Policy 2001

Workshop on Policies for Distributed Systems and Networks

29-32 January 2001

Hosted by HP Laboratories, Bristol, UK

Technical Co-sponsored by IEEE ComSoc

http://www-dse.doc.ic.ac.uk/events/policy-2001/

1. Workshop Program

2. Position Papers

3. Invited Presentations

Policy 2001
Workshop on Policies for Distributed Systems and Networks

29-31 January 2001

Hosted By HP Laboratories, Bristol, UK

Technical Co-sponsored by IEEE ComSoc

http://www-dse.doc.ic.ac.uk/events/policy-2001/

Program

Monday 29 January 2001

8.45 - 9.30: Registration and Coffee

9.30: Welcome

9.40 - 10.30 Keynote

***Chair**: Cheh Goh, HP Labs, Bristol*

Mechanized Policy, Fact or Fancy?
Joe Pato, Principal Scientist, Trusted E-Services Lab - HP Labs, Cambridge MA USA.

10.30 - 11.00 Coffee

11.00 - 13.00 Session 1 Policy Specification and Analysis
***Chair**: Peter Linington, University of Kent, UK*

Author Obliged to Submit Paper before 4 July: Policies in an Enterprise Specification
J. Cole[1] J. Derrick[2,] Z. Milosevic[1] and K. Raymond[1]
[1]DSTC, University of Queensland, Australia, [2]University of Kent, UK

The Ponder Policy Specification Language
N. Damianou, N. Dulay, E. Lupu and M. Sloman
Imperial College, UK

IPSec/VPN Security Policy: Correctness, Conflict Detection and Resolution
Z. Fu[1], F. Wu[1], H. Huang[2], K. Loh[2], F. Gong[2], I. Bladine[3], and C. Xu[3]
[1]North Carolina State University, [2]Nortel Networks, [3]MCNC, USA

Monitors for History-based Policies
J. Chomicki[1] and J. Lobo[2]
[1]SUNY, [2]Bell Laboratories, USA

13.00 - 14.00 Lunch

14:00 - 16.00 Session 2 RBAC and Security Policy
***Chair:** Ravi Sandhu, George Mason University, USA*

A Type/Domain Policy for Internet Transmission, Sharing and Archiving of Medical and Biological Data
R. Viviani
Uniklinik Ulm, Germany

Tower: A Language for Role Based Access Control
M. Hitchens and V. Varadharajan
University of Western Sydney, Australia

Translating Role-based Access Control Policy within Context
J. Bacon, M. Lloyd and K. Moody
University of Cambridge, UK

Model-Based Tool-Assistance for Packet-Filter Design
I. Lueck[1], C. Schaefer[2] and H. Krumm[2]
[1]Materna Information and Communications, [2]University of Dortmund, Germany

16.00 - 16.30 Tea

16.30 - 17.30 Session 3 Formal and Natural Language for Policy
***Chair:** Edgar Sibley, George Mason University, USA*

Role Based Constraints Language
R. Sandhu and G.-J. Ahn
George Mason University, USA

The Incorporation of Control Principles into Access Control Policies
A. Schaad and J. Moffett
University of York, UK

Event Centric Policy Specification for E-Commerce Applications
A. S. Abrahams and J. M. Bacon
University of Cambridge, UK

17.45 - 19.30 Reception at HP Laboratories

Tuesday 30 January 2001

9.00 - 10.30 Invited Talks
Chair: *Francisco Garcia, Agilent Technologies, UK*

Policy and the IETF – Theory and Practice
John Strassner
Past Chairman IETF Policy Working Group, Cisco, USA

Provisioning Your Future through Policy-based Management
Rick Roeling
OpenView PolicyXpert Architect, Hewlett-Packard Company, USA

10.30 - 11.00 Coffee

11.00 - 13.00 Session 4 Network Policy Realization
Chair: *Ed Elleson, LongBoard, USA*

Policy Based SLA Management in Enterprise Networks
D. Verma, M. Beigi and R. Jennings
IBM T.J. Watson Research Center, USA

Integrating Goal Specification in Policy-based Management
M. Bearden, S. Garg and W. Lee
Bell Laboratories, USA

Taxonomy of Policy Combination
Y. Kanada
Hitachi Central Research Laboratories, Japan

Issues in Managing Soft QoS Requirements in Distributed Systems Using a Policy Based Framework
H. Lutfiyya, G. Molenkamp, M. Katchabaw and M. Bauer
University of Western Ontario, Canada

13.00-14.00 Lunch

14.00 - 16.00 Session 5 Perspectives on Policy Architectures
Chair*: John Vicente, Intel, USA*

A Policy Based Management Architecture for Large Scale Active Communications Systems
I. Marshall and P. Mckee
BT Adastral Park, UK

Policy Driven Management of Agent Systems
A. Corradi[1], N. Dulay[2], R. Montanari[1] and C. Stefanelli[2]
[1]Universityof Bologna, Italy, [2]Imperial College, UK, [3]University of Ferrara, Italy

On Policy-based Extensible Hierarchical Network Management in QoS-enabled IP Networks
P. Flegkas, P. Trimintzios, G. Pavlou, I. Andrikopoulos and C. F. Cavalcanti
University of Surrey, UK

Towards Extensible Policy Enforcement Points
R. Boutaba[1] and A. Polyrakis[2]
[1]University of Waterloo, [2]University of Toronto, Canada

16.00 - 16.30 Tea

16.30 - 17.30 Session 6 Policy Applications
Chair*: Ian Marshall, BtexaCT, UK*

Policy-Based Storage Management
M. A. Carlson[1], M. Dutch[2], J. Gelb[3], G. Mueller[4], P. Spasic[5] and L. VanArsdale[6]
[1]Sun Microsystems, [2]Troika Networks, [3]IBM, [4]Storage Tek, [5]HP, [6]BMC Software,USA

Distributed Policy Management and Comprehension with Classified Advertisements
N. Coleman, R. Raman, M. Livny and M. Solomon
University of Wisconsin, USA

Policy-based Management: Towards Internet Service Provisioning
J. Vicente, L. Xie, H. Cartmill and G. Anavi
Intel Corporation, USA

19.30 Conference Dinner at the Roman Bath Pump Rooms, Bath
Bus leaves 18.00 from HP

Wednesday 31 January 2001

9.00 - 10.30 Invited Talks

Chair: *Jonathan Moffett, University of York, USA*

Trust Management and Security Policy
Matt Blaze, AT&T Research Laboratories, USA

On the Negotiation of Access Control Policies
Virgil Gligor, Himanshu Khurana
University of Maryland, USA

10.30 - 11.00 Coffee

11.00 - 13.00 Panel: Future Directions for Policy Research
Chair: *Morris Sloman, Imperial College, UK*

Policy based frameworks are comparatively new, mostly with vendor-specific tools to support policy specification. There is not much commonality in the approaches being followed in policy-based network management, role-based access control, security policies for operating systems, databases or firewalls. This panel will address some of issues we should be focusing on for future research.

- Is it feasible to have a common approach for management, security and enterprise policy specification?
- Is a standard information model such as the CIM Policy model sufficient?
- Do we need standard policy specification languages?
- What should be the starting point for policy specification – enterprise goals or service level agreements?
- What is missing from current approaches?
- What techniques and tools are needed for analysis?
- What is needed for large-scale policy based systems?

Panelists:

Ravi Sandhu, George Mason University, USA
to focus on RBAC and security issues.

John Strassner, Cisco, USA
to focus on information modelling

Ed Elleson, LongBoard, USA
to focus on what is required for network management.

Bret Michel, Naval Postgraduate School, USA
to focus on enterprise policy and trust.

Marek Sergot, Imperial College, UK
to focus on prospects of developing (appropriate) formal-logical languages and tools.

13.00 Close of Conference

Session 3
Formal and Natural Language for Policy

Session Chair:
Edgar Sibley, George Mason University, USA

Role Based Constraints Language

R. Sandhu, G.-J. Ahn, George Mason University, USA

The Incorporation of Control Principles into Access Control Policies

A. Schaad, J. Moffett, University of York, UK

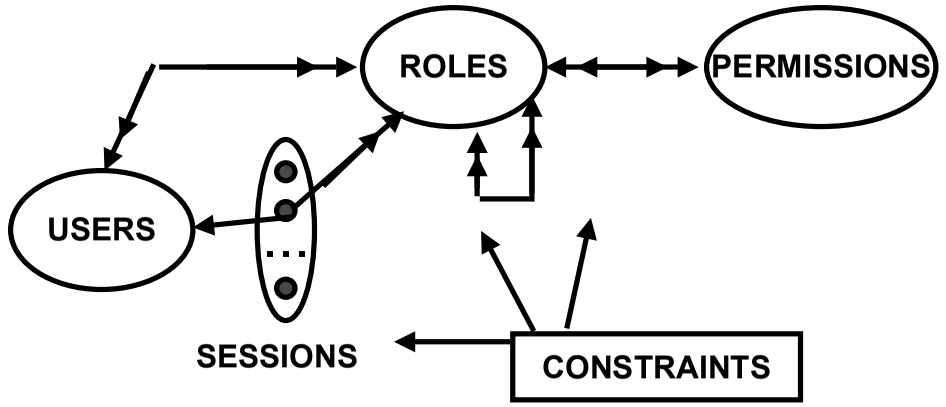
Event Centric Policy Specification for E-Commerce Applications

A. S. Abrahams, J. M. Bacon, University of Cambridge, UK

Role Based Constraints Language *RCL2000*

Ravi Sandhu
Gail-Joon Ahn
George Mason University
www.list.gmu.edu

RBAC96



RCL 2000

- **RCL 2000 (Role-based Constraints Language 2000)**
- **Specification Language**
  - **to formally express constraints in role-based systems**
- **Most components are built upon RBAC96**

PROHIBITION CONSTRAINTS

- **Forbid the RBAC component from doing (or being) something which is not allowed to do (or be)**
  - **Separation of duty constraints**

OBLIGATION CONSTRAINTS

- **Force the RBAC component to do (or be) something**
  - **Every faculty member must be assigned to at least one departmental committee**
  - **LBAC-RBAC, Chinese Wall-RBAC simulation**

EXAMPLES OF CONSTRAINT EXPRESSION

Conflicting roles cannot have common users

- $|roles(OE(U)) \cap OE(CR)| \leq 1$

Conflicting users cannot have common roles

- $roles(OE(OE(CU))) \cap roles(AO(OE(CU))) = \phi$

Users cannot activate two conflicting roles

- $|roles(sessions(OE(U))) \cap OE(CR)| \leq 1$

Users cannot activate two conflicting roles in a single session

- $| roles(OE(sessions(OE(U)))) \cap OE(CR)| \leq 1$

FORMAL SEMANTICS

- **Reduction Algorithm**
  - **to convert a constraint expression to a restricted form of first order predicate logic (RFOPL)**
- **Construction Algorithm**
  - **to construct a constraint expression from RFOPL**

SEPARATION OF DUTY CONSTRAINTS

- **Specification of SOD constraints identified by Simon and Zurko (1997) and formulated by Virgil et al (1998)**
- **Identify new SOD properties**
  - **Role-centric**
  - **User-centric**
  - **Permission-centric**

