organizational units for the grouping of resources and specifying management responsibility and authority boundaries; and (2) a clear

distinction between management policies/goals and the resources/activities being managed.

At each management level (figure 2(a)), the manager receives management goals from higher level mangers. A management goal is a statement about what is to be achieved. The manager achieves the management goals by taking actions which are single management operations. Actions available to the manager include issuing instructions to the resources being managed or to other managers involved in managing the resource, making reports (to the issuers of the management goals), and performing other internal operations.

Because deriving management plans from a set of management goals can be a complicated task, management policies are introduced as an intermediate step between management goals and management plans. Management policies are general statements about how the manager will achieve the management goals, and are used to ease decision making by restricting the set of solutions to a problem from the range given by the management goals to a much smaller handled size.

Management plans derived from management policies may be executed immediately to achieve some of the management goals (e.g., "turn on accounting"), or may be stored for execution in response to some situation as part of a persistent management goal (e.g., "if response time exceeds 50 ms, reroute").

The identified management phases are respectively processed and supervised by the manager' active entities illustrated in figure 2(b). These are : the Policy Maker, the Planner, and the Executive.

A manager can operate either proactively, stimulated by arrival of goals from higher level managers, or reactively on the detection of events or arrival of notifications from managed resources and/or lower level managers.

6.2 Table Lookup (2000)

This technique [88] was proposed by Verma in 2000 for use with network policies of the form (SrcIP, DstIP, SrcPort, DstPort, Protocol, ServiceLevel (kbps)), such as

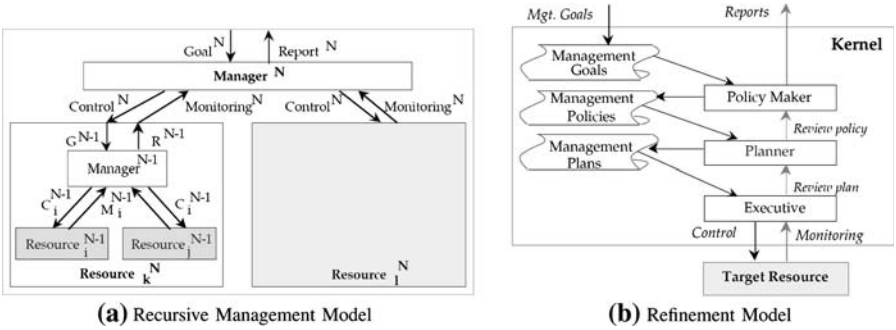

Fig. 2 Goal-driven management framework [64]

(1.4.0.0, 1.3.0.0, ANY, ANY, TCP, 20kbps) (guarantees 20kbps for all TCP traffic between 1.4.0.0 and 1.3.0.0). A policy represents a goal to achieve by the network.

The policy refinement module holds a table of policies to achieve. In order to perform the refinement, each policy in the table needs to be first mapped into an N-dimensional hyperspace object, where N is the number of configuration parameters in the policy. Each policy is considered to be a region in the hyperspace, which may be connected or disconnected. Each hyper-cube points to a set of goals or actions. The system needs then to match the hypercube corresponding to the policy being queried against all the hyper-cubes representing all the policies in the table. In the case of overlaps, the refinement process splits the policy into sub-policies (smaller hyper-cubes) with refined goals. The procedure continues until there every overlapping is removed.

6.3 Case-based Reasoning (2004)

The case-based reasoning (CBR) approach to policy management has been introduced in [89]. In this approach, the system learns experimentally from the operational behavior it has seen in the past. The system maintains a database of past cases, where each case is a combination of the system configuration parameters and the business objectives that are achieved by the specific combination of the system configuration parameters. When new configuration parameters needed for a new business objective, the case database is consulted to find the closest matching case, or an interpolation is performed between the configuration parameters of a set of cases to determine the appropriate configuration parameters that will result in the desired business objective.

The case-based approach to policy refinement has a number of weaknesses. Namely, the difficulty to populate the case database at bootstrap time. False acceptance is also possible due to the generalizations made based on wrongly constructed sets of cases.

The main advantage of a CBR system is that it becomes increasingly effective as its case database grows in size. For this reason, Beigi et al. [89] argue that a policy-based management architecture that uses CBR needs to employ a runtime policy transformation mechanism.

6.4 Goal Elaboration and the KAOS Methodology

The goal elaboration approach to policy refinement has received more interest recently. The approach is based on the KAOS goal refinement methodology proposed by Darimont et al. [90, 91].

KAOS is a formal approach to goal refinement and operationalization aimed at providing constructive formal support while hiding the underlying mathematics. The principle is to reuse generic refinement patterns from a library structured according to strengthening/weakening relationships among patterns. Once the

patterns are proved correct and complete, they can be used for guiding the refinement process or for pointing out missing elements in a refinement.

Goals can be of the form *Achieve* $(P \Rightarrow \diamond Q)$, *Maintain* $(P \Rightarrow \Box Q)$, *Cease* $(P \Rightarrow \diamond \neg Q)$, or *Avoid* $(P \Rightarrow \Box \neg Q)$. *Achieve* and *Cease* goals obey to system behaviors that require some target property to be eventually satisfied or denied respectively. *Maintain* and *Avoid* goals, on the other hand, restrict behaviors in that they require some target property to be permanently satisfied or denied respectively.

Goal elaboration in KAOS is based on the refinement of AND/OR structures by defining goals and their refinement/conflict links until implementable constraints are reached.

In order to complete goal operationalization, R. darimont [91] suggests the use of stimulus-response patterns. A *stimulus* is an event perceived by some agent which requires some action to be performed by the agent. A *response* is an agent reaction to some stimulus. This obviously resembles to an ECA policy although it was not specifically used in the scope of policy-based management.

The KAOS approach, however, is limited in the assignment decision step. It provides little support for formal reasoning about alternative assignments. The next three sections elaborate on this issue through the use of abduction, model checking techniques, and translation primitives respectively.

6.4.1 Goal Elaboration Using Event-calculus and Abductive Reasoning (2004)

Bandara et al. presented in [92, 93] an approach to policy refinement by which a formal representation of a system, based on the Event Calculus, is used in conjunction with *abductive reasoning* techniques to derive the sequence of operations that will allow a given system to achieve a desired goal. The refinement relies on using the KAOS methodology first for the process of refining the abstract goals into lower-level ones.

