# Harnessing Information Models and Ontologies for Policy Conflict Analysis

**Steven Davy B.A. (Mod.)**

**Supervisors: Dr. Brendan Jennings, Prof. John Strassner**

In partial fulfilment of the requirements for the award of

*Doctor of Philosophy*

Department of Computing, Mathematics and Physics

Waterford Institute of Technology
Ireland

Submitted to Waterford Institute of Technology, September 2008

# Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Doctor of Philosophy is entirely my own work and has not been taken from the work of others save to the extent that such work has been cited and acknowledged within the text of my work.

Signed:...................................................... ID: 20004309

Date: September 2008

# Harnessing Information Models and Ontologies for Policy Conflict Analysis

**Steven Davy B.A.(Mod.)**

**Supervisors: Dr. Brendan Jennings, Prof. John Strassner**

## Abstract

Policies are meant to govern the behaviour of the communications network. Unfortunately, most policy-based management systems do not contain conflict analysis processes. Therefore, if policy conflicts occur between deployed policies, there is no guarantee that the intended behaviour will be realised by the communications network. In addition, most policy-based management systems are designed to be used by a single constituency, so do not address the needs of different stakeholders that use the system in different ways. This thesis presents an approach to policy authoring that incorporates policy conflict analysis for communications networks. This approach is based on harnessing knowledge embodied in information models and ontologies to represent relationships between policy components that could indicate potential conflicts between policies. Harnessing the application information embodied in knowledge bases makes the approach independent from the application or use of policies because the application specific concerns are separated and thus can be designed independently. A model of a policy continuum is presented, which is used to maintain the relationships between different levels of policies used by different stakeholders. A policy authoring process is then developed to formally describe the interaction between stakeholders concerned with creating and modifying policies and the detection of policy conflicts. An efficient method of selecting appropriate subsets of policies that may conflict is developed. Experimental results show that this process results in a significant reduction in the number of policies that needed to be analysed.

## Acknowledgments

I would like to acknowledge my supervisory team of Dr. Brendan Jennings and Prof. John Strassner for their excellent advice and guidance throughout my time as a student. Their belief in my work and in my ability to complete this immense task is one of the major contributing factors to its final submission. I would like to also thank the broader research support available to me at Waterford Institute of Technology, especially Mícheál Ó Fóghlú, Dr. Willie Donnelly and Eamonn deLeaster who encouraged an atmosphere of research in WIT that inspired me to persue and complete this task. All the while, I was instilled with the comfort of knowing that they and their experience were always at hand during times of difficulty.

I would like to thank Dr. Sven van der Meer, Dr. Sasitharan Balasubramaniam and Dr. Dmitri Botvich for their advice, encourgement and support during my time as a student. They helped me consider new ways of addressing challenges in my research field. A portion of the systems and testbeds developed within this thesis would were carried out in coordination with Keara Barrett and Elyes Lehtihet, two former collegueas from WIT. Their technical knowledge of Policy-based Management (Keara) and Ontologies (Elyes) were of great help throughout each stage of the thesis development.

I would like to thank my fiancé Sinéad for her support and encourgement during this time consuming task. Last, but not least, to my family; my parents Irene and Cormac always encouraged me to try my best and my brothers understood I needed to unwind from time to time. My brother Alan provided great feedback and comments on my work and listened to its inner details for hours on end, as he too was going through the same process.

# Contents

# List of Figures

# List of Tables

# List of Algorithms

# Chapter 1

# Introduction

This thesis presents processes and algorithms to support the authoring and, in particular, conflict analysis of policies deployed as a policy continuum in an integrated and efficient way. Central to the approach is the harnessing of knowledge embodied in information models and associated ontologies to support the policy authoring and analysis processes.

Policy-based network management (PBNM) techniques are based on describing the intended configuration of a communications network in the form of rules termed "policies" (Strassner, 2003). The primary motivation for PBNM is that policies reduce the complexity associated with configuring a heterogeneous communications network. PBNM systems incorporate processes to support the configuration and monitoring of the communications network to enforce the behaviour defined by policies. Such processes include policy authoring, policy conflict detection and resolution, policy transformation and policy refinement.

A further step for PBNM systems is to enable high-level business policies to describe the intended behaviour of a communications network expressed in terms familiar to business users. To facilitate this functionality, algorithms are required to ensure deployed low-level network policies realise the high-level business policies. Realising business policies on a communications network not only requires the collaboration of separate constituencies of experts to define the policies, but also requires the configuration and monitoring of different sets of software and hardware resources. This business to network realisation of a PBNM has proven to be a difficult challenge (Jude, 2001; Chadha, 2006), due to the lack of integration between the processes and algorithms required for authoring and conflict analysis.

Policy conflicts occur as a consequence of new or modified policies being deployed into a PBNM system that leads the system to exhibit unintended behaviour. The enforcement of policies (business or network) should be conflict free, otherwise the intended behaviour defined by policies cannot be guaranteed (Moffett and Sloman, 1993). Current conflict analysis approaches are tightly coupled to the policy applications for which they were designed, are not integrated in other policy based management processes such as policy refinement and are not designed to cater for large numbers of policy sets.

A current bottleneck in the configuration of large scale communications networks is the manual alignment of network policies to business policies. There is a natural link between the policies relating to business concerns and the policies relating to system or network concerns; this link has been captured in the *policy continuum* as levels of associated policies (Strassner, 2003). The use of the policy continuum provides a method of associating business policies with network policies in an integrated way. The novelty of the approach presented by Strassner (2003) is the use of an information model of the related managed entities of an organisation to support the processes required by the policy continuum. However, the processes and algorithms required are only outlined by (Strassner, 2003) and are not formally specified. A formal specification of the policy continuum is required so that processes and algorithms can be developed to describe policy authoring and policy conflict analysis. The policy continuum introduces new complexities when associated hierarchies of policies are considered. As the policy continuum requires the integrated efforts of separate constituencies of systems experts to define policies, there must be a process to cater for the authoring of and detection of conflicts between the associated multiple levels of policies. As the policies may be defined at different levels of the organisation (i.e. business to network), the conflict analysis process must be able to accommodate the relationships between the policies defined at the same or different levels of abstraction.

There are challenges to realising an authoring process and conflict analysis process for policies deployed as a policy continuum. Specifically, there are complicated relationships between policies defined at different levels; when modifications are made a strict process should be followed that interacts with the multiple policy authors to maintain the consistency of the policy continuum. There is a complexity issue associated to selecting appropraite sets of policies that must be analysed for conflict at each policy continuum level that needs to be addresses, otherwise solutions will not be efficient.

## 1.1 Hypothesis

The hypothesis of this thesis is derived from an extensive examination of the literature covering the state of the art in policy based management processes, in particular looking at policy authoring and policy conflict analysis. It is the primary objective of this thesis to prove the validity of the following hypothesis.

*By leveraging semantic models (information models and ontologies) that have been enriched with application specific information, policy conflict analysis and policy selection processes, that are policy application agnostic, can be developed for use specifically with policies deployed as a policy continuum.*

The literature review that is presented in chapter 2 yields the following key contributing factors to the hypothesis:

- Language based policy conflict analysis is usable only in situations where the intended application of policies is well defined. However, such approaches do not make use of application information (be it structural or semantic) to make the analysis process more efficient, leading to difficulties when larger repositories of policies need to be analysed.

- Information model based policy conflict analysis takes into account application information (structural) in order to enable policy conflict analysis processes that are somewhat agnostic of the application for which policies are defined. This approach is not flexible in situations where the policy language or application constraints need to be altered. However, a larger variety of conflict types may be detected in comparison to language based analysis.

- Ontology based policy conflict analysis is the most flexile type of policy analysis, where application information, both structural and semantic, is represented separate to the policy language. Therefore, taking advantage of existing knowledge bases (information model and ontologies) can yield a more efficient policy conflict analysis process. Such processes are required when the policy language is substantially more complex, as is the case when a policy continuum is being analysed.

- There is no existing policy authoring process that ties policy conflict analysis to the multiple levels of the policy continuum.

## 1.2 Research Questions

The research questions addressed by this thesis are listed below. They highlight the challenges faced when designing processes and algorithms for the authoring and conflict analysis of policies within the policy continuum.

1. *How can a policy authoring process be defined that incorporates policy conflict analysis and that is specifically targeted at multiple constituencies of policy authors?*

   Multiple policy authors can create and modify policies at any level of the policy continuum. Policy conflicts can be manifested at any level of the policy continuum, where higher level policies as well as lower level policies can lead to the cause of a policy conflict. The relevant policy authors need to be alerted in a coordinated fashion so that the conflicts can be dealt with appropriately.

2. *What processes and algorithms need to be developed so that existing knowledge bases can be harnessed to aid in policy conflict analysis?*

   Knowledge embodied in information models and ontologies can describe in great detail the semantics and constraints associated to the management of a communications network. This thesis is interested in harnessing this knowledge automatically, to aid in policy conflict analysis. The reason for this is that with more knowledge of the system, more informed decisions can be made in analysing for policy conflicts.

3. *How can a policy conflict analysis process be developed that is independent of the nature (or purpose) of the policies?*

   Policies can be defined at multiple levels of the policy continuum, thus being described in different languages for application, services and resources of differing requirements. A policy conflict analysis process developed to analyse policies for a policy continuum has to be applicable to the range of different forms of policies that will be represented. Therefore, the process should be independent from the nature of the policies and thus can be used to analyse policies at multiple levels of the policy continuum.

4. *How can processes and algorithms developed for policy authoring and policy conflict analysis be developed so that they are made efficient when increasing numbers of policies are being considered?*

When large sets of policies require analysis, the algorithm for analysing for potential conflicts has to be efficient. This is an important criteria for and policy based management processes.

## 1.3 Main Contributions

This thesis contributes to the research area of policy based network management. Specifically, it addresses the development of the policy authoring and conflict analysis processes. The main contributions are:

- A formal policy continuum model that defines the relationships that exist between policies at different levels of abstraction, incorporating:
  - A policy authoring process, which is an iterative process that informs multiple policy authors about the effects of modifying policies at a given level of the policy continuum (Davy et al., 2007).
  - A discussion of how other policy analysis processes interact with policy conflict analysis to deliver an effective policy authoring process (Davy et al., 2008*c*).
  - Specific extensions to the existing DEN-ng policy information model as detailed originally by Strassner (2003), to facilitate the incorporation of policy conflict definitions and policy continuum concepts (Strassner et al., 2007*a*).
- An approach to policy conflict analysis for the policy continuum that makes extensive use of an information model of a communications network and associated sets of ontologies, incorporating:
  - An information model oriented policy conflict analysis algorithm that is specified for use with the policy continuum. The nature of the algorithm promotes the re-use of previous conflict analysis. Historical information is leveraged to improve performance for large scale policy deployments (Davy et al., 2008*a*).
  - A policy selection algorithm to be used with the policy conflict analysis algorithm that is capable of leveraging the information model to support the efficient selection of sets of deployed policies for analysis (Davy et al., 2008*b*).
  - An approach that leverages ontologies that are associated to the information model to aid in determining policy conflict that would otherwise have been difficult to detect, (Davy et al., 2008*d*).

## 1.4 Main Conclusions

This thesis presents processes and algorithms to support the authoring and, in particular, conflict analysis, of policies deployed as a policy continuum in an integrated and efficient way. The algorithms are integrated into the policy authoring process of the policy continuum and query an information model and associated ontology. The main conclusions of the thesis are:

- As there is neither a formal definition of the policy continuum nor a defined methodology to incorporate the effect a policy continuum has on policy authoring and conflict analysis, current policy based management systems find it cumbersome to enforce policies defined for a policy continuum.

- The formal policy continuum model and associated authoring process describe the interaction and dependency between policy refinement and policy conflict analysis, which has not been formally described before. This demonstrates that policy refinement and policy conflict analysis can be developed independently but that they both must be used in coordination with a policy authoring process.

- A policy conflict analysis algorithm that makes use of the application specific information encoded in an information model is able to cater for different conflict types if the definition of what constitutes a conflict is defined in the information model. The conflict analysis algorithm can be used to detect policy conflict at any level of the policy continuum due to the decoupling of the type of policies being analysed from the algorithm.

- The policy conflict analysis algorithm can be readily augmented to deal with new types of policies and types of relationships between policies. Furthermore, by describing the relationships between two policies in an ontology, significant improvements to the conflict analysis algorithm can be made in respect to policy selection.

- The policy relationships discovered during the conflict analysis algorithm can be saved as histories and readily re-used to improve the selection of policies that must be further analysed for conflict. By leveraging historical information, significant improvements in the performance of the conflict analysis algorithm are observed.

## 1.5 Thesis Outline

**Chapter 2** presents the background work related to network management and policy based management. It discusses the related work on policy conflict definition and analysis approaches concluding with an identification of the related open research issues and challenges concerning policy conflict analysis, together with a set of requirements for the work described in the remainder of the thesis.

**Chapter 3** presents the proposed formalisation of the policy continuum model and associated policy authoring process. The authoring process is designed to cater for the requirements of realising a policy continuum and maintaining the consistency of the policy continuum as policies are created or modified.

**Chapter 4** presents the policy conflict analysis algorithm that is a part of the policy authoring process. The conflict analysis algorithm is linked to and leverages an information model that describes a communications network. A crucial result is that by taking this approach, the algorithm can be used to detect potential conflict among policies defined at any individual policy continuum level. An implementation is described that demonstrates several case studies depicting the analysis of policy conflict at multiple levels of the policy continuum.

**Chapter 5** builds on the work of the previous chapter to address a challenge associated with physically distributed groups of policies. By incorporating the use of ontologies into the information model to further describe the relationships that can be established between policies, the conflict analysis algorithm can detect conflicts between distributed sets of policies defined to manage different devices having different functionality.

**Chapter 6** presents a policy selection algorithm that can be used with the policy conflict analysis algorithm to enable it to take advantage of historical information to enhance its performance. The policy selection algorithm promotes the re-use of previous computations during policy conflict analysis. The results are presented for theoretical and case study based policy repositories. The result indicate the potential savings in runtime complexity the selection algorithm offers.

**Chapter 7** summarises the contributions of this thesis and discusses future work. It also outlines future challenges associated with, in particular, the interdomain management aspects of using a policy continuum.

# Chapter 2

# Background and Literature Review

This chapter is comprised of three related portions. Firstly, the background work discusses network management approaches and the use of information modelling in network and distributed systems management in particular. This motivates the development of the policy continuum model in this thesis, indicating the complexity associated with communications network management. Policy based management is defined along with the challenges associated with the specification and enforcement of policy in communications networks. The use of other technologies such as ontological engineering is reviewed to investigate its current use within policy based management.

Secondly, a literature review of the work relating to policy based management and policy conflict analysis in particular is presented. The various definitions of policy conflict presented in the literature are reviewed. This provides a context for presenting current research on the processes developed to analyse policy. The different approaches to policy conflict analysis are categorised into language based, information model based and ontology based; an investigation into each approach reveals their respective advantages and disadvantages. Thirdly, conclusions are drawn from the literature review and the requirements of the approach are outlined.

## 2.1 Background

This section reviews the background of network management towards motivating the research and development of policy based network management. A brief introduction to communications network management is provided, highlighting the standard terminology and concepts used. Policy based management is described and reviewed from the perspective of its use in addressing the complexity associated with the management of large scale and dynamic communications networks. The Directory Enabled Network new generation (DEN-ng) policy information model (Strassner, 2003) is introduced to represent the state of the art in policy based management for networks. The use of not only structural information via information models, but semantic information via ontologies is also studied in the context of network management to support its use in policy based network management.

### 2.1.1 Network Management Approaches

According to Hegering et al. (1999): "The management of networked systems comprises all the measures necessary to ensure the effective and efficient operation of a system and its resources pursuant to an organisation's goals". Communications networks are complex and heterogeneous combinations of resources, services and applications and therefore must be managed in an integrated and coordinated fashion. The management of network resources concerns managing the physical routers, hubs, switches and other devices supporting network communications. Network management is also concerned with managing the distributed systems and services that run over the communications network, such as network operating systems, file systems and web servers.

The tasks of management can be dissected into five general areas as suggested by the Open Systems Interconnection (OSI) group's management architecture; standardised by the International Standards Organisation (ISO). These five general areas, typically referred to as FCAPS, are fault management, configuration management, accounting management, performance management and security management. These areas of management indicate what functionality must be provided by the management system. With such diverse responsibilities from the network management system, there is a requirement for a significant investment into systems that can coordinate the execution and monitoring of all the management tasks.

Management architectures have been developed to integrate the various management tasks in the form of distributed systems. Such tasks may typically be implemented on a vendor specific basis and may have defined unique interfaces to configure and monitor the associated entities. Vendor specific interfaces reduce the ability of a management system to be integrated, because a separate interface to perform all the management tasks would be required. A management architecture provides the basis for different vendors to develop interoperable management solutions largely independent of one another.

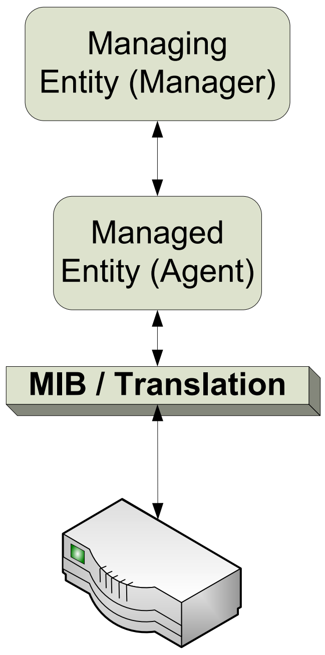


Figure 2.1: Manager-agent concept.

A managed object is a concept used in many management systems to abstract from the entity to be managed. The managed object is a management representation of an entity that can be managed by the management architecture. The managed object can therefore be a physical resource, a network service, a user account or even a domain structure. The key concept that must be realised is that the interface to the managed object is standardised and provides an abstraction above any vendor specific interface. Therefore, the managed object concept is critical to realising the management of heterogeneous entities in a communications network. To support the definition of these managed objects an information model is required. The information model is the heart of any management architecture as it defines the interfaces, behaviour and capabilities of all manageable entities in a network.

The proper use of managed objects requires that the “manager-agent” paradigm is followed, see figure 2.1. The manager is a piece of software that knows how to communicate management task related commands to an agent. The agent then knows how to perform the specified management task on the target resource and this is performed through a standards based interface.

The management architecture also requires an organisational model that is capable of representing the structure and relationships between entities that are managed. The organisational model must be able to represent roles and hierarchies of entities to reflect the nature of the organisation. In fact, policies are a natural method of representing an organisational model according to Moffett and Sloman (1991).

Management architectures also require communication models and functional models. The communication model defines the interaction between managed objects and the functional model dissects the management tasks into functional areas as described typically by FCAPS. From the perspective of this thesis the most important concepts are the potential uses of the information model to aid in policy based management. Although the other models of the management architecture are important, they are not considered as a method of enhancing policy based management processes. For this reason, examples of deployed management architectures are discussed from the perspective of their use of information models.

There are many examples of modern management architectures that share the concepts of information models. For example, the Simple Network Management Protocol (SNMP) management architecture was the most common method of management found on the Internet according to (Hegering et al., 1999), and is still widely supported. The latest version of SNMP was published in RFC 3411. The information model concepts for SNMP are based on the Structure of Management Information (SMI) and the Management Information Base (MIB) group of standards developed by the IETF. The manager-agent paradigm is followed to coordinate the management of networked services and resources. Currently, SNMP is typically used only for monitoring managed objects and not widely used for configuration management.

The OSI provide a description of a telecommunications network management architecture and a realisation of this architecture can be observed in the Telecommunications Management Network (TMN). The TMN is used to manage public networks and telecommunications networks of a large scale. The information model for the OSI management architecture apply an object-oriented approach. The OSI information model is an example of the importance of standardised structural information of a network for its use in management. The TMN adopted the OSI management architecture and standardised its use in the International Telecommunication Union, Telecommunication Standardisation Sector (ITU-T) specifically in the ITU-TM3xxx group of standards.

The Common Information Model (CIM) (DMTF, 2008) was developed by the Distributed Management Task Force (DMTF) as an information model to aid in the integrated management of desktop systems. The CIM uses an graphical and textual notation to describe the information model. The graphical notation illustrates the managed objects and relationships between managed objects. The graphical notation is based on the Unified Modelling Language but is more correctly referred to as the Meta-Object Facility (MOF).

The use of information models as a method of abstracting from the heterogeneity of communications networks is used in many management architectures as indicated in this section. However, there still remains a form of heterogeneity in the way the network resources and services are configured to realise business objectives. This point is addressed by policy based network management and is covered in the next section.

### 2.1.2 Policy Based Network Management

Policy based network management (PBNM) is a management paradigm that separates the rules governing the behaviour of a network from its functionality. Policy was originally described for access control of sensitive data as detailed by Bell and LaPadula (1973). It was in (Moffett and Sloman, 1991, 1993), that policies were first described for use in network management.

The use of policy in network management was attractive because it offered a solution to the management problem of large and heterogeneous networks (Strassner, 2003). To provide a clear definition and description of policy based management the concepts of a policy based management system and specification and enforcement of policy are dealt with in the next sections.

#### 2.1.2.1 Definition of Policy Based Management

According to the IETF policy terminology RFC (RFC 3198), and as described further by Strassner (2003), a policy based management (PBM) system controls the state of the system and objects within the system using policies. The control paradigm used is based on finite state automata. The PBM system is concerned with the installation, deleting and monitoring of policy rules, as well as ensuring the system operates in accordance to those policy rules. Policy based network management (PBNM) is PBM as applied to network management. In this case the managed objects are networking devices, resources and services. Ultimately, the PBNM system is used to control the provision of services across a network in a predictable way.

There are a number of terms associated with policy based management that are described in RFC 3198 and by Strassner (2003). The terms can be split into two portions, namely terms that describe a policy and terms that describe the management of those policies. The following descriptions are derived from Strassner (2003) and there is some overlap with the definitions defined in RFC 3198.

**Policy:** Policy is a set of rules that are used to manage and control the changing and/or maintaining of the state of one or more managed objects.

**Policy Rule:** A PolicyRule is an intelligent container. It contains data that define how the PolicyRule is used in a managed environment as well as a specification of behaviour that dictates how managed entities that it applies to will interact.

**Policy Group:** A PolicyGroup is a container that can aggregate PolicyRule and/or PolicyGroup objects.

**Policy Condition:** A PolicyCondition is represented as a Boolean expression and defines the necessary state and/or prerequisites that define whether the actions aggregated by the PolicyRule should be performed.

**Policy Action:** A PolicyAction represents the necessary actions that should be performed if the PolicyCondition clause evaluates to TRUE.

**Policy Event:** A PolicyEvent represents an occurrence of a specific event of interest to that managed system and can be used to trigger the evaluation of a PolicyRule.

The specific terms to describe the components used to manage the operation of a PBM are as follows:

**Policy Decision Point:** An entity that makes Policy Decisions for itself or for other entities that request such decisions.

**Policy Execution Point:** An entity that is used to verify that a prescribed set of PolicyActions have been successfully executed on a set of PolicyTargets.

**Policy Domain:** A PolicyDomain is a collection of managed entities that are operated on using a set of policies.

**Policy Repostory:** A PolicyRepository is an administratively defined logical container that is used to hold policy information.

**Policy Subject:** A PolicySubject is a set of entities that is the focus of the policy. The subject can make policy decision and information requests, and it can direct policies to be enforced at a set of policy targets.

**Policy Target:** A PolicyTarget is a set of entities that a set of policies will be applied to. The objective of applying policy is to either maintain the current state of the policy target or to transition the policy target to a new state.

The above specified definitions shall be referred to throughout the rest of the thesis unless the definitions are explicitly modified within the text.

#### 2.1.2.2 Specification and Enforcement

In the early 1990s PBNM was to revolutionise the way Quality of Service (QoS) would be provisioned across communications networks. QoS-enabled networks aimed to deliver differentiated services which were driven by business oriented goals and objectives (i.e. policies) (Jude, 2001). Research soon indicated that this objective proved difficult to achieve. The primary reasons being that there was little understanding on how policy should be specified and that the defined policies were difficult to enforced.

The original solutions developed to address QoS management via policy were predominately vendor specific and were not interoperable across large scale communications networks. Therefore, even if the management protocols were based on a management model that promoted interoperability (for example SNMP), the policy languages used to define the behaviour and execution environments used to enforce the policies, were not interoperable. This lack of interoperability between policy implementations occurred because there was no standard interpretation of what policy was aiming to achieve. In a move to increase interoperability, in RFC 3060 the IETF published the Policy Core Information Model (PCIM) following from the success that information models had on interoperability for network management. The PCIM is concerned with the specification of policy and not its enforcement.

Figure 2.2 depicts the primary components of the PCIM. The model describes the structure of policies but does not dictate the contents of policies. The semantics of a PCIM policy is defined to be “If a set of PolicyConditions are satisfied, then execute the appropriate set of PolicyActions”. Once the policy condition is evaluated to true a set of policy actions are executed.

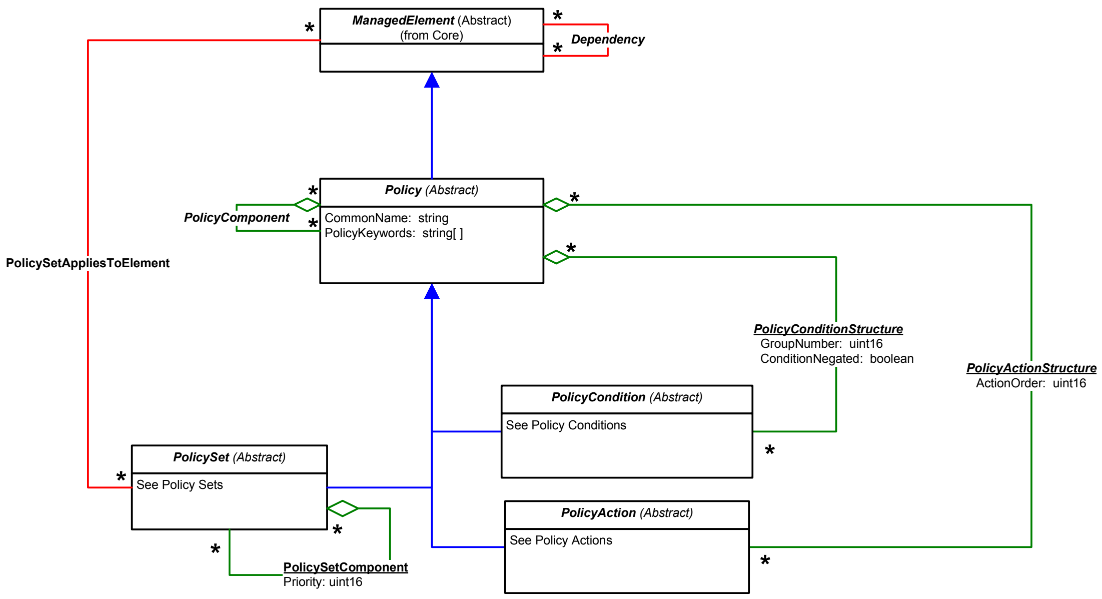


Figure 2.2: Simplified view of the PCIM (DMTF, 2008).

The PCIM is defined to be generic enough so that it can be readily extended to meet the requirements of any policy application. It was further extended by the IETF to meet the requirements of representing policy to manage quality of service and security. RFC3644, Policy Quality of Service Information Model (QPIM) (Snir et al., 2003), dictates how policy can describe the enforcement of differentiated IP services (DiffServ) and integrated IP services (IntServ). RFC3585, IPsec Configuration Policy Information Model (Jason et al., 2003), dictates how policy can describe the enforcement of IPsec security services. The execution semantics of such policies are well understood, in that the specified conditions are defined over IP header information of received IP traffic on a router interface and the specified actions either alter information in the packet header and/or control the processing of the packet.

Policy based management principles are equally applicable to domains other then network management. Policy for access control is concerned with defining the actions that should follow a request for access to a resource by some entity. The result may be to allow or deny access. Conceptually, the principle of policy should apply as the conditions of the policy describe the access rights of those entities that are allowed / denied access, and if the requesting entity meets the conditions, the corresponding actions are performed. There are many existing models for access control but a popular standard model for access control was developed by the National Institute of Standards and Technology (NIST), namely role based access control or RBAC (Ferraiolo et al., 2001). RBAC specifies that policies can be defined to manage access to resources in a generic way. The model they define is more specific to access control then that proposed by the IETF in the PCIM, as it also deals with concepts such as roles, users and groups. This model introduced new possibilities and associated complexities into the specification and enforcement of policy for access control. The concept of separation of duty and delegation could now be represented which was difficult to implement with the existing PCIM.

For distributed systems, a general policy based management toolkit, named Ponder (Damianou et al., 2001), mainly used for research was developed at Imperial College. The tools comprises of a policy specification language roughly based on the IETF PCIM, but including concepts of access control and a set of management processes that can perform some analysis of policies and enforce them. The tool was not directly aimed at network management but towards managing access to and operation of services in a distributed system. As Ponder combined two different aspects of policy it enabled policy administrators to specify different types of policies. Policies could now be directly related to groups of managed entities, and were subject or target based. In Ponder, *Positive Authorisation* (A+) policies dictate under what conditions could a subject perform an operation on a target. *Negative Authorisation* (A-) policies dictate the conditions when a subject could not perform an operation on a target. There are also positive and negative *Obligation* (O+/-) policies; these policies dictate when a subject must or must not perform a specific operation on a target. Where authorisation policies were akin to access control, obligation policies instruct the execution of operations (policy actions), similar to the original definition of policy by the IETF.

Recent technologies for policy based management relate to managing access to Web Services, such as XACML[1] Godik et al. and WS-Policy[2] (Bajaj et al., 2006). These technologies were developed to follow an access control paradigm. The specification languages are based on XML, and the enforcement of the policies follows the associated published standards so that multiple vendors can implement their own enforcement processes. The "Rei" (Kagal et al., 2003) and "KAoS" (Uszok et al., 2003) policy languages followed the avenue of Ponder representing access control concepts in policy which required the incor-

[1]eXtensible Access Control Markup Language
[2]Web Service Policy

poration of policy analysis. Their domain of application is that of pervasive environments and agent based systems, respectively.

With varying types of policies being specified, the enforcement of the policies also varies. High level access control policies are enforced by network attached servers, dedicated to processing access control requests, whereas low level network filtering policies would be enforced on the actual router interfaces spread throughout the network. These low level policies would essentially be evaluated on the arrival of each IP packet. The differing execution semantics of policy specified at different levels for different applications, suggests that there is no single view of policy. The next section discusses the architectures developed to support policy based management.

#### 2.1.2.3 Policy Architectures

The enforcement architecture for policy based management was standardised by the IETF in RFC2748, Common Open Policy Service (COPS) (Boyle et al., 2000). The designated architecture is depicted in figure 2.3. COPS is designed to be used with the PCIM to implement the decisions of the policies defined using PCIM and extended models. It dictates the use of a Policy Decision Point (PDP) and a Policy Enforcement Point (PEP). The decisions are the result of evaluating policy conditions. The enforcement of a decision is basically the execution of the associated policy actions. The COPS protocol is used to communicate policy decisions between the PDP and the PEP. However, there is a large portion of the architecture missing as this RFC only describes the communication between two of the core components and is not concerned with the interaction with policy authors creating, modifying or deleting policies.