ROLE-CENTRIC SOD CONSTRAINT EXPRESSION

- **Static SOD**
  - **: Conflicting roles cannot have common users**

    U = $\{u_1,u_2,\ldots u_n\}$, R = $\{r_1,r_2,\ldots r_n\}$,

    CR = $\{cr_1,cr_2\}$: $cr_1 = \{r_1,r_2,r_3\}$, $cr_2 = \{r_a,r_b,r_c\}$
  - **|roles(OE(U)) ∩ OE(CR)| ≤1**

PERMISSION-CENTRIC SOD CONSTRAINT EXPRESSION

- **SSOD-CP**
  - **|permissions(roles(OE(U))) ∩ OE(CP)| ≤1**
- **Variations of SSOD-CP**
  - **SSOD-CP ∧ |permissions(OE(R)) ∩ OE(CP)| ≤1**

USER-CENTRIC SOD CONSTRAINT EXPRESSION

- **SSOD-CU (User-centric)**
  - **SSOD-CR ∧ |user(OE(CR)) ∩ OE(CU)| ≤1**

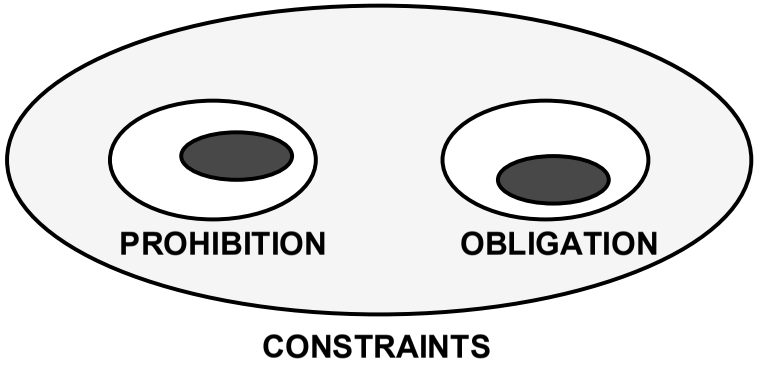
DYNAMIC SOD

- **User-based DSOD**
  - **|roles(sessions(OE(U))) ∩ OE(CR)| ≤1**
- **User-based DSOD with CU**
  - **|roles(sessions(OE(OE(CU)))) ∩ OE(CR)| ≤1**
- **Session-based DSOD**
  - **|roles(OE(sessions(OE(U)))) ∩ OE(CR)| ≤1**
- **Session-based DSOD with CU**
  - **|roles(OE(sessions(OE(OE(CU))))) ∩ OE(CR)| ≤1**

CASE STUDIES

- **Lattice-based access control**
  - **Ravi Sandhu (1993, 1996)**
- **Chinese Wall policy**
  - **Ravi Sandhu (1992)**
- **Discretionary access control**
  - **Sandhu and Munawer (1998)**

CONSTRAINTS CHARACTERIZATION



SIMPLE PROHIBITION CONSTRAINTS

- **Type 1**
  - $|expr| \leq 1$
- **Type 2**
  - $expr = \phi$ **or** $|expr| = 0$
- **Type 3**
  - $|expr1| < |expr2|$

SIMPLE OBLIGATION CONSTRAINTS

- **Type 1**
  - $expr \neq 0$ **or** $|expr| > 0$
- **Type 2**
  - **Set X = Set Y**
- **Type 3**
  - **obligation constraints ⇒ obligation constraints**
- **Type 4**
  - $|expr| = 1$
    - $|expr| = 1 \equiv |expr| \leq 1 \wedge |expr| > 0$

WHO IS THE USER OF RCL2000

- **Security Researcher**
- **Security Policy Designer**
- **Security Architect**

The Incorporation of Control Principles into Access Control Policies

Andreas Schaad[1] **& Jonathan D. Moffett**
Department of Computer Science, University of York
Heslington, York YO10 5DD, UK
{andreas, jdm}@cs.york.ac.uk

1 Introduction

Access control policies should be based upon the goals of an organisation, as expressed in its control principles, but the principles are not normally visible in the access control system (ACS). It would be desirable to represent them explicitly in the ACS so that they can be used in access control policies and rules.

In this paper we discuss common control principles and how they could be represented within an ACS. We have started with the control principle of separation of duties, and produced a prototype simulation tool which shows the effect of administrators' actions on the separation of duties constraints of a RBAC (Role Based Access Control) system.

2 Organisational Control Principles

2.1 Control Principles

In order to achieve and maintain control, organisations set out control principles which are used to guide its decisions, but they have not become explicitly represented in ACSs. This leads to the problem that proposed actions which would breach the principles are not recognised by the ACS, and may therefore be wrongly permitted.

2.2 Common Control Principles

Each organisation uses a different set of control principles as the individual control requirements are very diverse. Some common control principles are described below.

Separation of Duties: By partitioning critical transactions and assigning sub-tasks to different entities we prevent any one person from performing the whole transaction, thus reducing the risk of any error or fraud.

Delegation: Delegation is an important part of any working organisation, since the main task of management is to get work done through the efforts of other people. Delegation of authority can be seen as a specialisation of tasks and responsibilities, through which a superior delegates or transmits pieces of authority downward in the organisational chain along with the obligation to perform specific duties.

Supervision, Review and Audit: Supervision and review control whether delegated tasks are carried out as required. Supervision is a general activity carried out by a person in a superior position. Reviewing is task-specific and does not necessarily need to be performed by a superior position. Auditing in general serves as an activity of checking that a system performs its required function.

3 Security Policies and Control Principles

As shown in figure 1, the ADF makes its decision based on individual access rules, on information about system users, on the system state (e.g. time) and on fixed security policies. Both fixed security policies and mutable access rules are incorporated into the reference monitor. On the other hand control principles are used by human beings outside the access control system to determine fixed policies and access rules. This makes the enforcement of control principles difficult to achieve reliably, because it is carried out on an *ad hoc* basis by human beings who are liable to error. It would be desirable to incorporate them into the reference monitor, so that is becomes possible to detect, within the system, if they are being violated.

[1] Sponsored by the EPSRC under award no 99311141

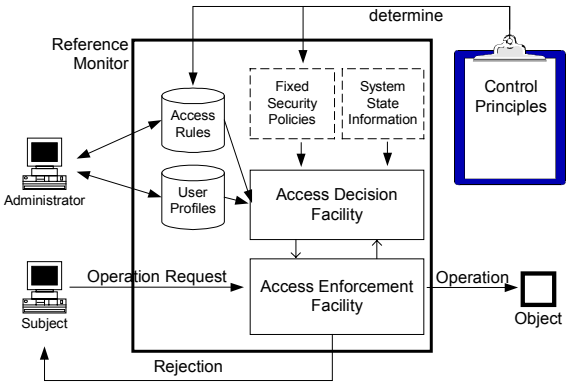


Figure 1: The Reference Monitor in MAC/DAC systems

4 Separation of Duties in Role-Based Environments

Role based access control (RBAC) systems, e.g. Sandhu's RBAC96 model [1], are a development of traditional MAC or DAC based systems, providing a more abstract approach to access control than their predecessors.

RBAC provides the mechanisms that are needed for the integration of separation of duties into an access control system, by introducing a set of pairs of mutually exclusive roles (conflict set).

4.1 Separation of Duties - Related Work

The two initial papers on issues of separation of duties are the Clark-Wilson [2] and Nash-Poland [3] papers, emphasising its importance, while not attempting to integrate it into a formal model.

Kuhn addresses the mutual exclusion of roles to implement separation of duty in a role-based access control system [4]. Simon et al. [5], show different variations of the separation of duty in role-based environments. The two categories of separation of duties that they identify are strong (Static) and weak (Dynamic) exclusion. Gligor et al. [6] use the observations made in [5] for a more formal description of separation of duties characteristics.

Nyanchama et al. [7] introduce a taxonomy of types of conflict of interest in their role graph model. It puts emphasis on the different types of conflict of interest in the three planes of users, roles and permissions and the relations between and among them.

4.2 Role Hierarchies and their Impact on Separation of Duties

Role hierarchies are partial orders, and are therefore transitive. Thus, if a user is a member of a pair of roles which is not in the conflict set, there may still be a violation of a separation of duties policy as expressed by the conflict set. The possible consequences of role hierarchies and their interaction with control principles is described in [8].

5 Animating Separation of Duties in a Role-Based Environment.

One of the aims of our research is to prove properties of experimental configurations of access control systems. Ideally, this would be done by formal proof but, unfortunately, currently available proof support is not able to deal with systems which are at all complex. We are therefore using simulation to examine the results of our experiments. Although it is not capable of providing positive proof of correctness, it can show, in many situation, that our design is wrong, or has unintended consequences. Indeed, it has already done so!

We wish to validate the state of an access control system with respect to separation of duties. We use Prolog and Visual Basic as the underlying technologies for simulation. The result is the SoDA (Separation of Duties Animator) tool that can be used to analyse role-based access control models for static separation of duties conflicts.

5.1 Using Prolog for the Simulation of Separation of Duties Properties.

We are using Prolog for modelling separation of duties properties because it handles recursive queries naturally.

We have used a Prolog database of facts to create the underlying role-based model. Upon these facts we build some rules. The model that we chose was Sandhu's RBAC96 ($RBAC_1$) model as it easy to implement, sufficiently formalised and provides us with the concept of role hierarchies.

Using a Prolog query interface we can ask our system about facts such as existing roles, users or permissions ①, all mutually exclusive roles ②, a certain pair of exclusive roles or all the roles a user is directly assigned to ③. We can then use

these basic queries and combine them in rules such as: asking for all roles that a user has also inherited as a result of being assigned to a role ④; or for a direct violation when a user is assigned to a pair of mutually exclusive roles ⑤. A combination of rules ④ and ⑤ enables us to find violations due to inheritance.

① *role*(R), *user*(U), *permission*(P).
② *exclusive*(Role1,Role2).
③ *ur_assignment*(User,Role).
④ *inherits_from(*Super_Role,Sub_Role*)*:-
*is_a(*Super_Role,Sub_Role*).*
*inherits_from(*Super_Role,Sub_Role*)*:-
*is_a(*Super_Role,Sub_Sub_Role*),*
*inherits_from(*Sub_Sub_Role,Sub_Role*).*
⑤ *show_direct_violation(*User,Role1,Role2*)* :-
*user(*User*), role(*Role1*), role(*Role2*),*
*ur_assignment(*User,Role1*),*
*ur_assignment(*User,Role2*),*
*exclusive(*Role1,Role2*).*

5.2 An Example System

Our example system is that of a software development company. Within that company their exist a variety of roles that company members can take.

Several people will be assigned to the role of a programmer whilst it is imaginable that the same person works as a requirements engineer or on the design of the graphical user interface. Also people work on different projects at the same time.

Certain roles are required to be exclusive, either directly, or by inheritance through the role hierarchy.

The mutually exclusive roles are represented in figure 2. A user must not be assigned to two roles which are directly connected.

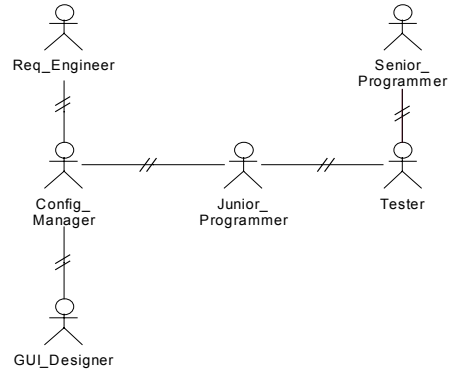


Figure 2: Mutually Exclusive roles

The role hierarchy is graphically represented in figure 3.