The relationship between the system description (SD) and the goals (G) is called a *Strategy* (S). It describes the mechanism by which the system represented by SD achieves the goals denoted by G, i.e. $(\mathrm{SD}, \mathrm{S}) \vdash \mathrm{G}$. Abductive reasoning allows to derive the *facts* that must be true for the desired system state (G) to be achieved given the SD. This consists in deriving a path in the statechart from some initial state to the desired one. Whether a strategy should be encoded as policy, or as system functionality, will depend on the particular application domain.

Users first provide the high-level policy they desire in the form "**on** event **if** condition **then** *achieve* goal". The KAOS approach is applied to elaborate the high-level policy, making use of both application-independent and application-specific refinement patterns. At each stage of elaboration, the system description and the goals are used to attempt to *abduce* a strategy for achieving the goal. If no strategy can be derived, then the preferred course of action is to further elaborate the goals. However, if the existing low-level goals are already expressed at the lowest level of abstraction in the system, it is not possible to elaborate the goals further. In this situation the system description must be augmented with more detail.

6.4.2 Goal Elaboration Using Model Checking (2005)

Instead of using an abductive reasoning engine, Rubio-loyola et al. [94] investigated the usage of *model checking* in order to derive operational policies from low-level goals. The refinement framework relies also on the KAOS methodology, the use of temporal logic, and the modelling of system behavior in terms of event-based labeled transitions (state charts).

Model checking is a technique of system verification that relies on the counterexample principle: In order to prove that property P can hold ($\Leftrightarrow$ P is a fluent), it is sufficient to find a counterexample to the property that states that P can never happen, i.e. $\Box!P$ (Always not P).

The counterexample produced by a model checker not only proves that P can hold but also provides a way to actually achieve it, which in turn is nothing but the low-level policy that needs to be encoded in order to achieve the original goal.

6.4.3 Goal Elaboration Using Translation Primitives (2006)

In [95], Rubio-loyola et al. provide a functional approach which extends their KAOS and model-checking-based refinement framework with the use of *design patterns for finite-state verification* [96] and *translation patterns* [97].

The key idea is to constrain the system to a particular behavior, from among the set of potential behaviors, based on a set of systematically identified refinement patterns. For example, if an achieve goal $G1 : (P \Rightarrow \Diamond Q)$ is refined using the milestone pattern to $(G11 : P \Rightarrow \Diamond R \wedge G12 : R \Rightarrow \Diamond Q)$, the subsequent decision is to constrain the system behavior to the new temporal relationship [95] "goal G11 *must be fulfilled before* goal G12".

While the refinement patterns used to elaborate goals describe the requirements for a system (e.g. both G11 and G12 must be fulfilled so that G1 is fulfilled), the patterns described in this approach deal with the translation of particular aspects of such requirements (e.g. G11 is achieved *before/after* G12) into formal specifications suitable for finite state verification tools, such as model checking.

Each pattern has an equivalent representation in linear temporal logic. For example, to specify that G12 (antecedent) must exist after G11 (consequence), which is a combination of an *existence* pattern with an *after* scope, the corresponding temporal formulae would be: $(\Box\neg G11 \mid \Diamond(G11 \wedge \Diamond G12))$.

Once the KAOS goal graph is elaborated and the temporal relationships between sub-goals identified, the refinement engine makes use of *translation primitives* in order to abstract policies from system trace executions. These primitives take advantage of the fact that the system trace executions indicate the pre- and post-conditions, and the actions taken by the involved managed objects.

Translation primitives specialize the *translation patterns* approach [97], which abstracts policy fields from system process specification, to generate policies automatically from system trace executions.

7 Policy Consistency Analysis

Once a policy set is generated after a refinement process, whether be it manual or automated, it is not sufficient to check the syntax of new policies before they are deployed within the system. The deployment of conflicting policies in a system causes abnormal behaviors that are generally difficult to diagnose and measure. Resolution of conflicts among policy rules has been long studied in many areas including artificial intelligence and databases [98–101].

7.1 Modality Conflicts (1999)

Modality conflicts have been studied by Lupu [102]. They are a special type of application-independent policy inconsistencies that can be syntactically detected. Using Ponder terminology [19], a modality conflict occurs when two policies are specified using the same subjects, targets and actions but are of opposite modality. There are three types of such conflicts: (i) *Obligation conflicts* (O+/O −) occur when the subject is obliged to do and obliged not to do an action, (ii) *Unauthorized Obligation conflicts* (O+/A −) occur when the subject is obliged to carry out an action it is not authorized to do, and (iii) *Authorization conflicts* (A+/A −) when the subject is both authorized and unauthorized to carry out an action.

No direct action can be taken at the detection of a modality conflict except to raise an exception to the system administrator to point towards a potential design error.

7.2 Formal Consistency Checking using Logic Programming (2000)

Chomicki and Lobo present in [58] a formal logic-based framework for detecting and resolving action conflicts in ECA policies that do not involve negated events. First, PDL is augmented with *action constraints* describing under which circumstances a set of actions cannot be executed simultaneously. *Monitors* are then defined as filters over policy actions. A monitor cancels some actions in order to meet the constraints.

Given a policy and a set of constraints on the simultaneous execution of its actions, the framework produces a monitor for the policy. Monitors work by filtering (canceling) policy actions and/or events. The aim is to make monitors behave as close as possible to the policy whose output they filter and that the effect of conflict resolution should be maximally transparent to the user. The framework uses stateless policy rules and imposes simultaneity restrictions on events and actions. Future work is considered in order to address those issues and propose more realistic conflict resolution monitors.

As an example, consider the first policy in listing 3 and the constraint ''***never*** *stop* ∧ *mailProduct*''. Listing 9 gives an event-cancellation monitor that can be used in order to enforce it.