The architecture suggested in the COPS RFC is built upon the definitions used to describe policy in the related RFCs, RFC 3198 (Terminology for Policy-Based Management) (Westerinen et al., 2001) and RFC 2753 (A Framework for Policy-based Admission Control) (Yavatkar et al., 2000). The descriptions are application independent and can therefore be adapted for use in many applications. For example, in an access control application that may be based on XACML, the PEP may be located separately from the PDP. The PDP in this case is the XACML server, and the PEP is a software component capable of communicating XACML requests to the server and subsequently allowing access to a managed resource. For a network filtering application, the PEP and PDP are hosted locally, where all decision logic is available to the filtering process and the filtering process

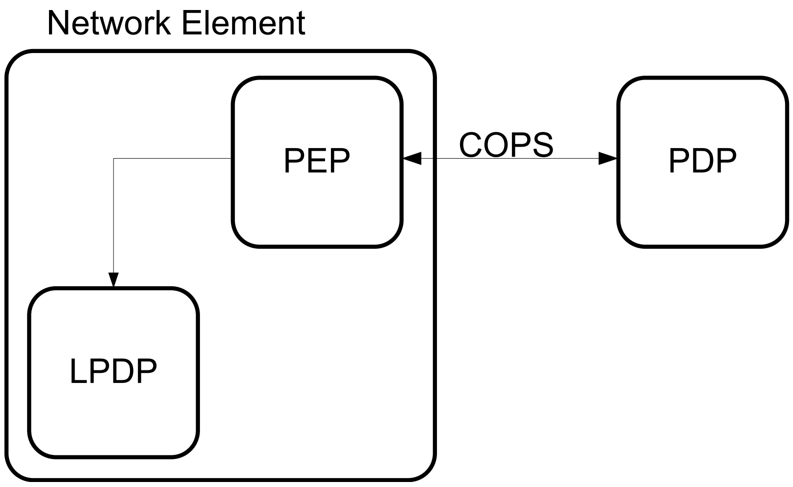


Figure 2.3: COPS architecture (Boyle et al., 2000).

is capable of carrying out the decisions. Both scenarios abide by the description of a policy based management system as detailed in the associated RFCs, but they have very different execution semantics, demonstrating the flexibility of the architecture. These two types of policy based management systems depict the distinction between the two approaches to implementing a PBM: distributed versus embedded.

In the distributed PBM, there is typically a policy server hosting the decision processes and the individual policy descriptions. Compatible services can then query the policy server for answers to policy related questions. The policy server can instruct the provision of configurations that amend the behaviour of services to perform in accordance with policy. The embedded PBM, such as used for network filtering services, stores policies locally and can make individual decisions based on these policies. The two approaches demonstrate the separation of high level policy from low level policy. This separation complicates the use of policy as a holistic management paradigm as it requires the coordination of policies among separate PBM systems. The processes required for such coordination have not been standardised. This thesis aims to contribute substantially in this domain.

Current network management research is investigating the use of policy based management from an autonomic network management perspective, as presented by Jennings et al. (2007). Policy based management is seen to be an important facilitator for autonomic network management due to its concept of separating the intended behaviour of a communications network from its functionality.

### 2.1.3 Information Models

Information models are a successful method of reducing the impact of managing heterogeneous networking resources. The Directory Enabled Networks (DEN) initiative (Strassner, 1999), enhanced the Common Information Model (CIM) with specific extensions so that it could be used specifically to manage networked services and resources. Also, DEN was designed with a data model to persist and store the objects defined within the information model. The data model was a distributed directory using Lightweight Directory Access Protocol (LDAP) (Howes and Smith, 1995). This directory provided applications and networking devices with a common interface to access management information. The problem now was not the management of network resources, but the management of associated applications and services that used the network.

The business of providing high value services over a network, including the Internet, was growing increasingly complex and existing management technologies were finding it difficult to cope with the requirements of managing a network to fulfill business objectives. Operation Support Systems (OSS) are systems developed to cater for the complex management of such networks, but were predominately a mash up of different "best of breed" applications to support various management functions. The cost involved in managing such large scale networks began to increase and was putting strains on the ability of businesses to turn profits. In an aim to reduce the associated complexity of OSS management systems, New Generation Operations Systems and Software (NGOSS) was developed by the TMForum (TMF) (TMForum, 2003).

The objective of NGOSS was to develop a set of information models and associated process models that would aid in establishing a better operations support system for the telecommunication industry. The approach NGOSS took was to integrate an information model that described all the applications and network resources that had to be managed, combined with a guide book for related business processes. The information model was called the Shared Information and Data Model (SID) in (TMForum) and the business process guidebook was called enhanced Telecom Operations Map (eTom) in TMForum. Together the SID and eTom defined the business processes and management information required to develop business aware and effective OSSs. The SID is separated out into a group of related information models each, defined for separate aspects of a business, as depicted in figure 2.4.

The SID is a standardised subset of another important information model, namely the

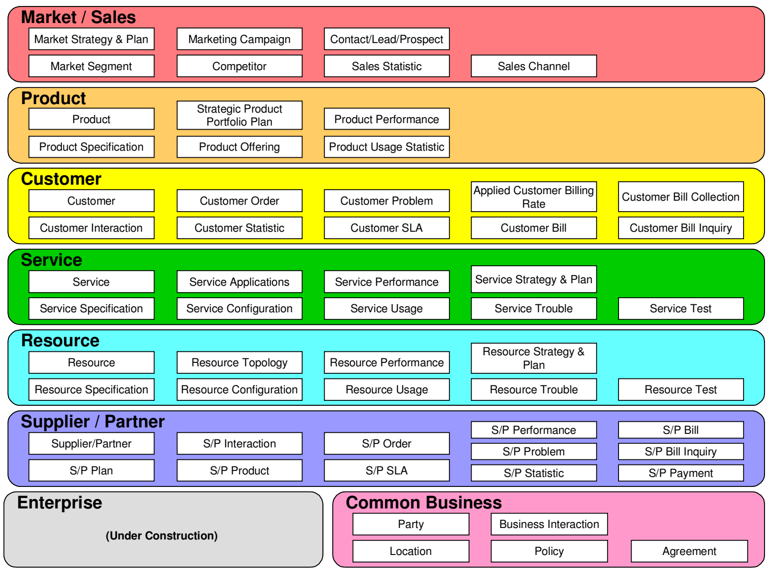


Figure 2.4: SID diagram (TMForum, 2008).

DEN-ng (new generation). DEN-ng was originally developed by John Strassner, as documented in (Strassner, 2003), and was submitted and adopted by the TMF and called the SID. The SID and DEN-ng have diverged in recent years with the DEN-ng focusing more in communications network management, and the SID on the business oriented aspects of management.

DEN-ng was designed to represent the management information of large scale communications networks, from a business through to implementation perspective. Not only are the relationships among networking components and network services represented, but it also represents business concepts and relationships. For example, the DEN-ng can represent information pertaining to a customer of an ISP service and its billing information and can also represent the services and resources required to enable the provision of the service within a communications network.

The DEN-ng is fully UML compliant and so can be used to automate the generation of management interfaces (via Model Driven Development (MDD)/ Model Driven Architecture (MDA)) to the services and resources of the network, and therefore can be made compatible with all existing management technologies. The primary enhancement DEN-ng has over all previous information models is that it did not just represent the structure of management information, it also defines a management methodology that instructed how the management of a large scale communications network should be carried out. Policy

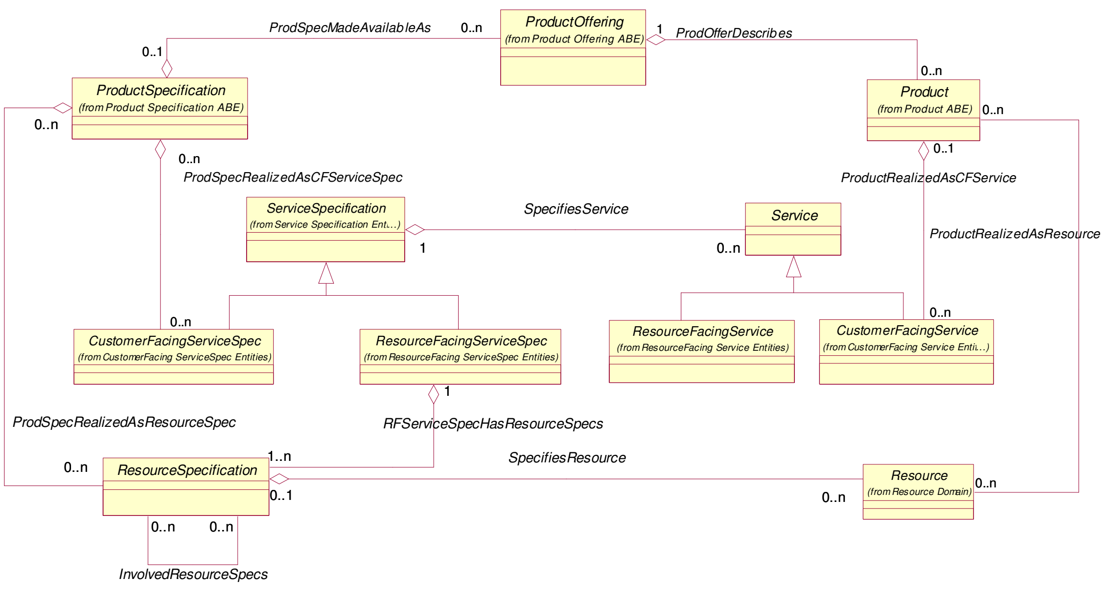

ProductOffering (from Product Offering ABE)
ProdSpecMadeAvailableAs
ProdOfferDescribes
ProductSpecification (from Product Specification ABE)
Product (from Product ABE)
ProdSpecRealizedAsCFServiceSpec
ProductRealizedAsCFService
ProductRealizedAsResource
ServiceSpecification (from Service Specification Ent...)
SpecifiesService
Service
CustomerFacingServiceSpec (from CustomerFacing ServiceSpec Entities)
ResourceFacingServiceSpec (from ResourceFacing ServiceSpec Entities)
ResourceFacingService (from ResourceFacing Service Entities)
CustomerFacingService (from CustomerFacing Service Enti...)
ProdSpecRealizedAsResourceSpec
RFServiceSpecHasResourceSpecs
SpecifiesResource
ResourceSpecification
Resource (from Resource Domain)
InvolvedResourceSpecs

Figure 2.5: DEN-ng product-service (TMForum, 2008).

based network management driven by an information model is one of the key differentiating factors that sets DEN-ng apart from other information models.

The components of the DEN-ng information model describe the relationships and attributes of common business to networking entities. It describes concepts related to products, services, customers down to routing protocols and QoS services. A typical information model diagram is depicted in figure 2.5 and illustrates the relationship between a product that can be purchased by a customer from an ISP, to a set of services that must realise that product in the communications network. The figure illustrates that a Product is made up of CustomerFacingServices or Resources. The associated specification classes describe the invariant characteristics associated to other classes.

### 2.1.4 DEN-ng Policy Information Model

The DEN-ng information model was designed with a policy information model built in. The reasoning behind this is that policy could be related to business managed entities down to network managed entities. As the DEN-ng information model can represent an array of different application domains, not just for network management, policy can be developed to manage these applications. The approach to interoperability among policy languages is the flexibility of the DEN-ng policy model in representing various types of policies. The primary distinction that differentiates the DEN-ng policy model from the

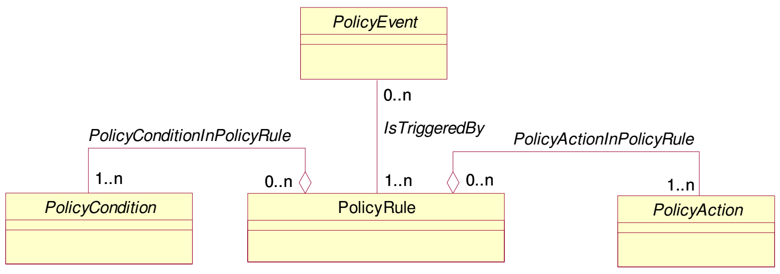


Figure 2.6: DEN-ng PolicyRule model (TMForum, 2008).

IETF PCIM policy model is that DEN-ng dictates the use of policy events as depicted in figure 2.6. The semantics of the policy are now, "On the occurrence of the Policy Events, if the Policy Conditions are satisfied, then execute the associated Policy Actions." This makes the enforcement of policy more scalable then in the previous model, as the policy conditions are only evaluated after a specific event of significance and not continuously evaluated as indicated in the original PCIM definition.

Another important distinction between the DEN-ng policy model and the IETF PCIM is that there is a policy management application hierarchy defined for DEN-ng. The DEN-ng policy model differentiates between using policy to manage applications and management applications used to manage and monitor the execution of policy. Figure 2.7 depicts the current DEN-ng view of policy management applications. The DEN-ng policy model recognises that it is not enough to just dictate the structure of policy but to also dictate how policy should be integrated into various applications domains. It dictates that a policy application is a PolicyServer, a PolicyBroker or a component of a policy server, e.g. a PolicyServerComponent. The PolicyBroker is designed to coordinate the distribution and enforcement of policies in a distributed environment. The PolicyServer is an application that controls the execution and verification of policy via the PolicyExecutionPoint (PXP) and the PolicyVerificationPoint (PVP). The PolicyDecisionPoint has the same functionality as that described within the IETF PCIM.

The DEN-ng information model is designed to describe the structure of business to network managed entities. One of the major advantages of this is that for the first time it is possible to realise policy derived from business goals and objectives to configure the behaviour of the communications network and services. The concept of the *policy continuum* is used to illustrate the processes involved in carrying out business driven network

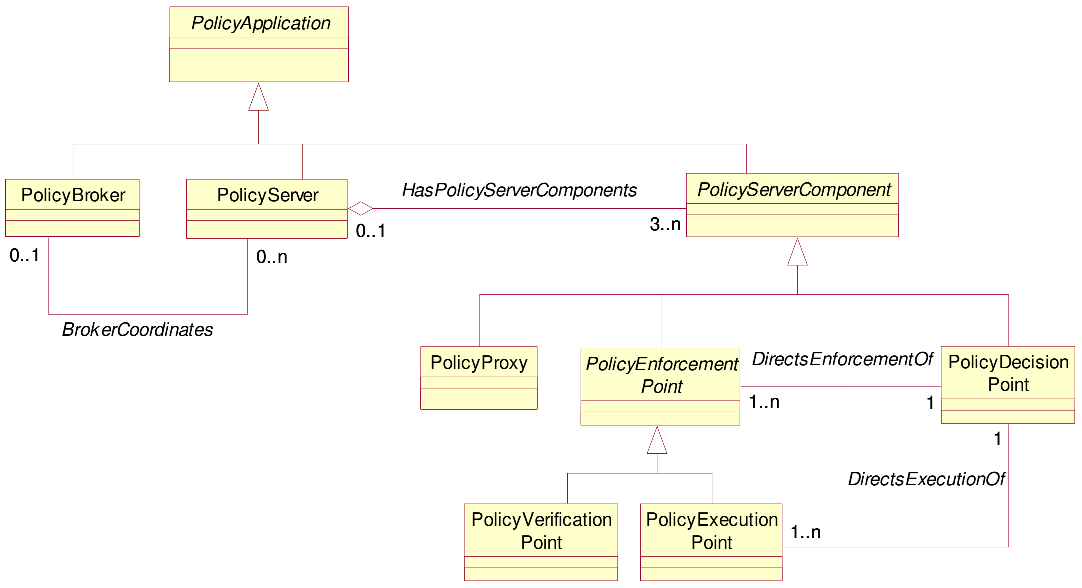


Figure 2.7: DEN-ng PolicyApplication model (TMForum, 2008).

management. It provides a means to related the various levels of policies that can exist, from business policies to network policies. However, even though the policy continuum offers this possibility, effective processes to realise this have not yet been developed. This is primarily due to the lack of effective policy management processes for policy transformation, policy refinement and policy conflict analysis when dealing with a hierarchical set of related policies and the relatively informal description of the policy continuum. In chapter 3 of this thesis, extensions to the policy information model of DEN-ng are proposed, and used throughout the thesis to support more expressive forms of policies and to more effectivly support the use of the policy continuum.

The policy continuum (Strassner, 2003) is described as a set of interrelated policy languages, where each policy language is expressed in the terminology most suitable to its respective constituency of user. The number of levels is arbitrary; however, five has been considered as appropriate for communications network management. The levels of the policy continuum as specified in DEN-ng are business, system, network, device and instance. One of the main contributions of this thesis is to formally define the policy continuum and accompanying policy authoring process.

The majority of the research published for using the DEN-ng policy information model is concerned with the use of policy to realise "autonomic" communications network management. The word autonomic refers to the autonomic nervous system of the human body,

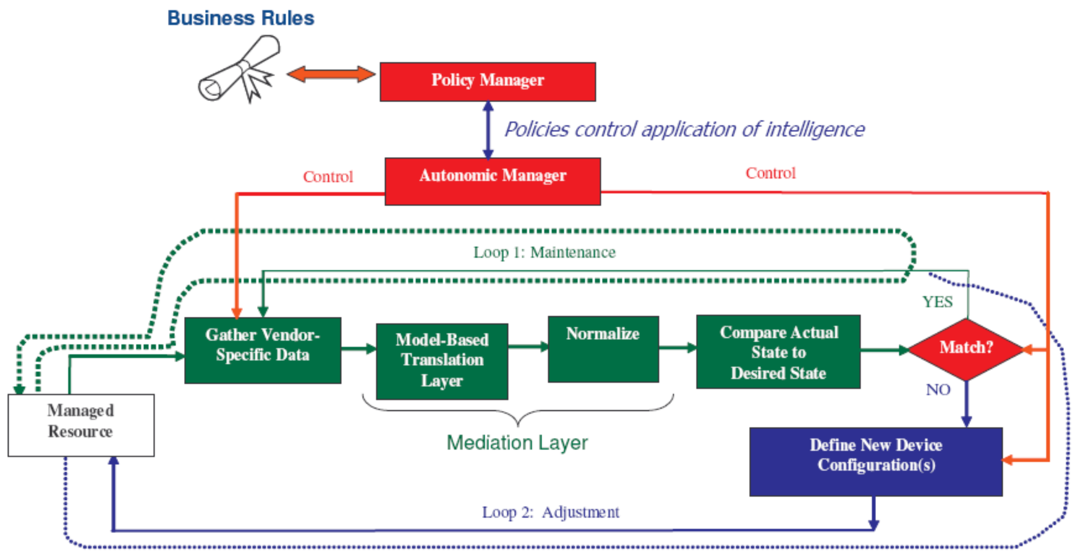

Business Rules
Policy Manager
Policies control application of intelligence
Autonomic Manager
Control
Control
Loop 1: Maintenance
YES
Gather Vendor-Specific Data
Model-Based Translation Layer
Normalize
Compare Actual State to Desired State
Match?
NO
Managed Resource
Mediation Layer
Define New Device Configuration(s)
Loop 2: Adjustment

Figure 2.8: FOCALE and policy interaction (Strassner et al., 2006).

where the body can self-regulate, taking care of breathing, and controlling body temperature among other tasks. The properties exhibited by this autonomic system are typically described as self-* (self-star) behaviour. The '*' can be replaced with the functionality desired, for example self-managed, self-healing, self-configurable. Strassner et al. (2006) introduce the FOCALE[3] autonomic networking architecture, which is heavily dependent on the DEN-ng information model and the associated policy continuum. In that paper, they refer to the policy continuum as being integral to the operation of an autonomic network, where business goals are translated into a form that can be use to configure network resources. The importance of the policy continuum as being central to the realisation of autonomic networking has also been stressed in later research by Strassner and collaborators. As depicted in figure 2.8, policy plays a role in managing the configuration of resources (loop 2), and managing the resource from a business perspective, i.e. maintenance (loop 1). Strassner et al. (2007*g*,*h*); Strassner (2007*b*,*d*); Strassner et al. (2007*f*) motivate the use of FOCALE supported by the policy continuum to enhance the management of wireless communications networks, specifically in the support of seamless mobility and autonomic communications network management.

The core challenge in realising the policy continuum still remains. The challenge being

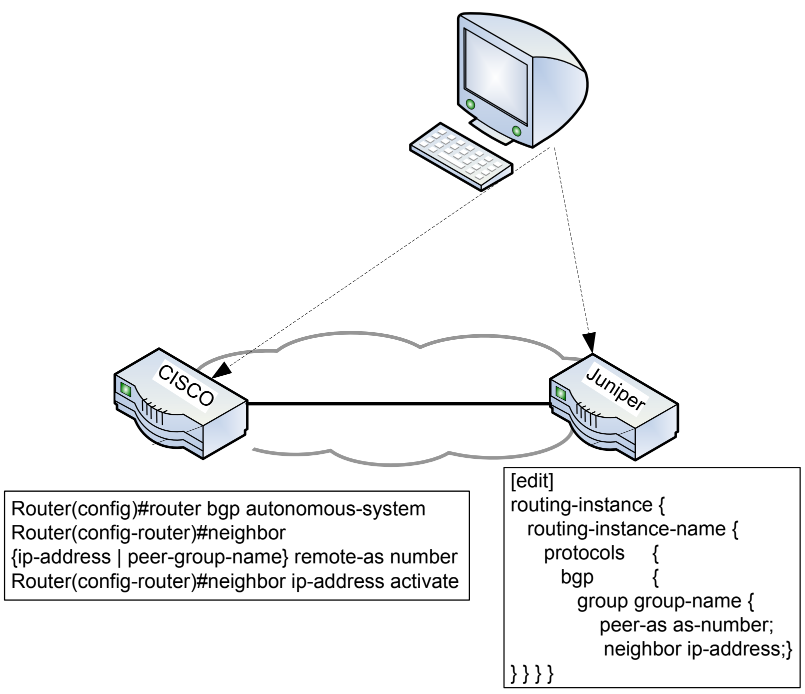
[3]Foundation, Observation, Comparison, Action, Learning and rEasoning

that there are no integrated and application independent processes to realise authoring of policies for the policy continuum. There has been significant related work in the area of enhancing the DEN-ng policy model towards making it more useful in a wider set of application areas that require the use of the policy continuum. For example, Fu and Strassner (2006) present an extension to the DEN-ng policy model to support role-based access control and authentication for a converged wireless network. They demonstrate the advantages of using a model based approach to management, which in this case increases the functionality of the management system to support seamless mobility in multiple wireless access networks. Strassner (2007*d*) enhanced the DEN-ng policy model with the ability to support a representation of context information. This enhancement enables DEN-ng policies to react to changes in context, effectively altering the set of relevant policies based on the state of the managed system. They discuss the enhancement to be critical to the advancement of the use of policy as a facilitator for autonomic networking.

The policy models designed to date do not consider that there are always going to be multiple levels of policy, where low level device or service configuration policies are inspired or derived from high level business policies that guide the behaviour of the communications network. Therefore, unless the a model of the policy continuum is created with an integrated set of application independent processes for policy authoring, and policy conflict analysis, then efforts such as FOCALE cannot be fully realised.

### 2.1.5 Ontological Engineering for Network Management

The DEN-ng information model is fully UML compliant and there is extensive documentation published by the OMG detailing the capabilities of modelling systems with UML. There are, however, some issues with using UML to describe management information for a system (Wong et al., 2005). Once the information model is described for a system it must then be used to build application specific and vendor specific data models. All information can then be mapped back to a common representation; however, the key problem arises when trying to compare different application or vendor specific representations together. Take for example the management of two routers, each manufactured by two different vendors. When configuring Border Gateway Protocol (BGP) peers (as illustrated in figure 2.9), the two routers must be configured correctly for routing to be successful, if not then there may be connectivity problems. The two configuration languages are completely dif-



```
Router(config)#router bgp autonomous-system
Router(config-router)#neighbor
{ip-address | peer-group-name} remote-as number
Router(config-router)#neighbor ip-address activate
```

```
[edit]
routing-instance {
   routing-instance-name {
      protocols     {
         bgp          {
            group group-name {
               peer-as as-number;
                neighbor ip-address;}
}}}}
```

Figure 2.9: Router configuration problem (Strassner, 2003).

ferent and it is difficult to map the capabilities of the two languages. This is especially true if commands exist in one language that are similar to groups of commands in the other language, or worse still, may not exist in the other language.

A possible solution to this problem is to augment the information models and data models with ontologies as described by Wong et al. (2005). Ontologies are capable of representing formal semantics that can be reasoned over and are capable of representing similarity relationships among modelled entities and other rich semantic relationships. The rest of this section is devoted to discussing how ontologies can benefit information modelling and policy based management.

#### 2.1.5.1 Types of Ontologies

Ontologies offer a formal method of defining a better understanding of information and according to Uschold and Gruninger (1996):

*"Ontology is the term used to refer to the shared understanding of some domain of interest which may be used as a unifying framework"*

By sharing ontologies, different parties can share how they perceive the meaning of the related information. Ontologies also offer methods of relating different parties' inter-

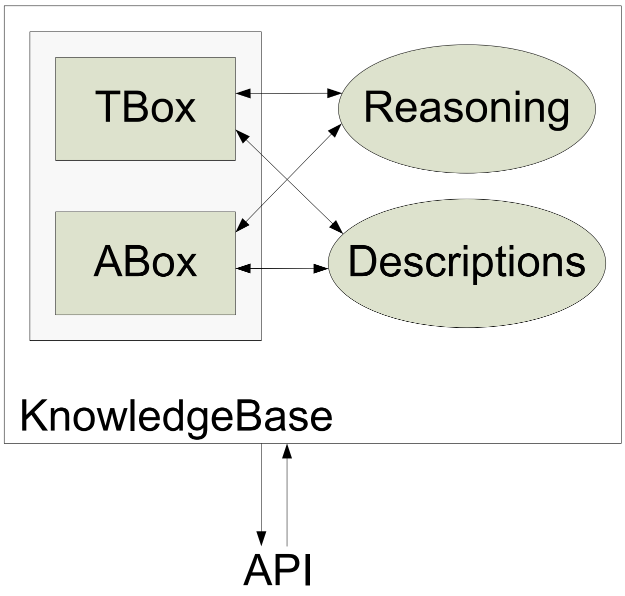


Figure 2.10: DL knowledge base (Baader et al., 2007).

pretation of information. For example, one party may describe a particular object, say a router, by its number of interfaces. Another may describe that router by its design goals (e.g. buffer size, capabilities and processor). By defining the concept of a router using the two ontologies, the two parties may begin to reason over each others' version of the router, in effect they will be describing the same router from different perspectives. This type of reasoning is very useful when trying to relate different data models together and even relate different programming models..

There are many techniques and technologies available to develop ontology models. Such techniques include using predicate calculus and frame logic (Gruber, 1993), but by far the most popular approach is using description logic (DL) (Baader et al., 2007). Description logic is a subset of first order logic and is concerned with the representation of concepts, relationships between concepts and the constraints on those relationships. There are essentially two aspects to DL, TBox concepts and ABox concepts. TBox dictates the terminology of the concepts being described and the constraints and relationships over those concepts. ABox dictates the individuals (assertions) that exist as types of concepts in the description logic system. Figure 2.10 depicts the relationship between the components of the DL knowledge base. An important property of DL that makes it so useful to representing ontologies is that individuals can be readily subsumed by classes other then the one they are originally typed by, thus allowing inference over the concepts defined in

| Notation | Description |
|---|---|
| $C \equiv D$ | Equivalence class relationship, C is equivalent to D |
| $B \equiv C \sqcap D$ | B is equivalent to C and D (C intersection D) |
| $B \equiv C \sqcup D$ | B is equivalent to C or D (C union D) |
| $B \equiv \exists x.D$ | B is equivalent to anything that has a property x with some value D (necessary) |
| $B \equiv \forall x.D$ | B is equivalent to anything that has a property x with value only in D (necessary and sufficient) |
| $A \sqsubseteq B$ | A is a sub type of B |

Table 2.1: Description Logic notation.

the knowledge base.

An example of a TBox statement is depicted in equation (2.1):

$$\mathit{SecureRouter} \equiv \mathit{Router} \sqcap \exists \mathit{hasCapability}.\mathit{Secure} \tag{2.1}$$

The statement defines the concept of a **SecureRouter**. It illustrates that if there exists a **Router** that has a value of **Secure** in its **hasCapability** property then that router is a **SecureRouter**. An example of an ABox statement is depicted in equation (2.2):

$$\begin{gathered} \mathit{Router}\,(CISCO3700) \\ \mathit{Secure}\,(SECDEV1) \\ \mathit{hasCapability}\,(CISCO3700, SECDEV1) \end{gathered} \tag{2.2}$$

More general terminology used for describing the TBox portion of the knowledge base is offered in table 2.1. The notation used represents the core axioms that can be used to build a description logic knowledge base. This knowledge base is used to represent the ontology. To query the knowledge, a new class type can be defined to investigate which individuals are subsumed into the class.

#### 2.1.5.2 Semantic Web Technologies

There has been a recent surge in the use of ontology technologies for use in describing the semantics associated to web services and documents on the Internet. There is a family of related technologies developed by the World Wide Web Consortium (W3C) namely: Resource Description Framework (RDF), RDF Schema (RDFS), Web Ontology Language (OWL) and Semantic Web Rule Language (SWRL). Each technology is a standard for representing semantic information in XML, where each is more expressive then the last.

RDF is able to describe metadata about documents on the Internet. It has a very simple structure, a triple of the form (subject, predicate, object). The subject entity is related to the object entity via some predicate logic statement. An example of its usage in depicted in (2.3)

$$\begin{gathered}[qn:Router1,\quad qn:Vendor,\quad ''Cisco'']\\ [qn:Router1,\quad qn:Model,\quad ''6509'']\end{gathered} \tag{2.3}$$

RDF is good at representing simple relationships associated to documents, but is too limited for some circumstances (Baader et al., 2007). RDFS is designed to extend the semantics of RDF adding the notion of classes and class hierarchies (classes can be associated together using properties and domain / range expressions). An example of its usage is depicted in (2.4), which specifies that an entity Router2 is of type ATMRoutingDevice, and that the new type ATMRoutingDevice is a sub class of type RoutingDevice.

$$\begin{gathered}[qn:ATMRoutingDevice,\quad rdf:subClassOf,\quad qn:RoutingDevice]\\ [qn:Router2,\quad rdf:Type,\quad qn:ATMRoutingDevice]\end{gathered} \tag{2.4}$$

The Web Ontology Language (OWL) (W3C, 2004) is by far the most well known technology for representing semantic information in the form of ontologies. A version of OWL, named OWL-DL is focused on using the notions from description logic to describe semantic information. OWL is built upon RDF/RDFS, there has been a number of inference and reasoning engines that have built in support for reasoning over the ontologies defined in OWL. Reasoners are for example Racer (2008), Fact++ (2008) and Pellet (2008). A popular editor for OWL, and RDF/RDFS is Protégé (Protégé, 2008). Protégé provides the user with a Graphical User Interface (GUI) and backend support for connecting to a reasoner so that subsumption can be performed. One of the attractions to using OWL-DL is that it remains computationally decidable as opposed to richer version of OWL such as OWL Full. OWL Full offers more extensive features to represent semantic information but can cause some performance complexity issues as it is not guaranteed to be decidable. For example, OWL Full offers the ability to treat classes as individuals and reasoning can be performed over classes as well as individuals of classes. The Semantic Web Rule Language (SWRL) (W3C, 2008*b*) is a rule language that can extend the semantic information described in an OWL ontology. It can for example re-classify individuals based on their relationships to other individuals in a more flexible way that cannot be done with OWL on its own.

The use of ontologies to represent management information for network management is well researched area, attracting considerable attention over the past few years (Lopez de Vergara et al., 2003; Wong et al., 2005; de Vergara et al., 2004). The primary focus was to enrich existing information models with semantic information to aid interoperability. Recently however, the use of ontologies in policy based management has gained momentum. A more thorough investigation into the use of ontologies in policy based management is given in the literature review section.

## 2.2 Literature Review

This section provides a review of research related to policy analysis from the perspective of how these processes perform along side a policy continuum. Firstly, a discussion of the existing definitions of policy conflict is presented. Depending on the application for which policy is being defined there are varying views on the matter. Secondly, a description and discussion of policy analysis processes that are required to deliver effective and dependable policy based management systems are presented. Policy conflict analysis is categorised into three approaches that reflect the current state of the art and from there a discussion of the current deficiencies of these approaches concerning operating within a policy continuum is presented.