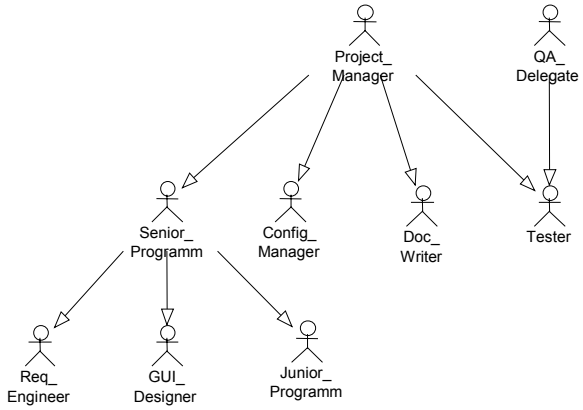


Figure 3: Roles and Role Hierarchy in the Company

5.3 The SoDA (Separation of Duties Animator) tool

The SoDA GUI is an extension to the Prolog query interface.

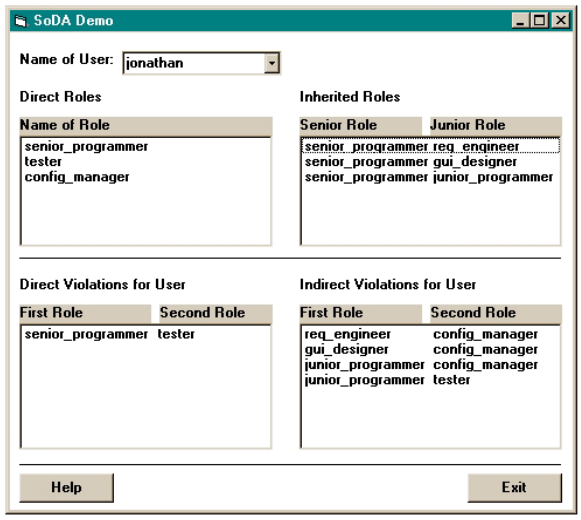


Figure 4: The SoDA user interface

Looking at figure 4, we can see that the tool has found a direct and indirect (by inheritance) violations of our mutual exclusion constraints for the user jonathan. As we deliberately assigned our user jonathan with the two exclusive roles of senior_programmer and tester the direct violation is easy to explain.

*ur_assignment(*jonathan,senior_programmer*).*
*ur_assignment(*jonathan,tester*).*

Of more interest is the fact that we also have an indirect violation for the user jonathan. He is directly assigned to the roles of the senior_programmer, config_manager and tester. We know which roles the role of the senior_programmer inherits (figure 3) and we can see that all of these are mutually exclusive to the role of the config_manager, and one of them to the tester role as well (figure 4). This explains the indirect violations as indicated in the lower right box.

6 Conclusion

Technology

We are developing a second prototype with a facility for integrating any ODBC supporting database in order to allow the basic facts to be held in a relational database. This would allow for the direct run-time manipulation of the system and a stronger separation of program logic from the facts.

Separation of Duties

For the future we plan on extending the tool to handle dynamic separation of duty constraints as they provide a more flexible approach than the static separation of duties. Also we are considering studying roles and their activation in different software development projects using the Chinese Wall approach [9].

Other Control Principles

The techniques that we have used on separation of duties appear to be possible to extend to the control principle of delegation by using delegate roles. It is perhaps more important, from a practical point of view, to provide some means of integrating the requirements of supervision, review and audit into a system. This complex task requires further work.

References

[1] Sandhu R., E. Coyne, H. Feinstein, and C. Youman, "Role-based access control models." *IEEE Computer*, vol. 29, pp. 38-47, 1996.

[2] Clark D. and D. Wilson, "A Comparison of Commercial and Military Security Policies." presented at IEEE Symposium on Security and Privacy, Oakland, California, 1987.

[3] Nash M. and K. Poland, "Some Conundrums Concerning Separation of Duty." presented at IEEE Symposium on Security and Privacy, Oakland, CA, 1990.

[4] Kuhn R., "Mutual exclusion of roles as a means of implementing separation of duty in role-based access control systems." presented at Proceedings of the second ACM workshop on Role-based access control, 1997.

[5] Simon R. and M. Zurko, "Separation of Duty in Role-Based Environments." presented at Computer Security Foundations Workshop X, Rockport, Massachusetts, 1997.

[6] Gligor V., S. Gavrila, and D. Ferraiolo, "On the Formal Definition of Separation-of-Duty Policies and their Composition." presented at IEEE Symposium on Security and Privacy, Oakland, CA, 1998.

[7] Nyanchama M. and S. Osborn, "The role graph model and conflict of interest." *Transactions on Information Systems Security*, vol. 2, pp. Pages 3 - 33, 1999.

[8] Moffett J., "Control Principles and Role Hierarchies." presented at 3rd ACM Workshop on Role Based Access Control (RBAC), George Mason University, Fairfax, VA, 1998.

[9] Brewer D. and M. Nash, "The Chinese Wall Security Policy." presented at IEEE Symposium on Security and Privacy, Oakland, CA, 1989.

Event-centric Policy Specification for E-commerce Applications

Alan S. Abrahams and Jean M. Bacon
University of Cambridge Computer Laboratory
{asa28, jmb}@cl.cam.ac.uk

1 Introduction

This paper introduces an event-centric paradigm for policy specification in electronic commerce applications. We first introduce the problem we are trying to solve and describe why events may help to address the problem. Events are defined and the event paradigm is compared briefly to object-orientation. We show the relationship between events and policy. A brief mapping from natural language to events is illustrated; we describe how some core and domain-specific events can be exposed in natural language specifications and provide some simple examples. Finally we explain how we intend to develop these ideas.

2 Background

The objective of our work is to allow for the executable specification of electronic commerce applications in a natural, English-like form. Current e-commerce applications are written in languages such as Java, Perl, and VBScript. These system-level languages make the application difficult for business users to read and modify, and bury the business procedures (policy) in code. Controlled, English-like, languages such as ACE [7] have been proposed for the executable specification of applications. ACE cannot be used to specify executable e-commerce applications. Aside from its lack of bindings to internet technologies such as HTTP and SMTP, ACE's controlled language explicitly forbids the usage of modal verbs (such as 'can' and 'must') which would be used in English to concisely specify norms: that is, authorizations and obligations. A thorough treatment of norms and norm violation is critically necessary in e-commerce for the definition of contracts between parties and also for the specification of internal procedures, practices, and workflows. Support for the definition of norms (authorization and obligation policies) is provided by policy notations like Ponder [6]. However, Ponder is an implementation-phase focussed technology – it omits analysis and design considerations and provides few pointers for mapping from a natural language specification to events or policies. Recently some work towards translating controlled English specifications into policy been undertaken at Cambridge [4]. However, this has been confined to the functionality provided by OASIS access control policy. Traditional event frameworks like CEA [3] and GEM [11] also omit detailed mappings to English.

We believe that translating an e-commerce application specification into events will eliminate the separation between (English) specification and (programming language) code: the specification, once mapped to descriptive and normative events, can be stored as data in an event store and can be used to control the behaviour of the application. This should bring benefits of ease of maintenance and understanding, and improve the credibility of systems by allowing the business practices implemented to be clear and unobscured by programming language syntax.

In *Section 3* we explain the major concepts of the event-centric paradigm. An explanation of what is meant by event subordination follows. *Section 5* lists the core types of events. *Section 6* explains how events may be found in natural language specifications, while *Section 7* gives sample mappings from real specifications.

3 The Event-Centric Paradigm: Events, Participants, and Parameters

Ponder, OASIS, CEA, and GEM are intended to be integrated with object-oriented technologies; The approach we use is event-centric rather than object-centric. The event-centric paradigm treats events as the primary abstraction. Uniquely identified referents may be participants in events such as classification events (specifying the type or class of the referent), normative events (specifying what the referent can and must do), and other (e.g. domain-specific) events. **Referents** [9] are entities denoted (or denotable) by a unique identifier, and may be things, places, concepts, roles, or other events. Referents are **participants** in the event and are bound to the event in **roles**. Roles may be:

- *event-specific roles*, which depend on the nature of the event itself. For example, 'writer' and 'written' are roles of participants in a 'write' event.
- *semantic or thematic roles* [1, 13], such as actor, patient, instrument, input (resource), or output (product / result) of the event, which pertain to many event types. For example, a pen may be the instrument of a 'write' event, the writer would be the actor, and item written would be the output of the event.

Events are regarded as any occurrence, process, wilful action or activity, or state, in which referents participate

and can be identified, inter alia, from *verbs*. Events may have **parameters** such as time (instant or period) and place.

Though it is unusual to treat the notion of events as incorporating the notion of states we have opted for this treatment as we believe both events and states denote a time-delimited relatedness between entities and are commonly denoted using verbs in English. Notably, Bach [2] treats 'eventualities' as covering both 'events' and 'states'; we have chosen to follow Bach here. For instance 'The unit trust manager *approves* the share sale' is typically considered an event whereas 'The share sale is *approved*' would be considered a follow-on state. In this case, we believe that the state 'approved' is merely a retrospective view of a past, successfully completed 'approve' event. Our contention is that both 'approving' and 'approved' refers to the self-same relatedness as was initially indicated by the 'approve' event - a relatedness between a 'unit trust manager' entity and a 'share sale' entity via an 'approves' event viewed from different perspectives. Barring any reverse action (e.g. cancelling the approval), the 'approved' state will hold. Forever in the future we could contend that the 'share sale was approved' though this may be somewhat misleading as it may imply, perhaps falsely, that the sale is still approved.

It is worthwhile to point out here that the event-centric approach does not make use of the traditional object-oriented notion of objects having attributes and methods. Instead **state** is determined by the currently applicable event bindings to a referent, and **behaviour** of a referent is determined by the ability (authorization) or obligation to:

- insert new events bound to the referent into actionable queues when composite events[1] are detected, or
- invoke external operations.

Referents have rights and responsibilities: responsibilities are obligations to which the referent is subjected, rights are authorizations and obligations in favour of the referent.

4 Subordination of Events

Traditional event frameworks, such as GEM and CEA mentioned earlier, do not provide for event subordination – a notion common in English grammar [12] – where events are relevant only in the context of other events. For instance, in 'administrator *authorizes* John to *read* file', it would seem that 'authorize' is an actual event (in the real world), whereas 'read' is an event that exists in a world of norms created by the 'authorization'. The sentence does not imply that any reading has occurred in the real world. Instead the 'read' event is subordinated to an 'authorize' event and therefore exists only in a sub-world; when interpreting this sentence it would be improper to take any action based on the 'read' event since it has not really occurred - it has only been mentioned in relation to a superordinate 'authorization'.