Listing 9 An event-cancellation monitor [103]

//The conflict rule

$block(stop) \vee block(mailProduct) \leftarrow exec(stop) \wedge exec(mailProduct)$

//The blocking rules:

$ignore(defectiveProduct) \leftarrow occ(defectiveProduct) \wedge block(stop)$

$ignore(orderReceived) \leftarrow occ(orderReceived) \wedge block(mailProduct)$

$ignore(orderReceived) \leftarrow occ(orderReceived) \wedge block(chargeCreditCard)$

//The accepting rules:

$accept(stop) \leftarrow occ(defectiveProduct) \wedge \neg ignore(defectiveProduct)$

$accept(mailProduct) \leftarrow occ(orderReceived) \wedge \neg ignore(orderReceived)$

$accept(chargeCreditCard) \leftarrow occ(orderReceived) \wedge \neg ignore(orderReceived)$

7.3 Anomaly Detection and Resolution in Firewall Policies (2003)

In [104] Al-Shaer and Hamed present the "firewall policy advisor" tool for the management of firewall policies which offers facilities for conflict detection and resolution. A policy is a set of simple rules of the form "*<order><protocol><s-ip><s-port><d-ip><d-port><action>*", where 's' stands for source and 'd' for destination. The action can be either an *accept* or a *deny*. Source and destination addresses support ranges and the * wild card. A simple priority mechanism is used where rules of a policy are listed sequentially with the first rules having precedence over the last.

The authors identify five types of intra-firewall policy anomalies: *shadowing* (rule never gets activated due to another policy with higher precedence shadowing it), *correlation* (two correlated rules with conflicting filtering actions), *generalization* (rule with lower precedence covering the range of packets of rule with higher precedence), *redundancy*, and *irrelevance*. Algorithms for intra-firewall anomaly discovery and resolution are presented in [104] and for inter-firewall anomalies in [105]. An intra-firewall policy anomaly defines the existence of two or more filtering rules that may match the same packet or the existence of a rule that can never match any packet on the network paths that cross the firewall. An inter-firewall anomaly may exist if any two firewalls on a network path take different filtering actions on the same traffic. Inter-firewall anomalies are classified into *shadowing, spuriousness* (upstream firewall permits traffic that is denied by the downstream firewall), *redundancy*, and *correlation* (correlated rules in upstream and downstream firewalls) anomalies. When an anomaly is discovered the user is informed in order to take the appropriate action among a set of alternatives presented by the firewall policy advisor.

7.4 Formal Consistency Checking Using Abductive Reasoning (2004)

Bandara presents in [86] a tool for policy analysis. The tool supports querying a set of policies for validation and review. *Validation queries* are supported in order to

determine the feasibility of a policy.. *Review queries* are used in order to help the administrator analyze the managed system specification and extract specific types of information.

In addition, Bandara suggests the use of abducting reasoning and the tool developed for event-calculus-based goal elaboration [92, 93] in order to query potential conflicts between policies. The example given is that of querying the obligation policy "Martin should always connect to the VPN server between 9 am and 5 pm" and the refrain policy "Martin should refrain from connecting to the VPN server when accessing the Internet with his PDA" for potential conflict (called a "conflict goal"). The abductive reasoning process manages to detect a case in which a conflict arises (Martin connects to the Internet with his PDA at 9 am). However, this technique remains limited in its applicability because of the inherent difficulty of automating the refinement of a conflict goal.

7.5 Attribute and Priority-based Policy Consistency Checking Algorithms (2005)

Agrawal el al. present in [106] a set of generic static policy consistency checking algorithms.

The first algorithm deals with the determination of whether a set of single attribute constraints do not reduce to empty the domain space of that attribute. The algorithm works for both totally ordered (integer, string, real) discrete or continuous domain types as well as for non ordered finite domain types.

The second algorithm deals with linear constraints over a set of variables and seeks to find a feasible non empty region for them. Constraints can be of the form $[\text{swap} \geq 2\,\text{RAM},\ \text{boot} = 1024,\ \text{boot} + \text{swap} < \frac{1}{4}\,\text{HD},\ \text{RAM} > 0]$.

The third algorithm deals with the solving of the satisfiability of compound boolean expressions. For example, if we want to check if two policies can be triggered at the same time where the condition on the first policy is $((X<10) \vee (X \geq 10 \wedge X + Y < 10))$ and that on the second policy is $(X > 12 \wedge X > 2Y)$; the algorithm will check if the conjunction of the two formulas is satisfiable. The determination of all solutions of a compound boolean expression is however difficult in the general case and the authors provide only a simplified solution based on a Prolog-like backtracking technique.

The fourth algorithm deals with the priority assignment problem and provides a solution to the precedence-based priority assignment. The algorithm has efficiency criteria that allow it to handle the assignment of priority values to a large number of policies, with low overhead properties in policy insertions and deletion.

7.6 Policy Optimization (2007)

Complementary to policy analysis, policy optimization has been stressed in [107] to emphasize the idea that a policy-based solution should provide, in addition to correct behavior, an adequate performance in order to justify its adoption in real-scale management systems. The specification of a set of a set of policies that work is

not sufficient. The issue of whether another set of policies might produce a better performance merits consideration. Furthermore, even for the same policy set, the runtime scheduling of policies may also impact the overall performance produced.

Analytical models, although useful, cannot be used at all times, mainly because of the complexity of real case scenarios. In this regard, simulation is a useful tool for testing the consistency and the performance of a policy solution.

It was shown in [107] that it is not always possible to maximize the service provider's business profit by considering a policy-based solution wherein all aspects of policy specification and behavior are fixed beforehand during a preliminary off-line refinement process. The investigation of runtime policy dynamics, through simulations in [107] allowed to significantly enhance policy performance and the overall business profit. The simulations were conducted using the $\mathcal{PS}$ policy simulator [108] designed to support a business-driven management that has policy support at its core. $\mathcal{PS}$ is a discrete simulation environment based on the process interaction world view and models policies according to the Event-Condition-Action paradigm. For the study conducted in [107], $\mathcal{PS}$ has also been useful in detecting some policy inconsistencies which were not easy to detect during preliminary static analysis.

8 Policy for Business-driven Management

Since the early 1990's, there has been an acceptance that policy can be specified and manipulated at different levels of abstraction [32, 109, 110], now known as the policy continuum [111]. The highest level being the *Business View* which defines policy in business terms only. This is because users do not need nor want to understand low-level system terminology, but still must be involved in the definition of policy. Policies of this type help translate SLAs as well as business procedures into a form that can be used to configure devices and control their deployment.