### 2.2.1 Policy Conflict

Policy conflict is one of the outstanding issues related to policy based management. Policy conflicts have different manifestations in different applications that make use of policies. Therefore, a policy conflict in one application may not necessarily be a policy conflict in another application. The careful definition of policy conflict should lead to the creation of purpose built policy conflict analysis processes that can effectively detect and cater for conflicts among policies in specific applications. However, even the definition of policy conflict is a challenge. Strassner (2003) describes policy conflict as:

*"A Policy conflict is when two of more policies applying to an overlapping set of managed entities are simultaneously triggered, and there conditions are satisfied, but their actions are contradictory."*

This definition describes that the shared resource (the overlapping set of managed en-

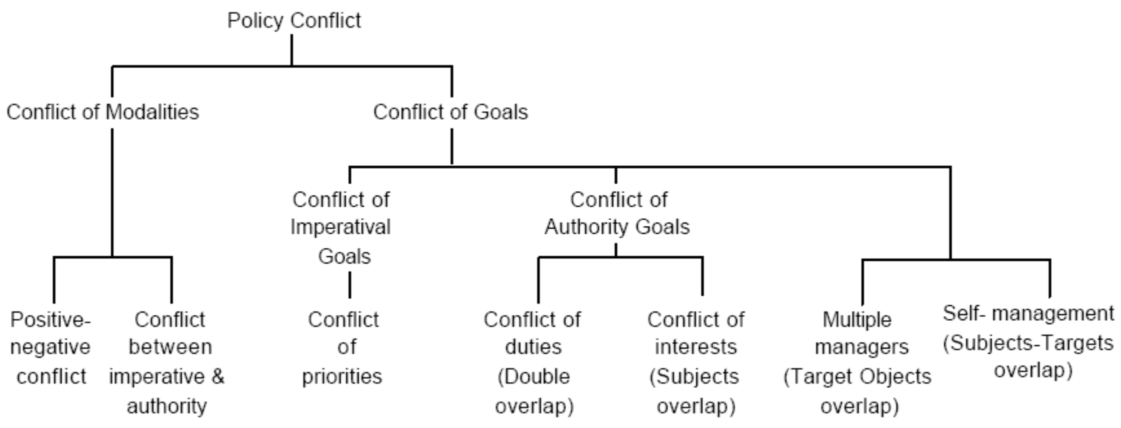

Policy Conflict
Conflict of Modalities
Conflict of Goals
Conflict of Imperatival Goals
Conflict of Authority Goals
Positive-negative conflict
Conflict between imperative & authority
Conflict of priorities
Conflict of duties (Double overlap)
Conflict of interests (Subjects overlap)
Multiple managers (Target Objects overlap)
Self- management (Subjects-Targets overlap)

Figure 2.11: Classification of policy conflicts (Moffett and Sloman, 1994).

tities), has multiple policies specifying behaviour for them (two or more policies applying). The overlapping set of policies are triggered simultaneously and are satisfied (conditions are true). The problem arises when the actions specified by the policies are contradictory. Different approaches to policy based management prescribe their own definition of conflict; however, in general they all are related to the enforcement of actions that ultimately may cause the managed system to act in a nondeterministic fashion.

Moffett and Sloman (1994) originally highlighted the issues associated to policy conflict; they define a policy conflict to occur when the objectives of two or more policies cannot simultaneously be met. They define policy conflict from the specific view point of management policies, which they consider to be positive or negative, authorisation or obligation policies. They describe that conflict can occur between individual policies if there is an overlap in the subjects, targets and actions of the policy and some other specific conditions are met. The types of conflict they define are depicted in figure 2.11. Conflicts can be associated to the "modality" of policy (authorisation/obligation) and the goals of the policy. Modality conflicts are when an action is both permitted and prohibited, or obligated and refrained (obliged not to). Goal conflicts are associated to the effect permitted actions have on the system when carried out simultaneously. An example of this is, if two policies require database access, but due to resource restrictions only one can access the database at a time. A conflict of duty occurs when the actions specified do not conflict in the traditional sense, but where their simultaneous execution is a breach in the use of the actions. For example, two policies may allow a bank clerk to both issue and approve cheques; obviously the organisation will not want such policies to exist. This is similar to the conflict of interest policy conflict. The multiple managers conflict occurs when two distinct subjects invoke contradicting actions on a common target; subsequently, the target cannot decide which manager to follow and nondeterministic behaviour may occur.

The above view of policy conflict has become quite popular and has been further published along with resolution strategies by Lupu and Sloman (1997, 1999). A similar definition of policy conflict has been adopted by Dunlop et al. (2001, 2002, 2003) who further classify policy conflict into dynamic and static cases. Dynamic policy conflict can only be truly detected at runtime, whereas static policy conflict can be detected at compile time. Policies for network management are specified at a system wide view of the network. This system view is in contrast to lower level policies such as those defined to be used on network interfaces, which have a network view.

Chomicki et al. (2003, 2000) formalise policy from a different perspective: they perceive policies as Event-Condition-Action triples. The subjects and targets of the policy are implicitly specified in the action component and describe that policy conflict is the overlap of events, conditions and actions, where the actions must contradict for a conflict to occur. This definition is similar to Strassner's definition, and can be associated to the system wide policies or network device specific policies, as the application is not defined.

Policies defined specifically for network traffic management are a relatively constrained type of policy and are usually defined over IP traffic match criteria, such as for firewall filtering policies, DiffServ classifying policies or IPsec VPN policies. Al-Shaer and Hamed (2003, 2004*a*,*b*) look specifically at policy conflict from this perspective and define conflict appropriately for their need: policy conflict from the perspective of network filtering policies can be defined as a specific ordering of policies to produce anomalous behaviour. Conflict or anomalies in network filtering policies come in the form of redundancy, contradictory, generalisation and correlation. Redundancy refers to when a filtering policy is made redundant due to the order of the policies. This is a problem as the rule is not required and can be removed. Contradictory, on the other hand is similar to redundancy except that the rule must not be removed; instead a re-ordering of firewall policies is required. A more indepth analysis of firewall policy conflicts is presented in section 2.2.3.

Using policy to define the access rights of requesting entities, or more specifically access control policies, can be a complex area. However, the definition of conflict in this circumstance is relatively straight forward. If contradicting access rights are granted to an individual entity, then a conflict has occurred. Ascertaining whether a conflict may occur

Table 2.2: Policy conflict classifications.

| Body of Research | Abstraction Level | Policy Type | Conflict Definition |
|---|---|---|---|
| Strassner (2003) | Network view | ECA | Incompatible policy actions, generic |
| Moffett and Sloman (1994) | System view | STA | Incompatible policy types, incompatible policy actions |
| Dunlop et al. (2001, 2002, 2003) | System view | STA | Role conflict, dynamic conflict, static conflict |
| Chomicki et al. (2000, 2003) | Network view | ECA | Incompatible policy actions, incompatible policy events |
| Al-Shaer and Hamed (2003, 2004*a*,*b*) | Network view | ECA | Incompatible policy actions, network filtering |
| Jajodia et al. (2001); Wijesekera and Jajodia (2003) | System view | STA | Incompatible policy actions, access control |

is address by research carried out by Jajodia et al. (2001) and Wijesekera and Jajodia (2003), who designed formal models of access control and devised algorithms to search for various inconsistencies in deployed policies.

It can be seen that depending on what the policies are being applied to (i.e. its application), then there are varying requirements for a policy conflict to occur, as summarised in table 2.2. In the table ECA refers to Event-Condition-Action policies and STA refers to Subject-Target-Action policies. Therefore, a single definition of conflict will not be applicable to all applications. The various approaches to policy conflict analysis for different application types are examined in this chapter. The approaches can be arranged into three areas:

1. **Language based policy conflict analysis**, which is devoted to examining the constructs of the policy language to determine cases for policy conflict. This method is predominately used for policy analysis on network filtering policies as the policy models and its associated execution semantics are well defined.

2. **Information model based policy conflict analysis** approaches refer to a system information model that describes both the policy model in use and the target managed system, to support the conflict analysis of policies.

3. **Ontology based policy conflict analysis** is a relatively recent approach. Well understood policy models are represented using ontologies and the inherent reasoning capabilities of ontologies aids in policy analysis.

### 2.2.2 Policy Analysis Processes

This section is concerned with reviewing the processes related to policy analysis in general and not specifically to policy conflict analysis. The objective is to investigate the dependency between the various analysis processes.

The power of policy based management is tied to the capabilities of the supportive processes to analyse policies towards ensuring the policies proper enforcement. Policy analysis is concerned with closely examining new and deployed policies so that they can be enforced in a well behaved (or at least non hazardous) way. By defining policy over a shared resource (e.g. router), each policy may specify some behaviour for the network that is contrary to the behaviour specified by other policies. The primary policy analysis processes (according to Verlaenen et al. (2007*a*)) to support the well behaved enforcement of policy are as follows:

- Policy conflict analysis: does policy A conflict with policy B?
- Policy refinement: can policy A be realised at a lower level by policy set B?
- Dominance checking: if policy B were removed will the system behaviour be altered?
- Policy optimisation: the alteration of policies to reduce computational complexity of the policies analysis or enforcement?
- Coverage checking: if a policy is defined for a particular condition, or specified over a set of managed entities?
- Policy combination: can two or more policies be replaced with less policies, where the resulting behaviour is the same?
- Policy deduction: can a policy can be deduced from a set of policies?

- Policy transformation: if policy A is transformed to policy B does it still meets its objectives?

According to Moffett and Sloman (1993), policy based management of very large (distributed) systems requires the generation of low level and system specific policies from general high level business oriented policies. They propose the generation of policy hierarchies, where low level policies are generated to represent the goals and objectives of high level policies. They describe the process of building the policy hierarchy as *policy refinement*. Their paper is a discussion paper highlighting some of the mains research challenges involved in automated policy refinement. They describe that as policies are continuously modified, so too are the associated policy hierarchies. They also mention that the refined policies must be analysed to ensure that they can be enforced onto the target managed system and that the refined policies meet the goals and objectives of their higher level policy counterparts. Policy conflict analysis should be performed in coordination with policy refinement, but the link between the two processes has not been formally investigated.

Later research lead to the development of a goal based policy refinement framework by Bandara et al. (2004) who present an approach to policy refinement based on a formal representation of the management system to aid in the derivation of low level policies. The system is represented in a logic formalism named Event Calculus (EC). EC, as presented in (Shanahan, 1999), is a formal language and is designed to represent and reason about dynamic systems (i.e. systems whose state changes over time). Bandara et al. (2004) began by representing the system in EC and described how low level actions affect the system by way of EC statements. Goal elaboration is then used to expand a policy into a set of low level actions that satisfy the goals of the original policy. It is then the responsibility of the system administrator to ensure that the correct sub-goals (or system actions) are chosen to enforce the high level goals (policies). Their approach is interesting as it dictates the incorporation of system specific information into the refinement process. They mention that this system specific information may be derived from a UML model and associated finite state machine.

An application of their approach is discussed by Bandara et al. (2006*a*) who describe policy refinement for an IP Differentiated Services application. They demonstrate that is it possible to represent the system in EC and define high level QoS policies for it. These high level policies are then refined into low level concrete policies that are used to actually provision the DiffServ classes of service. Their approach to policy analysis motivates and demonstrates the use of supplying to analysis processes extra information concerning the managed system. The approach they present is not formally tied to a conflict analysis process; however, they do address policy conflict in related works (Charalambides et al., 2005, 2006; Abedin et al., 2006). By not explicitly associating policy refinement with policy conflict analysis there is no guarantee that the refined policies do not conflict with themselves or conflict with the policies that have been previously deployed.

A alternative approach to policy refinement, but for the creation of network security policies, is proposed by Luck et al. (2002) and de Albuquerque et al. (2005). Their objective is to reduce the complexity associated to describing security policies for communications networks by first describing high level policies and then deriving low level concrete policies that can be enforced within the network. They use a model-based management approach where the management system is described using a set of tightly interrelated UML models of varying abstraction. They introduce the concepts of consistency and completeness of the refinement process. Their definition of a consistent refinement is when the resulting set of policies are not in conflict with each other, and their definition of completeness is that once the refinement is complete, all high level policies are satisfied by low level policies. These checks are important to a successful policy refinement process as they provide some measure of validation that the refinements are correct. Their approach is not automatic but is guided by a software tool used by the system administrator. They do not specify their policy conflict analysis process, but do highlight its importance in association to policy refinement.

Recent research into policy refinement, building upon the research of Bandara et al. (2006*a*), was carried out by Rubio-Loyola et al. (2005, 2006*a*,*b*). This work also used a logic based approach to policy refinement and incorporated a system model. The novelty introduced by their work is that they incorporate formal verification techniques; with verification being achieved using model checking techniques based on Linear Temporal Logic (LTL). With LTL, one can examine the temporal relationship between actions over a system. However, the process is not specifically designed to incorporate policy conflict analysis. They attempt to produce policies that are well specified with regards to the system specification, but existing deployed policies are not considered in the conflict analysis process.

Policy refinement is an active research area. It is clear that the introduction of information concerning the managed system can be leveraged to enhance the processes involved in policy refinement. The main outstanding challenge is to devise a formal relationship between the analysis processes for policy, as currently the process of policy refinement and policy conflict analysis are de-coupled, but clearly they are highly related.

Another process that is related to policy refinement and policy conflict analysis is that of policy transformation. Research carried out in (Cridlig et al., 2007) highlights the need to be able to ensure that once a policy has been transformed from one policy model to another that it still maintains consistent behaviour. They investigate the process of transforming access control policies across different policy models. They do not discuss the implications their process has in terms of other important policy analysis processes; therefore, it would be difficult to see how the process fits into a wider policy analysis process that caters for all policy analysis requirements such as those mentioned above.

### 2.2.3 Language based Policy Conflict Analysis

Network policies are devised to manage the behaviour of network services and vendor specific policy languages are typically used to describe network policies because, different application domains have well defined semantics for how policies control the services, and thus the policy languages have varying requirements. This section focuses on the analysis techniques developed to analyse firewall policies, IPsec VPN policies and routing policies. For each type of policy application, similar approaches to conflict analysis have been developed, but the algorithms developed depend heavily on the nature of the policy application. Such language based approaches are predominately focused on examining the syntax of the policy language to ascertain cases for conflict.

$$\langle IP.src\rangle\, \langle IP.dst\rangle\, \langle IP.srcPrt\rangle\, \langle IP.dstPrt\rangle\, \langle IP.tos\rangle \rightarrow \langle Forward/Drop\rangle \tag{2.5}$$

A typical firewall policy rule could be described as depicted in (2.5). The firewall policy can be understood to mean, if an IP packet arrives on a specified interface matching the fields as described, then the appropriate actions should be taken (forward or drop). This firewall policy specification follows the policy semantics of "If condition is true then action".

Recently however, it became apparent that so called *conflicts* or *anomalies* can occur when specific orderings of policies were deployed in combination. Early research into detecting anomalies in firewall policies was documented by Hari et al. (2000) who recognise

Table 2.3: Firewall policy anomalies.

| Anomaly Type | Description |
|---|---|
| Redundancy | Occurs if one rule is more general than another and the actions are the same |
| Contradictory | Occurs if one rule is more general than another and the actions are different |
| Generalisation | Occurs if one rule is more specific then another and the actions are the same |
| Correlation | Occurs if one rule overlap with another rule on some IP criteria and the actions are different |

that the inherent ordering of firewall policies may lead to security holes. Al-Shaer and Hamed (2003, 2004*a*,*b*) introduce a comprehensive firewall analysis toolkit that is capable of examining groups of deployed firewall policies and determining if such anomalies exist; the types of anomalies are listed in table 2.3 as documented by Al-Shaer and Hamed (2003).

The cause of the anomalies are down to the implicit execution semantics of firewall policies. More specifically, when an IP packet arrives on an interface and is sent to the firewall service to be processed, it is examined against each deployed firewall rule in sequence until one of the policies conditions are satisfied, the actions associated to this policy alone are executed. The issue is that each policy describes match criteria, and for example, the match criteria of an early policy may be more general then the match criteria of a later policy, the effect being that the later policy is never actually triggered. This is especially worrying if the later policy had a different intended action then the earlier policy, this is known as *contradictory* and can lead to undesired firewall behaviour.

The approach taken by Al-Shaer and Hamed (2004*a*) was to devise a custom model specifically to be used with firewall policies. The model allows them to specify relationships between firewall policies, namely : disjoint, equal, inclusive, partially disjoint and correlated. They also specified a method for determining the appropriate relationships among deployed policies. They devised a policy analysis algorithm that scans through each individual policy and adds it to a “policy tree” representing the relationships of previous firewall policies. Each level of the policy tree represents a different component of the match criteria of the policy: source port, destination port, source IP address, destination IP address, protocol and action. By storing the policies in such a data structure, the algorithm can search the tree for specific policy anomalies. In (Al-Shaer et al., 2005), they focused on a more complex variant of the above problem: the distributed scenario, which concerns detecting firewall policy conflict between physically distributed routers. The increased se-

Table 2.4: Distributed firewall policy anomalies.

| Anomaly Type | Description |
|---|---|
| Spuriousness | Occurs if an upstream firewall allows the traffic blocked by a down stream firewall |
| Shadowing | Occurs if an upstream firewall blocks the traffic allowed by a down stream firewall |
| Redundancy | Occurs if an upstream firewall blocks the traffic also blocked by a down stream firewall |
| Correlation | Occurs if an upstream firewall allows/blocks the overlapping traffic blocked/allowed by a downstream firewall |

curity required for large scale communications networks demanded the investigation into policy analysis when there were multiple firewalls to cater for. The situation became more complex as the semantics associated to the firewall policies changed. Now a single firewall policy could have an affect on the security of a path in the network. New forms of anomalies were proposed and are listed in table 2.4. The primary new challenges encounted when considering a distributed scenario is the increased scale of the problem. Effectively, the policy analysis algorithm must compare the firewall policies on one device with those deployed on associated devices in an efficient manner. Al-Shaer et al. (2005) proposed to base the discovery of related policies on pre-existing routes in the network; therefore, if the network is changed the algorithm can adapt. The key problem with their approach is that the conflict analysis process is completely decoupled from the policy authoring process or policy refinement process. The complexity of the resulting solution can be seen as a consequence of not analysing the policies at a higher level of abstraction first, before they are deployed to the networking devices.

Research carried out by Zhang et al. (2007) investigated the intelligent distribution of policy to avoid policy anomalies. They focus on addressing policy analysis at the authoring phase. By addressing the problem at the authoring phase, corrective action can be taken in the careful insertion of the new or modified firewall policy. The algorithm they developed is based on re-ordering and checking the correctness of deployed policies on each modification to the set of deployed policies. They reduced the complexity associated to the problem through abstraction.

Another approach to the problem, proposed by Bandara et al. (2006*b*), is to encode the firewall policies into a logic programming language and to use AI techniques to search for policy anomalies. They represent a firewall policy as a logic statement in Prolog (a logic programming language). They designed a logic program to investigate the automatic

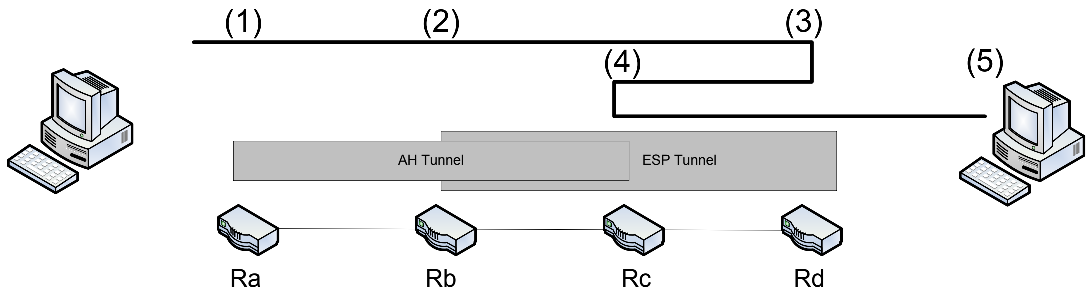


Figure 2.12: IPsec policy anomaly (Fu et al., 2001).

re-ordering of policies based on pre-defined precedence among the type of policies. The program was capable of detecting all anomalies as presented in Al-Shaer and Hamed (2003), and could resolve some anomalies by re-ordering the policies. Their approach was based on argumentation logic. However, they do not address distributed firewall problems, as their approach suffers from the same drawbacks as the approach presented by Al-Shaer et al. (2005).

IPsec (Kent et al., 1998) is a security extension to the IPv4 and IPv6 protocols enabling routing devices to secure IP traffic by adding encryption, authentication, integrity and non-repudiation. The ordering and execution semantics of IPsec policies are similar to that of firewall filtering policies. They are described against a traffic match criteria and a specific operation is applied to the packet. The operations fall under two categories: tunnel mode and transport mode. Tunnel mode encapsulates the original IP packet into the payload of a new IP packet originating from the router, thus masking the original source and destination addresses. Specific security services are used to encrypt and authenticate the payload of the IP packet. Transport mode modifies the original IP packet to apply a security service to the payload only. The main difference between firewall filtering policies and IPsec policies with respect to execution semantics is that IPsec policies must be path based. This means that IPsec policies are required to secure the original packet at the source router and un-secure the packet at the destination router. Therefore, policies must be coordinated across the network. This feature of IPsec policies creates an environment where policy analysis is critical to ensuring that the policies are correct and effective.

Fu et al. (2001) discusses the need for the analysis of IPsec policies and categorised a number of anomalies that can occur within the network. The anomaly they focus on is that given a specific ordering of IPsec policies across a number of routers, potentially sensitive IP traffic may be routed as clear text across the network. This anomaly occurs if an incorrect group of tunnels are defined by policy. An example of this is depicted in figure 2.12 which shows a network with four routers and a source and destination endpoint. There are two IPsec tunnels defined, one that encapsulates traffic from Ra to Rc in an Authentication Header (AH) tunnel, and one that encapsulates traffic from Rb to Rd in an Encapsulated Security Payload (ESP) tunnel. Therefore, the IP traffic should be secured from end to end. The process each IP packet goes through from Ra to Rd is as follows:

1. The traffic from the source is encapsulated at Ra and forwarded.
2. The traffic now arrives at Rb and is encapsulated again. As the traffic in now encrypted, Rc cannot de-capsulate the traffic encapsulated at Rb and so forwards the traffic to Rd.
3. At Rd the traffic is both encrypted and authenticated. The traffic is now decrypted but the protected payload belongs to the AH tunnel and must be de-capsulated at Rc.
4. The traffic is sent back to Rc, de-capsulated and forwarded on to Rd again.
5. The traffic is finally forwarded to the destination. In this scenario, the IP traffic passed as clear text from Rc to Rd thus breaching the defined policy and posing a critical security threat.

Fu et al. (2001) proposed a method of analysing the set of associated policies and re-defining the policies to eliminate overlapping tunnels. They noted that by eliminating tunnel overlaps, the severity of the problem is reduced. The approach to splitting tunnels was also addressed by Yang et al. (2004), who propose to retrieve all IPsec policies along a specific path and analyse each policy source and destination address; a specific algorithm is designed to determine if a split operation is required. If a split is required then some policies are generated and the IPsec tunnel is split in two. The difficulty here is in deciding what new policies should be created, and where they should be placed. These new policies must also be analysed to ensure that more overlap is not created. This demonstrates the difficulty involved in performing analysis when only the constructs of the language are available.

Lin et al. (2006) describe more cases of policy conflict, highlighting that the same shadowing and redundancy anomalies as documented for firewall filtering policies can also arise for IPsec policies. They also describe the problem of weakening security where a strong security service like ESP is weakened by then applying AH in tunnel mode. It showed that specific orderings of security services can degrade the security of a path in the network. They proposed a model for policies and an algorithm that can automatically re-order policies to eliminate specific cases of anomalies as documented. In common to the reviewed approaches to IPsec policy analysis is that all information about the anomalies is derived directly from the specification of the policies, and is thus language based. The language based approach works well here as the semantics of the policies are well defined and can be assumed when defining algorithms to discover potential anomalies.

One of the first domains where policy was applied is that of routing in networks. Routing policies are defined to determine how traffic is routed throughout a network and between autonomous systems (AS). Conflicts in routing policies may lead to under-performance or nondeterministic behaviour from networks. Policy for routing is used to control the path information being transferred between autonomous systems. Yang et al. (2007) classify a set of policy conflicts or anomalies that can exist between routing policies and IPsec policies. The nature of the conflict is that the paths in the network may be dynamic and the route that secure traffic takes may change from time to time. The consequence of this is that traffic may be susceptible to looping in the core of the network. They devised a specialised algorithm taking the specific behaviour of the problem into account to detect for conditions that may cause looping to occur when considering routing and IPsec policies. Again their approach makes assumptions about the implicit behaviour of the policies and depends on the specification of the policies to discover cases of conflict. As they perform analysis at such a low level, they must develop more complex algorithms to compensate for the lack of abstraction mechanisms. Analysis at a higher level may offer reduced complexity and easier solutions as suggested in Zhang et al. (2007).

The policy models discussed so far are typically described as Event-Condition-Action (ECA) policies. Agrawal et al. (2005) investigated generic policy analysis algorithms that can be applied to any policy model that is based on ECA. They treat the specialised cases of policy relationships as defined above for network management policies in more general terms. They also recognise that whether the policies are defined for firewall filtering, IPsec VPNs or routing, similar analysis capabilities are required by the conflict analysis process. Subsequently, they define four generic analysis processes, namely: domaince check, potential conflict check, coverage check and consistent priority assignment. Together the defined processes are part of an encompassing policy ratification process, defined to ensure that policies are well specified and consistent with each other. They specify that the majority of the processing is devoted to determining if the conditions of one policy are implied by the conditions of another. Ascertaining this relationship between two network filtering policies is readily computable as the condition components are related to IP address header information. However, in the general case determining implication relationships between two Boolean expressions is a NP problem. They define dominance checking as the process to determine if the conditions of one policy imply another, where the actions of the policies are equal. This is equivalent to redundancy for network filtering policies. They define potential conflict to occur if the conditions of one policy implies the condition of another, but where the actions of the policies contradict. They define contradicting actions as actions that set a common attribute to different values. Coverage checking is the process of determining if a set of target managed entities are all covered by at least one policy, the problem reduces to testing the coverage of Boolean expressions against a range of values.

The language based policy conflict analysis processes covered in this section treat the problem of detecting conflict as simple pair wise comparison of deployed policies. They do not take into account that the policies may have been refined from higher level policies. When policies are represented at a higher level of abstraction, it may be easier to ascertain cases of policy conflict, as more information may be available to the policy conflict analysis processes.

The main benefits of using language based approaches are that they suit analysis of highly specific policy languages as exemplified by firewall policies, IPsec VPN policies or routing policies. The behaviour of the policy models in use is well understood and documented; therefore, it makes the algorithms conceptually easier to understand and re-apply. The disadvantages of the language based approach is that the algorithms developed cannot be reused across different applications as they depend heavily on the nature of the policy language in use. The solutions are not inherently scalable as the policies are distributed across the network, and it is not straightforward to discover which policies should be retrieved for analysis. There is no consideration for the system constraints, which may limit the actual enforcement of policies at runtime.

### 2.2.4 Information Model based Policy Conflict Analysis

Information models are designed to describe a system in a technology independent way. There are a number of methods to describe a system, UML being prominent. This section discusses the research into policy conflict analysis that makes use of an information model to aid in the discovery of conflicts, as opposed to just using the constructs of the policy language. The information models are used to not only describe the application being managed but are also used to describe the policy language.

#### 2.2.4.1 Formal model approach

Bandara et al. (2003) use a model to describe the structure of a target managed system and further exploit this model to aid in policy conflict analysis. The information model they use is described in a formal logical notation, Event Calculus. As described earlier, instances of EC can represent the behaviour of dynamic systems. The EC object model (information model instances) express the service descriptions and systems constraints, for example it could represent a set of printers grouped into hierarchical management domains. It could also describe specific properties of those printers, for example whether they could print in colour or whether one was inkjet or laser. After the structural representation of the system is defined they then defined the behaviour of the system. They describe behaviour by associating pre- and post-conditions to the actions possible within the system. For example, a print action would render a printer busy for a period until the print job was finished.

The research looked at two aspects of policy conflict: application independent[4] and application specific. They define application independent policy conflict as conflict that occurs due to the types of policies, i.e. when one policy permits an action and another policy prohibits the same action. It is application independent as the semantics of the policy types do not change across applications. Application specific conflicts are defined to exist only within certain applications. This is because the actions causing the policy conflict are specifically associated to that application. For example, a "powerPrinterDown" operation and a "printDocumentX" operation are deemed to conflict within a printer management system. The knowledge representing which actions conflict are defined explicitly in the model of the specific application.

[4]In the original work, they refer to application independent as "domain independent". The definition is changed here for clarity and consistency.

$$
\begin{aligned}
&holdsAt(authConflict(Subj, Op), Tm) \leftarrow \\
&\quad holdsAt(permit(Subj, Op), Tm) \wedge holdsAt(deny(Subj, Op), Tm).
\end{aligned}
$$

Figure 2.13: Authentication conflict description axiom (Bandara et al., 2003).

$$
\begin{aligned}
&holdsAt(cwConflict(Subj, Target1, Action1, Target2, Action2), Tm) \leftarrow \\
&\quad holdsAt(permit(Subj, operation(Target1, Action1)), T1) \wedge \\
&\quad holdsAt(permit(Subj, operation(Target1, Action1)), T1) \wedge \\
&\quad holdsAt(permit(Subj, operation(Target2, Action2)), T2) \wedge \\
&\quad conflictingOps(conflictDuty, Ops) \wedge \\
&\quad Target1 != Target2 \wedge \\
&\quad memberOf(operation(Target1, Action1), Ops) \wedge \\
&\quad memberOf(operation(Target2, Action2), Ops) \wedge \\
&\quad T1 =< T2 =< Tm.
\end{aligned}
$$

Figure 2.14: Custom policy action conflict description axiom (Bandara et al., 2003).

A sample piece of code from Bandara et al. (2003) that tests for a simple application independent conflict is depicted in figure 2.13. The key word *holdsAt* is a phrase in EC and means that a specific fact is true at a given time (*Tm*). Therefore, it states that an *authConflict* holds if there exists a permit policy and a deny policy that reference the same subject (*Subj*) and the same operations (*Op*) at the same time (*Tm*). This statement does not refer to the model of the system and so is application independent.

A more complex piece of code from Bandara et al. (2003) that tests for an application specific conflict is depicted in figure 2.14. The type of conflict being detected is where the subjects are equal, the targets are different and the actions are also different, but in this application the specified actions are explicitly defined to conflict. The line referring to *conflictingOps* is model based information and is defined per application. The contrast here with language based policy conflict analysis is that the semantics of the operations being performed are not explicitly defined to conflict and therefore must be implicitly encoded into any algorithm used to detect for conflict. The approach presented by Bandara et al. (2003) decouples the requirement for implicit knowledge. This is especially useful, as the interaction among the actions is known to conflict by the system administrator, so that they can be encoded onto the model.

More recent research by Charalambides et al. (2005, 2006) explored similar concepts as those described by Bandara et al. (2003) but in the application of managing DiffServ aware networks. They defined a structural and behavioural model using EC that was based on the TEQUILA framework (Tequila IST, 2002). They propose the use of abductive reasoning to explain the sequence of actions that must occur in order for a specific conflict case to occur, thus also establishing its potential occurrence at runtime. They also describe a set of application specific conflicts that are specifically defined for their application. For example, a conflict occurs when there is an over provisioning of bandwidth after a particular sequence of actions are executed. The models defined using EC are static and cannot be readily altered at runtime; therefore, the system cannot be extended to cover more applications without being re-designed.

Chomicki et al. (2003) investigate the incorporation of so called *event monitors* and *actions monitors* to represent application specific information that should be taken into account when determining a case for policy conflict. The model for policy that Chomicki et al. (2003) considers is ECA and the language they use is Policy Description Language (PDL) as presented by Lobo et al. (1999). The approach they take to detect policy conflict is to determine if a specific sequence of actions are being executed together. The sequence of actions is specified in the *action monitor* list and represents a list of actions that should not be executed simultaneously. The *event monitor* offers a similar approach to the previous one but instead takes a more preventative approach. If a sequence of events as specified in the *event monitor* list occurs, then a conflict may potentially occur. In reaction to the occurrence of the highlighted events preventative action can be taken to prevent a conflict from occurring. The approach they takes depends on the system administer knowing the correct sequence of actions or events that lead to conflict. This may be difficult to know especially if the system functionality is continuously being extended.