5 Types of Events

The core event types we define are:

- **Factual** events: These can be user-interface events such as 'selects', 'clicks', and 'displays', or business events. Business events may be:
  - contractual events such as 'buys', 'leases', 'subscribes', or 'insures' which entail the acceptance of terms and definitions and the incurrence of obligations; or
  - workflow events such as 'charges', 'pays', 'approves' which typically entail the discharge of obligations or the exercising of rights (including the uptake of opportunities).
- **Descriptive and definitional policy** events: these include 'naming', 'classifying'[2], and 'quantifying' (e.g. 'counting') events. Depending on the mood of the specification utterance, naming and classification events may:
  - Be operational data: That is, they may be deemed to have occurred, indicating that a referents *is* so named or classified.
  - Indicate definitional policy: That is, they may be sub-ordinated to an *obligation* event, implying that any matching referents *must be* so named or classified.

  Referents may be multiply typed (classified) and types (classification events) may be mutually exclusive. Definitions ('define' events) are obligations to associate a name or classification with items that comply with a description (criteria or constraints). For example:

  Name/Classification: Wealthy Londoners
  are *defined* as (i.e. is a classification that must be given to):
  Description/Criteria: People with yearly income > £100k per year and telephone number beginning with 0207 or 0208.[3]

  There may then be certain prescriptive policies – e.g. related to web-page content personalization – which pertain to referents *classified* as 'wealthy Londoners'.

  Type-determination in the event-centric paradigm is supported through the triggering of a *classification* event when a relevant composite event is detected. In the event-centric development paradigm, the type

[1] Refer to [3] for a discussion of composite events and the operators available for composite event detection.

[2] All event occurrences are typed: typing of events is achieved by the participation of an event in one or more classification events.

[3] This definition would more appropriately be atomized into separate definitions for 'wealthy' and 'Londoner'.

of a concept is determined by events which the (extensionally or intensionally) identified referents have enacted or could enact (as *willing* actors or *controlled* instruments) or to which they have been, or could be, subject (as *voluntary or involuntary* patients). So, the event-centric paradigm supports at least two modes of type determination:

1. Type determination based on *factual event-history*.
2. Type determination based on *pattern of permissible future invocations* (derivable from *normative event history*).

Again, the **behaviour** of referents that are of a particular type – their reaction to events – is determined by what events *must* or *can* be inserted into the event store (or what external operations must or can be invoked) for referents of that type.

Unlike object-orientation, the event-centric paradigm makes direct provision for subjectivism via optionally specified actors of classification, naming, and definition events.

In the event paradigm, both specifications (normative events) and data (factual events) are stored in the event store. Code and data are thus uniformly treated and can manipulated and queried using a common language.

- **Prescriptive policy (normative)** events: these include 'authorize', 'oblige', and 'forbid'. Informally, we can say that prescriptive policy allows us to define what can or must (or cannot or must not) do what to what and when.

Box 1 illustrates some verb frame templates that may be used for the input of descriptive and prescriptive policy events. The syntax of factual events can, in part, be based on verb frames from WordNet [15].

Box 1: Some core events useful across varied application domains

```
{[somebody]} authorizes {[somebody] to} [qualified event]
{[somebody]} obliges {[somebody] to} [qualified event]
{[somebody]} forbids {[somebody] to} [qualified event]
{[somebody]} classifies [something] as [concept] denoted by [symbol]
{[somebody]} names [something] [symbol]
{[somebody]} defines [some-classification] as [criteria]
{[somebody]} quantifies [something] as [measure]
```

'Symbol' is any word or sequence of letters from the language. 'Somebody' is any referent classified as an agent. 'Something' is any referent. 'Concept' is an abstract referent with unique semantics denotable by one or more symbols (synonyms). Where the subject is omitted, the verb is stated in the passive form.
In all cases, referents can be extensionally or intensionally defined: that is, they must be either one or more unique identifiers, or they can be '*'+*criteria* indicating any referent that conforms to specified criteria. 'Qualified event' is any event conforming to stated criteria. Criteria are constraints on the values or types of participants or parameters to the event.

6 Exposing Events in an English Language Specification

A variety of mechanisms are available for exposing events. These can be briefly summarized as follows:

- Contractual and workflow events can typically be identified from verbs and their morphological forms in specifications. For instance a 'subscribe' event may appear as 'subscribes', 'subscribing', or as the deverbative noun 'subscription'. It is important that all these be referenced to a single canonical form (e.g. 'subscribe') if they co-refer to a given verb sense. Possessive forms (e.g. "'s" and "of") may imply the existence of authorizations and obligations (rights and responsibilities) that surround ownership. Roles — which often end in –er, –or, –ant, or –ent in English[4] — frequently indicate underlying domain-specific events. For example, the 'manager' role implies a 'manages' event. Underlying normative events (e.g. oblige, authorise) that describe the *responsibilities* and *privileges* of the role may also be implied.
- Modals such as 'can', 'must', 'should', 'have to' and the suffices '-able' and '-ible' may imply 'authorize', 'forbid', or 'oblige' events.
- Nouns and adjectives imply 'classifying' events; proper nouns imply 'naming' events. Adverbs may imply 'classify' events upon referents that are events.
- Determiners, anaphora, and modifiers or qualifiers (such as 'the', 'that', and 'which') may imply 'selecting' (find and filter) events, as may conjunctions ('and') and disjunctions ('or').
- Quantifiers (such as the cardinals '1', '2', 'one', etc. and words such as 'all', 'none', 'a', and 'some'), and singular, plural, and collective forms may imply 'counting' events.
- Ordinals (such as 'first', '3rd', and 'last'), comparatives and superlatives (e.g. ending in '-er' and '-est'), and prefixes like pre-, post-, and suc- may imply 'sorting' events.
- Negatives may imply negation, failure, forbiddance, or non-occurrence of events, rejection (non-conformance) events, or reversal of the effects of a previous event (i.e. retraction of assertions).
- Tenses and temporal function words (like 'before', 'after', 'within' and 'during') may denote temporal relations [1, 8, 10] between events.
- Speech acts (moods) [5, 14] may imply events: imperative and commisive speech acts imply 'oblige' events; declarative speech acts imply 'naming' or 'classification' events; interrogative speech acts may imply 'selection' (output of finding and filtering according to criteria) events.

[4] E.g. Farmer, Actor, Participant

7 Sample mappings from English to events

Following are two examples which demonstrate simple policies common in e-commerce websites and their implementation in an event-centric paradigm:

7.1 E-commerce Example 1

In the example:

"Customers can buy goods if they are registered".

we can see that:

- The modal 'can' implies an 'authorize' event.
- The nouns 'customers' and 'goods' imply 'classification' events. Specifically they imply that we are looking to match the events: 'referents classified as customer' and 'referents classified as goods'.
- The anaphor 'they' is ambiguous here; being plural, it could imply the selection of either the referents classified as 'customers' or the referents classified as 'goods'. In this case it is the former, and we can substitute 'the customers' for 'they' to disambiguate the sentence. Alternatively, a reordering of the sentence as "Customers, if they are registered, can buy goods" may resolve the ambiguity. Identifying and catering for ambiguity in original phrasings is useful though, as alternative, ambiguity-controlled, phrasings may have a more stilted reading and so would often not be the first-choice of system analysts who write specifications.
- The past tense 'registered' implies we are looking for referents – customers – that have in the past (relative to the policy usage event), participated in a 'register' event[5].
- The function word 'if' implies an event (in this case an 'authorize' event) must be added to the event store when a certain condition is met.

The full translation to an event-centric implementation would be:

"If there is a referent in the event store that is *classified* as a customer as a result of its participation as an actor in a *register* event (i.e. an event occurrence classified as 'register'[6] and temporally before the time of policy usage), then add to the event store an *authorize* event, with the afore-mentioned referent as a participant (beneficiary). This authorization event authorizes 'buy' events to be added to the event store with the afore-mentioned referent (i.e. customer) as an agent of the buy event – i.e. as the buy*er*. A *violation* event may be triggered if there is an attempt by a non-registered customer (i.e. referent classified as customer but not classified as registered) or by a non-customer to buy goods."

7.2 E-commerce Example 2

In the example:

"Purchase orders need three management signatures before they may be approved".

we can see that:

- The modal 'need' implies an 'oblige' event.
- The deverbative noun 'signature' implies a 'sign' event.
- The modal 'may' implies an 'authorize' event.
- The cardinal 'three' implies that we need to trigger a 'counting' event.
- 'Before' implies that the authorization ('authorize' event) may only be triggered after the obligation to count three management signatures for that purchase order has been fulfilled. Furthermore, it implies that there should be a default 'forbid' in the absence of the necessary explicit authorization.

So here, where we encounter a referent *classified* as a purchase order, the system is obliged to, by default, forbid approval events on said purchase order. The system is also obliged to count *sign* (signature) events with actors *classified* as managers and of which that referent (i.e. purchase order) is a patient. If this count yields an output of three, then we can *authorize* approval events which take that referent as a patient (by adding a relevant 'authorize' event to the event store, which overrides the default 'forbid'). In the absence of explicit authorization events, attempts to add any unauthorized approval to the event store are a violation of the default forbiddance on such events.

8 Future Work

Work on the current framework and architecture is ongoing. We are currently beginning implementation of the event-centric development and execution environment and thereafter will attempt to execute a set of real-life specifications which we have gathered. Insights gained will be used to adapt and refine the framework. Events in the current implementation will be stored in a relational database and transferred as text over TCP/IP, SMTP, or HTTP. Supporting tools used will be WordNet, the Cambridge International Dictionary of English electronic edition, as well as a part-of-speech tagger and lemmatizer, the latter being used to identify the canonical form of verbs.

---

[5] Technically we would need to specify that the register event was one that resulted in the referent being classified as a customer, since it would not be sufficient if the referent was classified as 'registered' – e.g. as a user – and then subsequently, in an unrelated event, classified as a 'customer'.

[6] Event occurrences themselves are actually stored as referents and then *classified* as being events of event types. Note that further classifications of the event occurrence may be necessary in order to disambiguate polysemous verbs and arrive at a single sense or meaning for the verb.

9 References

[1] Allen J. *Natural Language Understanding: Second Edition*. Benjamin/Cummings. 1995. pp 1-41, 248

[2] Bach E. The Algebra of Events. *Linguistics and Philosophy*. 9 (1986). pp. 5-16

[3] Bacon J, Moody K, Bates J, Hayton R, Ma C, McNeil A, Seidel O, Spiteri M, Generic Support for Distributed Applications. *IEEE Computer*. March 2000, pp 68-76.

[4] Bacon J, Lloyd M, Moody K, Translating Role Based Access Control Policy within Context. *Policy Workshop 2001*. Bristol, January 29-31, 2001.

[5] Cruse A. *Meaning in Language: An Introduction to Semantics and Pragmatics*. Oxford University Press, Oxford, UK. 2000. Chapter 16, pp 331-346.

[6] Damianou N, Dulay N, Lupu M, Sloman M. *Ponder: A Language for Specifying Security and Management Policies for Distributed Systems. The Language Specification. Version 1.11*. Imperial Collect Research Report DoC 2000/1. 18th January 2000.

[7] Fuchs NE, Schwertel U, Schitter R. Attempto Controlled English (ACE) Language Manual, Version 3.0. Insitut fur Informatik der Universitat Zurich. August 1999. Available at: http://www.ifi.unizh.ch/groups/req/projects/attempto/rerg_projects_vtclp.html

[8] Hayes PJ. *A Catalog of Temporal Theories*, Tech report UIUC-BI-AI-96-01, Beckman Institute and Departments of Philosophy and Computer Science, University of Illinois 1995. Available at: http://www.coginst.uwf.edu/~phayes/TimeCatalog1.ps Accessed on: 1 May 2000

[9] Kamp H, and Reyle U. *From Discourse to Logic*. Kluwer Academic Publishers. 1993.