8.1 IBM Autonomic Computing Initiative (2001)

In 2001, IBM presented the theme and importance of autonomic computing as the solution to the increasing challenge of complexity that is facing the information technology industry [112]. As it became no unordinary that some computing systems weigh in millions of lines of code, and with the increasing need of the interconnection and integration of heterogeneous IT systems, the initiative is intended to enable large-scale computing systems that are self-managed in accordance with high-level guidance from humans.

Central to the IBM architecture is the Autonomic Manager (AM) which implements an intelligent control loop. An autonomic manager is responsible for managing its own behavior in accordance with policies, and for interacting with other autonomic managers to provide or consume computational services. For that to be achieved, an autonomic system needs to possess enough abilities of

self-management, self-configuration, self-optimization, and self-protection. These are now commonly known as the self-* properties.

As shown in Figure 3 [112], the architecture dissects the AM loop into four parts that share knowledge: (1) The *monitor* function provides the mechanisms that collect, aggregate, filter and report details (such as metrics and topologies) collected from a managed resource. (2) The *analyze* function provides the mechanisms that correlate and model complex situations (for example, time-series forecasting and queuing models). These mechanisms allow the autonomic manager to learn about the IT environment and help predict future situations. (3) The *plan* function provides the mechanisms that construct the actions needed to achieve goals and objectives. The planning mechanism uses policy information to guide its work. (4) The *execute* function provides the mechanisms that control the execution of a plan with considerations for dynamic updates.

The structuring and internals of AMs are similar to the recursive policy framework of figure 2(a) as well as the internal structure of the policy managers (figure 2(b)) presented in [32]. In the latter model a manager receives monitoring status events input from the resources/sub-managers it is responsible of. Depending on the nature of the event, the manager may trigger predefined control actions, or review its plans for generating more appropriate control, or else start an internal policy adjustment that still falls within the goals set to it by its superior manager. The sensor/effector interfaces of an AM are hence comparable to the control/monitor interfaces of the manager (figure 2(b)). Using sensor and effector interfaces for the distributed infrastructure components enables these components to be composed together in a manner that is transparent to the managed resources [112].

Of central importance to autonomic system behavior is the ability for high-level, broadly-scoped directives to be translated into specific actions to be taken by elements. This is achieved by the use of policies. For autonomic computing, the focus is specifically on policy-based self-management. At the lowest level of specification are *action policies*, which are typically of the form *if condition then action*, e.g. *if* (ResponseTime $>$ 2 sec) *then* (Increase CPU share by 5%). At the next level are goal policies, which describe the conditions to be attained without

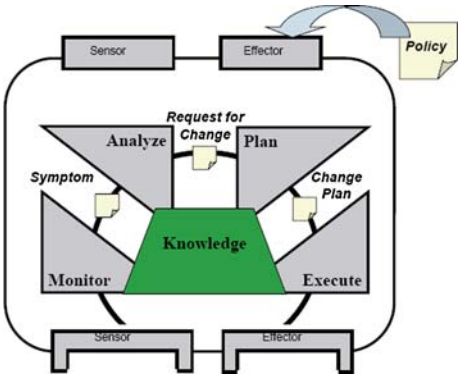

Fig. 3 Structure of an IBM Autonomic Manager [113]

specifying how to attain them, e.g."Response time must not exceed 2 sec". At the highest level are utility function policies, which specify the relative desirability of alternative states. Utility functions are even more powerful than goal policies because they automatically determine the most valuable goal in any given situation [114].

8.2 The Business-driven Management Framework (2004)

The objective of the business-driven management framework (BDMF), introduced in [115], is to drive the management of system resources and services from the business perspective rather than from the low-level technician's point of view. As shown in figure 4, BDMF is divided into two main layers. The high-level business layer hosts the long term control of the system based on the business objectives of the service provider. Beneath it is a resource control layer that hosts the real time logic for the reactive short term control of the resource infrastructure. The low-level *policy control loop* concerns the execution of policies that were predefined through a refinement process, hence has a faster reactivity to system events. The *policy business loop* provides the PDP with high-level knowledge of the system business and helps in elaborating a maximization of business profit based on runtime context.

Business-driven management is still at a preliminary stage. Few other attempts to enable policies at the business level that are worth mentioning are the DEN-ng policy information model inclusion in the TMF NGOSS SID [116], and the policy-driven business performance management framework presented in [117].

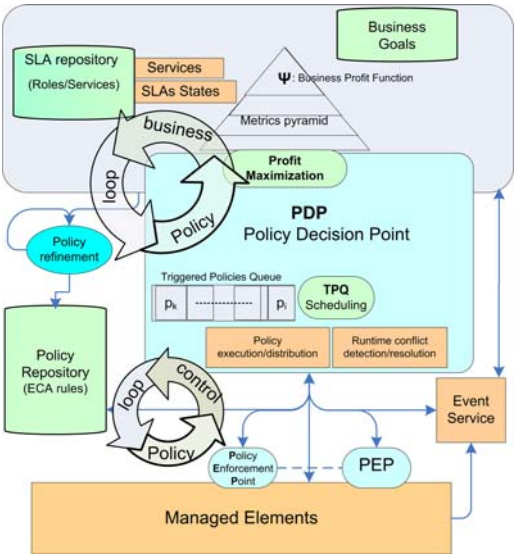

Fig. 4 The Business-Driven Management Framework [108]

9 Conclusion

This paper presented a chronological view of the evolution of policy-based management as a management paradigm for networks and distributed systems. Earliest works, in the late 1960's and 1970's, focused on security policy. Works in the mid to late 1980's, most notably those of Estrin at MIT and Sloman's research group at ICL, paved the road for the recognition of policy as a high-level management paradigm. It was not before the late 1990's early 2000's that the policy-based management concept received a significant attention. Several factors contributed to this observation. First, a significantly higher number of research papers were presented at management and security conferences and published in journals at that period. Second, several policy standardization initiatives were launched such as the ones by the IETF and the DMTF. Third, major industries such as IBM, HP, Intel, Sun, Microsoft, Lucent, and Motorola became involved in the development of policy-based networking and management solutions. Fourth, a policy special interest group was formed and the first of a series of policy workshops was organized. From a technical point of view, the late 1990's and early 2000's were marked by a strong focus on policy specification languages and to a lesser extent on policy refinement and policy conflicts detection and resolution. A plethora of policy-based management architectures, specification languages, testbeds, and tools have been proposed for as many environments as differentiated services Internet, enterprise networks, utility computing, data centers, wireless networks, optical networks, and middleware systems. However, there is still no real large-scale deployment of policy-based management as of today. A methodology of how and when to use policies and how to separate them from hard-coded functionality, support tools for policy refinement and analysis, as well as simulation environments for evaluating large-scale policy-based management solutions are necessary research steps towards policy deployment in practice.