Kikuchi et al. (2007) use a temporal logic to represent the behaviour of the target managed system separate to policy. They then employ model checking techniques to ensure that the specified policies do not elicit bad behaviour from the target managed system. Instead of focusing on detecting conflict among policies they are interested in validation and verification of policies. Validation is concerned with analysing individual policies to ensure that they do not contradict the intended behaviour of the system. Verification is concerned with ensuring the policies perform as planed when taking into account the effect other policies have on the system.

Both Baliosian and Serrat (2004) and Vidales et al. (2005) make use of finite state transducers to represent the behaviour of policies. The motivation of the work is to investigate if existing processes and algorithms designed to combine finite state transducers can be reused to discover and potentially resolve policy conflicts. They developed a spe-

$$\begin{array}{ll} policy\{id =' A' & policy\{id =' B' \\ \quad event\{8am\} & \quad event\{8am\} \\ \quad target\{/terminals\} & \quad target\{/DHCP\} \\ \quad action\{enable()\} & \quad action\{disable(); update()\} \\ \} & \} \end{array}$$

Figure 2.15: Sample conflicting policies (Kempter and Danciu, 2005).

cific formal representation based on temporal logic and finite state transducers that can model policies specified in the Ponder policy language. The main novel contribution of their research is the "tautness" functions that can indicate a precedence ordering among policies by examining the components of the policies (events, conditions and actions). The algorithms and processes developed for policy conflict analysis make heavy use of the tautness function. However, the functionality and specification of what the tautness function measures between two policies can be defined per-application. Therefore, the algorithms and processes can be re-used for different applications where the tautness function can be customised. This approach puts the focus on defining complex tautness functions which is still a very difficult problem.

The specification of the system structure in the above listed approaches used predominately non-standard methods. The use of non-standard methods of system descriptions limits the widespread adoption of the work. In the next section, standards based approaches are discussed that make use of standardised information modeling techniques. Following a standard way of representing a system information model will not only improve uptake of the techniques developed, but will also encourage reuse and tool development.

#### 2.2.4.2 UML approach

Kempter and Danciu (2005) investigate the use of information models to be used as input to a policy conflict analysis process. They argue that policy conflicts can exist, specific to a target system, which may be undetectable when deployed policies are analysed using previous approaches (based on language analysis only). They contend that this occurs because implicit knowledge about the system is unavailable to the conflict analysis process. Representing implicit knowledge about a system explicitly in UML is central to their approach to policy conflict analysis. The policy model they assume is ECA.

According to their paper, the policies depicted in figure 2.15 represent a conflict that is undetectable using language based policy analysis. The policies are triggered simultane-

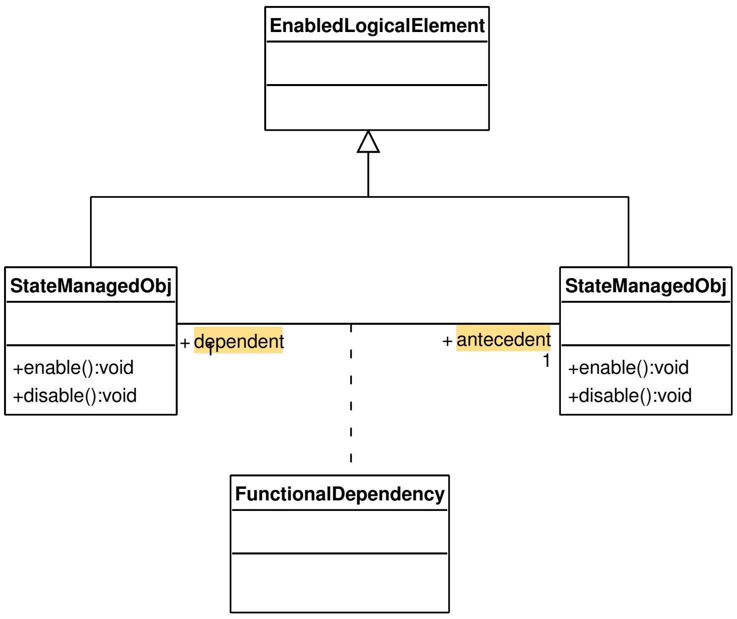


Figure 2.16: UML model of functional dependency (Shankar et al., 2005*a*).

ously but refer to distinct target managed entities. However there is a relationship between the target of policy A and policy B, in that the terminals are functionally dependent on the Dynamic Host Configuration Protocol (DHCP) server. The dependency is that the terminals cannot be enabled unless the DHCP server is also enabled. This implicit information is not modelled in the policies and therefore cannot be taken into consideration during policy analysis if the approach is based on language construct analysis only. The solution offered is to incorporate information defined within a system information model into the policy conflict analysis process. By considering invariants during policy analysis, constraints imposed by the model can affect the outcome. Therefore, the implicit relationships described for the policies depicted in figure 2.15 are modelled as an invariant in the model of the system. They go on to define that a conflict occurs when two or more policies breach the invariants imposed by the model of the system. They also consider pre- and post-conditions defined for actions that can be performed in the system, where breaches in post-conditions can also be defined as a conflict. Figure 2.16 illustrates the functionalDependency class that is an association class between two *stateManagedObj* classes. The Object Constraint Language (OMG, 2008) can be now used to further express the meaning of the functional dependency to ensure that no state managed object can be disabled while it is functionally dependent on another state managed object.

UML based information models as described in section 2.1.3 also have the added advantage of supporting invariants, preconditions and postconditions via OCL. Shankar et al. (2005*a*) investigate the use of OCL to augment policy analysis for pervasive computing environments. They define policies as ECA-P (postcondition) where a postcondition is specified along with the policy. The postcondition then specifies the values the objects' attributes should have once the policy has been enforced. They defined a conflict to occur if two or more policies yield contradicting postconditions. The algorithm they define to detect for policy conflict reduces to determining if postconditions contradict or not. The approach they take reduces the requirement on knowing which actions conflict beforehand. However, their approach does not take into account constraints concerning the state the system can be in. Invariants and action pre-conditions can be used to represent such information, failing to do so ignores the limitations in functionality of the managed system.

#### 2.2.4.3 Summary of Information Model based Policy Conflict Analysis

The main advantage of model based approach is that the information that describes the system is separated from the policies that define the behaviour of the system. The added information can be leveraged for use in policy analysis processes. Therefore conflict analysis algorithms can be designed to be application agnostic, taking application specific information in from an information model. Policies can be defined over multiple abstractions of a single managed system if the information model supports it. The added advantage here is that more complex entities can be represented at a higher level of abstraction, subsequently more expressive and powerful policy conflict analysis processes can be developed.

The disadvantages of using such an approach is that there is an overhead involved in designing the system information model, however this can be traded off against the return in more powerful analysis processes. The information model is relatively static, in that once it has been designed it cannot be readily altered if new information about the structure of the system becomes known. The model may need to be re-engineered along with supported tools. Another disadvantage is that when multiple administrative domains need to distribute the management of the communications network, which invariably occurs, they must agree on a common information model. Although standards aim to eliminate this, they can only encourage uptake.

### 2.2.5 Ontology based Policy Conflict Analysis

Ontology based policy conflict analysis aims to take advantage of the ability of ontologies to describe and reason about the semantics of the target managed system. As opposed to the information available to language based policy analysis, ontology based approaches can be made to leverage relationships defined and inferred between language constructs. Ontologies also offer a more expressive method of describing a target managed system over UML as there is no formal reasoning capabilities built into UML. Following is a review of the most popular ontology based policy approaches and how they make use of reasoning techniques to improve policy conflict analysis.

#### 2.2.5.1 Description Logic (DL) approach

The KAoS policy language and an accompanying set of policy services were developed by Uszok et al. (2003). KAoS is designed to manage agent based software systems but can be extended to manage grid based and web based services. It uses the DARPA Agent Markup Language, W3C (2001), a description-logic-based ontology language to describe the structure of the target managed system and to describe policies. By basing the languages (structural and policy) on DAML it allows KAoS to be inherently extensible as it is based on an extensible technology. KAoS policies are similar to Ponder policies as KAoS also has notions of authorisation and obligation. Currently, policy analysis for KAoS is capable of only detecting application independent conflict. Their approach allows for the use of inferencing and subsumption during the evaluation and enforcement of policies as KAoS is integrated with the Java Theorem Prover (JTP, 2008) A significant advantage of this is that KAoS has the ability to reason over different abstractions of policies to infer specific relationships.

Campbell and Turner (2007) investigate the use of OWL based ontologies to describe a policy language for a call control system. The policy language they developed is called APPEL. The APPEL policy language is built to be applicable to any domain so long as a domain specific ontology is defined and is compatible with the APPEL policy language, which is based on the ECA policy paradigm and is susceptible to application specific conflicts. The approach taken to detect conflict is similar to that proposed in Chomicki et al. (2003, 2000), where special rules are defined to trigger when particular sequences of actions are performed simultaneously. Additionally, an optional resolution strategy can be enforced to prevent the policy conflict from affecting the managed system. The main advantage of using ontologies within APPEL is that the language can be configured to operate for specific domains, and their method of conflict analysis can be configured. By extending the base ontology to include concepts from the specific target application, the policy language can be tailored, and their conflict analysis rules can be extended to cover the introduced concepts.

Verlaenen et al. (2007*b*) propose a generic policy model based on an ontology that is designed to be extended for use in any policy application by extending the generic ontology. They offer an example of a telecoms policy scenario, where the information describing the managed system is described in an ontology that is subsequently linked to the generic policy model. Policy conflict analysis is not strictly covered in their paper but they specify that the existence of the domain ontology is critical to supporting policy conflict analysis.

Determining the similarity of policies can aid in the policy conflict analysis between policies deployed to different management domains, as Lin et al. (2007) demonstrate. They propose to use policy similarity algorithms as a fast check to determine how similar two policies are, so that policies can be rapidly eliminated from more extensive policy analysis. The policy similarity process analyses two policies and with the aid of ontological representations of the two policies can quickly establish a value representing policy similarity. Their approach demonstrates that there is a strong need to reduce the complexity associated with policy analysis processes that are based on computationally expensive analysis algorithms.

#### 2.2.5.2 Logic Programming (LP) approach

The Rei policy language, presented by Kagal et al. (2003), is described as a policy language for pervasive computing environments. Rei is defined using OWL-Lite and is based on logic programming principles as opposed to description logic principles. For this reason, the inherent reasoning capabilities of OWL are not used; instead, the policies are transformed into Prolog and reasoning is performed in a logic programming system. Rei dictates that the description of the domain over which policy is being defined is also defined using OWL; therefore, the policy language can be attached to different domains. It follows the deontic concepts as proposed in Ponder (Damianou et al., 2001), and therefore suffers from similar forms of policy conflicts. Rei uses meta policies to describe how policy conflict should be resolved; for example, always prefer authorisation over prohibition in a conflict situation. Application specific policy conflict is not addressed in Rei explicitly; however it does include concepts of preconditions and postconditions of policy actions, and so may be extended to support application specific conflicts.

#### 2.2.5.3 Rule based approach

Kaviani et al. (2007*b*,*a*) describe recent work carried out by the REWERSE (2008) Framework Programme 6 (European Comission, 2008) Network of Excellence project related to representing policies using rule languages. They proposed a rule language named R2ML and they provide mappings form R2ML to Rei and KAoS policies for interoperability. Their primary motivation for this work is to combine the properties of description logic based approaches with logic programming based approaches. They claim that by doing this the resulting approach is able to perform better policy conflict analysis and the policies are easier to enforce is distributed domains where multiple policy languages are being used. The approach dose not offer any new methods of detecting policy conflict, it just extends the reach of existing approaches to detecting policy conflict among multiple policy languages.

Verlaenen et al. (2007*a*) propose the use of a hybrid policy analysis engine that makes use of description logic reasoning and logic programming reasoning to establish relationships among policies, including policy conflict. The types of relationships among policies that they are interested in establishing are as follows:

- Policy equivalence: policy A is equivalent to policy B.
- Policy containment: if policy A's condition is met then policy B's condition is also met.
- Policy incompatibility: if policy A's condition is met then policy B's condition cannot be met.
- Policy conflict: policies A and B are in conflict if they cannot both be satisfied simultaneously.
- Policy incoherence: policy A cannot be triggered or ever evaluated to true.
- Dominance checking: if policy B were removed will the system behaviour be altered.
- Coverage checking: are all possible conditions met by atleast one policy

- Policy combination: where two or more policies can be replaced with fewer policies, where the resulting behaviour is the same.

- Policy deduction: where a policy can be derived from a set of policies.

Their approach involves the description of custom rules that provide more powerful reasoning than ontologies alone would provide. By more powerful reasoning they argue that inorder to establish the forms of policy relationships as described above, customised rules are required that ontologies on their own do not provide. For example, checking a value constraint on each element in a collection, or checking the non-existence of an element in a collection. They offer an implementation where each relationship can be described in a logic programming language called Drools (2008), which is a forward chaining rule engine. Their research can be contrasted to the research presented by Agrawal et al. (2005) who determine similar relationships but focus specifically on the Boolean conditions of the policy. However, Agrawal et al. (2005) do not take into account ontological relationships to aid in establishing policy relationships.

#### 2.2.5.4 Summary of Ontology based Policy Conflict Analysis

Using ontologies to describe the structure of a system allows policy analysis processes to take advantage of the capabilities ontologies offer to improve analysis functionality. Ontologies offer a method of expressing more extensive semantic information that can be extended at runtime, therefore making the analysis processes more dynamic. Ontologies also support the integration of policies where concepts defined in one ontology can be mapped to concepts in another. This enables system designers to bridge the semantic difference between independently created system ontologies. The classification and subsumption algorithms developed to support reasoning in ontologies can be reused to aid in the analysis or relationships among policies, and concepts defined in the system ontology.

The down side of using ontologies to support policy conflict analysis is more technical for the moment then conceptual. As the use of ontologies within network management is recent, the efficiency of classification and subsumption has not been developed to operate at runtime, or on a large scale. There are also no standards for representing policies in an ontology, although there has been considerable efforts in this direction. This issue may be overcome by using the semantic mapping capabilities of ontologies as discussed by Lin et al. (2007); Cridlig et al. (2007), however there is currently no standard way of transforming between policy languages.

## 2.3 Summary and Conclusions

This chapter reviewed the background and related works representing the state of the art in policy based management and specifically policy conflict analysis. The use of information modelling was highlighted to demonstrate its use in improving the management processes of the communications network by abstracting the heterogeneity of the network. PBM research focuses on investigating the critical issues of policy analysis, such as refinement and conflict analysis, but solutions remain application specific and heterogeneous. Crucially, there are no integrated approaches that take account of multiple levels of policy that exist in a policy continuum.

Language based policy conflict analysis is best suited to applications where the enforcement semantics of policies are well understood. This is observed by Hari et al. (2000), Al-Shaer and Hamed (2003, 2004*a*,*b*), Al-Shaer et al. (2005) and Bandara et al. (2006*b*), as the algorithms developed depend on some sort of implicit knowledge of the target managed system, for example, that the policies are ordered and only one in a set is enforced. This approach can be very efficient for centralised applications, but may not scale to distributed systems without augmenting the algorithms (Al-Shaer et al., 2005). This approach is useful when the policy language used exhibits well defined (but implied) semantics.

Information model based policy conflict analysis is used with higher level policy languages (i.e., not device specific), where the target application or target managed system cannot not be known at design time. The application specific information is described separately to the policy language and is used to augment policy analysis processes, as presented by Bandara et al. (2003); Charalambides et al. (2005, 2006); Chomicki et al. (2000, 2003); Kikuchi et al. (2007). Information critical to enhancing policy conflict analysis processes are the relationships defined in the model and the preconditions, postconditions and invariants defined for policy actions. Specifically, constraint information is used to aid in determining the potential states of the system (Kempter and Danciu, 2005). Information model based approaches may be scalable; however, the system model cannot be readily adapted at runtime, nor can the policy language be modified. Approaches that make extensive use of an information model are readily able to detect application specific conflicts as well as application (domain) independent conflicts. A drawback of current approaches is that the conflict detection algorithms are designed to retrieve a well-defined set of information from the information model, and thus need to be coupled to a specific type of information model. This limitation reduces that more widespread use of the algorithms as existing information needs to be re-coded into proprietary formats.

Ontology based policy conflict analysis offers the maximum flexibility to policy based management systems. As the application specific information is described in an inherently extensible technology, this information can be readily extended at runtime to extend the coverage of the PBM system. Policy languages described using ontologies can also be extended to describe policies specifically tied to the target application (Uszok et al., 2003; Campbell and Turner, 2007; Verlaenen et al., 2007*b*; Kagal et al., 2003). The use of ontologies falls into two categories: those that augment the process with rules, versus those that solely rely on the reasoning capabilities of ontologies. The semantic web rule language (SWRL) is extended from OWL, and illustrates the momentum more general purpose reasoning engines are gaining. The significance of this can be observed for example, in Verlaenen et al. (2007*a*) who indicate that policy analysis can be enhanced by using more expressive rule languages. However, research into the use of ontologies for policy analysis is not extensive and typically does not always address application specific conflicts (Uszok et al., 2003; Kagal et al., 2003).

The reviewed policy conflict analysis approaches fail to adequately address the integration of the policy continuum, and thus fail to deliver a language integrated (or policy continuum level integrated) approach to policy based management in an efficient way. Furthermore, the thesis proposes that policy conflict analysis should be defined within the policy authoring process and not as a separate process. The following hypothesis, presented originally in chapter 1 is reiterated:

*Hypothesis: The definition of a policy authoring process that harnesses existing knowledge bases, which is cognisant of the intricate requirements associated to the policy continuum must be developed to control the efficient operation of policy conflict analysis.*

The following set of requirements for policy continuum based policy conflict analysis has subsequently been derived form the examination of the shortcomings of the related work and the posed research question from chapter 1.

**Requirements for a policy continuum based conflict analysis process** As the policy continuum must represent policies at varying levels of abstraction/granularity, there must also exist policy conflict analysis processes that can operate and coordinate across multiple levels of the policy continuum. This gives rise to the following high level requirements for a policy conflict analysis process for the policy continuum.

1. Be able to meet the needs of multiple constituencies of policy authors
2. Be extensible so that it can be used in multiple policy applications
3. Be information model based to take advantage of a system knowledge and to ensure that the process is flexible enough to be used with various type of policies
4. Be adaptable to the varying application requirements that may be difficult to represent in information models
5. Be computationally efficient to meet the demanding performance requirements of large scale communications networks where large numbers of policies are considered

These requirements arise due to the nature of the managed system that the policy continuum is managing. The multiple views of the managed system are from the perspective of users with business concerns, to administrators concerned with network performance and security. Therefore, a policy conflict analysis approach targeted at a single layer of the policy continuum is not desirable as conflicts can occur at multiple levels of the policy continuum. This is why an approach that is integrated across the levels of the policy continuum is preferable. Multiple constituencies of users will continuously be updating the policies of the policy continuum. As policies at one level can affect policies at multiple levels, policy conflict analysis should be capable of being performed not just from high level to low level policies, but also from low level to high level policies. Currently no policy conflict analysis processes take these issues related to the policy continuum into consideration.

Scalability of policy conflict analysis algorithms presented (i.e., their ability to handle high numbers of policies), have not been evaluated for the majority of cases. An exception is Al-Shaer et al. (2005); however, the approach they present is described specifically for firewall policy conflict and the performance of the algorithm is not presented. Another exception is the work on policy similarity carried out by Lin et al. (2007), who introduce a filtering step to eliminate policies from being considered for further expensive analysis. Lacking in current implementations of policy conflict analysis is a pre-analysis of deployed policies so that an appropriate subset of policies can be selected for more extensive analysis. Such a process should be integrated into a generic process for policy analysis for a policy continuum.

The requirements specified above fall within the scope of the research questions outlined in chapter 1. Question 1 states "*How can a policy authoring process be defined that incorporates policy conflict analysis, which is specifically targeted at multiple constituencies of policy authors?*" The requirements indicate that the defined conflict analysis process be able to relate policies defined for different applications (multiple stakeholders), associated to different policy continuum levels and defined by different policy authors together.

Question 2 states "*What processes and algorithms need to be developed so that existing knowledge bases can be harnessed to aid the policy authoring and policy conflict analysis processes?*" The requirements outlined fall within the scope of this research question, as they clearly state that system knowledge and ontologies be integrated into and be leveraged by policy conflict analysis. Question 3 states *"How can a policy conflict analysis process be developed that is independent of the nature of the policies?"*. The requirements outlined fall within the scope of this research question, as the requirements state that the conflict analysis process be extensible to many applications, be usable by multiple policy authors with varying concerns and make use of language based and ontology based analysis algorithms to maximise the use of system knowledge. Question 4 states *"How can the results produced by the processes and algorithms developed for policy authoring and policy conflict analysis be leveraged to be make the processes and algorithms more efficient when large numbers of policies are being considered?".* The requirements outlined also fall with the scope of this research question, the need for efficiency is paramount when faced with large numbers of policies are is the case for large communications networks.

This thesis presents a generic and extensible policy conflict analysis process and policy selection algorithm that are both part of an encompassing policy authoring process. These algorithms are defined in reference to a policy continuum and are tightly coupled to system information models and ontologies. An approach to policy conflict analysis is required that can be used across any application; such a conflict analysis process can take advantage of the benefits offered by the approaches presented in this chapter. Central to many of the approaches presented in this chapter is the requirement of being able to ascertain a set of relationships among policies. This is demonstrated across all approaches presented in this chapter.

In order to integrate the policy conflict analysis process with the multiple levels of related policies defined in the policy continuum, a process defined to coordinate the analysis is required. The policy continuum and associated analysis processes are defined in chapter 3, where the policy conflict analysis algorithm developed, as a central component to the policy conflict analysis process, is presented in chapter 4. The policy selection process is presented in chapter 5 where ontologies are used extensively. Finally, an efficient enhancement to the selection process is presented in chapter 6 to deal with large sets of deployed policies.

# Chapter 3

# Policy Continuum Model and Policy Authoring Process

The notion of a policy continuum, as presented by Strassner (2003), should be a fundamental component of any policy based management system, but as of yet it has no formal operational semantics. A policy continuum model and accompanying policy authoring process are presented in this chapter. This process demonstrates the key properties that set a policy based management approach based on using a policy continuum apart from other policy based management approaches. There are currently two challenges facing the realisation of the policy continuum:

1. There is currently no formal (structural or operational) representation of the policy continuum, which makes it difficult for standard tools to be developed for policy authors that wish to use it, and

2. Policy conflict analysis across multiple levels of the policy continuum during the policy authoring process has not been addressed. In particular, the realisation of such an analysis process requires the integration of policy refinement and policy verification processes.

The objective of this chapter is to define a formal policy continuum model and a formal policy authoring process that can be used with the policy continuum to guide the authoring and analysis of policies at compile time. The DEN-ng information model is the starting point of the model developed in this chapter to represent the policy continuum. In order to achieve the objective set out, a portion of the DEN-ng information model referencing the structure of policy and the policy continuum has to be extended so that an effective policy authoring process can be defined. The policy model is derived from the existing DEN-ng policy model as presented in chapter 2 and can be extended to meet the needs of specialised applications.

The chapter is structured as follows. Section 3.1 describes extensions to the DEN-ng policy information model to allow for more effective policy authoring in the context of the policy continuum. Section 3.2 introduces a formal notation to describe the structure of policy and the policy continuum, and illustrates the use of the formal representation of these concepts to develop processes and algorithms to describe policy authoring and conflict analysis. Finally, the contribution of this chapter is summarised.

## 3.1 Policy Model Extensions for Conflict Analysis

This section describes the DEN-ng policy model enhancements to model the extra information required to represent policies in the policy continuum. There are limitations to the existing DEN-ng policy model that reduce the flexibility of its use in policy analysis processes, specifically in representing policies that describe deontic behaviour.

### 3.1.1 Extensions to the DEN-ng Policy Model

DEN-ng describes policy authoring and conflict analysis processes at a conceptual level. For example, it specifies that when a policy is being authored from a policy editor Graphical User Interface (GUI), the author must first authenticate, and must then be authorised to edit the policy. Only then can a new policy be created or an existing policy be modified (Fu and Strassner, 2006). The DEN-ng model also describes two specific policy application classes that should be used to aid in the detection and resolution of policy conflicts. The `LocalServerConflictDetectionComponent` is used to examine policies for conflict on a global level, where a conflict is defined to occur when two policies can be triggers and satisified at the same time and where their actions contradict. The `PDPConflictDetectionComponent` is used to detect conflicts at a device specific level, where the capabilities of devices should be taken into account during the conflict detection process. Note that DEN-ng does not offer any conflict detection or resolution algorithms. Essentially, the DEN-ng policy model does not specify *how* policy conflict analysis should be carried out, only *where* it should be carried out.

The definition of policy conflict by Strassner (2003) is:

*"A Policy conflict is when two of more policies applying to an overlapping set of managed entities are simultaneously triggered, and there conditions are satisfied, but their actions are contradictory."*

This definition does not take into account the requirements of different types of applications, services and resources, where different situations arise to establish policy conflict. An extension to the original DEN-ng policy information model is now defined so that policy conflict detection can be supported more effectively, and so that more policy application types can be catered for. Specifically, deontic and access control policies are explicitly modelled so that they can be represented at the business level of the policy continuum. Currently, it is cumbersome to define such types of business level policies using the existing policy model.

The extensions to the DEN-ng policy information model are to the `PolicyRule, Policy` and `PolicyApplication` classes. The first subtle change is the renaming of the `Policy` class to `PolicyConcept`. This change was done to avoid confusing policy authors as well as ontology-based applications about what a policy is. This new class name conveys that all of its subclasses are in fact concepts relating to policy and not complete "policies". Without this change users could confuse the Policy and PolicyRule classes, and think (for example) that they are used interchangeably- this is incorrect.

The next extension is to the `PolicyRule` class. It is important to remember that there are many different types of policy rules. Besides DEN-ng, no policy model to date has actually tried to represent more than one basic type of policy (from a structural point of view). The DEN-ng model must be able to represent all types of relevant policy rules, from business policy rules to network policy rules. Hence, the specific structural representation of a policy rule must be separated from the label "PolicyRule". For example, a `PolicyRule` is represented using a traditional event-condition-action form, then how is a policy rule from the DMTF or IETF represented, since both only use condition-action representations and do not contain an event. A new class type is defined called `PolicyRuleStructure` which extends `PolicyConcept`, this can be then extended to describe policies that take different forms. Therefore, the `ECAPolicyRule` class is built to define a particular type of `PolicyRuleStucture` - one that has an {event, condition, action} structure; this is depicted in figure 3.1. This class is essentially identical to the previous PolicyRule as defined in the original DEN-ng model with one important difference: there is no explicit

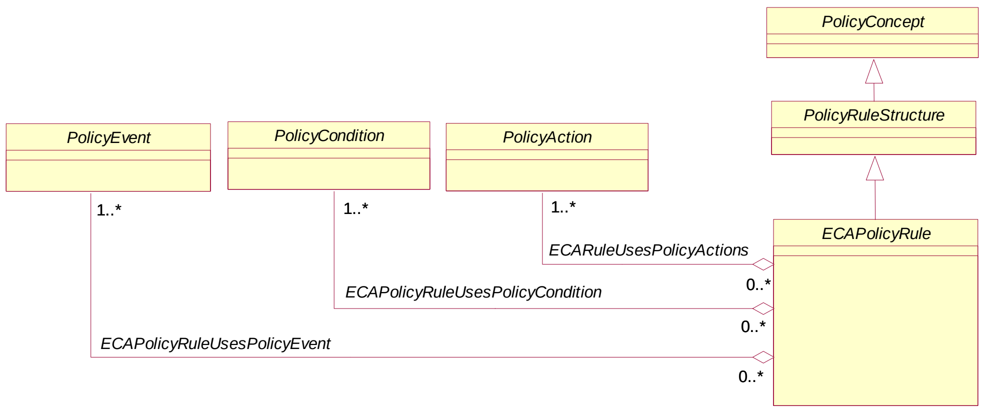


Figure 3.1: ECAPolicyRule.

relationship to the managed entities that partake in the operation of the policy (i.e. the subject and target of the policy).

The next extension is to introduce explicit links to `PolicySubject` and `PolicyTarget`. The objective of applying policy is either to maintain the current state of the `PolicyTarget`, or to transition the `PolicyTarget` to a new state. In the original DEN-ng model, these classes were directly associated with a `PolicyRule`. In the modified DEN-ng policy model, this is no longer done, as this would have the unfortunate effect of binding the usage of `PolicySubject` and `PolicyTarget` to an `ECAPolicyRule`. Instead, depicted in figure 3.2, a `ManagementPolicy` class is introduced to separate the relationships between the `ECAPolicyRule` and the `PolicySubject` and `PolicyTarget`. One of the main advantages of this is that other types of policies can be created that may be only subject based or target based. In addition, this decouples the notion of `PolicySubject` and `PolicyTarget` from the representation of the `ECAPolicyRule`. In order to represent popular policy types such as deontic policies, new `ManagementPolicy` subclasses are created. Policy conflict analysis processes can benefit from these extensions, as previously there was no method of representing deontic policies in DEN-ng. Defining policies at varying levels of abstraction such as is required in the policy continuum demands this flexibility in the information model. A case study on conflict analysis based on deontic policies is presented in chapter 4 in section 4.3.3.

Previously, deontic policies could only be represented by extending the `PolicyRule` class. While that was possible, the functionality of the deontic policies was limited, as

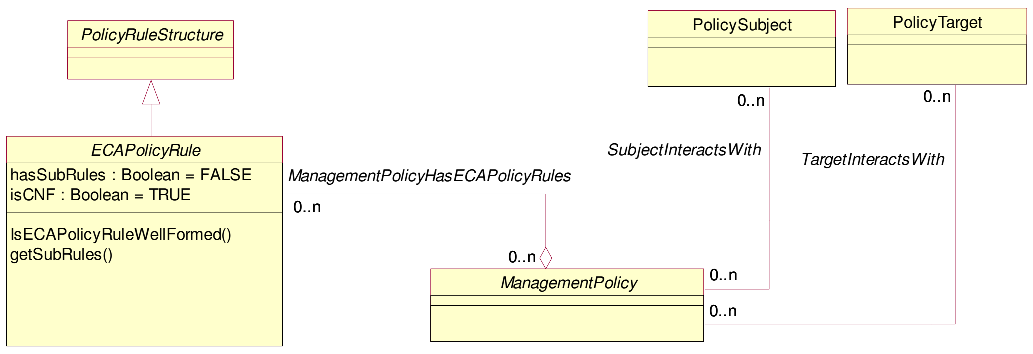


Figure 3.2: Improved ManagementPolicy.

they had to reference all policy components (events, conditions, actions, subjects and targets). However, with the introduction of the `ManagementPolicy` class, the relationships are now more flexible. Deontic policy types such as those depicted in figure 3.3 can now be readily described using the approach. A new class called `DeonticPolicy` is derived from ManagementPolicy, therefore inheriting the optional association to `PolicySubject` and `Policy`Target.

AuthorisationPolicy embodies the concept of "is permitted to" or "is allowed to" Strassner et al. (2007*a*). Deontic logicians assign the concept "may" to authorisation. A subject-based authorisation policy is a policy that is enforced by a subject, and defines the actions that a subject is permitted to perform on one or more targets. Conceptually, subject-based authorisation policies are designed to define actions that can be performed on the target by the subject that will not adversely affect the subject. In contrast, target-based authorisation policies are enforced by the target, and define the actions that a target permits a subject to perform on the target.

ProhibitionPolicy defines the set of actions that an entity is forbidden to execute on another entity Strassner et al. (2007*a*). Deontic logicians assign the concept "may not" to authorization. Hence, subject-based prohibition policies are enforced by the subject, and define actions that the subject is forbidden to perform on a target, because performing such actions would jeopardise the subject. Target-based prohibition policies are enforced by the target, and define the actions that a target forbids a subject to execute on that target.