[10] Ladkin PB. *The Logic of Time Representation*. PhD Thesis. Group in Logic and Methodology of Science, University of California at Berkeley. November 1987.

[11] Mansouri-Samani M and Sloman M. GEM: A Generalised Event Monitoring Language for Distributed Systems. *IEE/IOP/BCS Distributed Systems Engineering Journal*, Vol. 4 No. 2, June 1997. Extended version of a paper presented at ICODP/ICDP '97 conference, Toronto, May, 1997.

[12] Palmer F. *Grammar*. Penguin Books. 1984.

[13] Sowa JF. Knowledge Representation: Logical, Philosophical, and Computational Foundations. Brooks/Cole. Pacific Grove, California, USA. 2000. Revisions to the chapter on *Thematic Roles* available at: http://www.bestweb.net/~sowa/ontology/thematic.htm

[14] Thomas J. *Meaning in Interaction: An Introduction to Pragmatics*. Addison Wesley Longman Limited. 1998. pp 1-54

[15] *WordNet – A Lexical Database for English*. Available from: http://www.cogsci.princeton.edu/~wn/online/

Session 6
Policy Applications

Session Chair:
Ian Marshall, BtexaCT, UK

Policy-Based Storage Management

M. A. Carlson[1], M. Dutch[2], J. Gelb[3], G. Mueller[4], P. Spasic[5] and L. VanArsdale[6], [1]Sun Microsystems, [2]Troika Networks, [3]IBM, [4]Storage Tek, [5]HP, [6]BMC Software, USA

Distributed Policy Management and Comprehension with Classified Advertisements

N. Coleman, R. Raman, M. Livny and M. Solomon
University of Wisconsin, USA

Policy-based Management: Towards Internet Service Provisioning

J. Vicente[1], L. Xie[1], H. Cartmill[1] and G. Anavi[2],
[1]Intel Corporation, USA, [2]Intel Corporation Israel

Policy-based Storage Management

Authors:

Mark A. Carlson, *Sun Microsystems*
Mike Dutch, *Troika Networks*
Jack Gelb, *IBM*
Gary Mueller, *StorageTek*
Predrag Spasic, *HP*
Lynne VanArsdale, *BMC Software*

SNIA Policy Work Group

Abstract

Enterprises seek automated mechanisms to manage networked storage, driving solutions based on policies. This policy-based storage management presents a new set of issues that need to be evaluated when proposing policy standards. This paper presents some new terminology and use cases that address those issues, as well as some suggestions for future directions.

Keywords:

Storage, SAN, storage management, policy terminology

SNIA Background

The SNIA is a not-for-profit organization, made up of more than 150 companies and individuals formed to promote the growth and development of the market for storage and networking systems and technologies.

SNIA members share a common goal: to set the pace of the industry by ensuring that storage networks become efficient, complete, and trusted solutions across the IT community. To this end the SNIA is uniquely committed to delivering standards, education, and services that will propel open storage networking solutions into the broader market.

Policy and New Paradigms for Storage

Policy-based storage management (PBSM) has the potential to significantly improve the way the world manages storage. Today, multiple applications and computing hosts must share and exchange data. The current high-level of interest in storage networking indicates the need to manage data at the storage level, rather than the host or application level. Enabling management at the storage level requires automation for security, cost/labor and quality reasons. Hence the Storage Network Industry Association (SNIA) is working on standards for policy-based networked storage management.

The purpose of PBSM is to enable management of storage as a service so that it meets business goals of the organization. As any management framework, PBSM needs to enable control configuration and behavior of entities such as hosts, applications, SAN and storage components. Approaches so far have been to assume that applications have knowledge of all individual entities. The scalability is unpredictable. This scalability problem becomes more complex in an ever growing environment such as Storage Service.

Early studies of the SNIA Policy Work Group includes an informal survey of the policy standards available or in-progress today. A major effort in this area is the Distributed Management Task Force (DMTF) Common Information Model (CIM) policy model.

The SNIA Policy Work Group has found this model to have some biases toward message network issues and that storage networks require additional considerations. The differences lie in the fact that storage networks deal not only with data on-the-move, but also data in place. When considering storage networks, location characteristics become more important and the size of data being moved is on average larger. Accessibility becomes of primary importance, whereas in message networks, techniques such as alternate pathing can help work around many access issues. Persistence is much more of an issue with storage networks than message networks. In general stored data and its management is more closely related to business process and policy than message passing, hence more easily allowing an opportunity to automate and enforce business policies at an IT level.

These differences have inspired the SNIA Policy Work Group to revise and add to the current DMTF policy definitions. The goal of the work group is to increase interoperability and industry efficiency. The work group has also begun an effort to create standard policy-related metrics to allow interoperability of policy-based storage management systems. The group is also considering the generation of common representations and APIs to enable higher-level integration (above model level) in the future.

Definitions for Storage-related Policy

The SNIA Dictionary

The SNIA Education Committee has standardized a Dictionary of Storage Networking Terminology. Compiled through the cooperative efforts of the SNIA's 150 member companies and individuals, the Dictionary presents standard definitions of nearly 1,000 commonly used terms as the leading participants in the storage industry have agreed upon them.

The Dictionary is available at *http://www.snia.org/English/Resources/Dictionary.html*, and is a hypertext document, which includes comprehensive definitions and internal links. It presents definitions for a broad range of storage terminology, including both technically detailed terms and higher-level storage networking concepts.

Policy Terminology

One of the work items of the SNIA Policy Work Group is to extend the SNIA Dictionary to include terms and their definitions that relate to policy based management of storage networks. Towards this end, the group is leveraging the work of the networking industry and the Internet Engineering Task Force (IETF) standards body. The IETF is in the process of creating its own set of standard terms for policy based network management.

The Policy working group has been actively involved in the IETF terminology effort to ensure that the terms that are defined are also applicable to storage networking. The IETF Policy terminology document is available in draft form at: *http://www.ietf.org/internet-drafts/draft-ietf-policy-terminology-00.txt*

The goal of this work is then to leverage both the existing networking terminology standards and the existing SNIA Dictionary to define a set of terms that are self consistent and will advance the quality of standards in this area.

The terms will be drawn from the use cases that the group has developed and will be checked against those use cases for validity and usefulness.

Conceptual Distinctions

It is important to draw a distinction between the above description of policies and the concept of “attributes”. While policies are rules that can be modeled by condition-action tuples, attributes specify intrinsic or extrinsic characteristics of the entities they describe. While these two terms may be considered independent of one another, it is also reasonable that they may be inter-related; i.e. policies may cause the (re-)setting of attributes, and changing attribute values may cause the initiation of policies. Both concepts are valuable in characterizing the behavior of networked storage.

Another area of confusion is distinguishing between the management of storage and the management of data *on that storage*. Policies and attributes associated with storage control the functioning (or “serving ability”) of the storage devices themselves,

as well as the networking of such devices (be it SAN, LAN, WAN, or other). This includes such functions as capacity management, configuration, performance ("feeds and speeds"), and reliability. Data management, however, is concerned with the control of stored data from a client perspective: data deliverability, recoverability, accessibility, etc.

In our work on policy management for storage networking, we will endeavor to further clarify and quantify the above distinctions, and examine the synergistic relationships among them.

Use Cases

Use cases provide the basis to evaluate current and emerging policy standards from the DMTF, IETF and ANSI. They provide examples of how a policy-based storage management system might be used to manage storage and provide a method for creating storage-related policies and policy-based management tools.

Use cases provide examples of how policies are used and are described by their goals, metrics and mechanisms.

Goals: What is to be achieved?

Metrics: What measurements are required to determine if the goals are being achieved?

Mechanisms: What actions are taken to achieve the goals?[1]

Categories of Use

Business policies and the use cases they describe may be organized into categories according to their intent.

- **Legal**
- **Cost saving**
- **Cost-avoiding**
- **Strategic**
- **Risk management**
- **Business continuity**
- **Asset deployment optimization**
- **Personnel deployment optimization**
- **Competitive**
- **Partner**

SLAs, SLOs and Metrics

"IT needs to reorient its attention towards business requirements and away from technical issues, such as systems availability and staff efficiency. Efficiency and effectiveness per se have no meaning if not achieved towards a business end." IDC, 1999

Although SLAs as means of specifying business level requirements,are well established and accepted by the business community, they are not so widely used in storage management applications, partially due to the lack of appropriate standards and verification mechanisms. SNIA Policy Work Group is working towards establishing these standards.

A Service Level Agreement (SLA) codifies the requirements and expectations of all parties, which usually are service provider and a customer. In our context, SLAs represent contractual aspect of the service specification. Service Level Objectives (SLO) specify desired storage and application system service levels so that they can be mapped into Element Policies that manage each of the elements of the service. Metrics for SLA verification and enforcement are also specified.

SLA performance specification for storage might be specified in terms of:

- Application level performance
- Storage Level Performance

Challenges

- Existing policy models do not adeqately address specific requirements of Storage Service management. Extension of the

[1] ***Notification and Escalation***Notification and escalation are common mechanisms available for all of the categories. A policy architecture may wish to factor out notification and escalation policy into a single location by defining category- independent severity and basing actions on that severity.

model to include data managemnt aspect is not a trivial task.

- Although Information models that describe policies in a platform/protocol/device independent model are available, devices still require interpretation of policies and metrics that are stored in the repository to the level that can be executed by the target.
- Application level performance SLOs are much closer to business requirements and therefore easier to map to general policy specifications, emerging Storage Service business might not have control over the hosts, or hosts might not be in the administrative domain of Storage Service Provider

Future Standards for Interoperability and Quality

Near-term the SNIA Policy Work Group plans to apply the use cases to the proposed DMTF CIM Policy Model. The group plans to forward any recommendations resulting from this work to the DMTF for inclusion with CIM version 2.4. We plan to submit our ideas regarding a common understanding of metrics for storage policies to appropriate standards organizations, as well. Future work will include further contributions toward Information Models and standards for policy interoperability end-to-end in complex storage environments.

Information Models

Information models are used to abstract the capabilities of multiple possible implementations and provide a common meaning to the concepts that require management. The classic example is a device's cooling mechanism: some implementations may call it a fan, some a cooling unit, while others use the term temperature control. These are all semantically equivalent and need a common abstraction that allows for the management of cooling mechanisms regardless of the implementation.

Similarly, in order for policy-based mechanisms to automate the management of devices and services, there is a need for abstractions of the metrics, conditions, actions and goals inherent in this management. The policy working group will work to refine the information models associated with policy-based management of network storage, drawing from the core models for networking quality of service and extending and refining the models as appropriate for interoperability.

End-to-End Policy Management

Policies for managing network storage must inter-operate with policies that control data, the network fabric, and the host environment (e.g. workload, connectivity, availability). As an example, policies that prescribe data performance for a particular instance of a specific application on a given host must be cognizant of – and interact with – policies that control network performance, and the performance of the storage device on which the data exists. The seamless integration of such policies is essential to balancing enterprise throughput, as well as simplifying its administration. The actuality that heterogeneous products are providing support for a multiplicity of policies with unique implementations must not burden the customer. Hence, it is our goal to seek standards and promote interoperability among providers to satisfy this requirement.