References

1. Keller, A., Al-Shaer, E., Hegering, H.-G. (eds.): Integrated Network Management X (IM 2007), moving from bits to business value. Munich, Germany, May 21–25, 2007. IFIP/IEEE
2. Anderson, J.P.: Computer security technology planning study. ESD-TR-73-51, AD-758 206, ESD/AFSC, Deputy for Command and Management Systems, HQ Electronic Systems Division (AFSC), Hanscom AFB, October 1972
3. Samarati, P., De Capitani di Vimercati, S.: Access control: Policies, models, and mechanisms. In: FOSAD '00: Revised versions of lectures given during the IFIP WG 1.7 International School on Foundations of Security Analysis and Design, pp. 137–196. Springer-Verlag, Bertinoro, Sept 2000
4. Lampson, B.W.: Dynamic protection structures. In: Proc. AFIPS Conf.35 (1969 FJCC), pp. 27–38 (1969)
5. Lampson, B.W.: Protection. In: Proceedings of the 5th Annual Princeton Conference on Information Sciences and Systems, pp. 437–443. Princeton University (1971). Reprinted in ACM SIGOPS Operating Systems Review **8**(1), 18–24 (Jan 1974)
6. Dennis, J.B., Van Horn, E.C.: Programming semantics for multiprogrammed computations, p. 46 (1966)
7. Fabry, R.S.: Capability-based addressing. Commun. ACM (1974)
8. Bell, D.E., LaPadula, L.J.: Secure computer systems: Mathematical foundations. Technical report esd-tr-278, MITRE Corporation, Bedford (1973)

9. Biba, K.J.: Integrity considerations for secure computer systems. ESD-TR-76-372, ESD/AFSC, Hanscom AFB, Bedford, April 1977. NTIS ADA039324
10. Clark, D.D., Wilson, D.R.: A comparison of commercial and military computer security policies. In: IEEE Symposium on Security and Privacy, p. 184. Los Alamitos (1987). IEEE Computer Society
11. Brewer, D., Nash, M.: The Chinese Wall security policy. In: Security and Privacy, 1989. Proceedings, 1989 IEEE Symposium on, pp. 206–214. Oakland, May 1989
12. Bell, D.E.: Looking back at the Bell-la Padula model. 21st Annual Computer Security Applications Conference, December 2005
13. Estrin, D.: Inter-organizational networks: stringing wires across administrative boundaries. Comp. Networks ISDN Syst. **9**(4), 281–295 (1985)
14. Estrin, D.: Inter-organization networks: implications of access control: Requirements for interconnection protocol. In: SIGCOMM '86: Proceedings of the ACM SIGCOMM conference on Communications architectures & protocols, pp. 254–264. ACM Press, New York (1986)
15. Ferraiolo, D.F., Kuhn, D.R., Chandramouli, R.: Role-Based Access Control. Artech House, Computer Security Series (2007)
16. Ferraiolo, D., Kuhn, R.: Role-based access controls. In: 15th NIST-NCSC National Computer Security Conference, pp. 554–563 (1992)
17. Sloman, M., Lupu, E.: Security and management policy specification. IEEE Network **16**(2), 10–19 (2002)
18. InterNational Committee for Information Technology Standards (formerly NCITS). Information technology – role based access control. Ansi/incits standard, ANSI/INCITS 359-2004, 03-Feb 2004, 56 pp
19. Damianou, N., Dulay, N., Lupu, E., Sloman, M.: The ponder policy specification language. In: POLICY '01: Proceedings of the International Workshop on Policies for Distributed Systems and Networks, pp. 18–38. Springer-Verlag, London (2001)
20. Sun Microsystems Anne Anderson: XACML profile for role based access control (RBAC), committee draft 01. OASIS Open cs-xacml-rbac-profile-01, OASIS, XACML Technical Committee, 13 February 2004
21. Mockapetris, P.: Domain names-concepts and facilities. RFC 0882, Internet Engineering Task Force, Network Working Group, Dec 1983. Obsoleted by RFC1034
22. Mockapetris, P.: Domain names-concepts and facilities. RFC 1034, Internet Engineering Task Force, Network Working Group, Nov 1987
23. Levine, P.H.: The Apollo domain distributed file system. In: Paker, Y., et al. (eds.) Distributed Operating Systems: Theory and Practice, volume F28 of NATO ASI series, pp. 241–260. Springer Verlag, August 1986
24. Robinson, D.C.: Domains: A Uniform Approach to Distributed System Management. PhD thesis, Dept of Computing, Imperial College of Science and Technology, University of London, March 1988
25. Robinson, D.C., Sloman, M.: Domains: A new approach to distributed system management. In: Proceedings, Workshop on the Future Trends of Distributed Computing Systems in the 1990s, 1988., pp. 154–163. ieeexplore.ieee.org, 14–16 Sep 1988
26. Robinson, D.C., Sloman, M.: Domain-based access control for distributed computing systems. Software Eng. J. **3**(5), 161–170 (1988)
27. Moffett, J.D., Sloman, M.: Management domains. Technical Report DOC 88/6, University of London, Imperial College of Science and Technology, June 1988
28. Sloman, M., Moffett, J.D.: Domain model of autonomy. In: 3rd workshop on ACM SIGOPS European workshop: Autonomy or interdependence in distributed systems? pp. 1–4. Cambridge, Sept 18–21 1988. ACM Press, New York. portal.acm.org
29. Moffett, J.D.: Delegation of Authority Using Domain Based Access Rules. PhD thesis, Department of Computing, Imperial College, University of London, London, July 1990
30. Twidle, K., Sloman, M.: Domain based configuration and name management for distributedsystems. In: Distributed Computing Systems in the 1990s, 1988. Proceedings, Workshop on the Future Trends of, pp. 147–153. 14–16 Sep 1988
31. Twidle, K.: Domain Services for Distributed Systems Management. PhD thesis, Department of Computing, Imperial College, London (1993)
32. Boutaba, R., Benkiran, A.: A framework for distributed systems management. In: Raghavan, S.V., von Bochmann, G., Pujolle G. (eds.) NETWORKS '92: Proceedings of the IFIP TC6 Working