ObligationPolicy defines the set of actions that one entity must perform on another entity Strassner et al. (2007*a*). Deontic logicians assign the concept "must" to obligation.

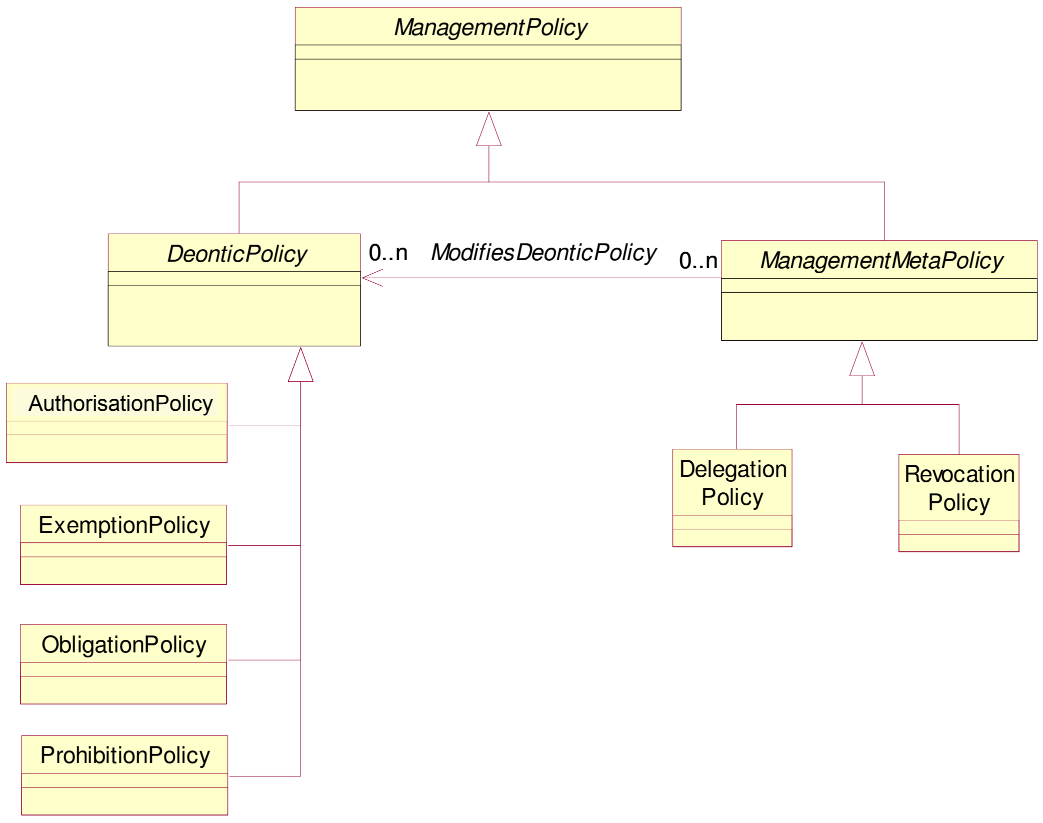


Figure 3.3: Deontic management policies.

Subject-based obligation policies are enforced by the subject, and define the set of actions that a subject must do on a target. Target-based obligation policies are enforced by the target, and define the set of actions that the subject must do on a target.

ExemptionPolicy is, literally, immunity or release from an obligation Strassner et al. (2007*a*). Deontic logicians assign the concept "need not" to exemption. Subject-based exemption policies are enforced by the subject, and define the set of actions that the subject need not perform on the target. Target-based exemption policies are enforced by the target, and define the set of actions that the subject need not perform on the target.

Delegation defines the ability for a sender to confer some function or privilege to a receiver Strassner et al. (2007*a*). Subject-based delegation policies are enforced by the subject, and apply a subject-based policy to a receiver. Similarly, a target-based delegation policy is enforced by the target, and applies a target-based policy to a receiver. Revocation policies are used to retract functionality that was previously delegated. Subject-based revocation policies are enforced by the subject, and retract a subject-based policy from a receiver. Target-based revocation policies are enforced by the target, and retract a target-based policy from a receiver.

As the DEN-ng policy model can now represent an increased amount of policy types, it can therefore be used in a wider set of policy application domains and is more appropriate for representing the various types of policies that can exist in the policy continuum. The new information can also be leveraged for policy analysis processes, in particular for policy conflict analysis.

## 3.1.2 Extensions to the DEN-ng Policy Continuum Model

Just as the model of the `Policy` and `PolicyRule` has changed, so too is the DEN-ng model of the policy continuum. Policies need to be related together across the levels of the policy continuum, for example, it must be possible to determine the system level policies associated to a given business level policy. Also it must be possible to determine those business level policies that rely on the behaviour offered by a system level policy (or policies). The existence of this relationship has been added to the DEN-ng policy continuum model and amounts to building a composite pattern to extend `ManagementPolicy` (see figure 3.4). A composite pattern is a modelling technique used to describe hierarchical structures. The base class is extended with an atomic version of the class and an composite version of the class. The composite version is associated to the base version of the class, that way

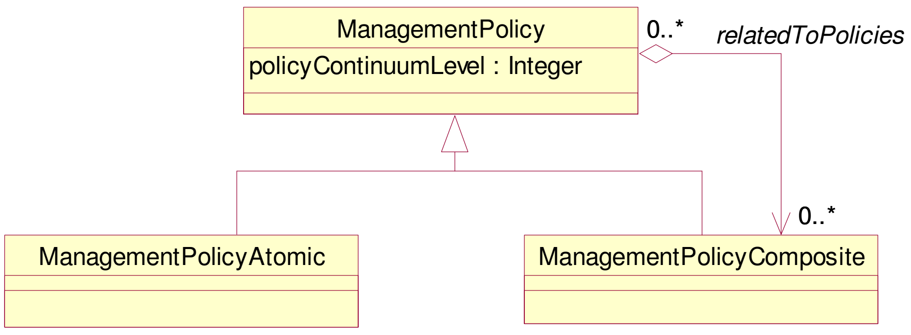


Figure 3.4: Policy Continuum aware ManagementPolicy.

is can aggregate both atomic and composite versions of the base class. The composite pattern enables the representation of hierarchical structures of the base class, and is very well suited to the representation of a policy continuum.

The name of the composite relationship is “relatedToPolicies”. It is a bi-directional relationship between the abstract class `ManagementPolicy` and the class `ManagementPolicyComposite`. As the relationship is bi-directional, policies can be related to policies at higher levels and lower levels. A `ManagementPolicyConposite` can be related to many `ManagementPolicies`, and a `ManagementPolicy` can be related to many `ManagementPolicyComposites`. The structure that can be built from this can reflect the levels of the policy continuum as any `ManagementPolicy` can also be a `ManagementPolicyComposite`. The information model now contains an attribute that can be queried and searched when a policy continuum is being used. However, the processes to support the proper use of these attributes and relationships are currently ill defined. A constraint that must be imposed on this structure that is not supported by the pattern is that if a policy is defined at the highest level of the policy continuum, it should not be related to any higher level policies, and the relationship should not be used. A decision was made therefore to set the relationship to 0..* and not 1..*. A consequence of this decision is that it is more difficult to impose that fact that a policy not at the highest level of the policy continuum must be associated to a higher level policy, effectively enforcing a 1..* association.

## 3.2 A Formal Policy Continuum Model

This section builds on the extensions to the DEN-ng information model outlined in section 3.1 to model a policy continuum using a formal notation approach. By using a formal notation, algorithms and processes for policy authoring and policy conflict analysis can be readily described from the perspective of how they operate with a policy continuum. First, the notation used - the Vienna Development Methodology (VDM) notation (Bjorner and Jones, 1978)- is described. Following that, the assumptions the approach is based on are discussed along with the base model query interface used to leverage the information model. Finally, the policy rule and policy continuum formal models are presented.

### 3.2.1 Formalisms

The notation used throughout the thesis to describe formalisms is based on the VDM formal specification language (Bjorner and Jones, 1978). VDM is an implementation independent model for describing the structure and operation of software systems. It is based on the mathematical theory of sets and maps. Sets are used in VDM to describe different types of entities that can exist in a system and can be thought of as objects in the object oriented world. For example, a set containing all entities of type ManagementPolicy can be defined. There is a very well defined method of searching and transforming the elements of this set using the formal approach taken. Maps are used to describe relationships between sets of entities. Maps are used to relate various Sets together, where the maps may have specific properties (e.g. be transitive, reflexive, symmetric). For example, a map can be defined that relates policies to events, in order to determine the set of those policies that are associated to specific events, if policy sets and event sets are defined. The notation used throughout is outlined in table 3.1.

The structural representation of a set of related entities in VDM is straightforward, once the entities are well known. However, the functions defined to transform and analyse the structural representation are of most interest. Functions can be defined using the function notation described in table 3.1. For example, a set of natural numbers (A), another set of the letters of the alphabet (B) and a map function that maps a letter to its numeric value (C), is depicted below.

Table 3.1: Notation for VDM.

| Notation | Description |
|---|---|
| $a \in S$ | $a$ is an entity belonging to the set $S$ |
| $S \backslash a$ | The set $S$ less the element $a$ |
| $S \cup a$ | The set $S$ union the element $a$ |
| $S \subset T$ | The set $S$ is a subset of the set T |
| $\mathbb{P}S$ | The power set of $S$, i.e. a set containing sets of all the permutations of the elements of the set $S$ |
| $b \in \mathbb{P}S$ | $b$ is a set of entities from the set $S$ |
| $Q = S \rightarrow T$ | $Q$ is a map that maps entities from the set $S$ to entities in the set $T$. $S$ is the domain of the map, $T$ is the range of the map |
| $dom \circ Q$ | The domain of map Q. The domain is those elements that can be input into the map $Q$ |
| $rng \circ Q$ | The range of map Q. The range is those elements that the input is mapped to the map $Q$ |
| $q(e)$ | Map index function, if e$\in dom \circ q$, then return what $e$ is mapped to in the map $q$ |
| $Q \sqcup [s \rightarrow t]$ | Like union, but instead add a new mapping to a map |
| $Q \dagger [s \rightarrow t]$ | Overrides an existing mapping |
| $(R \times S \times T)$ | A tuple consisting of an entity from each set specified |
| $\pi^i$ | Tuple index function, returns the element of a given tuple at index i |
| $\bar{\pi}^i$ | Tuple remove function, returns the tuple with the associated index removed |
| $(I \rightarrow f)$ | A map transform function, the function $f$ is applied to the range of the map. $I$ is the identity function in that the domain of the map is unchanged. |
| $(I \rightarrow \lhd(b))$ | A map transform function, the function $b$ is a Boolean function and is applied to each element of the range of the map, if a **true** is returned then the element is kept, otherwise it is discarded |
| $(I \rightarrow \bar{\lhd}(b))$ | A map transform function, the function $b$ is a Boolean function and is applied to each element of the range of the map, if a **false** is returned then the element is kept, otherwise it is discarded |
| $funct : (A) \rightarrow (B)$ | A function specification, the parameter is from the set A and the result if from the set B |
| $\triangleq$ | The function specification symbol |
| $Q^{-1}$ | The inverse map of Q |
| $\forall a \in S : f\,(a)$ | For each entity a in set S, apply function $f$ |

$$\begin{aligned} a &\in A = \mathbb{N} \\ b &\in B = \{A..Z\} \\ c &\in C = A \rightarrow B \end{aligned} \tag{3.1}$$

This is a simple example that shows how to apply the map C indexed by the associated number, and is outlined in equation (3.2). The complexity of the function is not so relevant, but the notation used to formulate the function is used through the rest of the thesis.

$$\begin{aligned} GetLetter &: (A \times C) \rightarrow B \\ GetLetter\,(a, c) &\triangleq c(a) \end{aligned} \tag{3.2}$$

### 3.2.2 Assumptions

There are key assumptions made in the development of the policy continuum model and associated algorithms and processes. The assumptions are concerned with the form and structure of the information model (noting that they are satisfied by DEN-ng)

- The information model can represent the basic concepts of object-oriented modelling. Specifically, it should model inheritence hierarchies, associations, operations and attribites. Additionally the information model should be able to represent invariants, as well as pre- and post- conditions relating to the state of attribites before and after operation invocations;
- The information model encodes knowledge regarding the structure of and relationships between managed entities. The degree to which the model reflects the actual managed system constrains the ability of algorithms developed for policy analysis.
- Policies are themselves modelled in the information model, in a manner such that their relationships with each other can with managed entities can be ascertained.
- Policies are modelled as event-condition-action tuples, with the semantics of "on event(s), if condition(s), do action(s)". Also the subjects and targets of the policies should be specified.
- Operations defined for entities within the information model correspond to actions of policies.
- Any modelling technology used should provide an interface for querying and searching the information within the information model.

The assumptions outlined are important because they restrict the type of information model that can be used to those that can represent useful information that can be queried. The query interface is important because the specified algorithms and processes need to have access to a generic interface to the information model.

### 3.2.3 Information Model Querying

This section describes some information model queries specified in VDM that may be used to look up some information about a specific class. Note that the queries are implementation independent as they are specified in VDM; therefore, for a runtime system a query language that is capable of represent queries with similar semantics to the queries presented here should be used. First, some simple sets can be defined to represent some of the information contained within the information model. A class is the basic unit of modelling for most modelling technologies; the basic structure which is defined in (3.3). The map *ClassDetails* describes that the element *cld* is a "class-to-details" mapping, where the details is a tuple of sets. The map function returns a tuple of class related details when provided with a class identifier. The basic information that can be maintained about each class is therefore its operations, properties, associations, invariants and a parent super class. All information about a class is not mandatory. For example, parent super class can be left out. Obviously, different modelling technologies may contains more information about a class; however, if for all classes a minimum set of details as listed in (3.3) is described, then regardless of the modelling method used, that modelling method is deemed compatible with the approach required for representing information models.

$$\begin{gathered} c \in Class \\ cld \in ClassDetails = \\ Class \rightarrow (\mathbb{P}Operation \times \mathbb{P}Property \times \mathbb{P}Association \times \mathbb{P}Invariant \times Class) \end{gathered} \tag{3.3}$$

Next, some information model queries are defined to illustrate the type of capabilities the query interface can provide any specified algorithms that make use of an information model.

**GetClassParent** This function (3.4) will return the parent class of a given class. Every class is at least associated with a root class by default. This function can be used to discover common parent classes between two given classes within the information model. The returned information is the fifth tuple member that the *ClassDetails* map returns when given a class identifier. In this function, the map index operator, $\pi^5$, is used to retrieve the appropriate tuple entry.

$$\begin{aligned} &GetClassParent : (Class \times ClassDetails) \rightarrow Class \\ &GetClassParent\,(c, cld) \triangleq \pi^5 \circ cld(c) \end{aligned} \tag{3.4}$$

**GetClassAssociations** This function will retrieve a list of associations for a particular class. This set may be further traversed to find an association that links to a specific class. The function uses the *ClassDetails* map function and indexes it to return the set of associations.

$$\begin{aligned} &GetClassAssociations : (Class \times ClassDetails) \rightarrow \mathbb{P}Association \\ &GetClassAssociations(c, cld) \triangleq \pi^3 \circ cld(c) \end{aligned} \tag{3.5}$$

**GetObjectClass** This function will retrieve the class associated with this specific object. This function begins to demonstrate the ability of coupling the information model with the instances defined against it. Once the object's class is retrieved, the class's details and associations can also be retrieved. This type of function can be used to facilitate model-focused approaches to defining algorithms, where the information model is used both as a design tool as well as for an inventory function that can track instances of model elements at runtime.

$$\begin{aligned} &GetObjectClass : (Object \times ObjectClass) \rightarrow Class \\ &GetObjectClass(o, objCl) \triangleq objCl(o) \end{aligned} \tag{3.6}$$

### 3.2.4 Policy Rule Formalism

This section outlines the policy model using the VDM formal notation. By defining the policy model this way, it is easier to specify algorithm that need to be able to query information about policies and their relationships with each other. A formal approach to modifying policies is required to help better describe the complex processes of policy authoring and policy conflict analysis that require the ability to modify policies.

#### 3.2.4.1 Policy Specification

For a policy language to be compatible with the algorithms and processes defined in this thesis, they must at least support the concepts defined here. The formalised policy model presented is most aligned to the `ManagementPolicy` class of the modified DEN-ng policy information model, as presented in section 3.1.1. `PolicyRuleComponents` as defined in the DEN-ng policy model are used here to illustrate the component parts of a policy rule.

The definitions depicted in equation (3.7), show that all defined policy rule components are subsets of the set *Object*, meaning that the defined policy components are sub classes of the *Object* class.

$$\begin{gathered} ob \in Object \\ e \in Event \subset Object \\ c \in Condition \subset Object \\ a \in Action \subset Object \\ t \in Target \subset Object \\ s \in Subject \subset Object \\ p \in PolicyRule \subset Object \end{gathered} \tag{3.7}$$

Next the basic components of a policy rule, equation (3.8), are defined. A policy rule is made up of sets of events, conditions, actions, subjects and targets as dictated in the DEN-ng information model. A *PolicyMap* is used to associate this information to a policy rule. When this map is applied to a policy rule, a tuple of information is returned. The associated tuple contains sets of policy rule component identifiers, therefore policy rule components are defined separate to policy rules and can be shared among policy rules.

Following is a description of formally specified functions that can be implemented to create and modify a single policy. In essence, the `PolicyRuleComponent` association of the `PolicyRule` class in DEN-ng is represented here as a tuple. In doing so, some structural information is lost, but this is traded against improved operational semantics derived from the formal function defined from here on. The following listed functions should be used by external processes and algorithms as an interface to create and modify policies used to manage the target system (communications network).

$$\begin{aligned} pm \in PolicyMap = PolicyRule \rightarrow \\ (\mathbb{P}Event \times \mathbb{P}Condition \times \mathbb{P}Action \times \mathbb{P}Subject \times \mathbb{P}Target) \end{aligned} \tag{3.8}$$

**CreatePolicyRule** When a policy rule is created using the function defined in (3.9), it is not associated with any sets of events, conditions, actions, subjects or targets. When a policy rule is initially created it is just a container and is incapable of invoking any sort of behaviour within the managed system. In the function, $p$ is the new policy rule identified and $pm$ is the policy information map, *PolicyMap*. Therefore the function results modifies *PolicyMap*, by adding a new mapping where $p$ is mapped to a tuple. The component's that comprise $p$'s tuple are empty sets, demonstrating that there are currently no events, conditions, actions, subjects or targets associated to this new policy rule.

$$\begin{aligned} &CreatePolicyRule : (PolicyRule \times PolicyMap) \rightarrow (PolicyMap) \\ &CreatePolicyRule\,(p, pm) \triangleq [p \rightarrow (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset)] \sqcup pm \end{aligned} \tag{3.9}$$

**AddPolicyEventToPolicyRule** An event can be added to an existing policy rule. This function is useful when the policy author needs to add a specific trigger to the policy rule so that it will be evaluated in different circumstances. In this function, $se$ refers to a new set of policy events. The union of this new set of policy events, with the existing set of policy events creates an updated set of policy events for this policy rule. The function effectively copies all existing components of the policy without modification. The operator used in this circumstance is the map override operator which will override the tuple that policy $p$ is currently mapped to and subsequently maps it to a new tuple of components. Note that adding an event to a policy rule makes that policy rule more generic, and hence care should be taken to redo any previous policy analysis to ensure that no new conflicts have been added.

$$\begin{aligned} &AddPolicyEventToPolicyRule : (PolicyRule \times \mathbb{P}Event) \rightarrow (PolicyMap \rightarrow PolicyMap) \\ &AddPolicyEventToPolicyRule\,(p, se)\,pm \triangleq pm\dagger \left[ p \rightarrow \begin{pmatrix} se \cup \pi^1 \circ pm(p), \\ \pi^2 \circ pm(p), \\ \pi^3 \circ pm(p), \\ \pi^4 \circ pm(p) \\ \pi^5 \circ pm(p) \end{pmatrix} \right] \end{aligned} \tag{3.10}$$

**RemovePolicyEventFromPolicyRule** Similarly, an event can be removed from an existing policy, by modifying only the set of events associated with that policy rule. This function may be used if the policy author needs to reduce the circumstances for which the policy rule is evaluated. Here the set of events are subtracted from the existing set of events currently associated to the policy rule, thereby reducing the event set. Note that removing an event from a policy rule makes the policy rule less generic and hence care should be taken to redo any previous policy analysis to ensure that no new conflicts have been added.

$$
\begin{aligned}
&\mathit{RemovePolicyEventFromPolicyRule} : (\mathit{PolicyRule} \times \mathbb{P}\mathit{Event}) \rightarrow (\mathit{PolicyMap} \rightarrow \mathit{PolicyMap}) \\
&\mathit{RemovePolicyEventFromPolicyRule}\,(p, se)\,pm \triangleq pm\dagger \left[ p \rightarrow \begin{pmatrix} se \backslash \pi^1 \circ pm(p), \\ \pi^2 \circ pm(p), \\ \pi^3 \circ pm(p), \\ \pi^4 \circ pm(p), \\ \pi^5 \circ pm(p), \end{pmatrix} \right]
\end{aligned}
\tag{3.11}
$$

**AddPolicyConditionToPolicyRule** This function adds a condition element to the set of conditions associated to the policy rule, therefore constraining its applicability. A potential consequence of this operation is that it may logically contradict with an existing policy condition, therefore making the policy rule unsatisfiable.

$$
\begin{aligned}
&\mathit{AddPolicyConditionToPolicyRule} : (\mathit{PolicyRule} \times \mathbb{P}\mathit{Condition}) \rightarrow (\mathit{PolicyMap} \rightarrow \mathit{PolicyMap}) \\
&\mathit{AddPolicyConditionToPolicyRule}\,(p, sc)\,pm \triangleq pm\dagger \left[ p \rightarrow \begin{pmatrix} \pi^1 \circ pm(p), \\ sc \cup \pi^2 \circ pm(p), \\ \pi^3 \circ pm(p), \\ \pi^4 \circ pm(p), \\ \pi^5 \circ pm(p) \end{pmatrix} \right]
\end{aligned}
\tag{3.12}
$$

**RemovePolicyConditionFromPolicyRule** This function removes a condition element from the set of conditions associated to the policy, therefore reducing the number of conditions that must succeed before this policy rule can be executed. Care should be taken to redo any previous policy analysis to ensure that no new conflicts have been added by removing this policy condition.

$$RemovePolicyConditionFromPolicyRule : (Policy \times \mathbb{P}Condition) \rightarrow (PolicyMap \rightarrow PolicyMap)$$

$$RemovePolicyConditionFromPolicyRule\,(p, sc)\,pm \triangleq pm\dagger \left[ p \rightarrow \begin{pmatrix} \pi^1 \circ pm(p), \\ sc \backslash \pi^2 \circ pm(p), \\ \pi^3 \circ pm(p), \\ \pi^4 \circ pm(p) \\ \pi^5 \circ pm(p) \end{pmatrix} \right] \tag{3.13}$$

**AddPolicyActionToPolicyRule** This function adds a policy action to the existing set of policy actions associated to the policy rule. Therefore the added policy action must be carefully analysed before the addition is finalised, as it may cause undesirable effects within the managed system.

$$AddPolicyActionToPolicyRule : (PolicyRule \times \mathbb{P}Action) \rightarrow (PolicyMap \rightarrow PolicyMap)$$

$$AddPolicyActionToPolicyRule\,(p, sa)\,pm \triangleq pm\dagger \left[ p \rightarrow \begin{pmatrix} \pi^1 \circ pm(p), \\ \pi^2 \circ pm(p), \\ sa \cup \pi^3 \circ pm(p), \\ \pi^4 \circ pm(p), \\ \pi^5 \circ pm(p) \end{pmatrix} \right] \tag{3.14}$$

**RemovePolicyActionFromPolicyRule** This function removes a policy action from the existing set of policy actions associated to the policy rule. Again this function may invalidate a policy rule if there are no actions left after the function completes.

$$RemovePolicyActionFromPolicyRule : (PolicyRule \times \mathbb{P}Action) \rightarrow (PolicyMap \rightarrow PolicyMap)$$

$$RemovePolicyActionFromPolicyRule\,(p, sa)\,pm \triangleq pm\dagger \left[ p \rightarrow \begin{pmatrix} \pi^1 \circ pm(p), \\ \pi^2 \circ pm(p), \\ sa \backslash \pi^3 \circ pm(p), \\ \pi^4 \circ pm(p), \\ \pi^5 \circ pm(p) \end{pmatrix} \right] \tag{3.15}$$

**AddPolicySubjectToPolicyRule** This function adds a policy subject to the set of policy subjects associated with the policy rule. This results in increasing the number of subjects that impose this policy rule. Effectively, this means that the policy rule can be invoked by a larger set of managed entities.

$$AddPolicySubjectToPolicyRule : (PolicyRule \times \mathbb{P}Subject) \rightarrow (PolicyMap \rightarrow PolicyMap)$$
$$AddPolicySubjectToPolicyRule\,(p, ss)\,pm \triangleq pm \dagger \left[ p \rightarrow \begin{pmatrix} \pi^1 \circ pm(p), \\ \pi^2 \circ pm(p), \\ \pi^3 \circ pm(p), \\ ss \cup \pi^4 \circ pm(p), \\ \pi^5 \circ pm(p) \end{pmatrix} \right] \tag{3.16}$$

**RemovePolicySubjectFromPolicyRule** This function removes a policy subject from the set of policy subjects associated to the policy rule. This results in a decrease of the number of entities that represent the authority imposing this policy rule. Effectively, this means that the policy rule can only be invoked by a smaller set of managed entities.

$$RemovePolicySubjectFromPolicyRule : (PolicyRule \times \mathbb{P}Subject) \rightarrow (PolicyMap \rightarrow PolicyMap)$$
$$RemovePolicySubjectFromPolicyRule\,(p, ss)\,pm \triangleq pm \dagger \left[ p \rightarrow \begin{pmatrix} \pi^1 \circ pm(p), \\ \pi^2 \circ pm(p), \\ \pi^3 \circ pm(p), \\ ss \backslash \pi^4 \circ pm(p), \\ \pi^5 \circ pm(p) \end{pmatrix} \right] \tag{3.17}$$

**AddPolicyTargetToPolicyRule** This function increases the number of entities that are the targets of this policy rule, thereby widening its scope. Hence, care should be taken to redo any previous policy analysis to ensure that no new conflicts have been added.

$$AddPolicyTargetToPolicyRule : (PolicyRule \times \mathbb{P}Target) \rightarrow (PolicyMap \rightarrow PolicyMap)$$

$$AddPolicyTargetToPolicyRule\,(p, st)\,pm \triangleq pm\dagger \left[ p \rightarrow \begin{pmatrix} \pi^1 \circ pm(p), \\ \pi^2 \circ pm(p), \\ \pi^3 \circ pm(p), \\ \pi^4 \circ pm(p), \\ st \cup \pi^5 \circ pm(p) \end{pmatrix} \right] \tag{3.18}$$

**RemovePolicyTargetFromPolicyRule** This function decreases the number of entities that are the targets of this policy rule, thereby restricting its scope.

$$RemovePolicyTargetFromPolicyRule : (PolicyRule \times \mathbb{P}Target) \rightarrow (PolicyMap \rightarrow PolicyMap)$$

$$RemovePolicyTargetFromPolicyRule\,(p, st)\,pm \triangleq pm\dagger \left[ p \rightarrow \begin{pmatrix} \pi^1 \circ pm(p), \\ \pi^2 \circ pm(p), \\ \pi^3 \circ pm(p), \\ \pi^4 \circ pm(p), \\ st \backslash \pi^5 \circ pm(p) \end{pmatrix} \right] \tag{3.19}$$

#### 3.2.4.2 Policy Model Manipulation

The functions defined to manipulate policy rules do not take into account how the modification of individual policy rules affect either the target managed system, or other policies currently deployed to manage the system. For example, the modification of a policy's actions may seem like a trivial task, but the introduction of a new action into a policy must be analysed carefully. From examining the policy model as currently presented, there is no way of knowing which action can or cannot be added to a policy as a policy conflict or inconsistency may arise. This is also true for all other modifications to a policy rule. However, the information model that the policy model is a subset of can describe, in detail, the associations and constraints among modelled types of events, conditions, actions, and managed entities.

#### 3.2.4.3 Information Model Integration

The policy model is a subset of the information model, as each event, condition, action, subject and target is linked to modelled entities (classes) in the information model. The links to the information model enable the design of specific algorithms and processes that, given a policy object as input, can explore the information model and retrieve information relating to the components of the policy. In effect, the information model is being used as a data dictionary to capture relationships, attributes and methods that exist for the managed system.

In the case of events, an information model such as DEN-ng describes a relevant set of event types that can be observed from a telecommunications network, along with information pertaining to their execution frequency and association to other events. The associations and hierarchies defined within the information model can be augmented further with constraints. However, it is up to the information model architect to decide how much should be modelled in the information model.

The information model may capture a constraint between two specific types of events in an event hierarchy wherein the two event types cannot occur within a specific time interval. Given this information, an algorithm can be devised to detect if a policy rule is relying on the occurrence of these event types within a small time interval (i.e. simultaneously), thus violating the constraint. This specific type of information would otherwise not be available to the policy author who may not have intricate knowledge of the nature of the event types as captured in the information model. This type of violation should be detected as a semantic error, as the policy may be syntactically correct but the policy has an inconsistent meaning according to the constraint information in the model.

Similarly for action types, constraint information may prohibit two action types from being performed simultaneously. Therefore, a policy rule describing such behaviour would be deemed in breach of the information model defined constraints. This information can be leveraged by algorithms that could modify policies if violations are detected.

A subject or target may be any modelled entity defined in the information model. Subjects and targets may be linked to roles or individual elements. Constraint information defined across multiple roles can specify which roles are incompatible and which roles are only applicable in specific contexts. This type of information can be used to augment policies so that a clearer picture can be formed as to their applicability in certain circumstances. For example, a policy rule can be defined to enable a service use a specific router interface to provide encryption; however, if the interface cannot perform in the encryption due to an information model based constraint, then the policy is invalid and should be modified or removed.

Access to the information model is invaluable for creating consistent and correct policies, and these important uses require a formal link between the policy subset of the information model and the part of the information model that describes the structure of the managed system.

### 3.2.5 Policy Continuum Formalism

The policy continuum represents the concept that policies can be represented differently for different constituencies of policy authors or domains of expertise. This is outlined pictorially in figure 3.5. More specifically, policies related to business level concepts such as customers, products and service contracts can be represented at a high-level of abstraction; this is amenable to a business domain expert. In contrast, policies related to network level concepts, such as routers, switches and storage devices can be represented at more detailed levels of abstraction suitable to network administrators. The policy continuum is therefore predominately used as an abstraction tool to help policy authors link low level policies to high-level goals. However, this work focuses on realising a policy continuum structure capable of maintaining the relationships among policies at multiple levels and maintaining consistency as those relationships change over time. The main output from this section is the definition of a formal model to represent the policy continuum.