References

a) DMTF Networks Working Group. Information can be obtained from: http://www.dmtf.org/info/networks.html.

b) CIM V2.x Schema available from http://www.dmtf.org/spec/cims.html.

c) DMTF SLA Working Group. Information can be obtained from: http://www.dmtf.org/info/sla.html.

d) IETF Policy Framework Working Group. Information can be obtained from: http://www.ietf.org/html.charters/policy-charter.html

Distributed Policy Management and Comprehension with Classified Advertisements (Extended Abstract)

Nicholas Coleman, Rajesh Raman, Miron Livny and Marvin Solomon
University of Wisconsin, 1210 West Dayton Street, Madison, WI 53703
{ncoleman,raman,miron,solomon}@cs.wisc.edu

December 6, 2000

Abstract

Federated distributed systems present new challenges to resource management, which cannot be met by conventional systems that employ relatively static resource models and centralized allocators. Matchmaking paradigms based on identifying compatible classified advertisements placed by providers and requesters of services work well in such environments. We describe a matchmaking framework which realizes a federated distributed policy model in Condor, a production-quality distributed system. However, due to the distributed polices and dynamics of such federated environments, understanding why some classads are not matched while others are can be a very complex task. We therefore present algorithms which not only identify problematic aspects of a policy, but also suggest modifications.

1 Introduction

A fundamental challenge of distributed federated systems is collaboration in highly dynamic, heterogeneous and untrusted environments. In such environments the service providers and requesters which comprise the federation must be able to specify policies which define the conditions under which they will collaborate. The difficulties involved in accommodating such a framework include a notation for *specifying* such usage policies, a method for *discovering* entities compatible with the stated policy, a simple yet flexible and scalable architecture to *implement* policies and analysis technologies to *comprehend* the implications of policies.

In this paper, we describe distributed policy management and comprehension in the context of Condor, a production-quality distributed high throughput computing system architected on a federated model, developed at the University of Wisconsin-Madison. The emphasis on policy management in Condor is on the specification and implementation of *resource allocation policy*. Condor operates in highly dynamic environments characterized by distributed management and distributed ownership. Distributed management introduces *resource heterogeneity*: Not only the set of available resources, but even the set of resource types is constantly changing. Distributed ownership introduces *policy heterogeneity*: Each provider and requester has a unique idiosyncratic notion of "compatibility."

Condor solves these problems by adopting a Matchmaking paradigm: providers and customers of services (usually compute services) send the Matchmaker structures called *classified advertisements* (classads), which are declarative descriptions of the principal's characteristics, constraints and preferences. The Matchmaker uses a generic policy-neutral algorithm to discover and notify compatible agents. The notified agents then activate a claiming protocol to establish a collaboration. Thus, in contrast to many conventional resource management systems, Condor does not impose a monolithic allocation and scheduling model on the resources in its purview. Instead, providers and requesters independently and dynamically define allocation policies, and form dynamic collaborations when matched, realizing an *opportunistic computing* paradigm.

The simplicity and flexibility of this distributed policy approach has been validated in practice—Condor has been successfully deployed in both academic and industrial environments as a production quality system. Experience has shown that the classad-based Matchmaking framework enables the description of sophisticated policies to accurately represent the expectations of the system's users. However, we have also discovered that understanding why certain classads are not matched (while others are) can be a very complex task in the presence of complex policies and environment dynamics. In response to this problem, we have developed algorithms which not only identify problematic aspects of a principal's policy, but also suggest modifications.

2 Matchmaking

The underlying ideas of the matchmaking paradigm are intuitive and very simple. In this section, we briefly describe the fundamental processes and components of our matchmaking framework.

In our framework, human users who participate directly or indirectly in the system are called *principals*, and the software programs that represent principals in the system are called *agents*. Server and customer agents requiring matchmaking services express characteristics, constraints and preferences to a Matchmaker (illustrated as Step 1 in Figure **??**). We call these agent descriptions *classified advertisements* in analogy to their newspaper counterparts. The task of the Matchmaker is to detect compatible advertisements in a generic manner (Step 2), which is performed by checking the constraints specified in the respective advertisements. When compatibilities are discovered, the Matchmaker notifies the respective advertising agents, discards the matched classads and relinquishes any further responsibility for the match (Step 3). Matched agents then establish an allocation through a claiming process that does not involve the Matchmaker (Step 4).

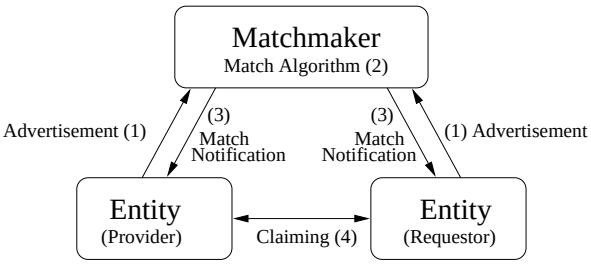


Figure 1: Actions involved in the Matchmaking process

Our matchmaking framework can be naturally decomposed into the following components:

1. A language for specifying the characteristics, constraints and preferences of agents. Our framework uses the *classified advertisement* (classad) language for this purpose. Classads are semi-structured sets of $(name, expression)$ pairs which may be thought of as "attribute lists" that describe agents. The language has special **undefined** and **error** values, as well as special operator semantics to operate with these values. Classads may be nested, in which case lexical scoping is employed to resolve attribute references: if an attribute is not found in a nested classad, the attribute is looked up in successive enclosing scopes till the reference is resolved. (If the reference could not be resolved, the reference evaluates to **undefined**.)

2. The *Matchmaker Protocol* describes how agents communicate with the Matchmaker to post advertisements and receive notifications.

3. The *Matchmaking Algorithm* is used by the Matchmaker to create matches. In the abstract, the matchmaking algorithm transforms the contents of submitted advertisements and the state of the system to the set of matches created.

4. *Claiming Protocols* are activated between matched parties to confirm the match and establish a working relationship.

The simplicity, flexibility and expressiveness of the classad language greatly contributes to the effectiveness of our Matchmaking framework. Figures **??** shows a classad describing a workstation in the University of Wisconsin-Madison Condor pool.[1]

Most attributes of the workstation (e.g, `Name`, `Memory`, `OpSys`) describe the machine's characteristics. The `Requirements` and `Rank` attributes are of special interest to the Matchmaker since these attributes identify the advertising agent's constraints and preferences. When testing the compatibility and preferences of two advertisements A and B, the Matchmaker places the two advertisements in an evaluation environment such that in classad A, the reference `other` evaluates to B, and *vice versa*. Thus, the workstation in Figure **??** has the following policy: Jobs belonging to user "riffraff" are never accepted, and jobs are only serviced when the machine has a low load average and its console has been idle for at least fifteen minutes. Furthermore, jobs with small image sizes are preferred between 9 a.m. and 5 p.m. Similarly, the job shown in Figure **??** requires an INTEL workstation running the LINUX operating system with at least 128 MB of memory. Among all such workstations, the job prefers machines with better KFLOPS ratings and more memory.

Many interesting and useful policies may be easily defined within this framework; interested readers are referred to reference for more sophisticated examples derived from the policies of real-world users of the Condor system.

3 Classad Analysis

Due to the dynamics of large distributed systems and the policies that may be specified in classads, understanding why some matches fail (while others succeed) can be a complex task. The match failure may be due to a mistake made in the user's configuration files, or the user may not

[1]The Wisconsin Condor pool is currently composed of over 675 nodes, running nine different architecture/operating system combinations. The pool is used continuously as a production system to provide computation services for several research projects.

```
[
  Type        = "Machine";
  Activity    = "Idle";
  KeybrdIdle  = '00:23:12'; // h:m:s
  Disk        = 323.4M;     // mbytes
  Memory      = 256M;       // mbytes
  State       = "Unclaimed";
  LoadAvg     = 0.042969;
  Mips        = 104;
  Arch        = "INTEL";
  OpSys       = "LINUX";
  KFlops      = 21893;
  Name        = "foo.cs.wisc.edu";
  Subnet      = "128.105.175";
  Rank        = DayTime() >= '9:00' &&
            DayTime() <= '17:00' ?
            1/other.ImageSize : 0;
  Requirements= other.Type == "Job"
            && other.Owner != "riffraff"
            && LoadAvg < 0.3
            && KeybrdIdle>'00:15'
]
```

Figure 2: Classad describing a Machine

```
[
  Type         = "Job";
  QDate        = 'Mon Jan 11 10:53:31
                   1999 (CST) -06:00';
  CompletionDate = undefined;
  Owner        = "raman";
  Cmd          = "run_sim";
  WantRemoteSyscalls = true;
  WantCheckpoint = true;
  Iwd          = "/usr/raman/sim2";
  Args         = "-Q 17 3200 10";
  Memory       = 31m;
  Rank         = KFlops/1E3 +
                   other.Memory/32;
  Requirements = other.Type == "Machine"
                   && other.Arch=="INTEL"
                   && other.OpSys=="LINUX"
                   && other.Memory >=128
]
```

Figure 3: Classad describing a Job

be aware of the qualities of the various services or their allocation policies. There is therefore a necessity to analyze a classad's policy to know if it may be successfully matched in an environment, and if not, why.

In this section, we discuss algorithms built to analyze match failure for jobs submitted to the Condor system. In some cases, typographical mistakes are made in their job description files, while in other cases, some Condor users define job policies which cannot be satisfied, while in still other cases, the job does find machines that it requires, but the resulting machines do not accept the job. While there are other causes of match failure, we mainly focus on these issues for the purposes of this discussion.

In our model of classad analysis, we consider Requirements expressions which may be decomposed into subexpressions which are then evaluated in the contexts of the various machine classads. A subexpression that consistently causes the requirements expression to evaluate to **false** can be tagged as problematic.

A requirements expression is for all practical purposes a quantifier free formula in first order logic. Since it can be expressed as a series of smaller subexpressions joined by logical operators (e.g. `&&`, `||`, `!`) we call it a *molecular formula*. The smallest such subexpressions are *atomic formulas* or atoms. If we treat these atoms as propositions then we can convert any arbitrary molecular formula into a formula in disjunctive normal form (DNF) and use this form as the basis for our analysis.

3.1 An Algorithm For Analysis

Let us examine the case in which a job's requirements expression evaluates to **false** in multiple machine classads (machine classads). For the reasons stated above we can restrict our algorithm to expressions in DNF. DNF consists of a series of subexpressions joined by `||` operators. Each one of these subexpressions is a series of atoms joined by `&&` operators. We shall call such a subexpression a *profile*.

Since our requirements expression evaluates to **false** it follows that all of the profile subexpressions evaluate to **false** as well. We need only modify one of these profiles to evaluate to **true** in order for the entire expression to evaluate to **true**. Our algorithm for analyzing requirements expression shall focus on these subexpressions in *profile form*:

$$\begin{aligned} \mathit{Profile} &\Rightarrow \mathit{Atom}\ \mathbf{\&\&} \ldots \mathbf{\&\&}\ \mathit{Atom} \\ \mathit{Atom} &\Rightarrow (\text{atomic formula}) \end{aligned}$$

Our goal is to determine which atoms must be removed from the expression so that the job matches at least one machine. We would like to remove the fewest number of atoms possible. If there is more that one combination of atoms that satisfy these criteria, we would like to chose the combination whose removal will result in matching the most machines possible. The more machines we can match with a job, the better chance we have of finding a machine whose *rank* expression evaluates to a high value.