Conference on Computer Networks, Architecture, and Applications, volume C-13 of IFIP Transactions, pp. 287–298. Trivandrum, India, 28–29 Oct 1992
33. Clark, D.D.: Policy routing in internet protocols. IETF Network Working Group, RFC 1102, May 1989
34. Mills, D.L.: Exterior gateway protocol formal specification. IETF Network Working Group, RFC 904, April 1984
35. Lougheed, K., Rekhter, J.: A border gateway protocol (BGP). IETF Network Working Group, RFC 1105, June 1989
36. Kunzinger., C.: ISO/IEC 10747, protocol for the exchange of inter-domain routing information among intermediate systems to support forwarding of ISO 8473 PDUs. IETF Network Working Group, Internet Draft, April 1994, Expired October 1994
37. Steenstrup, M.: An architecture for inter-domain policy routing. IETF Network Working Group, rfc 1478, June 1993
38. Stone, G.N., Lundy, B., Xie, G.G.: Network policy languages: A survey and a new approach. IEEE Network **15**(1), 10–21 (2001)
39. Waters, G., Wheeler, J., Westerinen, A., Rafalow, L., Moore, R.: Policy framework architecture. Internet-draft, IETF, Network Working Group, Feb 1999
40. DMTF. CIM Policy Model. White paper, February 2001
41. Westerinen, A., Schnizlein, J., Strassner, J., Scherling, M., Quinn, B. Herzog, S., Huynh, A., Carlson, M., Perry, J., Waldbusser, S.: Terminology for policy-based management. IETF RFC 3198, November 2001
42. IETF. Policy framework. http://www.ietf.org/html.charters/OLD/policy-charter.html
43. Rajan, R., Verma, D., Kamat, S., Felstaine, E., Herzog, S.: A policy framework for integrated and differentiated services inthe internet. IEEE Network **13**(5), 36–41 (1999)
44. Flegkas, P., Trimintzios, P., Pavlou, G.: A policy-based quality of service management system for IP diffservnetworks. IEEE Network **16**(2), 50–56 (2002)
45. Ahmed, T., Mehaoua, A., Boutaba, R.: Dynamic QoS adaptation using COPS and network monitoring feedback. In: MMNS '02: Proceedings of the 5th IFIP/IEEE International Conference on Management of Multimedia Networks and Services, pp. 250–262. Springer-Verlag, London, October 6–9 2002. Journal version published in Elsevier Computer Communications (ComCom), A measurement-based approach for dynamic QoS adaptation in DiffServ networks **28**(18), 2020–2033 (2005)
46. Strassner, J.: DEN-ng: achieving business-driven network management. In Network Operations and Management Symposium, 2002. NOMS 2002. 2002 IEEE/IFIP, pp. 753–766, April 15–19 2002
47. Durham, D., Boyle, J., Cohen, R., Rajan, R., Herzog, S., Sastry, A.: The COPS (common open policy service) protocol. IETF Network Working Group, rfc 2748, January 2000
48. Yavatkar, R., Pendarakis, D., Guerin, R.: A framework for policy-based admission control. IETF Network Working Group, rfc 2753, January 2000
49. Reichmeyer, F., Seligson, J., Yavatkar, R., Smith, A.: COPS usage for policy provisioning (COPS-PR). IETF Network Working Group, rfc 3084, March 2001
50. Chan, K.H., Sahita, R., Hahn, S., McCloghrie, K.: Differentiated services quality of service policy information base. IETF Network Working Group, rfc 3317, March 2003. First version published in Michael Fine et al. Differentiated services quality of service policy information base. IETF Internet Draft, Expired Sep 2001, March 2, 2001
51. Sahita, R., Hahn, S., Chan, K.H., McCloghrie, K.: Framework policy information base. IETF Network Working Group, RFC 3318, March 2003
52. 3GPP TSG SA WG2. IP multimedia subsystem (IMS); stage 2 (release 8). Technical Report SP-36 v8.1.0, 3GPP, 19 June 2007. Version SP-10 v1.4.0 published in 10 Jan 2001
53. Nossik, M., Welfeld, F., Richardson, M.: PAX PDL: A Non-Procedural Packet Description Language. Technical report, Sept. 30, 1998
54. Brownlee, N.: SRL: A language for describing traffic flows and specifying actions for flow groups. IETF Internet draft, Expird February 2000, Aug 1999
55. Damianou, N.: A Policy Framework for Management of Distributed Systems. PhD thesis, Imperial College of Science, April 2002
56. Lobo, J., Bhatia, R., Naqvi, S.: A policy description language. In: AAAI '99/IAAI '99: Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative Applications of Artificial Intelligence conference, pp. 291–298. American Association for Artificial Intelligence, Menlo Park (1999)