#### 3.2.5.1 Policy Continuum Model Specification

The policy continuum is modelled as a tree of policy objects where the definition of a policy object is specified in the previous section in equation (3.8).

$$\begin{gathered} pr \in \mathit{PolicyContinuum} = \mathit{PolicyRule} \rightarrow (\mathbb{N} \times \mathbb{P}\mathit{PolicyRule}) \\ \mathit{dom} \circ pr = p \end{gathered} \tag{3.20}$$

In the policy continuum as defined in (3.20), $pr$ is a map function that maps policy rules to other policy rules. The PolicyContinuum maps a policy to:

1. a natural number representing a policy continuum level; and

2. a set of policies that represents related policies the next lower level of the policy

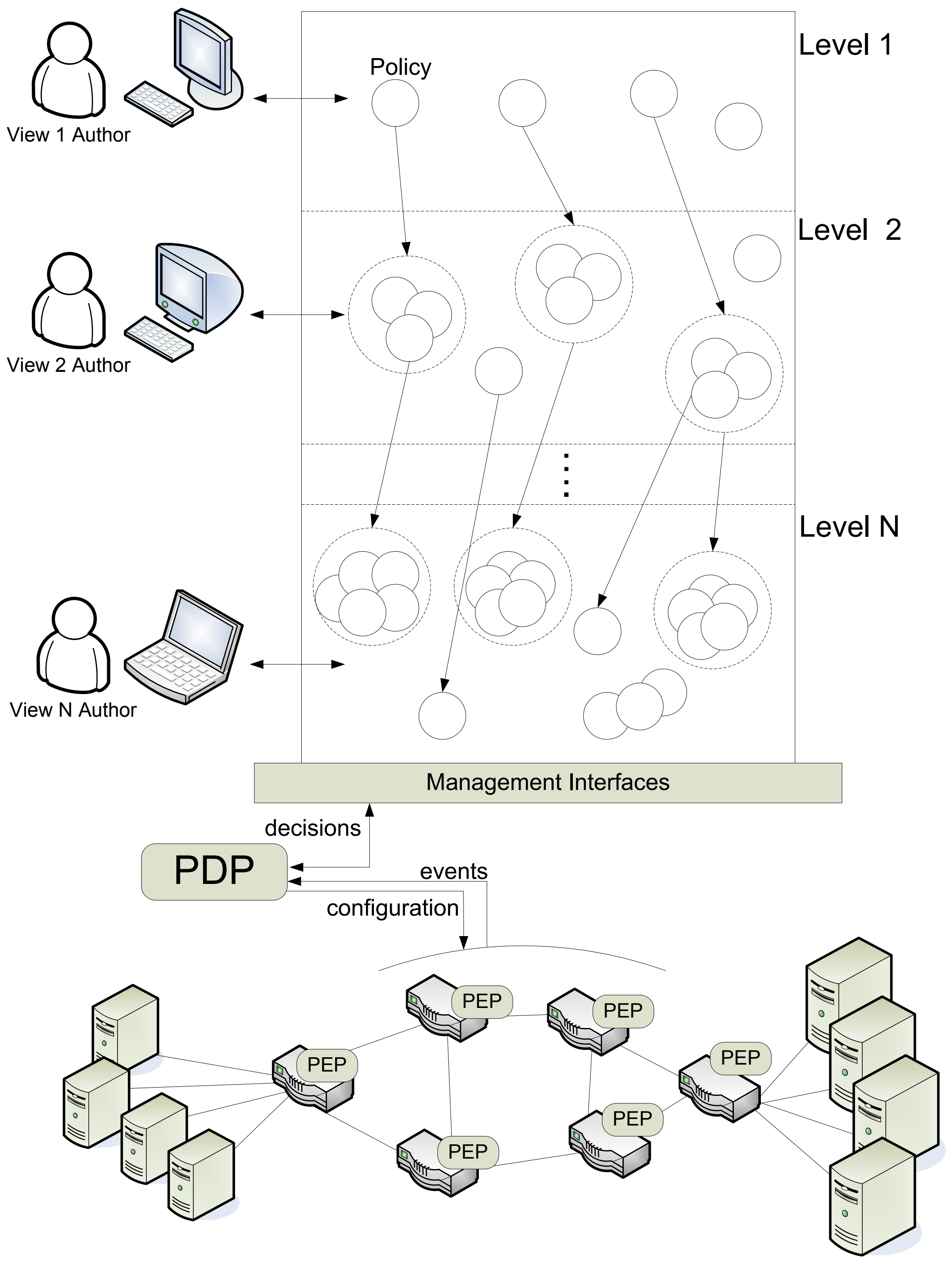


Figure 3.5: Conceptual policy continuum

continuum.

This facilitates the ability for a policy to reference policies in lower levels of the policy continuum. A policy can only be associated directly with policies defined at other continuum levels. A policy at a specific level may be referenced by more than one policy at a higher level and may reference one or more policies at lower levels. All policies within the policy continuum must at least be accessible at the top level of the PolicyContinuum.

#### 3.2.5.2 Basic Continuum functions

Once the structure of the policy continuum has been defined, some basic functions must be formally described to enable processes to make use of the structure to store and retrieve policies. These basic functions do not yet describe how the policy continuum is affected by the addition or modification of policies. These functions are used later as part of the authoring process.

**AddPolicy** A crucial requirement for the policy continuum is the ability to add a given policy to a specific level of the policy continuum. This function therefore simply adds a policy at the given level and associates it to a null set of policies, as the policy is not yet associated with policies at any other level of the policy continuum. In the function defined in (3.21), $n$ refer to a continuum level, $p$ is a policy rule identifier and $pr$ is the policy continuum to which the policy mapping is being added. The result of the function is a modified PolicyContinuum.

$$\begin{aligned}&AddPolicy : (\mathbb{N} \times PolicyRule) \rightarrow (PolicyContinuum \rightarrow PolicyContinuum)\\&AddPolicy(n, p)pr \triangleq pr \cup [p \rightarrow (n, \emptyset)]\end{aligned} \tag{3.21}$$

**AssociatePolicy** Associating policies in the policy continuum involves changing the set of policies associated to the given policy. If the policy is updated, then the goals of the higher level policy may be impacted. Essentially the policy $p$ and a set of policies $ps$ are inserted into the map, where the policy rule set $ps$ overrides the existing set of policy rules associated to $p$. The associated policy set is a set of policies that exist at the lower level of the policy continuum.

$$\begin{aligned}&UpdatePolicy : PolicyRule \times \mathbb{P}PolicyRule \rightarrow (PolicyContinuum \rightarrow PolicyContinuum)\\&UpdatePolicy(p, ps)pr \triangleq pr\dagger\left[p \rightarrow \left(\pi^1 \circ pr(p), ps\right)\right]\end{aligned} \tag{3.22}$$

**GetPolicyChildren** The child policies of a specific input policy can be recursively retrieved (3.23). This is useful to see the impact a specific policy rule may have on the policy continuum. The function is recursive and is called on each of the child policy rule associated to a policy rule $p$. The function terminates when there are no child policies left to call. The function returns an enumerated set of child policies. As this function just retrieves policies from the policy continuum, there are no side affects caused. The function operates as follows. $\left(I \rightarrow \pi^2\right)pr$ changes the structure of the policy continuum mapping to map input policies to associated child policies only. When the input policy $p$ is mapped through, then a set of associated policies are returned. For each associated policy returned (represented by $p_n$), the *GetPolicyChildren* function is called. The result is a union of all associated policies.

$$\begin{aligned}&GetPolicyChildren : PolicyRule \rightarrow (PolicyContinuum \rightarrow \mathbb{P}PolicyRule)\\&GetPolicyChildren(\emptyset) \triangleq \emptyset\\&GetPolicyChildren(p)pr \triangleq\\&\quad \forall p_n \in \left(\left(I \rightarrow \pi^2\right)pr\right)(p) : GetPolicyChildren(p_n)\\&\quad \cup \left(\left(I \rightarrow \pi^2\right)pr\right)(p)\end{aligned} \tag{3.23}$$

**GetPoliciesAtLevelN** All policy rules at a specific level of the policy continuum (3.24) can be retrieved. The policy continuum map *pr,* is restricted to only those policies that are at the level equal to the level specified as an argument to the function. The domain of this reduced map is a set of policies at a specific level. The specified map transformation reduces a map function to only those members that satisfy the Boolean expression ( i.e. are specified to exist at continuum level $n$). As any policy can be input into the policy continuum map to access its continuum level and associated lower level policies, then the domain of the restricted map are policies at a specific policy continuum level.

$$\begin{aligned} &GetPoliciesAtLevelN : \mathbb{N} \rightarrow (PolicyContinuum \rightarrow \mathbb{P}PolicyRule) \\ &GetPoliciesAtLevelN(n)pr \triangleq dom(I \rightarrow \lhd \left[\pi^1 = n\right])pr \end{aligned} \tag{3.24}$$

**GetAllPolicies** All policies in the policy continuum can be retrieved via (3.25). All policies must at least be expressed in the domain of the policy continuum map. This function can be used by other functions that must iterate over all existing deployed policies.

$$\begin{aligned} &GetAllPolicies : PolicyContinuum \rightarrow \mathbb{P}PolicyRule \\ &GetAllPolicies(pr) \triangleq dom(pr) \end{aligned} \tag{3.25}$$

**GetPolicyParents** A policy can be referenced by multiple other policies at a higher level of the policy continuum, these higher level policies can be retrieved using this function. It is used to trace up the policy continuum given a policy at a specific level. The function firstly reduces the policy continuum map *pr* to only those policies specified at a higher level to the given policy *p*. This map is then transformed to a "policy-to-policy map". The inverse of this map is a map between policy rules and their associated parent policy rules. When this new inverse map is indexed by *p*, the parent policy rules of *p* are returned.

$$\begin{aligned} &GetPolicyParents : PolicyRule \rightarrow (PolicyContinuum \rightarrow \mathbb{P}PolicyRule) \\ &GetPolicyParents(p)pr \triangleq \\ &\quad \left(\left(\mathrm{I} \rightarrow \pi^2\right) GetPoliciesAtViewN\left(\pi^1 \circ pr\,(p) - 1\right)\right)^{-1}(p) \end{aligned} \tag{3.26}$$

**GetCommonPolicies** This function retrieves those policy rules specified at lower levels of the policy continuum that are in common to two given policy rules (3.27). This function makes use of the GetPolicyChildren function, where the child policy rules of the supplied policy rules p$_a$ and p$_b$ are retrieved, the intersection of which are common policies.

$$\begin{aligned} &GetCommonPolicies : (PolicyRule \times PolicyRule) \rightarrow (PolicyContinuum \rightarrow \mathbb{P}PolicyRule) \\ &GetCommonPolicies(p_a, p_b)pr \triangleq \\ &GetPolicyChildren(p_a) \cap GetPolicyChildren(p_b) \end{aligned} \tag{3.27}$$

**GetEventAssociatedPolicyRules** The function depicted in (3.28) returns the set of all policies that are associated to the same, or a super set of, the events of the input policy. As VDM is implementation independent the evaluation of the super set relationhship is not defined for the moment, but has to be defined for any implementation. This function is useful for analysing the dependency among policies relating to the use of events. The function operates as follows. The policy continuum map is restricted to contain only those policies that reference the same, or super set of, the events referenced to by the input policy. The domain of this restricted map is a set of policies that are event associated to the input policy.

$$\begin{aligned} &GetEventAssociatedPolicyRules : PolicyRule \rightarrow \\ &\quad ((PolicyMap \times PolicyContinuum) \rightarrow \mathbb{P}PolicyRule) \\ &GetEventAssociatedPolicyRules(p)pm, pr \triangleq \\ &\quad dom \left(\lhd[\pi^2 \circ pm(p) \subseteq \pi^2 \circ pm]pr\right) \end{aligned} \tag{3.28}$$

**GetPoliciesWithEvent** This function (3.29) returns all those policies that are triggered by a specific input event. This is useful for "what if" analysis to retrieve those policies that would be evaluated when a given event is triggered. This function restricts the policy continuum to only those policies that reference the input event. The domain of the restricted continuum is a set of policies. This function illustrates the ability of querying the policy continuum via specific policy events. Another example of a usage of this function would be to analyse the policy continuum to investigate the popularity of use of certain events.

$$\begin{aligned} &GetPoliciesWithEvent : Event \rightarrow \\ &\quad ((PolicyMap \times PolicyContinuum) \rightarrow \mathbb{P}PolicyRule) \\ &GetPoliciesWithEvent(e)pm, pr \triangleq dom \left(\lhd[e \subseteq \pi^2 \circ pm]pr\right) \end{aligned} \tag{3.29}$$

**GetConditionAssociatedPolicyRules** The function depicted in (3.30) returns the set of all policies that are associated to the same set or a super set of the conditions of the input policy. This function is useful for analysing the dependency among policies relating to the use of conditions. For example, a policy rule can be passed into the function, and a set of policies will be returned that share the condition component of the input policy.

$$\begin{aligned}
&GetConditionAssociatedPolicyRules : PolicyRule \rightarrow \\
&\quad ((PolicyMap \times PolicyContinuum) \rightarrow \mathbb{P}PolicyRule) \\
&GetConditionAssociatedPolicyRules\,(p)\,pm, pr \triangleq \\
&\quad dom\left(\lhd\left[\pi^2 \circ pm\,(p) \subseteq \pi^2 \circ pm\right] pr\right)
\end{aligned} \tag{3.30}$$

**GetPoliciesWithCondition** This function (3.31) returns all those policies that reference a given condition. This is useful for detecting those policies dependent on conditions that may for example have a problem associated with them (i.e. the condition can no longer be evaluated or there is an anomaly detected in the conditions evaluation). This function differs from the previous function as a generic policy condition is used as an input and may not necessarily be associated to an existing deployed policy.

$$\begin{aligned}
&GetPoliciesWithCondition : Condition \rightarrow \\
&\quad ((PolicyMap \times PolicyContinuum) \rightarrow \mathbb{P}PolicyRule) \\
&GetPoliciesWithCondition(c)pm, pr \triangleq dom\left(\lhd[c \subseteq \pi^3 \circ pm]pr\right)
\end{aligned} \tag{3.31}$$

**GetActionAssociatedPolicyRules** This function (3.32) returns the set of all policies that are associated with the same or super set of actions of the input policy. This function can be used, for example, to retrieve a set of policy rules that are associated to an action that is failing for some reason. This function restricts the policy continuum to only those policies that reference the same or more actions as referenced by the input policy.

$$\begin{aligned}
&GetActionAssociatedPolicyRules : PolicyRule \rightarrow \\
&\quad ((PolicyMap \times PolicyContinuum) \rightarrow \mathbb{P}PolicyRule) \\
&GetActionAssociatedPolicies\,(p)\,pm, pr \triangleq \\
&\quad dom\left(\lhd\left[\pi^3 \circ pm\,(p) \subseteq \pi^3 \circ pm\right] pr\right)
\end{aligned} \tag{3.32}$$

**GetPoliciesWithAction** This function (3.33) returns the set of all policies that are associated with the input action. This function differs from the previous function as it does not depend on an input policy, but instead takes as input an arbitrary policy action that may or may not be associated with deployed policies.

$$\begin{aligned}
&GetPoliciesWithAction : Action \rightarrow \\
&\quad ((PolicyMap \times PolicyContinuum) \rightarrow \mathbb{P}PolicyRule) \\
&GetPoliciesWithAction\,(a)\,pm, pr \triangleq \\
&\quad dom\left(\lhd \left[a \subseteq \pi^3 \circ pm\right] pr\right)
\end{aligned} \tag{3.33}$$

**GetSubjectAssociatedPolicyRules** This function (3.34) returns the set of all policies that are associated by subset or equality with the input policy via its referenced subject components. By associating two policies by subset or equality, the function can determine if the two policies share some subject components. This function can therefore retrieve those deployed policies that have common subject components to the input policy.

$$\begin{aligned}
&GetSubjectAssociatedPolicyRules : PolicyRules \rightarrow \\
&\quad ((PolicyMap \times PolicyContinuum) \rightarrow \mathbb{P}PolicyRule) \\
&GetSubjectAssociatedPolicies\,(p)\,pm, pr \triangleq \\
&\quad dom\left(\lhd \left[\pi^4 \circ pm\,(p) \subseteq \pi^4 \circ pm\right] pr\right)
\end{aligned} \tag{3.34}$$

**GetPoliciesWithSubject** This function (3.35) returns the set of all policies that are associated with the input subject. The difference between this function and the previous function is that the subject input into the function is independent from the currently deployed policies, whereas the previous function depends on an input policy.

$$\begin{aligned}
&GetPoliciesWithSubject : Subject \rightarrow \\
&\quad ((PolicyMap \times PolicyContinuum) \rightarrow \mathbb{P}PolicyRule) \\
&GetPoliciesWithSubject\,(s)\,pm, pr \triangleq \\
&\quad dom\left(\lhd \left[s \subseteq \pi^4 \circ pm\right] pr\right)
\end{aligned} \tag{3.35}$$

**GetTargetAssociatedPolicyRules** This function (3.36) returns the set of all policies that are associated with the input policy via its referenced targets. This function can be used to retrieve those deployed policies in the policy continuum that share common target policy components with the input policy. For example, this function can be used to discover policies that apply to a common set of targets.

$$\begin{aligned}&GetTargetAssociatedPolicyRules : PolicyRule \rightarrow\\&\quad((PolicyMap \times PolicyContinuum) \rightarrow \mathbb{P}PolicyRule)\\&GetTargetAssociatedPolicies(p)pm, pr \triangleq\\&\qquad dom\left(\lhd[\pi^6 \circ mpi(p) \subseteq \pi^6 \circ pm]pr\right)\end{aligned} \tag{3.36}$$

**GetPoliciesWithTarget** This function (3.37) returns the set of all policies that are associated with the input target. This function differs from the previous function, as it is independent from any input policy. Therefore, an arbitrary policy target can be input into this function and all policy rules that reference this target are returned.

$$\begin{aligned}&GetTargetAssociatedPolicies : Policy \rightarrow\\&\quad((PolicyMap \times PolicyContinuum) \rightarrow \mathbb{P}PolicyRule)\\&GetTargetAssociatedPolicies\,(p)\,pm, pr \triangleq\\&\quad dom\left(\lhd\left[\pi^5 \circ pm\,(p) \subseteq \pi^5 \circ pm\right] pr\right)\end{aligned} \tag{3.37}$$

### 3.2.6 Policy Authoring Process

The information model is tightly integrated with policy model and the policy continuum model. Any of the processes that must examine information concerning policy element types can interrogate the information model directly. The advantages of the information model integration become apparent when devising algorithms that depend on discovery of relationships between policies within the policy continuum. This is especially the case for conflict analysis and policy refinement, as outlined in this section where a policy authoring process is defined. The authoring process must be aware that multiple constituencies of policy authors may be involved in the deployment of a single policy. The process a policy author must go through to create or modify a policy that is merged into the policy continuum is the same across all levels. The process description in algorithm 1 depicts modification of a candidate policy. It is referred to as the candidate policy to distinguish it from policies that are already contained in the policy continuum. The authoring process is split into three steps.

The first step traces up the policy continuum to verify that the modification of the candidate policy does not invalidate the goals of higher-level deployed policies. The second step analyses the policies at the same level of the policy continuum as that of the candidate policy, to ensure that no potential conflict exists with the candidate policy. The third step

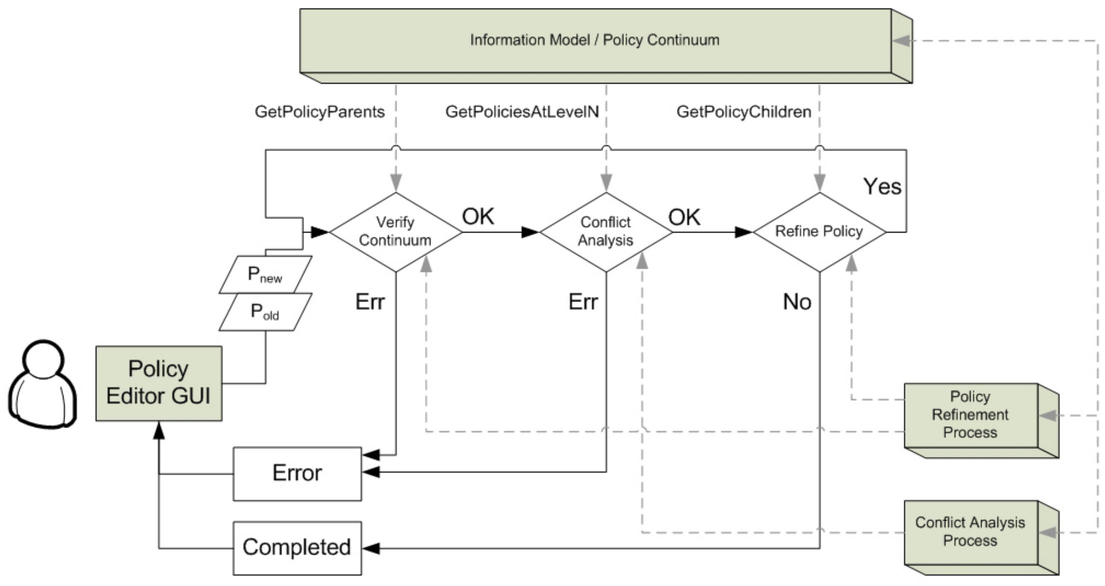


Figure 3.6: Policy authoring steps.

invokes a refinement process to derive a set of lower level policies from the candidate policy that must be recursively verified and tested for conflict and validity. For each policy produced during refinement, they need to be integrated into the policy continuum. If there are no policies produced from the refinement step then the process finishes.

Figure 3.6 depicts the steps involved in policy authoring, explaining the process from the perspective of a policy author.

The process is described by presenting three essential component processes, modifying a policy in the policy continuum, creating a policy in the policy continuum and removing a policy from the policy continuum. Table 3.2 describes the terms used within the algorithm. The modification of a policy begins by ensuring the modification to $p_{old}$ (the existing policy), represented by $p_{new}$ (the modified version of the policy), satisfies all of the higher level policy refinements that $p_{old}$ was related to. The old policy may have been created as a refinement to a higher-level policy; therefore, if it is modified the process needs to be able to ensure that the related high-level policies can still meet their specified objectives. The process retrieves a set of parent policies by calling *GetParentPolicies* on $p_{old}$ and for each parent policy it verifies that it is still consistent (e.g. goals are still satisfied) when using the modified version of the policy (i.e. $p_{new}$). If consistency with each parent policy is satisfied then the algorithm continues, otherwise the candidate policy $p_{new}$ is causing inconsistencies within the policy continuum and should not be committed. This result is

Table 3.2: Terms used.

| Term | Meaning |
|---|---|
| $p_{old}$ | The policy being modified before modification |
| $p_{new}$ | The policy being modified after modification |
| $pc$ | The policy continuum map |
| $p_{parent}$ | The parent policy of the policy currently being modified |
| $p_{cnf}$ | A policy that conflicts with the new policy |
| $p_{parcnf}$ | The parent policy of a conflicting policy |
| $p_{ref}$ | A policy that is refined from the policy being modified |
| $p_{ref}$ | An old policy that is moified due to a refinement from the current policy being modified |

then passed to the current policy author. Assuming the candidate policy has passed the previous test, it must now be analysed for conflict against currently deployed policies at the same continuum level.

A selection algorithm could be used here to reduce the runtime complexity associated with comparing the candidate policy against all deployed policies at the same policy continuum level.

If a conflict is detected, the set of associated parent policies that are indirectly involved in the conflict can to be retrieved so that more information about the conflict can be relayed back to the current candidate policy author. This information may be used to establish a strategy to resolve the conflict. If no conflict is detected at the current policy continuum level, $p_{new}$ is refined into a set of lower level modifications that may consist of create, modify or remove operations to lower level policies. For each policy that must be created, modified or removed, appropraite process is carried out. If refinement is not needed then the process returns with a successful commit. The process continues until all refinement operations are successful thus enabling it to commit $p_{new}$ to the policy continuum. Similarly, there is a *CreatePolicy* algorithm (algorithm 2) and *DeletePolicy* algorithm (algorithm 3). Together these processes and the associated algorithms comprise the authoring process.

The policy authoring process makes extensive use of three algorithms that work independently of each other but are combined to deliver an effective solution. Specifically, the algorithms are *AnalysePolicyConflict*, *RefinePolicy* and *VerifyPolicyContinuum*. The

**Algorithm 1** Modify a policy in the policy continuum.

```
$\mathbf{ModifyPolicy} : (\mathit{PolicyRule} \times \mathit{PolicyRule} \times \mathit{PolicyContinuum}) \rightarrow \mathbf{B}$
$\mathbf{ModifyPolicy}\,(p_{old}, p_{new}, pc) \triangleq$
$\forall p_{parent} \in \mathbf{GetPolicyParents}\,(p_{old})\,pc :$
    if $not$ $\mathbf{VerifyPolicyContinuum}\,(p_{parent}, p_{new}, pc)$
    then
        $\mathbf{NotifyCurrentAuthor}\,(p_{parent})$
        $return$ false
if $\mathbb{P}\mathbf{p_{cnf}} = \mathbf{AnalysePolicyConflict}\,(p_{new}, pc)$
then
  $\forall p_{cnf} \in \mathbf{PotentialConflictList}\,(p_{new})\,pc :$
    $\forall p_{parcnf} \in \mathbf{GetPolicyParents}\,(p_{cnf})\,pc :$
        $\mathbf{NotifyCurrentAuthor}\,(p_{parcnf})$
    $\mathbf{NotifyCurrentAuthor}\,(p_{cnf})$
    $return$ false
else
  $\forall p_{\mathrm{oldcld}} \in \mathbf{GetPolicyChildren}\,(p_{old}, pc) :$
    $\mathbf{DeletePolicy}\,(p_{oldcld}, pc)$
  $\forall p_{ref} = \mathbf{RefinePolicy}\,(p_{new}, pc) :$
    $\mathbf{AddPolicy}\,\left(\pi^1 \circ pc\,(p_{new}), p_{ref}\right) pc$
  $\mathbf{VerifyPolicyContinuum}\,(p_{new}, p_{ref}, pc)$
$\mathbf{CommitChange}\,(p_{old}, p_{new}, pc)$
```

interfaces defined to the information model and the policy continuum provide such algorithms with the ability to examine the dependencies among policies and to modify these dependencies as they see fit.

Policy conflict analysis is one of the integral processes that maintains the consistency of the policy continuum as it is modified by multiple constituencies of policy authors. There are essentially two component processes that work together to deliver an effective policy analysis process. They are policy selection and policy conflict analysis. The policy selection process deals with selecting a subset of policies that are currently deployed that must be analysed against given the current candidate policy. The purpose of this process is to effectively reduce the runtime complexity associated to policy conflict analysis by only analysing those policies that have a high probability of conflict with the candidate policy.

The conflict analysis algorithm takes as input two policies, the candidate policy and one of the selected deployed policies, and performs a set of functions to ascertain a case for potential conflict. If a conflict is detected it is highlighted and information pertaining

**Algorithm 2** Create a policy in the policy continuum.

```
$\mathbf{CreatePolicy} : (PolicyRule \times PolicyContinuum) \rightarrow \mathbf{B}$
$\mathbf{CreatePolicy}\,(p_{new}, pc) \triangleq$
$\forall p_{parent} \in \mathbf{GetPolicyParents}\,(p_{new}, pc)$ :
    if $not$ $\mathbf{VerifyPolicyContinuum}\,(p_{parent}, p_{new}, pc)$
    then
        $\mathbf{NotifyCurrentAuthor}\,(p_{parent})$
        $return\ false$
if $\mathbf{AnalysePolicyConflict}\,(p_{new}, pc)$
then
  $\forall p_{cnf} \in \mathbf{PotentialConflictList}\,(p_{new}, pc)$ :
    $\forall p_{parcnf} \in \mathbf{GetPolicyParents}\,(p_{cnf} pc)$ :
        $\mathbf{NotifyCurrentAuthor}\,(p_{parcnf})$
    $\mathbf{NotifyCurrentAuthor}\,(p_{cnf})$
    $return\ false$
else
  $\forall p_{ref} = \mathbf{RefinePolicy}\,(p_{new}, pc)$ :
    $\mathbf{AddPolicy}\,(\pi^1 \circ pc\,(p_{new})\,, p_{ref})\ pc$
  $\mathbf{VerifyPolicyContinuum}\,(p_{new}, p_{ref}, pc)$
$return$ $\mathbf{CommitChange}\,(p_{new}, pc)$
```

to the conflict is passed back to the policy author The policy conflict analysis algorithm for use within the policy authoring process is the topic of the next chapter.

## 3.3 Summary and Discussion

The policy continuum as described by Strassner (2003) can be automated if well defined policy authoring and analysis processes exist. The concept of a policy hierarchy was introduced by Moffett and Sloman (1993), who identify a need for high level business policies to be translated or refined into lower level policies that carry out the high level objectives. Policy hierarchies are also investigated by Wies (1995) who specifically identified a need for a refinement process to automate the task of associating and creating effective policy hierarchies. The main difference between the hierarchies of policy defined by Moffett and Sloman (1993) and Wies (1995), and the policy continuum as defined by Strassner (2003) is that, the former two authors do not consider the policy authoring process occurring for multiple levels of policy. There are non-trivial issues to be aware of when trying to coordinate edits to a group of related policies that from a policy continuum, such as: creating,

**Algorithm 3** Delete a policy in the policy continuum.

```
$\mathbf{DeletePolicy} : (\mathit{PolicyRule} \times \mathit{PolicyContinuum}) \rightarrow \mathbf{B}$
$\mathbf{DeletePolicy}\,(p_{old}, pc) \triangleq$
$\forall p_{parent} \in \mathbf{GetPolicyParents}\,(p_{old}, pc)$ :
    if $\mathit{not}$ $\mathbf{VerifyPolicyContinuum}\,(p_{parent}, pc)$
    then
        $\mathbf{NotifyCurrentAuthor}\,(p_{parent})$
        $\mathit{return}\ pc$
  $\forall p_{\mathrm{ref}} \in \mathbf{GetChildPolicies}\,(p_{old})\,pc$ :
    $\mathbf{DeletePolicy}\,(p_{ref}, pc)$
$\mathit{return}\,\mathbf{CommitChange}\,(p_{old}, pc)$
```

modifying or removing a policy at a specific policy continuum level, policy refinement, policy conflict analysis and policy verification.

It is possible to expand on the functionality of existing approaches to policy refinement by incorporating refinement into a holistic policy authoring process that is capable of alerting the policy author when conflict may occur as a result of a policy refinement. Until now, a formal model of the policy continuum has not existed that enables different levels of policy to be related to each other and that is sensitive to the intricate relationships that exist among policies at multiple levels. the relationships that can be established between policies depends on the information model.

The policy model used to represent policies in the policy continuum is very flexible and can be used to represent many of the current policy languages in use at present. Popular policy languages such as Ponder (Damianou et al., 2001), Rei (Kagal et al., 2003), and KAoS (Uszok et al., 2003), which are used primarily for creating deontic policies and XACML (Godik et al., 2003), which is primarily used for a subset of deontic actions (access control), all have concepts of event, condition and action, though they vary in their semantics. They also define subject and target managed entities (again , their semantics differ). However, none of them use a policy continuum. The assumptions on the required components of a compatible policy language for the policy continuum are therefore satisfied by these policy languages.

This chapter initially presented extensions to the existing DEN-ng information model for the purpose of making the links between levels of the policy continuum more explicit. This was followed by the development of a formal policy continuum and associated policy authoring process. The formal model specifies the operational semantics of the policy continuum and the steps the policy author must go through to deploy a newly created or modified policy into the policy continuum. An important component of the policy authoring process is policy conflict analysis. The authoring process explicitly mentions policy conflict analysis and relates its role in the authoring process with that of policy refinement and policy verification. Research question 1, outlined in chapter 1 asks how the policy author can interact with the policy continuum in a consistent manner across policy continuum levels, while being alerted if a policy conflict arises. The policy authoring process, such as defined in this chapter addresses this research question as it is defined to operate with a well specified policy continuum model. The conflict analysis process is the focus of the next chapter. The conflict analysis process is a pair-wise analysis process that compares pairs of policies in order to establish a case for conflict. The complexity of the process lies in the fact that there is no single definition of policy conflict making it difficult to design a generic and policy nature independent algorithm based on existing approaches. The approach taken in this thesis harnesses the knowledge embodied in information model to aid in ascertaining a case for conflict among pairs of policies.

Chapter 4

# Application Independent Policy Conflict Analysis Algorithm

Policy conflict analysis is an integral part of the policy authoring process as presented in chapter 3. The goal of this chapter is to define an application independent and extensible policy conflict analysis algorithm to operate as part of the policy authoring process. This algorithm should be capable of detecting potential policy conflict while being independent of the application domain to which the policies relate and the continuum level to which policies are defined. The presence of an information model which models the structure and relationships between the system managed entities and the policies that effect their management is assumed. Given the presence of such an information model, and making minimal assumptions regarding its nature this chapter demonstrates that it is possible to define an application independent and extensible policy conflict analysis algorithm.