To accomplish this goal we build a table to store the results of evaluating each of the individual atoms in the

different machine classads. The columns of the table represent atoms and the rows machine classads. By examining the total number of **true** values in each column and row we can select the best subset of atoms to be removed. If there are no **true** entries in a column, this means that the corresponding atom is not satisfied in any machine classad.

After removing all such atoms we turn to the total **true** values for each of the rows. If the total number of **true** values in any one row is equal to the number of remaining atoms, the resulting expression will evaluate to **true** in the corresponding machine classad, and our goal is satisfied.

Otherwise, we seek out the rows with the highest total **true** tallies. We then find the most common configuration of values in these rows. Every non-**true** value in this set corresponds to a atom that we can remove so that the profile evaluates to true. Since the rows with this set of values have the highest total **true** tallies, we end up removing the fewest number of atoms possible. Having done so, we are guaranteed to remove the combination of atoms that result in matching the most machines by choosing the most common configuration of values in these rows. Our goal is now satisfied.

3.1.1 A Simple Example:

One common case of mismatched classads is user error, as exemplified in the following expression, where the user has accidentally requested the non-existent "SPARK" architecture.

```
Requirements = (Arch==``SPARK'') &&
       (OpSys==``SOLARIS2.7'')
```

The evaluation tableau of the above expression is:

Machine Classad	**(Arch == "SPARK")**	(OpSys == "SOLARIS2.7")	Total True
1	**F**	T	1
2	**F**	F	0
3	**F**	F	0
4	**F**	T	1
5	**F**	T	1
6	**F**	F	0
7	**F**	T	1
8	**F**	F	0
9	**F**	T	1
10	**F**	T	1
11	**F**	T	1
12	**F**	F	0
13	**F**	T	1
Total True	**0**	8	

Analysis: Atom 1 always evaluates to **false** in all machine classads. Atom 2 evaluates to **true** in some machine classads. If atom 1 is removed, the expression will evaluate to **true** in some machine classad.

Suggestion: Remove atom 1.

4 Conclusions

As the matchmaking process is used to deal with larger groups of resources, and the classad language is used to represent more complex policies, the need for classad analysis will only increase. A framework is needed that will be optimized for the common case (profile form) but be powerful enough to analyze expressions in any form. It follows that a classad analysis tool must be very robust, but it must also be usable. If the presentation of the results of the analysis is not clear and concise the tool will be of no help to the user.

The classad analysis framework that is currently being developed at the University of Wisconsin will address these issues. A preliminary version of classad analysis has been implemented in the Java programming language. The API is flexible enough to support either a graphical user interface or a command line interface.

Classad analysis may also be extended beyond the purview of requirements expressions of jobs and machines. Condor uses classads in a variety of ways including job scheduling and the negotiation process. In many of these cases it may be useful to provide analysis results that are useful to an automated process, rather than to a user.

Policy-based Management: Towards Internet Service Provisioning

John Vicente, Lilin Xie, Harold Cartmill
Information Technology, Intel Corporation
1900 Prairie City Road
Folsom, CA 95630
USA

Gil Anavi
Information Technology, Intel Corporation
IDC1-1B, MTM-Scientific Ind. Center POB 1659
Haifa, Israel

Extended Abstract

Index words: QoS, policy, policy-based management (PBM), service provisioning

1 INTRODUCTION

Current trends in corporate and Internet networks are shifting from best-effort, vertical network architecture towards a more intelligent, end-to-end, service-aware network paradigm. As evidenced over recent years, the need for enhanced network services such as virtual private networks (VPN), quality of service (QoS), security, collaboration, and directory technologies demonstrates that customers are demanding more from the core infrastructure for enabling productivity, flexibility, service differentiation, isolation, privacy, and manageability. Moreover, critical network resources must be aligned with business objectives where networks are i) more content or application-aware; ii) provide dynamic features for service creation; iii) observe and enforce network-wide policies; and finally, iv) enable control from the network provider to the administrator to the end-user.

The changing models for enterprise business communication and collaboration are forcing more virtualization of infrastructure and services, yet requiring more sophistication in management and networking technology to enable seamless transmission integration and control. This shift requires a tremendous effort on the part of information technology and service provider organizations to adopt and transform existing operational, network management and provisioning tools, skills and processes to match this paradigm. Key areas we believe policy-based management (PBM) can improve in order to help facilitate this transformation include infrastructure abstraction and simplification, policy validation, and network management integration.

In this paper, we present initial findings from Intel's Information Technology (IT) research and development activities supporting policy-based management. Our PBM research agenda was geared towards IT operational integration and enterprise deployment feasibility. More specifically, our R&D work focused on the integration of QoS mechanisms and policy administration to facilitate service differentiation and bandwidth management. Based on our experience and learning, we provide an operational perspective on existing commercial solutions and propose technology extensions based on our findings. Moreover, we view policy-based management as a viable technology to provide greater control and management of underlying networks via the creation and coordinated distribution of abstract policies; thus allowing for a diverse range of administration and network management automation. We propose extending the existing policy-based network management systems towards Internet service provisioning and present a discussion on the operational and management requirements necessary to achieve this perspective.

2 INFORMATION TECHNOLOGY TRIALS EXPERIENCE AND USAGE

Within Intel IT, our research agenda was to qualify the suitability of policy-based management technology for the corporate enterprise environment as well as to define the transitional models to support e-Business. In what follows, we provide an operational perspective of our findings, our developments in support of key operational shortcomings observed, and present a case study of a successful deployment of policy-based QoS provisioning.

2.1 Operational Shortcomings

As stated earlier, our initial technology evaluation objectives primarily focused on QoS and provisioning to enable service differentiation and bandwidth management. To facilitate these objectives, we developed a Quality of Service Network for Emerging Technologies ("QoSNET"). QoSNET is a production-level and policy-managed network environment to support proof-of-concept QoS and policy-based management technologies. QoSNET brings together these enhanced technologies along with emerging applications to investigate intelligent bandwidth management capabilities. Using commercially available PBM tools, we observed that the current solutions revealed similar operational shortcomings. First, existing PBM tools lacked open, plug-n-play features for network management integration. Secondly, QoS provisioning lacked QoS abstraction consistency and end2end completeness. Third, deploying QoS policies can have significant ramifications for the operational environment, and it was apparent policy impact verification was not given appropriate consideration in most of the solutions. Finally, policy deployment in terms of operational and business process integration is a daunting challenge for traditional IT and ISP environments. During the time of this writing, most of the tools lacked a rigorous or consistent methodology associated with traffic analysis and characterization, policy creation, deployment and validation. In other words, the policy administrator required a high-degree of skill, multi-tasking and operational risk in deploying QoS. In our opinion, it is the combination of the above issues, which we perceive to be the key challenges for more pervasive adoption of today's policy-based management solutions.

2.2 IT Research & Development

The above issues served as motivation for our research and prototype development activities – PBM Operational Console and the Network Traffic Controller (NetTC). The PBM Operational Console, as a mock-up prototype, was developed to drive requirements for PBM technology. The Network Traffic Controller is a prototype of an end-system based QoS provisioning

model that i) integrates policy, QoS, metering; ii) allows both static and dynamic QoS provisioning; and iii) correlates event with policy validation.

2.2.1 PBM Operational Console

The PBM Console prototype contains two components: a schema stored in a relational SQL server and a policy administrative GUI implemented as a Microsoft MMC* snap-in. The motivation behind the schema were operationally driven with the design goals set to achieve: i) two levels of QoS provisioning policy abstraction; ii) business SLA, operational and infrastructure integration; and iii) policy validation.

Within the PBM Console schema, there exists a *Policy* entity associated with a *ServiceLevel* entity and a policy *Rule* entity. The *ServiceLevel* entity represents abstract service levels (e.g., Gold Service) and associates with a *QoS* entity that encompasses network resources such as bandwidth partition, priority, rate and other QoS attributes. The *Rule* entity is associated with *Route* entity and *RuleCondition* entity. The *Route* entity associates with the *Interface* entity for network nodes and the *NetEvent* entity for policy correlation with bandwidth usage and network event conditions. *Interface* entities associate with a *Node* entity for network devices, which are linked to the *Event* entity to indicate network device status. The *RuleCondition* entity is associated with a conditional resource entity or *ResourceGroup* entities to identify application, physical location, user, and time. To define a policy, the user needs to understand the service level and required conditions in terms of application, site, user, and the times for which the policy would apply. The schema would automatically associate the underlying network nodes associated with the QoS path and QoS specification necessary to enforce the policy. The status of deployed policies would be reflected on the console based on dynamically updated events. Figure 1 is a 'snap shot' of the GUI console that demonstrates a sample presentation of the Console model. As shown, it provides a simple business view of the presentation of QoS policy information including service level, status, and rules associated with the policy. Although not shown, the PBM console supports an Infrastructure view, which details policy rules and conditions on a more granular view of the infrastructure resources, and correlates (yet hides their schema relation) the business policies with the infrastructure policies, including component resource QoS specifications. Also not shown, an Operational view depicts both the infrastructure and business status information in terms of policy rules and associated status, infrastructure usage and network event conditions. Thus, operational manageability is afforded for both business and infrastructure views and correlated in schema and presentation format.

With the PBM Operational Console prototype, we have attempted to simplify the provisioning model for both the sophisticated and unsophisticated PBM administrator

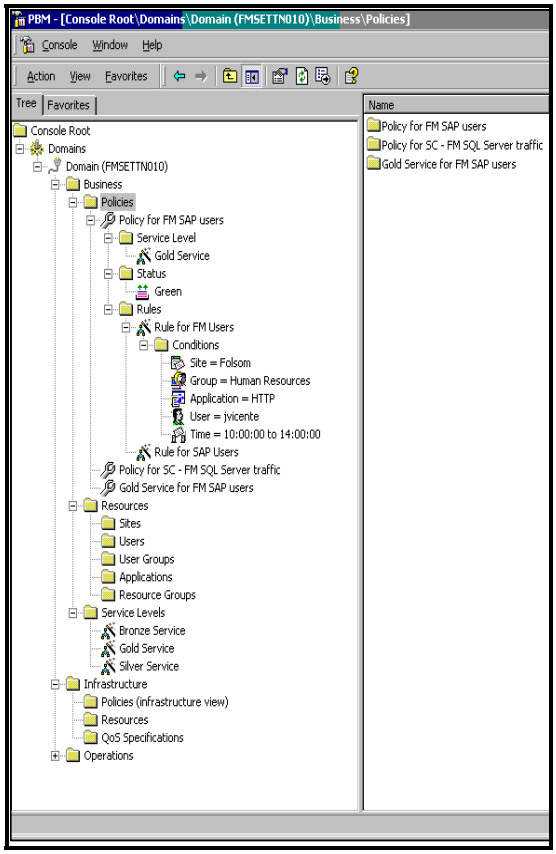


Figure 1: PBM Operational Console GUI

or user, correlate business policy with infrastructure policy and demonstrate a means to validate deployed policies.