57. Virmani, A., Lobo, J., Kohli, M.: Netmon: network management for the SARAS softswitch. In: Network Operations and Management Symposium, 2000. NOMS 2000. 2000 IEEE/IFIP, pp. 803–816. Honolulu (2000)
58. Chomicki, J., Lobo, J., Naqvi, S.: A logic programming approach to conflict resolution in policy management. In: Cohn, A.G., Giunchiglia, F., Selman, B. (eds.) 7th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR2000), pp. 121–132. Morgan Kaufmann, San Francisco (2000)
59. Damianou, N., Dulay, N., Lupu, E.C., Sloman, M.: Ponder: A language for specifying security and management policies for distributed systems. Imperial College Research Report DoC 2000/1, 2000
60. Moffett, J.D., Sloman, M.S.: The representation of policies as system objects. In: Proceedings of the Conference on Organizational Computer Systems (COCS), ACM SIGOIS Bulletin, 12(2–3), pp. 171–184. Atlanta, Georgia, 5–8 Nov 1991
61. Sloman, M., Moffett, J., Twidle, K.: Domino domains and policies: An introduction to the project results. Technical report, Imperial College of Science, Technology and Medicine, London (1992)
62. Marriott, D.: Policy Service for Distributed Systems. PhD thesis, Imperial College London, October 1997
63. Polyrakis, A., Boutaba, R.: The meta-policy information base. Network IEEE **16**(2), 40–48 (2002)
64. Boutaba, R.: A methodology for structuring management of networked systems. In: Proceedings of the IFIP TC6/WG6. 4 International Conference on Advanced Information Processing Techniques for LAN and MAN Management, pp. 225–242. North-Holland Publishing Co, Amsterdam (1993)
65. Sloman, M.: Policy driven management for distributed systems. JNSM **2**(4), 333–360 (1994)
66. Wies, R.: Policies in network and systems management-Formal definition and architecture. J. Network Syst. Manage. **2**(1), 63–83 (1994)
67. Hasan, M.Z.: An active temporal model for network management databases. In: Proceedings of the Fourth International Symposium on Integrated network management IV, pp. 524–535. Chapman & Hall, Ltd, London (1995)
68. Jajodia, S., Samarati, P., Subrahmanian, V.S.: A logical language for expressing authorisations. In: IEEE Symposium on Security and Privacy (1997)
69. Ortalo, R.: A flexible method for information system security policy specification. In: ESORICS '98: Proceedings of the 5th European Symposium on Research in Computer Security, pp. 67–84. Springer-Verlag, London (1998)
70. James, H., Pandey, R., Levitt, K.: Security policy specification using a graphical approach. Technical Report CSE-98-3, University of California, Davis Department of Computer Science (1998)
71. Koch, T., Krell, C., Kramer, B.: Policy definition language for automated management of distributed systems. In: Systems Management, 1996., Proceedings of IEEE Second International Workshop on, pp. 55–64. Toronto, June 19–21, 1996
72. Dini, P., v. Bochmann, G., Koch, T., Krämer, B.: Agent based management of distributed systems with variable polling frequency policies. In: Proceedings of the fifth IFIP/IEEE international symposium on Integrated network management V: Integrated management in a virtual world, pp. 553–564. Chapman & Hall, Ltd., London (1997)
73. DMTF.: CIM core model, version 2.4. White Paper (2000)
74. Moore, B., Ellesson, E., Strassner, J., Westerinen, A.: Policy core information model – version 1 specification. IETF RFC 3060, February 2001
75. Moore, B. (ed.): Policy core information model (PCIM) extensions, rfc 3460, January 2003
76. Snir, Y., Ramberg, Y., Strassner, J., Cohen, R., Moore, B.: IBM. Policy quality of service (QoS) information model. IETF Internet standards track protocol, November 2003
77. Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., Weiss, W.: An Architecture for Differentiated Services. IETF Differentiated Services group, RFC 2475, Dec 1998
78. Shenker, S., Wroclawski, J.: General characterization parameters for integrated service network elements. IETF Internet Draft 2215, Integrated Services group, Dec 1998
79. Lobo, J.: CIM Simplified Policy Language (CIM-SPL). Specification DSP0231 v1.0.0a, Distributed Management Task Force (DMTF), 10 Jan 2007
80. International Business Machines (IBM). Autonomic computing policy language. White paper, IBM Trivoli, Nov 2005
81. Godik, S., Moses, T.: eXtensible Access Control Markup Language (XACML) version 1.0. OASIS Standard Document identifier: oasis-xacml-1.0.pdf, OASIS, XACML Technical Committee, 18 February 2003

82. Anderson, A.: A Brief Introduction to XACML. Posted to the XACML TC mailing list, 14 March 2003
83. Agrawal, D., Lee, K.-W., Lobo, J.: Policy-based management of networked computing systems. IEEE Commun. Mag. **43**(10), October 2005
84. IBM. Autonomic computing: Creating self-managing computing systems. http://www.researchweb.watson.ibm.com/autonomic/, 2004
85. Agrawal, D., et al.: Autonomic computing expression language. IBM DeveloperWorks Tutorial, Mar 2005
86. Bandara, A.: A Formal Approach to Analysis and Refinement of Policies. PhD thesis, University College London, University of London, July 2005
87. Boutaba R., Znaty S. (1995) An architectural approach for integrated network and systems management. SIGCOMM Comput. Commun. Rev. **25**(5), 13–38
88. Verma, D.C.: Policy-Based Networking: Architecture and Algorithms. Technology series. SAMS, 2000 edition, Novermber 14, 2000
89. Beigi, M.S., Calo, S., Verma, D.: Policy transformation techniques in policy-based systems management. In: Fifth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'04), p. 13. IEEE Computer SocietyLos Alamitos, CA, USA, 2004.
90. Delcourt, B., van Lamsweerde, A., Dardenne, A., Dubisy, F.: The KAOS project: Knowledge acquisition in automated specification of software. In: AAAI Spring Symposium Series, Track: "Design of Composite Systems", pp. 59–62. Stanford University, March 1991
91. Darimont, R., van Lamsweerde, A.: Formal refinement patterns for goal-driven requirements elaboration. In: SIGSOFT '96: Proceedings of the 4th ACM SIGSOFT symposium on Foundations of software engineering, pp. 179–190. ACM Press, New York (1996)
92. Bandara, A.K., Lupu, E.C., Moffett, J., Russo, A.: A goal-based approach to policy refinement. In: Policies for Distributed Systems and Networks, 2004. POLICY 2004. Proceedings. Fifth IEEE International Workshop on, pp. 229–239, June 7–9 2004
93. Bandara, A.K., Lupu, E.C., Russo, A., Dulay, N., Sloman, M., Flegkas, P., Charalambides, M., Pavlou, G.: Policy refinement for DiffServ quality of service management. IEEE eTrans. Network Ser. Manage. (eTNSM), **3**(2), 12, 2nd uarter 2006
94. Rubio-Loyola, J., Serrat, J., Charalambides, M., Flegkas, P., Pavlou, G., Lluch-Lafuente, A.: Using linear temporal model checking for goal-oriented policy refinement frameworks. In: POLICY, pp. 181–190. IEEE Computer Society (2005)
95. Rubio-Loyola, J., Serrat, J., Charalambides, M., Flegkas, P., Pavlou, G.: A functional solution for goal-ooriented policy refinement. In: POLICY '06: Proceedings of the Seventh IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'06), pp. 133–144. IEEE Computer Society, Washington (2006)
96. Dwyer, M., Avrunin, G.S., Corbett, J.C.: Property specification patterns for finite-state verification. In: Ardis, M. (ed.) Proc. 2nd Workshop on Formal Methods in Software Practice (FMSP-98), pp. 7–15. ACM Press, New York (1998)
97. Danciu, V., Kempter, B.: From processes to policies – concepts for large scale policy generation. In: Boutaba, R., Kim, S.-B. (eds.) Managing Next Generation Convergence Networks and Services, pp. 17–30. IFIP/IEEE, IEEE Publishing, apr 2004
98. Brownston, L., Farrell, R., Kant, E., Martin, N.: Programming Expert Systems in OPS5: An Introduction to Rule-based Programming. Addison-Wesley Longman Publishing Co., Inc., Boston (1985)
99. Ioannidis, Y.E., Sellis, T.K.: Supporting inconsistent rules in database systems. J. Intell. Inf. Syst. **1**(3), 243–270 (1992)
100. Büning, Hk., Löwen, U., Schmitgen, S.: Inconsistency of production systems. Data Knowl. Eng. **3**(4), 245–260 (1988)
101. Jagadish, H.V., Mendelzon, A.O., Mumick, I.S.: Managing conflicts between rules (extended abstract). In: PODS '96: Proceedings of the fifteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems, pp. 192–201. ACM Press, New York (1996)
102. Lupu, E.C., Sloman, M.: Conflicts in policy-based distributed systems management. IEEE Trans. Software Eng., Special Issue Inconsistencies Manage. **25**(6), 852–869 (1999)
103. Chomicki, J., Lobo, J., Naqvi, S.: Conflict resolution using logic programming. IEEE Trans. Knowl. Data Eng. **15**(1), 244–249 (2003)
104. Al-Shaer, E.S., Hamed, H.H.: Firewall policy advisor for anomaly discovery and rule editing. In: Integrated Network Management, 2003. IFIP/IEEE Eighth International Symposium on, pp. 17–30, March 24–28, 2003