The algorithm defined in this chapter should be deployed as part of the policy authoring process to ascertain if a newly created or modified policy (the candidate policy) potentially conflicts with already deployed policies. The policy authoring process controls which policies are passed to the policy conflict analysis algorithm via a selection process. The candidate policy is analysed on a pair wise basis with currently deployed policies, with the conflict analysis algorithm being split into two phases. In the first phase, the algorithm uses the information model to identify pertinent relationships between the candidate policy rule and the deployed policy rule (for example, the policies may reference the same target entity). In the second phase these relationships are examined in the context of an application specific conflict signature matrix extracted from the information model. The separation of the two phases reflects the fact that different applications utilise differing policy execution models and hence there is no universal criteria to define how policies conflict.

The rest of the chapter is structured as follows, section 4.1 presents the policy conflict analysis algorithm. The policy based management implementation is described and discussed in section 4.2. Section 4.3 presents set of non trivial case studies used to illustrate the flexibility of the algorithm in detecting policy conflict at different levels of the policy continuum and for different applications. It demonstrates how the detection of conflict depends on the contents of the accompanying information model. Finally, section 4.4 summarises and concludes the chapter.

## 4.1 Policy Conflict Analysis Algorithm

Research question 3 states, "*How can a policy conflict analysis process be developed that is independent of the nature of the policies?*" A policy conflict analysis algorithm should be equally applicable to a range of policy based applications so that the algorithms do not need to be continously re-developed and re-engineered for each application of policy. The only criterion for the policy language used is that it is compatible with the generic form of policy that was presented in chapter 3. Also, the algorithm should be independent of application and policy language, so that it can be used across heterogeneous applications and be equally applicable to any constituency of policy author that uses the policy continuum. Note, however, that this algorithm assumes that an information model is available that can represent the structure and constraints of the system that is being managed by policy and that there is a query interface to the information model consistent with that specified in chapter 3 in section 3.2.3.

### 4.1.1 Algorithm Overview

The difference between existing policy conflict analysis algorithms is largely in how they define what a "policy conflict" is, and subsequently how they measure success the of a policy conflict analysis algorithm. As different application domains have differing policy semantics, each application domain has different criteria for determining when two policies conflict. The algorithm presented here decouples the criteria for conflict from the analysis of the policies, thus enabling the association of multiple definitions of policy conflict. Furthermore, it is assumed that as part of the policy authoring process presented in chap-

**Algorithm 4** Select policies then analyse for conflict.

```
$\textbf{analysePolicyConflict} : (PolicyRule) \rightarrow$
    $(PolicyContinuum \times Ontology \times InformationModel) \rightarrow \mathbf{B}$
$\textbf{analysePolicyConflict}\,(p_{cnd})\,pc, ot, in \triangleq$
$let\ p_{list} = \textbf{selectPolicies}\left(p_{cnd}, GetPoliciesAtLevelN\left(\pi^1 \circ pc\,(p_{cnd})\right) pc\right) ot, \text{in} \triangleq$
    $\forall p_{dep} \in p_{list}$ :
        $\textbf{analyseConflict}\,(p_{cnd}, p_{dep})$
```

ter 3 there is a separate selection algorithm that determines which appropriate subset of policies must be analysed for conflict. In this chapter the selection algorithm is assumed to be a simple pair-wise selection algorithm, in which a candidate policy is analysed for conflict against all deployed policies. The role of the selection algorithm in policy conflict analysis is shown in algorithm 4. After a policy has been modified by the user at the policy authoring GUI, that policy is used to aid in the selection of those deployed policies that must be further analysed for conflict. For each selected policy, it is input into the conflict analysis algorithm. The result of the algorithm is fed back to the author via a policy conflict indication, from which the policy author should act upon. The interaction of the algorithm with the policy continuum is also discussed.

Algorithm 5 illustrates the main steps of the 'analyseConflict' algorithm. There are essentially two phases to the algorithm. The first phase performs a pair-wise examination of the two input policies being analysed and creates a policy relationship matrix to define how different aspects of policies are related to each other. The matrix is established by comparing the contained policy rule components of the policies against each other. The second phase retrieves a conflict matrix from the information model representing a set of those relationships that must exist between the two policies for a conflict to potentially occur. The conflict matrix is matched against the policy relationship matrix for similarity; if the matrices satisfy the match operation (denoted by $\circledast$ in algorithm 5) then a conflict can occur.

This approach decouples the application domain from the conflict analysis algorithm by retrieving application specific data via a well defined interface to the information model. The policy conflict types are decoupled because checking for conflict is based on a conflict relationship matrix that can be adjusted to detect different types of relationships among different pairs of policies. In addition, the algorithm does not depend on a particular policy language, only that the language is compatible with the policy model defined in chapter

**Algorithm 5** Analyse for conflict.

```
**analyseConflict** : $(PolicyRule \times PolicyRule) \rightarrow \mathrm{B}$
**analyseConflict** $(p_1, p_2) \triangleq$
        $//Phase1$
        $let\, m =$ **relatePolicies** $(p_1, p_2)$

        $//Phase2$
        **if** $m \neq null$ **then**
          $\forall\, cf \in$ **retrieveConflictPattern** $(p_1, p_2)$ :
          **if** $m \circledast cf = 1$ **then**
             "Flag for Conflict"
```

$$\begin{bmatrix} ssb & ssp & seq & scor & 0 \\ tsb & tsp & teq & tcor & 0 \\ esb & esp & eeq & ecor & emux \\ csb & csp & ceq & ccor & cmux \\ asb & asp & aeq & acor & actd \end{bmatrix}$$

Figure 4.1: A policy relationship matrix

3. Both phases of the algorithm are now discussed in detail, along with the extensions to the DEN-ng information model to represent policy relationships and conflict matrices.

### 4.1.2 Phase 1: Policy Relationship Analysis

The approach taken is to initially create a matrix that relates policies to each other in different ways; this is the policy relationship matrix. From examining the related work in the area, policies can be related in a number of different ways depending on the policy rule component type being analysed. For example, when determining if two access control policies conflict, there must be an overlap among the subjects, targets and actions (Lupu and Sloman, 1999). Therefore, policies can be associated by subject, target and action as a first step, and those policies that can be seen to overlap after this step can be flagged to signify potential conflicts. These are potential conflicts, as the relationships only indicate requirements for conflict; thus, the conflict may or may not happen at runtime. However, in other cases, the combination of {subject, target, action } overlap may not be of interest, but an {event, condition, action} overlap may be of interest; this is the case for detecting conflict in firewall filtering policies as (Al-Shaer et al., 2005). The relationship matrix dictates whether two policies potentially conflict or not, where different types of relationships are relevant to different types of policy conflicts.

Table 4.1: Policy relationship descriptions.

| Code | Description |
|---|---|
| ssb | $p_a$.subject $\subset$ $p_b$.subject |
| ssp | $p_a$.subject $\supset$ $p_b$.subject |
| seq | $p_a$.subject $=$ $p_b$.subject |
| scor | $p_a$.subject $\cap$ $p_b$.subject $\neq\emptyset$ |
| tsb | $p_a$.target $\subset$ $p_b$.target |
| tsp | $p_a$.target $\supset$ $p_b$.target |
| teq | $p_a$.target $=$ $p_b$.target |
| tcor | $p_a$.target $\cap$ $p_b$.target $\neq\emptyset$ |
| esb | $p_a$.event $\subset$ $p_b$.event |
| esp | $p_a$.event $\supset$ $p_b$.event |
| eeq | $p_a$.event $=$ $p_b$.event |
| ecor | $p_a$.event $\cap$ $p_b$.event $\neq\emptyset$ |
| emux | $p_a$.event $\Rightarrow \neg p_b$.event |
| csb | $p_b$.condition $\Rightarrow$ $p_a$.condition |
| csp | $p_a$.condition $\Rightarrow$ $p_b$.condition |
| ceq | $p_b$.condition $\Leftrightarrow$ $p_a$.condition |
| ccor | $p_b$.condition $\vee$ $p_a$.condition |
| cmux | $p_b$.condition $\Rightarrow \neg p_a$.condition $\wedge p_a$.condition $\Rightarrow \neg p_b$.condition |
| asb | $p_a$.action $\subset$ $p_b$.action |
| asp | $p_a$.action $\supset$ $p_b$.action |
| aeq | $p_a$.action $=$ $p_b$.action |
| acor | $p_a$.action $\cap$ $p_b$.action $\neq\emptyset$ |
| actd | $\neg\,(p_a.action \wedge p_b.action)$ |

**Algorithm 6** Policy relationship comparisons.

```
$\textbf{relatePolicies} : (PolicyRule \times PolicyRule \times PolicyMap) \rightarrow Matrix$
$\textbf{relatePolicies}\,(p_1, p_2, pm) \triangleq$
                    $m = \textbf{zeroMatrix}$
                    $m = \textbf{associateBySubject}\,(p_1, p_2, pm) \circ$
                    $\textbf{associateByTarget}\,(p_1, p_2, pm) \circ$
                      $\textbf{associateByEvent}\,(p_1, p_2, pm) \circ$
                        $\textbf{associateByCondition}\,(p_1, p_2, pm) \circ$
                          $\textbf{associateByAction}\,(p_1, p_2, pm)\, m$
```

**Algorithm 7** AssociateBySubject.

```
$\textbf{associateBySubject} : (PolicyRule \times PolicyRule \times PolicyMap) \rightarrow (Matrix \rightarrow Matrix)$
$\textbf{associateBySubject}\,(p_1, p_2, pm)\, m \triangleq$
    if $isSubjectSubset\,(p_1, p_2, pm)$
    then
        $markSSB\,(p_1, p_2) \circ markSSP\,(p_2, p_1)\, m$
    elseif $isSubjectSuperset\,(p_1, p_2, pm)$
    then
        $markSSP\,(p_1, p_2) \circ markSSB\,(p_2, p_1)\, m$
    elseif $isSubjectEqual\,(p_1, p_2, pm)$
    then
        $markSEQ\,(p_1, p_2) \circ markSEQ\,(p_2, p_1)\, m$
    elseif $isSubjectCorrelated\,(p_1, p_2, pm)$
    then
        $markSCOR\,(p_1, p_2) \circ markSCOR\,(p_2, p_1)\, m$
```

The initial step of the algorithm is called *RelatePolicies*, as illustrated in algorithm 6. This step involves examining the candidate policy with a deployed policy in order to discover whether specific relationships exist. Relationships can be established via event, condition, action, subject or target; the approach is flexible and allows other policy component types to be included or replaced in the matrix should the policy model need to be extended. The initially zeroed matrix is modified by a group of operations that populate the matrix with ones or zeros depending on the result. Figure 4.1 illustrates a policy relationship matrix, where the codes describe the associated significance of the entry in the matrix. The codes used in figure 4.1 represent the associated relationships considered. The first letter of the codes indicate which component is being analysed, and are as follows: *"s"* prefix for subject, *"t"* prefix for target, and similarly "*e*" for event, "*c*" for condition and "*a*" for action. The end of the codes indicate the type of relationship being established, and are as follows: *"sb"* suffix for subset, *"sp"* suffix for superset, *"eq"* suffix for equal, *"cor"*

**Algorithm 8** markSSB.

```
**markSSB** : $(PolicyRule \times PolicyRule) \rightarrow (Matrix \rightarrow Matrix)$
**markSSB** $(p_1, p_2)\, m \triangleq m[p_1][p_2][S][SB] = 1$
```

suffix for correlation, *"mux"* suffix for mutually exclusive and *"ctd"* suffix for contradiction. Each row is dedicated to holding relationship information about two policies concerning a specific policy component type. For example, the first row describes subject based relationships between two policies. There are four relationships that can be established: subset, superset, equal and correlated. A subset relationship is established is the subjects of one policy are a subset of the subjects of the other policy, similarly for superset and equal. Correlated means that a subset, superset and equal relationship cannot be established but that there are some shared subject members to both policies. Table 4.1 details the meaning of the entries of the policy relationship matrix.

#### 4.1.2.1 AssociateBySubject / AssociateByTarget

The subject and target tests are similar because both policy components refer to managed entities. For example, the *associateBySubject* (algorithm 7) operation takes as input two policies and tests for a set of relationships that exist among the subjects of the policies. Each policy is analysed using four different subject-based tests, where the tests examine the subject of the policies for subset, superset, equality or correlation (intersection) relationships. If for example $p_1$ and $p_2$ (the two policies being compared respectively) return true for the function *isSubjectSubset*, marks are placed in the input matrix at the positions that signify the subject of $p_1$ is a subset of the subject of $p_2$, and inversely the subject of $p_2$ is a superset of the subject of $p_1$. The functions used to mark the matrix are called "mark" functions. The name of the function indicates the index into the matrix that is marked for any given input policies. Algorithm 8 shows the setting of a relationship in the matrix.

The *isSubjectSubset* is specified in algorithm 10; it specifies that two policies are input, along with map functions to discover the components of the policies and the related classes defined in the information model. The subject components of the two policies are retrieved and compared. The comparison required is an element subset membership operation.

If the information model allows subjects of a policy rule to be individually managed entities or members of selected management domains, the class information of the subject entities has to be examined in order to establish if it has a containment relationship. The

**Algorithm 9** isTypeOf and getMembers functions.

```
**isTypeOf** : $(Class \times Class \times ClassDetails) \rightarrow \mathbb{P}Object$
**isTypeOf** : $(c_1, c_2, cld) \rightarrow \mathbb{P}Object$
**if** $c_1 = c_2$ **then**
  $return\, true$
**else if isTypeOf** $\left(\pi^5 \circ cld\,(c_1)\,, c_2, cld\right)$
**else** $return\, false$
⋮
**getMembers** : $\begin{pmatrix} PolicyRule \times PolicyRule \times PolicyMap \times ObjectClass.. \\ ClassDetails \times Integer \end{pmatrix} \rightarrow \mathbb{P}Object$
**getMembers** : $(p_1, pm, objCl, cld, parentType, i) \rightarrow \mathbb{P}Object$
**let**   $s_1 = \bigcup \forall s_e \in \pi^i \circ pm\,(p_1)$ :
**if** $isTypeOf\,(objCl\,(s_e)\,, parentType, cld)$ **then**
  $getMembers\,(s_e)$
**else** $s_e$
$return\, s_1$
```

inputs to the function are the *ObjectClass* map and the *ClassDetails* map. The algorithm makes use of the well defined interfaces to the information model as specified in chapter 3 to simplify the computation of these functions.

The *isTypeOf* function, as specified in algorithm 9, compares two classes together that are specified from within the information model and can determine if one is equal to or inherits from the other. The *getMembers* function, also specified in algorithm 9, enables an entity of a particular type to be searched for all of its members. This function is recursive, so that nested domains are also expanded and searched. Using these two helper functions, a complete set of domain entities that represent the subjects of both input policies can be found; a test for inclusion will determine if one set is a subset of the other. This information can be used to establish is one if a superset of the other.

The function for establishing subject equality is specified in algorithm 11. This function is similar to the algorithm 10 except the discovered subject members of the two policies compared for equality.

Similarly, operations can be performed for testing for subject superset membership (see algorithm 12) and subject correlation (see algorithm 13). After the relationship matrix has been populated with information pertaining to subjects, additional operations can be performed to populate the matrix with results of target, event, condition, and action overlap in a similar manner.

**Algorithm 10** isSubjectSubset.

```
$\textbf{isSubjectSubset} : \begin{pmatrix} PolicyRule \times PolicyRule \times PolicyMap \times ObjectClass.. \\ ClassDetails \times Class \end{pmatrix} \rightarrow \mathrm{B}$
$\textbf{isSubjectSubset}\,(p_1, p_2, pm, objCl, cld, parentType) \triangleq$
 $\textbf{let}\, s_1 = getMembers\,(p_1, pm, objCl, cld, parentType, 4)$
 $\textbf{let}\, s_2 = getMembers\,(p_2, pm, objCl, cld, parentType, 4)$
$return\,(s_1 \subset s_2)$
```

**Algorithm 11** isSubjectEqual.

```
$\textbf{isSubjectEqual} : \begin{pmatrix} PolicyRule \times PolicyRule \times PolicyMap \times ObjectClass.. \\ ClassDetails \times Class \end{pmatrix} \rightarrow \mathrm{B}$
$\textbf{isSubjectEqual}\,(p_1, p_2, pm, objCl, cld, parentType) \triangleq$
 $\textbf{let}\, s_1 = getMembers\,(p_1, pm, objCl, cld, parentType, 4)$
 $\textbf{let}\, s_2 = getMembers\,(p_2, pm, objCl, cld, parentType, 4)$
 $return\,(s_1 = s_2)$
```

**Algorithm 12** isSubjectSuperset.

```
$\textbf{isSubjectSuperset} : \begin{pmatrix} PolicyRule \times PolicyRule \times PolicyMap \times ObjectClass.. \\ ClassDetails \times Class \end{pmatrix} \rightarrow \mathrm{B}$
$\textbf{isSubjectSuperset}\,(p_1, p_2, pm, objCl, cld, parentType) \triangleq$
 $\textbf{let}\, s1 = getMembers\,(p_1, pm, objCl, cld, parentType, 4)$
 $\textbf{let}\, s_2 = getMembers\,(p_2, pm, objCl, cld, parentType, 4)$
 $return\,(s_1 \supset s_2)$
```

**Algorithm 13** isSubjectCorrelated.

```
$\textbf{isSubjectCorrelated} : \begin{pmatrix} PolicyRule \times PolicyRule \times PolicyMap \times ObjectClass.. \\ ClassDetails \times Class \end{pmatrix} \rightarrow \mathrm{B}$
$\textbf{isSubjectCorrelated}\,(p_1, p_2, pm, objCl, cld, parentType) \triangleq$
 $\textbf{let}\, s_1 = getMembers\,(p_1, pm, objCl, cld, parentType, 4)$
 $\textbf{let}\, s_2 = getMembers\,(p_2, pm, objCl, cld, parentType, 4)$
 $return\,(s_1 \neq s_2 \wedge s_1 \not\subset s_2 \wedge s_1 \not\supset s_2 \wedge s_1 \cap s_2 \neq \emptyset)$
```

#### 4.1.2.2 AssociateByAction

Events, conditions and actions can be related in similar ways to subjects and targets. However, there are other ways of relating these components of policy, since they indicate the actual behaviour that the policy is proposing to enforce. The *associateByAction* (see algorithm 14) operation is exemplary of these: it checks for any relationships between actions defined in the policies that are being compared. The basic comparisons between policy actions are the same as those defined for subjects: subset (algorithm 15), equality (algorithm 16), superset (algorithm 17) and correlated (algorithm 18). Along with these eelationships, the *isActionContradiction* operation examines if any of the actions specified in the candidate policy contradict with any of the actions specified in the selected deployed policy. Note that a policy conflict may still not exist even if the actions of two policies contradict. Typically a policy conflict is manifested only when the policies can be triggered at the same time and the conditions of the policies can both be satisified. Therefore, if only the relationship concerning the actions of the policies is established, then not enough information is available to indicate a potential policy conflict. Policy conflict depends on many more relationships among policies to be in place before flagging the policy author can be justified. For example, the candidate policy may specify that a system be shut down, whereas a deployed policy may specify that a system run a backup process. The two actions contradict if the backup process requires a system to be running and not shut down; however, depending on how and when the policies are triggered and evaluated, they may occur during non overlapping time intervals.

To determine whether two given policy actions actually contradict information specified within the information model about those operations may be leveraged. Any pre- and post-conditions of class operations are made available through well specified interfaces to the information model. One of the assumptions specified concerning the information model is that policy actions are defined as operations in the information model. The function *doActionsContradict* is based on the premise that:

1. Two operations are acting on the same type or sub type of an object,
2. The operations' pre-conditions can be satisfied simultaneously and,
3. The operations' post conditions are 'incompatible' or cannot be satisfied simultaneously and,

**Algorithm 14** associateByAction.

```
$\textbf{associateByAction} : (PolicyRule \times PolicyRule \times PolicyMap \times Matrix) \rightarrow Matrix$

$\textbf{associateByAction}\,(p_1, p_2, pm, m) \triangleq$
    $\textbf{if}\ isActionSubset\,(p_1, p_2, pm)$
    $\textbf{then}$
     $markASB\,(p_1, p_2) \circ markASP\,(p_2, p_1)\, m$
    $\textbf{elseif}\ isActionSuperset\,(p_1, p_2, pm)$
    $\textbf{then}$
     $markASP\,(p_1, p_2) \circ markASB\,(p_2, p_1)\, m$
    $\textbf{elseif}\ isActionEqual\,(p_1, p_2, pm)$
    $\textbf{then}$
     $markAEQ\,(p_1, p_2) \circ markAEQ\,(p_2, p_1)\, m$
    $\textbf{elseif}\ isActionCorrelated\,(p_1, p_2, pm)$
    $\textbf{then}$
     $markACOR\,(p_1, p_2) \circ markACOR\,(p_2, p_1)\, m$
    $\textbf{elseif}\ doActionsContradict\,(p_1, p_2, pm)$
    $\textbf{then}$
     $markACLF\,(p_1, p_2) \circ markACLF\,(p_2, p_1) \circ m$
```

**Algorithm 15** isActionSubset.

```
$\textbf{isActionSubset} : \left( \begin{array}{c} PolicyRule \times PolicyRule \times PolicyMap \times ObjectClass.. \\ ClassDetails \times Class \end{array} \right) \rightarrow \mathrm{B}$

$\textbf{isActionSubset}\,(p_1, p_2, pm, objCl, cld, parentType) \triangleq$
 $\textbf{let}\ a_1 = getMembers\,(p_1, pm, objCl, cld, parentType, 3)$
 $\textbf{let}\ a_2 = getMembers\,(p_2, pm, objCl, cld, parentType, 3)$
 $return\,(a_1 \subset a_2)$
```

**Algorithm 16** isActionEqual.

```
$\textbf{isActionEqual} : \left( \begin{array}{c} PolicyRule \times PolicyRule \times PolicyMap \times ObjectClass.. \\ ClassDetails \times Class \end{array} \right) \rightarrow \mathrm{B}$

$\textbf{isActionEqual}\,(p_1, p_2, pm, objCl, cld, parentType) \triangleq$
 $\textbf{let}\ a_1 = getMembers\,(p_1, pm, objCl, cld, parentType, 3)$
 $\textbf{let}\ a_2 = getMembers\,(p_2, pm, objCl, cld, parentType, 3)$
 $return\,(a_1 = a_2)$
```

**Algorithm 17** isActionSuperset.

```
$\textbf{isActionSuperset} : \left( \begin{array}{c} PolicyRule \times PolicyRule \times PolicyMap \times ObjectClass.. \\ ClassDetails \times Class \end{array} \right) \rightarrow \mathrm{B}$

$\textbf{isActionSuperset}\,(p_1, p_2, pm, objCl, cld, parentType) \triangleq$
 $\textbf{let}\ a_1 = getMembers\,(p_1, pm, objCl, cld, parentType, 3)$
 $\textbf{let}\ a_2 = getMembers\,(p_2, pm, objCl, cld, parentType, 3)$
 $return\,(a_1 \supset a_2)$
```

**Algorithm 18** isActionCorrelated.

```
$\textbf{isActionCorrelated} : \begin{pmatrix} PolicyRule \times PolicyRule \times PolicyMap \times ObjectClass.. \\ ClassDetails \times Class \end{pmatrix} \rightarrow \mathrm{B}$
$\textbf{isActionCorrelated}\,(p_1, p_2, pm, objCl, cld, parentType) \triangleq$
 $\textbf{let}\, a_1 = getMembers\,(p_1, pm, objCl, cld, parentType, 3)$
 $\textbf{let}\, a_2 = getMembers\,(p_2, pm, objCl, cld, parentType, 3)$
 $return\,(a_1 \neq a_2 \wedge a_1 \not\subset a_2 \wedge a_1 \not\supset a_2 \wedge a_1 \cap a_2 \neq \emptyset)$
```

**Algorithm 19** doActionsContradict.

```
$\textbf{doActionsContradict} : \begin{pmatrix} PolicyRule \times PolicyRule \times \\ PolicyMap \times \\ ObjectClass \times \\ ClassDetails \times \\ OperationConstraints \times \\ ActionOperation \end{pmatrix} \rightarrow \mathrm{B}$
$\textbf{doActionsContradict}\,(p_1, p_2, mpi, objCl, cld, op, actOp) \triangleq$
$\forall a_1, a_2 \wedge a_1 \in \pi^3 \circ pm\,(p_1)\,, a_2 \in \pi^3 \circ pm\,(p_2) :$
$\textbf{if}\ \left(I \rightarrow \pi^1\right) cld^{-1}\,(actOp\,(a_1)) \cap \left(I \rightarrow \pi^1\right) cld^{-1}\,(actOp\,(a_2)) \neq \emptyset$
$\textbf{then}$
    $\textbf{if}\ \forall pre_1, pre_2 \wedge pre_1 \in \left(I \rightarrow \pi^1\right) op^{-1}\,(actOp\,(a_1))$
    $\wedge pre_2 \in \left(I \rightarrow \pi^1\right) op^{-1}\,(actOp\,(a_2)) :$
    $compatible\,(pre_1, pre_2)$
     $\textbf{then}$
       $\textbf{if}\ \forall post_1, post_2 \wedge post_1 \in \left(I \rightarrow \pi^2\right) op^{-1}\,(actOp\,(a_1))$
       $\wedge post_2 \in \left(I \rightarrow \pi^2\right) op^{-1}\,(actOp\,(a_2)) :$
       $incompatible\,(post_1, post_2)$
       $\textbf{elseif}\ \forall post, inv \wedge post \in \left(I \rightarrow \pi^2\right) op^{-1}\,(actOp\,(a_1))$
       $\cup \left(I \rightarrow \pi^2\right) op^{-1}\,(actOp\,(a_2))\,, :$
                 $inv \in \left(I \rightarrow \pi^4\right) cld^{-1}\,(actOp\,(a_1))$
                   $\cup \left(I \rightarrow \pi^4\right) cld^{-1}\,(actOp\,(a_2))$
       $incompatible\,(post, inv)$
```

4. Any invariants defined also hold true through the life of each operation.

The pre-conditions specify the constraints over the attributes of the object before the operation can be performed and the post-conditions specify the constraints over the attributes of the object after the operation has completed. The pre-conditions for the operations in question must be true, because if they are not then the actions will not be performed simultaneously. “Incompatible post-conditions” means that the values specified constraining the objects attributes are contradicting for two or more post-conditions at the same time. In effect, this would dictate the object to be in two different states at the same time. Finally, each action can have a number of invariants, which define a set of global constraints for the objects which must be preserved by all operations of that class. Therefore, the true power of pre-conditions, post-conditions, and invariants is that collectively, they help specify the expected behaviour of an object. Hence, any variation from the specified pre-conditions, post-conditions and invariants may result in unpredictable behaviour. This implies that no side effects are present – everything that is required is specified in the invariants, pre-conditions, and post-conditions. The operation is specified in algorithm 19. It begins by discovering the associated classes in the information model that correspond to the actions being performed by the two policies. If there exists any two actions that correspond to operations on the same class then point 1 is satisified. Next, the pre-conditions corresponding to the associated operations are discovered and checked for compatibility. If compatible then point 2 is satisified. Then, the post conditions are discovered and checked for incompatibility, if incompatible, then contradicting actions are being performed. The invariants as checked if the post-conditions are compatible. If the invariants are incompatible with the post-conditions, then there are operations being specified that contradict the behaviour of the associated classes.

#### 4.1.2.3 AssociateByEvent

If a policy A is triggered by some events that also trigger another policy B, but that policy B depends on more events before it can be triggered, then policy A is related to the policy B by an event subset relationship. An assumption can be made that when policy B is triggered, then policy A is also be triggered.

For example, relating policies via events can give insight into whether the two policies might be executed simultaneously. However, there are more interesting ways to compare events; depending on the information supplied that define the content and relationships of

**Algorithm 20** associateByEvent.

```
$\textbf{associateByEvent} : (PolicyRule \times PolicyRule \times PolicyMap \times Matrix) \rightarrow Matrix$
$\textbf{associateByEvent}\,(p_1, p_2, pm, m) \triangleq$
    if $isEventSubset\,(p_1, p_2, pm)$
    then
     $markESB\,(p_1, p_2) \circ markESP\,(p_2, p_1)\,m$
    elseif $isEventSuperset\,(p_1, p_2, pm)$
    then
     $markESP\,(p_1, p_2) \circ markESB\,(p_2, p_1)\,m$
    elseif $isEventEqual\,(p_1, p_2, pm)$
    then
     $markEEQ\,(p_1, p_2) \circ markEEQ\,(p_2, p_1)\,m$
    elseif $isEventCorrelated\,(p_1, p_2, pm)$
    then
     $markECOR\,(p_1, p_2) \circ markECOR\,(p_2, p_1)\,m$
    elseif $isEventMUX\,(p_1, p_2, pm)$
    then
     $markEMUX\,(p_1, p_2) \circ markEMUX\,(p_2, p_1) \circ m$
```

**Algorithm 21** isEventSubset.

```
$\textbf{isEventSubset} : \begin{pmatrix} PolicyRule \times PolicyRule \times PolicyMap \times ObjectClass.. \\ ClassDetails \times Class \end{pmatrix} \rightarrow \mathrm{B}$
$\textbf{isEventSubset}\,(p_1, p_2, pm, objCl, cld, parentType) \triangleq$
 let $e_1 = getMembers\,(p_1, pm, objCl, cld, parentType, 1)$
 let $e_2 = getMembers\,(p_2, pm, objCl, cld, parentType, 1)$
 $return\,(e_1 \subset e_2)$
```

**Algorithm 22** isEventEqual.

```
$\textbf{isEventEqual} : \begin{pmatrix} PolicyRule \times PolicyRule \times PolicyMap \times ObjectClass.. \\ ClassDetails \times Class \end{pmatrix} \rightarrow \mathrm{B}$
$\textbf{isEventEqual}\,(p_1, p_2, pm, objCl, cld, parentType) \triangleq$
 let $e_1 = getMembers\,(p_1, pm, objCl, cld, parentType, 1)$
 let $e_2 = getMembers\,(p_2, pm, objCl, cld, parentType, 1)$
 $return\,(e_1 = e_2)$
```

**Algorithm 23** isEventSuperset.

```
$\textbf{isEventSuperset} : \begin{pmatrix} PolicyRule \times PolicyRule \times PolicyMap \times ObjectClass.. \\ ClassDetails \times Class \end{pmatrix} \rightarrow \mathrm{B}$
$\textbf{isEventSuperset}\,(p_1, p_2, pm, objCl, cld, parentType) \triangleq$
 let $e_1 = getMembers\,(p_1, pm, objCl, cld, parentType, 1)$
 let $e_2 = getMembers\,(p_2, pm, objCl, cld, parentType, 1)$
 $return\,(e_1 \supset e_2)$
```

**Algorithm 24** isEventCorrelated.

```
$\textbf{isEventCorrelated}: \begin{pmatrix} PolicyRule \times PolicyRule \times PolicyMap \times ObjectClass.. \\ ClassDetails \times Class \end{pmatrix} \rightarrow \mathrm{B}$
$\textbf{isEventCorrelated}\,(p_1, p_2, pm, objCl, cld, parentType) \triangleq$
 $\textbf{let}\; e_1 = getMembers\,(p_1, pm, objCl, cld, parentType, 1)$
 $\textbf{let}\; e_2 = getMembers\,(p_2, pm, objCl, cld, parentType, 1)$
 $return\,(e_1 \neq e_2 \wedge e_1 \not\subset e_2 \wedge e_1 \not\supset e_2 \wedge e_1 \cap e_2 \neq \emptyset)$
```

**Algorithm 25** isEventMUX.

```
$\textbf{isEventMUX}: \begin{pmatrix} PolicyRule \times PolicyRule \times PolicyMap \times ObjectClass.. \\ ClassDetails \times Class \end{pmatrix} \rightarrow \mathrm{B}$
$\textbf{isEventMUX}\,(p_1, p_2, pm, objCl, cld, parentType) \triangleq$
 $\textbf{let}\; e_1 = getMembers\,(p_1, pm, objCl, cld, parentType, 1)$
 $\textbf{let}\; e_2 = getMembers\,(p_2, pm, objCl, cld, parentType, 1)$
 $\textbf{if}\; \forall inv_1, inv_2 \wedge inv_1 \in \left(I \rightarrow \pi^1\right) cl^{-1}\,(objCl\,(e_1))\,,$
  $inv_1 \in \left(I \rightarrow \pi^1\right) cl^{-1}\,(objCl\,(e_1)) :$
   $incompatible\,(inv_1, inv_2)$
```

the objects that make up the events in the information model. Mutually exclusive events are those events that cannot occur at the same time, and two policies depending on the occurrence of two such events will not be evaluated simultaneously. A test for mutually exclusive events can be specified that takes into account the invariants associated to the related event classes.