2.2.2 Network Traffic Controller

The Network Traffic Controller prototypes a central policy enforcement administration engine and distributed client enforcement agents working in cooperation as a scalable and feedback-driven control system. As illustrated in Figure 2, the NetTC architecture consists of four major components including a client component "NetTC Agent", two server components "NetTC Administrator Traffic Control" and "NetTC Administrator Segment Collector", and a Microsoft MMC* console. The NetTC administrator *automates* QoS provisioning 'refinement' and enforcement through central, dual-threaded measurement-based algorithms and a policy data store. A SQL server database stores

* Other trademarks and brands are the property of their respective companies.

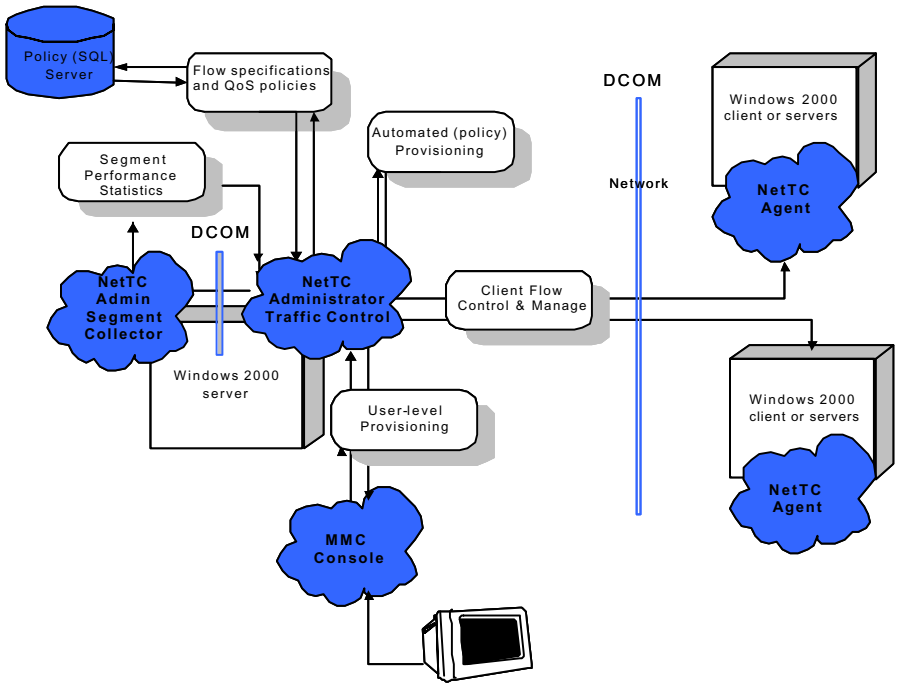


Figure 2: Network Traffic Controller Architectural Components

relevant configuration, policy and flow specification information, which is used by NetTC Administrator to control and refine QoS at host systems where the QoSAgent runs, and to manage LAN network bandwidth.

The NetTC Administrator also provides user-driven, manual QoS provisioning through the console. By implementing NetTC's manual QoS provisioning, we investigated host-based QoS control capability using the QoS Traffic Control APIs* (i.e., QoS flows & flow specs, flow filters, QoS functions & structures) made available through Windows 2000*. The client agent runs as a DCOM* enabled service called QoSAgent on a Windows 2000 client (or any QoS-enabled Windows client, e.g., Windows 98* system). It uses the Traffic Control API* (TCAPI) to manage flows, flow specs and filters on the QoS-enabled client. If an application is designed as QoS-aware (i.e., GQoS API*) using the RSVP service mechanisms, e.g., NetMeeting*, the application can allow the RSVP service provider to manage traffic flows generated by this application. Alternatively, the QoSAgent can also make legacy applications QoS-aware through the TCAPI. In both cases, an application communicates with the QoS Packet Scheduler*, which is a service running on Windows 2000* systems, to have its packets marked, filtered according to the application specifications or in the case of NetTC, based on the NetTC Administrator's instruction.

One of the unique aspects of the NetTC Traffic Control prototype is the use of performance feedback information on a local flow-level and global network usage level. In addition to manual invocations made by way of the MMC* administrative console, feedback and automated provisioning decisions can be made on a policy-driven and dual timescale to set policies, reset thresholds, police "greedy" flows or better manage the QoS needs of applications (their flows) as well as the stability of the shared network resources. The latter is achieved by setting appropriate thresholds at different timescales and augmenting per flow specifications where appropriate to reduce or avoid potential contention or spiky congestion behavior.

2.3 Case Study – QoS Deployment

The Packeteer's QoS solution was deployed to Intel Haifa site in Israel. A single PacketShaper 4000* device was installed on the link interconnecting the LAN to the WAN. The Packeteer's cyclic QoS provisioning model was followed as defined by four phases: i) network traffic discovery & classification; ii) traffic analysis; iii) traffic shaping & control; iv) traffic behavior reporting.

Initially, network traffic discovery was set in passive mode, allowing the device to discover and classify critical WAN traffic. This phase was run for about a month in two main iterations – first, without manual manipulation, and secondly after differentiating between different link traffic and application types (e.g., web, email, etc).

During the analysis phase we reached device analysis limits due to the large number of WAN traffic connections, and narrowed our focus to key applications such as heavy network consumers, latency sensitive and mission critical applications. This actually proved to be a more effective approach to QoS analysis. For example, on one of the local links we concluded that 90% of the traffic was being consumed by two primary applications: Microsoft Exchange* and a notebook backup utility (code named: CNB). CNB is used in Intel Israel to backup all the notebooks' local drive data.

During the traffic shaping & control phase the active mode was set, and we applied QoS policies to solve the aforementioned problems. One good example for QoS provisioning was the policies applied to the CNB application. The application uses the WAN to backup the data into a central data center located in Haifa from four major remote sites. Once a day, the application randomly tries a backup of the notebook data to the data center. If the backup fails (i.e., the notebook is offline or timeout occurs) the application will retry at a later time. This usually generates more traffic over the WAN link. Two QoS policies were implemented for the CNB applications via PacketShaper*. One policy enforces the overall bandwidth used by CNB to not exceed 25% the of the link bandwidth. The second policy guarantees 100 Kbps for each backup session. The first policy protects the links from being overloaded by the CNB traffic, and further protects all other bandwidth competing applications. The second policy provides the CNB users with a reasonable backup response time. As such, we formulated that only 10 users could perform simultaneous backups with reasonable response time. Other CNB users would receive a timeout when trying to perform the backup, and, as stated, the application would retry at a later time.

Based on our experience in Israel, we formulated three operational modes that could be used to achieve operational benefit from policy-driven QoS technology: a) long term QoS strategy & deployment; b) short-term ad hoc solution; c) troubleshooting & monitoring. The long-term strategy would have QoS policies enforced to meet strategic objectives and would be reviewed twice a year. The short-term approach would be used for deployment of QoS policies to meet short-term ad hoc demands such as managing videoconference meetings or distance learning sessions over the WAN. Finally, the troubleshooting mode of operation would allow deployment of QoS policies to resolve immediate problems on the network such as greedy traffic scenarios.

3 TOWARDS INTERNET SERVICE PROVISIONING

Moving beyond today's slow, vertical, and proprietary networks towards an open platform for rapid and seamless service provisioning will require more sophisticated middleware services to realize it. Policy-based management, as a middleware environment, provides the necessary abstraction glue to formalize Internet service provisioning. While the complexities of today's networks continue to converge between public and private boundaries, greater emphasis will be placed on abstractions to facilitate end-to-end services – service differentiation, bandwidth management, security, user and network administration. While QoS provisioning and security administration are the primary policy deployment models today, the evolving requirement is service provisioning and automation of more complex and bundled network services. Key services, which are helping to transform the next-generation Internet, include VPN, distance learning, VoIP, user collaborative environments, multimedia conferencing and other, complex virtualized services. This translates into further partitioning and management complexity of QoS, security, address space, and network administration. Such complexities will require a myriad of policies that not only support a similar level of abstraction for specific bandwidth services (e.g., QoS), but can also associate with corresponding policies for privacy, authorization, user-specific customization as well as associations with business directives.

Therefore, while the operational management paradigm of today's information technology environments is based on traditional models of systems and network management, moving to a service provisioning orientation within the context of the next-generation Internet will require more sophisticated IT operational infrastructure. We propose the following operational and management requirements to support the integration and advancement of current policy-based network management systems:

> *Open network management.* The migration and long-term sustainment of a policy-driven environment will require well-defined API's and a common directory schema that allow plug-n-play devices and network management tools to co-exist seamlessly. An operationally centric data model (schema) is appropriate to correlate policy actions that match the business rules of the organization to infrastructure policies, to operational availability and performance. Thus, the correlation of policy, event, configuration and usage data is key to comprehensive end-to-end management.

Operational agility and validation. We envision a consolidation of existing processes and tools through the integration of change and problem management with PBM systems. By streamlining processes for change and problem management groups, we will enable data sharing and true interdependence. Thus, we believe the generation of customer resource management databases and tools should speed up and personalize the change/problem management system to provide customers and stakeholders a dynamic view of the relevant data that supports their activities.

Moreover, the shift to PBM will require IT organizations to have the tools and motivation to manage to tighter service levels on service performance, security, reliability, and customized agreements. This will require a more dynamic and granular auditing and reporting function for the underlying services and the managed environment. Additionally, the reporting function must be designed to refer to the negotiated SLA and provide a meaningful report that supplies the customer with validation that services are being delivered as agreed.

Active, secure and dynamic directories. A directory service capable of supporting 'dynamic' data types is useful to policy-based network operations, supporting service provisioning or delivery of network state information to applications. An active association between a user or application context stored in the directory and the network can be discovered and used. New possibilities enabled by this technology follow by: i) providing secure management of information from a variety of sources, including applications and network devices; ii) defining, registering, and providing publish/subscribe features for network events; iii) processing network events for applications, devices, and users; iv) exposing APIs to applications to take advantage of directory-based services; v) maintaining state information for devices, users, and applications to support policy-based management. The core directory service acts as the single point of administration for all resources, including users, files, peripheral devices, databases, Web access, and other objects.

Ubiquitous network security. The requirements for policy-based security focus on the integration of policies across administratively separate or heterogeneous domain boundaries. Security has to allow individual consumers, corporate suppliers, and acquisitions/mergers as well as corporate business partners to operate under a shared, and what is perceived to be, borderless infrastructure. End-end security will require tighter integration of policies across separate security realms, which are traditionally disjointed across networks, applications, and servers. Finally, dynamic policies will be a requirement for the administrator, providing him/her the means to perform on-the-fly control or automation of short-lived policies to secure content and communications (e.g., inter-company video conference or online supplier training sessions). Enabling such a paradigm will require a higher level of abstraction, automation, and integration across infrastructure elements. Policy-based management solutions would work here by aiding in the definition of allowable security associations, integrating policy abstractions across administrative or heterogeneous security boundaries, facilitating encryption parameterization, and by rapid and dynamic configuration of boundary devices (e.g., VPN, firewall, and proxy services).

4 CONCLUSION

In this paper, we presented results and findings from Intel's Information Technology (IT) research and development trials supporting policy-based management. We presented our operational findings, internal prototype developments on the PBM Operational Console and the Network Traffic Controller capability; and finally, presented a case study of QoS deployment. We proposed extending existing policy-based network management systems towards Internet service provisioning and presented key operational and management requirements necessary to achieve this motivation.