105. Al-Shaer, E., Hamed, H., Boutaba, R., Hasan, M.: Conflict classification and analysis of distributed firewall policies. IEEE J. Sel. Areas Commun. **23**(10), 2069–2084 (2005)
106. Agrawal, D., Giles, J.,won Lee, K., Lobo, J.: Policy ratification. In: IEEE, editor, IEEE International Workshop on Policies for Distributed Systems and Networks (Policy 2005), IBM Thomas J. Watson Research Center, Yorktown Heights, New York, June 2005
107. Aib, I., Boutaba, R.: Business-Driven optimization of Policy-Based Management Solutions. In: Integrated Network Management, 2007. IM '07. 10th IFIP/IEEE International Symposium on, pp. 254–263. Munich, May 21–25, 2007. IEEE
108. Aib, I., Boutaba, R.: PS: A policy simulator. IEEE Commun. Mag. **45**(4), 130–137 (2007)
109. Maullo M.J., Calo, S.B.: Policy management: An architecture and approach. In: Systems Management, 1993, Proceedings of the IEEE First International Workshop on, pp. 13–26. Los Angeles, 14–16 Apr 1993. IEEE
110. Moffet, J.D., Sloman, M.: Policy hierarchies for distributed systems management. IEEE J. Sel. Areas Commun., Special Issue Network Manage.t **11**, 1404–14 (1993)
111. Strassner, J.: How policy empowers business-driven device management. In: IEEE, editor, IEEE Third International workshop on Policies for Distributed Systems and Networks (POLICY'02), pp. 214–217 (2002)
112. Horn, P.: Autonomic Computing: IBM's Perspective on the State of Information Technology. IBM Corporation, 8 March 2001
113. IBM. An architectural blueprint for autonomic computing. White paper 4, IBM, June 2006
114. White, S.R., Hanson, J.E., Whalley, I., Chess, D.M., Kephart, J.O.: An architectural approach to autonomic computing. In: Autonomic Computing (ICAC), 2004. Proceedings. International Conference on, pp. 2–9, May 2004
115. Aib, I., Salle, M., Bartolini, C., Boulmakoul, A., Boutaba, R., Pujolle, G.: Business-aware policy-based management. In: The First IEEE/IFIP International Workshop on Business-Driven IT Management (BDIM 2006), in conjunction with NOMS 2006, pp. 55–62. IEEE, April 7 2006. Previously published in HP technical report HPL-2004-171, Oct 2004, Bristol
116. TMF John Strassner.: Shared information/data (SID) model, Common Business Entity Definitions – Policy NGOSS Release 4.0. GB922 Addendum – 1-POL TMF Approved Version 1.1, TMF, August 2004
117. Jeng, J.-J., Chang, H., Bhaskaran, K.: Policy-driven business performance management. In: Proc. Of 151h IFIP/IEEE International Workshop on Distributed Systems, Operations, and Management, DSOM 2004, pp. 52–63. Springer LNCS 3278, 2004

Author Biographies

Raouf Boutaba received the MSc. and PhD. degrees in Computer Science from the University Pierre & Marie Curie, Paris, in 1990 and 1994 respectively. He is currently a Professor of Computer Science at the University of Waterloo. His research interests include network, resource and service management in multimedia wired and wireless networks. Dr. Boutaba is the founder and Editor-in-Chief of the IEEE Transactions on Network and Service Management and on theeditorial boards of several other journals. He is currently a distinguished lecturer of the IEEE Communications Society, the chairman of the IEEE Technical Committee on Information Infrastructure and the IFIP Working Group 6.6 on Network and Distributed Systems Management. He has received several best paper awards and other recognitions such as the Premier's research excellence award.

Issam Aib received the MSc. and PhD. degrees in Computer Science from the University of Pierre & Marie Curie, Paris, France, in 2002 and 2007 respectively. He is currently a Postdoctoral fellow at the school of computer science of the University of Waterloo (Canada) where he is conducting research on policy-based and business-driven management of networks and distributed systems since 2005. He is the recipient of the best student-paper award of the tenth IFIP/IEEE International Symposium on Integrated Network Management (IM 2007) for his work on the optimization of policy-based management solutions [107].