Algorithm 21 checks two policies to discover an event subset relationship. Algorithm 22 checks two policies to discover an event equality relationship. This means that two policies are triggered by exactly the same event(s). Algorithm 23 checks two policies to discover an event superset relationship. This is similar to the event subset relationship but in the opposite direction. Algorithm 24 checks two policies to discover an event correlation relationship. This relationship has the same meaning for correlation as defined for subject and target correlation. There is an overlapping set of events that the two policies share. Algorithm 25 checks two policies to discover an events mutually exclusive relationship. To ascertain this relationship, class information about the events is retrieved from the information model and the invariants defined for the associated classes are checked for incompatibility. If the invariants of two events are incompatible (i.e. the state the system must be in so that the events can be triggered), then the event cannot occur at simultaneously.

#### 4.1.2.4 AssociateByCondition

Similarly, conditions can be tested for subset, superset, equality and correlation relationships. Therefore, in relating policies via the condition component, conclusions can be drawn as to which policies imply the condition components of other policies.

Mutual exclusivity is not confined to events. For example, conditions that logically contradict can never be satisfied simultaneously and should be flagged as mutually exclusive. A common example is temporal conditions that evaluate to true only during specific non over lapping time intervals. AssociateByCondition is specified in algorithm 26. In fact, related work carried out by Agrawal et al. (2005) concerning policy ratification and by Lin et al. (2007) concerning policy similarity highlight the challenges associated to comparing the components of policies together. Specifically, Agrawal et al. (2005) states that determining boolean expression implication relationships can be a computationally expensive problem. In that paper they do not solve the complexity issue, but instead reduce the expressivness of the boolean expressions used. Their approach to specifying conditions is adopted in the work presented in this chapter, where conditions are assumed to be single attribute value restrictions. The consequence of this is that determining relationships between condition components of policies is simpler, as the issues associated to detecting complex implication relationships is eliminated.

Algorithm 27 checks two policies to discover a condition subset relationship. Algorithm 28 checks two policies to discover a condition equality relationship. This means that two policies are satisified simultaneously. Algorithm 29 checks two policies to discover a condition superset relationship. Algorithm 30 checks two policies to discover a condition correlation relationship. This relationship means that there are parts of the two conditions components that are true simultaneously, but not all parts. Algorithm 31 check two policies to discover a conditions mutually exclusive relationship. This relationship means that there is no possible way that the two conditions can be satisified simulataneously, or there the a logical contradiction between the conditions components of the two policies.

**Algorithm 26** associateByCondition.

```
$\textbf{associateByCondition} : (PolicyRule \times PolicyRule \times PolicyMap \times Matrix) \rightarrow Matrix$
$\textbf{associateByCondition}\,(p_1, p_2, pm, m) \triangleq$
    $\textbf{if}\ isConditionSubset\,(p_1, p_2, pm)$
    $\textbf{then}$
     $markCSB\,(p_1, p_2) \circ markCSP\,(p_2, p_1)\, m$
    $\textbf{elseif}\ isConditionSuperset\,(p_1, p_2, pm)$
    $\textbf{then}$
     $markCSP\,(p_1, p_2) \circ markCSB\,(p_2, p_1)\, m$
    $\textbf{elseif}\ isConditionEqual\,(p_1, p_2, pm)$
    $\textbf{then}$
     $markCEQ\,(p_1, p_2) \circ markCEQ\,(p_2, p_1)\, m$
    $\textbf{elseif}\ isConditionCorrelated\,(p_1, p_2, pm)$
    $\textbf{then}$
     $markCCOR\,(p_1, p_2) \circ markCCOR\,(p_2, p_1)\, m$
    $\textbf{elseif}\ isConditionMUX\,(p_1, p_2, pm)$
    $\textbf{then}$
     $markCMUX\,(p_1, p_2) \circ markCMUX\,(p_2, p_1) \circ m$
```

**Algorithm 27** isConditionSubset.

```
$\textbf{isConditionSubset} : \left( \begin{array}{c} PolicyRule \times PolicyRule \times PolicyMap \times ObjectClass.. \\ ClassDetails \times Class \end{array} \right) \rightarrow \mathrm{B}$
$\textbf{isConditionSubset}\,(p_1, p_2, pm, objCl, cld, parentType) \triangleq$
 $\textbf{let}\ c_1 = getMembers\,(p_1, pm, objCl, cld, parentType, 2)$
 $\textbf{let}\ c_2 = getMembers\,(p_2, pm, objCl, cld, parentType, 2)$
 $return\,(c_1 \subset c_2)$
```

**Algorithm 28** isConditionEqual.

```
$\textbf{isConditionEqual} : \left( \begin{array}{c} PolicyRule \times PolicyRule \times PolicyMap \times ObjectClass.. \\ ClassDetails \times Class \end{array} \right) \rightarrow \mathrm{B}$
$\textbf{isConditionEqual}\,(p_1, p_2, pm, objCl, cld, parentType) \triangleq$
 $\textbf{let}\ c_1 = getMembers\,(p_1, pm, objCl, cld, parentType, 2)$
 $\textbf{let}\ c_2 = getMembers\,(p_2, pm, objCl, cld, parentType, 2)$
 $return\,(c_1 = c_2)$
```

**Algorithm 29** isConditionSuperset.

```
$\textbf{isConditionSuperset} : \left( \begin{array}{c} PolicyRule \times PolicyRule \times PolicyMap \times ObjectClass.. \\ ClassDetails \times Class \end{array} \right) \rightarrow \mathrm{B}$
$\textbf{isConditionSuperset}\,(p_1, p_2, pm, objCl, cld, parentType) \triangleq$
 $\textbf{let}\ c1 = getMembers\,(p_1, pm, objCl, cld, parentType, 2)$
 $\textbf{let}\ c_2 = getMembers\,(p_2, pm, objCl, cld, parentType, 2)$
 $return\,(c_1 \supset c_2)$
```

**Algorithm 30** isConditionCorrelated.

```
$\textbf{isConditionCorrelated} : \left( \begin{array}{c} PolicyRule \times PolicyRule \times PolicyMap \times ObjectClass.. \\ ClassDetails \times Class \end{array} \right) \rightarrow \mathrm{B}$
$\textbf{isConditionCorrelated}\,(p_1, p_2, pm, objCl, cld, parentType) \triangleq$
 $\textbf{let}\, c1 = getMembers\,(p_1, pm, objCl, cld, parentType, 2)$
 $\textbf{let}\, c_2 = getMembers\,(p_2, pm, objCl, cld, parentType, 2)$
 $return\,(c_1 \neq c_2 \wedge c_1 \not\subset c_2 \wedge c_1 \not\supset c_2 \wedge c_1 \cap c_2 \neq \emptyset)$
```

**Algorithm 31** isConditionMUX.

```
$\textbf{isConditionMUX} : \left( \begin{array}{c} PolicyRule \times PolicyRule \times PolicyMap \times ObjectClass.. \\ ClassDetails \times Class \end{array} \right) \rightarrow \mathrm{B}$
$\textbf{isConditionMUX}\,(p_1, p_2, pm, objCl, cld, parentType) \triangleq$
 $\textbf{let}\, c_1 = getMembers\,(p_1, pm, objCl, cld, parentType, 2)$
 $\textbf{let}\, c_2 = getMembers\,(p_2, pm, objCl, cld, parentType, 2)$
                 $return \neg\,(c_1 \leftarrow c_2)$
```

### 4.1.3 Phase 2: Conflict Matrix Match

The manner in which policies are analysed for potential conflict is highly dependent on the application for which those policies are defined. For example, when determining if two access control policies conflict, there must be an overlap among the subjects, targets and actions. Therefore, policies can be associated by subject, target and action as a first step, and those policies that can be seen to overlap after this step can be flagged to indicate potential conflicts. On the other hand, for filtering policies (such as firewall rules) subject, target, action overlap are not of interest, but instead event, condition and action overlap are relevent. Clearly, the relationship matrix between two policies gives an indication of the potential for conflict; however, different relationships are relevant in different application contexts in respect to detecting conflict.

In phase 2 another relationship matrix is used, but this one is not derived from the examination of two policies, but is instead a pre-defined matrix. This is known as the conflict matrix and represents a set of relationships that if exist between two policies may indicate a potential policy conflict. The conflict matrix is defined per application and is tailored to list only those relationships that must exist for a conflict to occur for that application. There may be multiple conflict matrices per application. The conflict matrix is used to make the decision whether to flag the policies as potentially conflicting. A simple matrix comparison operation is used that matches the relationship matrix produced in phase 1 with a conflict matrix that represents the set of relationships that may or must hold for conflict to exist. The matrix combination operation combines the values of two matrices by using logical AND and OR operations and is specified in equation 4.1. The conflict matrix specifies the requirements for conflict. Thus, the specification of conflict is decoupled from the analysis algorithm and can accommodate disparate conflict definitions. This is a very important property of the presented approach as it enables the information model architect to define the requirements of policy conflict on a per application basis, as opposed to having to encode the causes of policy conflict into specialised algorithms.

For each row in both matrices the appropriate entries are first logically ANDed together; therefore, if a “1” is present on the relationship matrix (indicating that this relationship exists) and also present in the conflict matrix (indicating that this relationship can lead to a conflict), then that relationship satisfies the requirement for a policy conflict to occur. After the logical AND operation is completed the results are logically ORed together. The effect of this for the row is that if one of the highlighted relationships is present, then ORing

$$\begin{bmatrix} ssb & ssp & seq & scor & 0 \\ tsb & tsp & teq & tcor & 0 \\ esb & esp & eeq & ecor & emux \\ csb & csp & ceq & ccor & cmux \\ asb & asp & aeq & acor & actd \end{bmatrix}$$

Figure 4.2: Phase 2 conflict matrix legend.

that row will produce a "1" for that row (i.e. conflicting via a specific policy component type, now check the other component types). As each row represents a component type, then a "1" for a particular stage signifies that the two policies satisfy the conditions for conflict with respect to that policy component type.

This process is repeated for each row in the matrix, resulting in a "0" or "1" for each policy component type. Finally these results are logically ANDed together, establishing that only if each policy component type satisfies the case for conflict that a "1" is returned.

$$\begin{aligned} &(M \circledast M)_{i,j} \rightarrow \mathrm{B} \\ &(a \circledast b)_{i,j} \triangleq \bigwedge_{p=0}^{i} \bigvee_{q=0}^{j} (a_{p,q} \wedge b_{p,q}) \end{aligned} \tag{4.1}$$

Figure 4.2 depicts a sample policy relationship matrix that is examined in this phase. Each row represents the relationships of a component type of a policy. To describe a condition for conflict, a "1" is placed in the associated position in the conflict matrix. For example, if a specific type of conflict requires subject subset membership, then a "1" would be placed in the upper left position of the conflict matrix. Note that the conflict matrix may signify multiple relationships for a single policy component type, indicating that any one of the relationships may hold for a conflict to be considered.

It is important to note that the set of relationship types given in the matrix can be expanded on. This thesis considers a set of obvious relationship types among policy components, but there are certainly more ways to relate two policy components. Both the conflict matrix and the policy relationship matrix can be increased in the number of columns and rows so that new ways of relating policies can be addressed. A possible extension to the matrix is to support relationships between policies that are related to the types of policies in use. For example, the Ponder policy language specifies five distinct policy types; they are: positive and negative authorisation, obligation, refrain and delegation. A new row can be easily defined to handle these types of relationships. However, the relationships presented in the matrix are typical of the majority of current policy language implementations.

Compiling the correct entries for the conflict matrix is a combined task of the policy specification expert and the information model architect. This is because the conflict matrix pattern is encoded in the information model, but the policy specification expert (policy author) is aware of what constitutes a policy conflict for their specific application of policies. The procedure for compiling the pertinent relationships for the conflict matrix is as follows. Note also that this is an offline process and can be done before any policies are actually specified.

1. The policy author must identify the target application that policies shall be used to manage. This may be for example access control, deontic permissions, firewalls, IPsec VPNs or routing domains. There are well defined cases for conflict in each of these applications, some of the cases even overlap as outlined in chapter 2.
2. Compile a list of policy components that are required to represent the policy for that applications. The available components include subject, target, event, condition and action. Not all of these components may be needed if a limited policy language is in use. For other cases new policy components may be required. For example, in the case of deontic permissions, a policy deontic type component is required. For the case of goal policies (policies that indicate a condition and action, but no event), then the policy event component clause is not required.
3. An empty matrix can be built the considers all possible relationships between each of the policy component types. The matrix is populated with '1's or '0's depending on whether a specific relationship may contribute to a potential conflict or not. A set of questions can be answered about each entry in the matrix. For example, do the subjects of the policies contribute to conflict, depending on application this will be true (1) or false (0).
4. Does the information model need to be extended to respresent extra semantics so that a specific relationship can be discovered. If so, then the information model should be extended. A description of how the information model is extended, specifically for DEN-ng, is given in the next section.
5. For each relationship that needs to be evaluated in the conflict matrix, the logic of the evaluation is designed so that it can be implemented and evaluated at runtime.

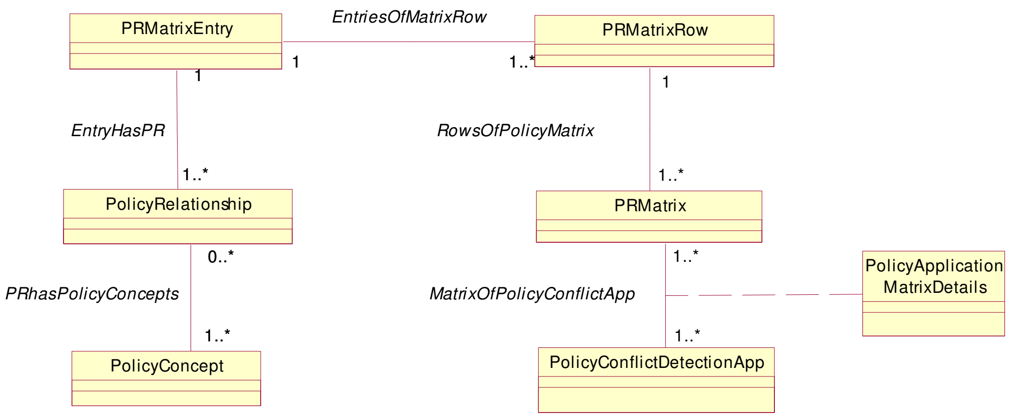


Figure 4.3: PolicyRelationship extensions to DEN-ng.

### 4.1.4 DEN-ng Extensions to Represent Policy Relationships

The algorithm presented depends heavily on the ability of the information model to describe the relationships that can exist between the various policy component types. To illustrate that an information model can be extended to describe policy relationships, DEN-ng is adapted. As the DEN-ng policy model has no facility to explicitly represent policy component type relationships, a set of modelling extensions are required to meet the needs of the approach taken.

Depicted in figure 4.3 are the primary new classes defined in the information model to support policy component type relationships. The `PolicyRelationship` class is an abstract class and is used to represent a relationship type between two or more `PolicyConcept` classes. Remember that a `PolicyConcept` class is a super class of the various component types of policies and policy types. Therefore, this class can be extended to represent a relationship between any combination of policy component types (or `PolicyConcept` subclasses). The concept of the policy relationship matrix is represented by the `PRMatrix` class, where the “PR” is short for policy relationship. Instances of this matrix, as described in the conflict detection algorithm, are used to store a group of relationships between two or more policy components. A `PRMatrix` is made up of a number of `PRMatrixRow` classes via the association *RowOfPolicyMatrix*. Each `PRMatrixRow` represents a group of relationships that are associated to the same policy component type. For example, a `PRMatrixRow` may represent a group of relationships that relate the PolicyEvent component of two policies. The `PRMatrixRow` is associated with a `PolicyRelationship` via the `PRMatrixEntry` class. The

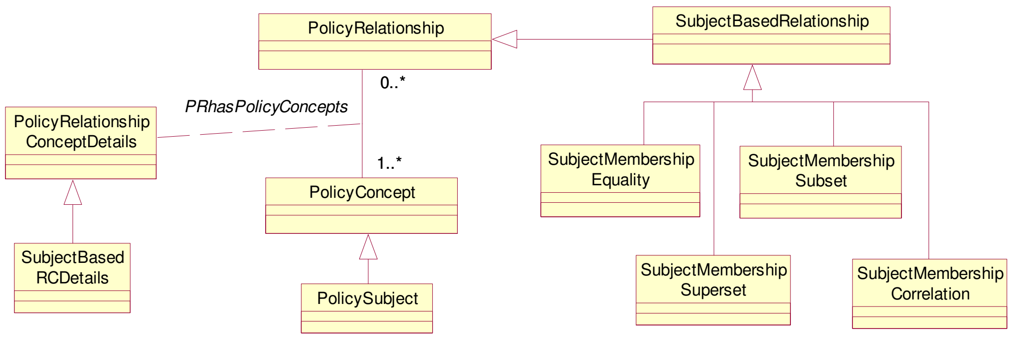


Figure 4.4: PolicyRelationship extension for PolicySubject.

`PRMatrixEntry` class is associated to a single `PolicyRelationship`. As the `PolicyConcept` class is very abstract, it is also used to subclass `ManagementPolicy`, `ECAPolicy` etc. This thesis does not consider relationships that can exist between policy types and focuses instead on relationships between policy component types.

The policy relationship matrix as discussed in phase 1 of the algorithm can be built from querying the `PRMatrix` classes in an information model. The `PRMatrix` class holds details on the number of rows the matrix should have and which relationship must be tested to populate each row. The logic for ascertaining the status of relationships between the various policy component types can be held in the individual `PolicyRelationship` subclasses. An example of extending the `PolicyRelationship` class is depicted in figure 4.4.

The `SubjectBasedRelationship` is a subclass of the `PolicyRelationship` class and is constrained to only relate `PolicySubjects` together. Specialised types of relationships are then defined, such as `SubjectMembershipEquality` and `SubjectMembershipSubset`. These two specialised classes are used to define very specific types of relationships that can exist between two `PolicySubjects`. The logic to ascertain this relationship is defined in the implementation of the class, and is used to develop the policy relationship function as used in phase 1 of the conflict detection algorithm.

For phase 2 of the algorithm, a conflict matrix is used and compared to the policy relationship matrix in order to ascertain a potential policy conflict. The conflict matrix is similar to the policy relationship matrix, but instead represents a group of desired relationships that could be ascertained in a particular application between two policies. The extension to the DEN-ng policy model depicted in figure 4.5 shows how the conflict matrix,

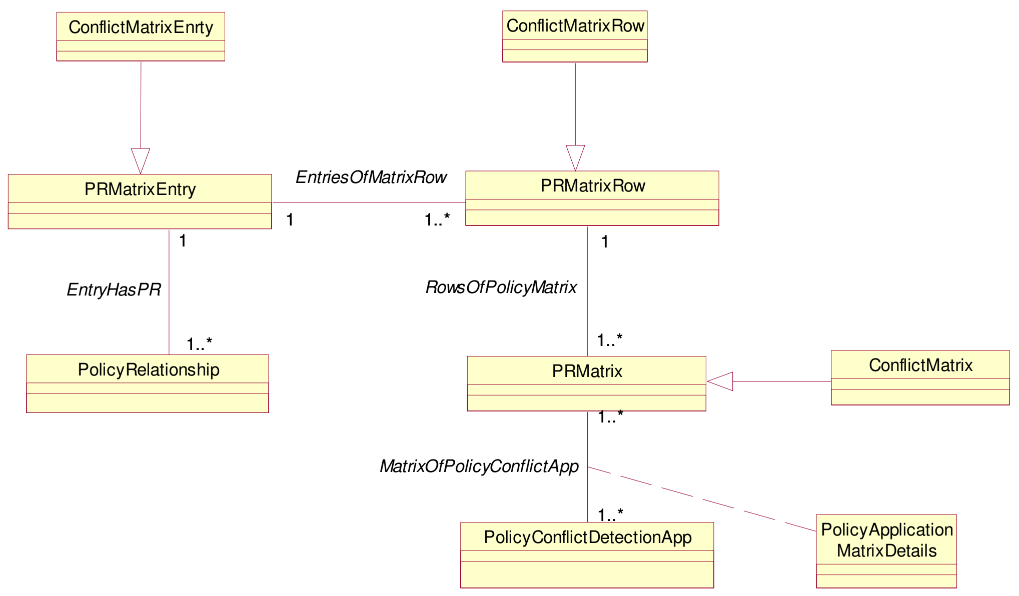


Figure 4.5: ConflictMatrix extension.

represented by the class `ConflictMatrix`, is an extension of the `PRMatrix` class. Similarly, the `ConflictMatrixRow` and `ConflictMatrixEntry` are extensions of the `PRMatrixRow` and `PRMatrixEntry` classes respectively. In describing an application specific conflict matrix, the information model architect should extend these provided classes and establish the associations to `PolicyRelationships` in order to represent the specific details of the application using policy conflict detection. The `ConflictMatrixEntry` should describe if the associated `PolicyRelationship` is required, optional or not required. This is catered for by describing an attribute in the class, if the attribute is set, then the relationship is required. If it is unset, the relationship is optional. If the class is not associated to in the matrix then the relationship is not required. The `ConflictMatrixRow` should describe if the group of `ConflictMatrixEntry` classes are required, optional or not required. The resulting `ConflictMatrix` is used in phase 2 of the conflict detection algorithm. There is also a selection process, where the appropriate `ConflictMatries` are chosen to be compared to associated `PRMatrices`. To aid in this selection process, all matrix classes are associated with specific sets of applications via the `PolicyApplicationMatrixDetails` association class, which represents the semantics of the `MatrixOfPolicyConflictApp` association. Essentially, the specific design of the matrix is tied to the application, and is independent from the conflict detection algorithm. It (as well as other classes described as part of this extension) are defined in the information model in order to define, in a generic fashion, the notion of policy conflict detection. This enables different applications to use the same set of concepts to define policy conflict.

## 4.2 Testbed Implementation

In order to evaluate the processes and algorithm documented in this thesis a policy based management testbed was implemented. The testbed has been designed to manage the resources and services of a simulated communications network modelled using OPNET$^{TM}$(OPNET, 2008). The design goals of the implementation are as follows:

- To be driven by a policy language tightly coupled to the information model of the managed system.
- To be compatible with the policy continuum interfaces defined in chapter 4.
- To enable multiple policy authors access to different management domains.
- To be flexible enough so that it can be readily extended to more policy applications.
- To support the validation and testing of the policy authoring process.

In light of these high level requirements the test bed was developed using model driven development (MDD) techniques. MDD enables the harnessing of the information contained within the UML model to aid in the automated generation of support tools and domain specific languages for use within the test bed. There are advantages to taking a MDD approach: 1) it alleviates the need to develop software tailored to a specific application domain of policy, as the information model dictates the application concepts; 2) there is reduced overhead involved in software development as code can be semi-automatically generated, therefore more time can be put into developing the information model, and subsequently reducing the time required to develop the software; and 3) the information model is used to drive the generation of the tools and languages; therefore, the information model is tightly coupled to the tools and hence can be rapidly updated should the model need modification.

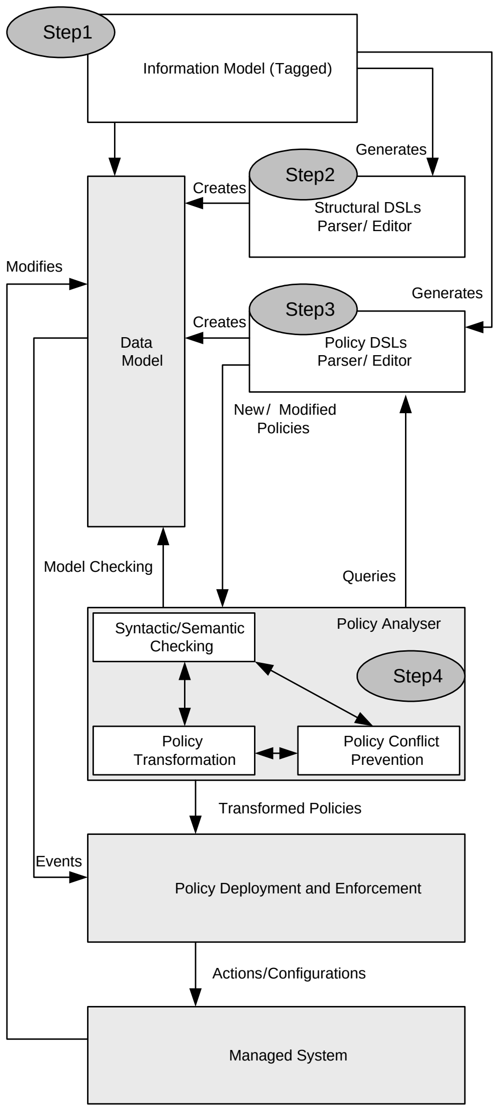


Figure 4.6: MDD process steps.

### 4.2.1 Model Driven Development based Process

A generic process was developed to follow in the construction of the testbed implementation. The process steps are depicted in figure 4.6. Each step indicates a part of the development process where a clear decision can be made as to the functionality that is required from the resulting testbed implementation. At step 1, an appropriate subset of the information model is identified against which the generation of support tools and languages will be carried out. Step 2 describes the generation of a structural domain specific language, which can be used to build instances of the information model that directly represent entities of the actual target managed system. Step 3 describes the generation of a policy language, the functionality of which depends on the subset of the information model used. Step 4 describes the generation of base support tools that aid in policy authoring, analysis and deployment. After all the support tools and languages have been developed, the implementation must be further extended to bind with the specific applications, services and/or resources that must be managed. Next a more detailed description of each step in the process is provided:

**Step 1 - Information Model Tagging.** Once a relevant information model subset is identified it must be marked so that this subset of the model alone is used for tool generation. For example, when using UML-based information models UML tagged value pairs can be used to identify the information model subset. Tagged values represent a simple extension to the meta-attributes of UML model elements. They add information to existing model elements for the benefit of back-end tools, such as code generators, report writers and simulators. Thus, the utility of the information model can be increased. A model element may have numerous markups (e.g. tagged value pairs), as each tagged value may be relevant for more than one tool.

**Step 2 - Structural DSL Generation.** To manually create an object model that represents the managed system (e.g. a communications network) using concepts specified in the information model, a domain specific language (DSL) is generated. This DSL is called a structural DSL as it is designed expressly to build groups of interrelated managed objects that represent the applications, services and resources that make up the system. Such a structural DSL, and its accompanying parser and editor, is cognisant of the types of entities that can be linked to one another and in precisely what manner. Multiple structural

DSLs, along with associated parsers and editors can be generated from separate, or possibly overlapping, identified and tagged subsets of an information model. This enables different object models to be defined that correspond to the different policy continuum levels of the managed system.

**Step 3 - Policy DSL Generation.** DSLs from step 2 are used to represent the structure, but not the behaviour of a managed system. The policy model subset of an information model (e.g. the modified DEN-ng policy model as defined in this thesis) includes entities to represent various components of policy rules that are used to specify part of the behaviour of a system. Similar to the generation of the structural DSL, policy DSLs are generated and along with associated parsers and editors. The policies defined using a policy DSL orchestrates the behaviour of managed entities described within the system, which has been populated using structural DSLs. A benefit of this is that the policy DSL editors can prevent users from defining policy instances over entities or entity types that do not exist within the managed system and that are inconsistent with the constraints defined in the information model.

**Step 4 - Support Tools Generation.** The editors generated for the policy DSL and structural DSL are not capable of performing in depth policy analysis, refinement, transformation, conflict detection and resolution or other similar processes demanded by most policy based management solutions. Therefore, some foundational tools are required to provide access to the code generated and the instantiated objects produced by the DSLs. The interfaces constructed can be leveraged by specially developed policy authoring and analysis processes and it is these interfaces that are built upon to implement the defined algorithms. The interfaces can be used to aid in the construction of databases for back end storage of policies. Also, the information model query interfaces and policy query interfaces can be generated from the information model.

### 4.2.2 Testbed Architecture

Following the process outlined in section 4.2.1, the policy based management testbed was developed as depicted in figure 4.7. The diagram illustrates that a structural DSL and editor GUI were generated. The structural DSL editor is used to populate the object model (instances of the information model classes) with information detailing those entities that exist in the managed system. The policy model defined in the information model can be

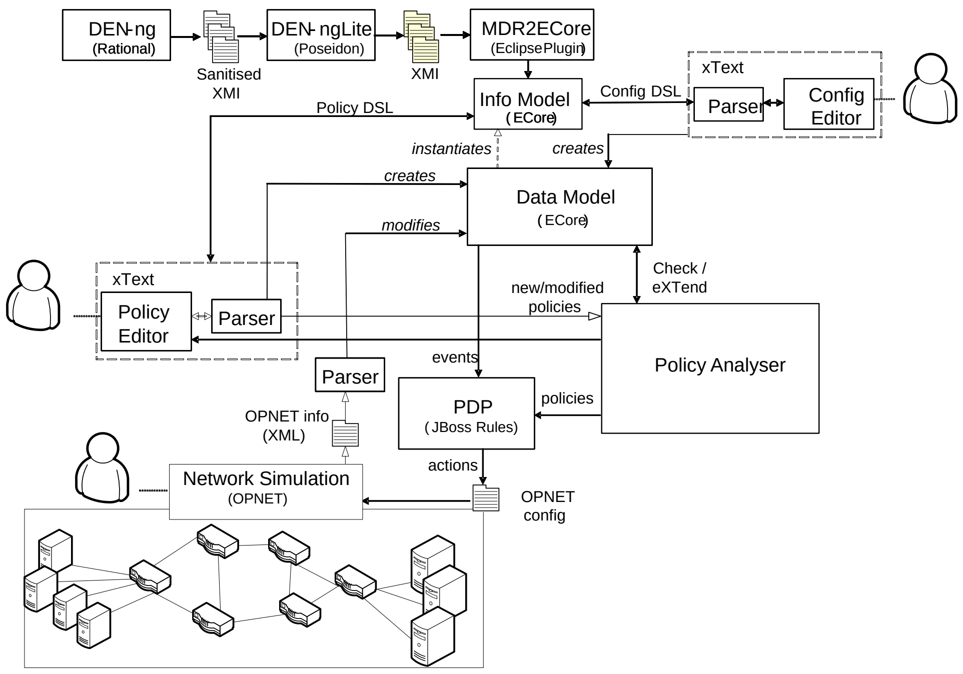


Figure 4.7: Implementation diagram.

used to generate a policy language that can in turn be used by policy authors to define policies. The type of policy language generated is determined by the portion of the policy information model used. Multiple policy languages can be generated if only specific aspects of the policy information model are used. For example, if the policy information model and its associations to customer and product are used to generate a policy DSL, then that policy DSL can be used to describe the behaviour of customers and products. The policy authoring GUI was also generated, along with an associated DEN-ng compatible policy language that can be used to define management policies over the entities populating the object model. Also generated were the interfaces to the object model to enable querying of policies and entities.

To follow the steps, a version of the DEN-ng information model was exported from Rational Rose (Rational Rose, 2003) (which is the tool used to define the DEN-ng information model) and imported into Poseidon UML (Poseidon, 2008). At the time of development, Poseidon was a free tool, and made the information model more accessible to the open source community. Note that only the portion of the DEN-ng information model that was of relevance to the management of the communications network described above was imported. From here, tagged values were added to the classes defined in the information model. A tool developed at the TSSG named MDR2Ecore was used to transform the tagged DEN-ng information model from UML to an Ecore representation (Barrett et al., 2007). Ecore is a platform specific modelling language developed to support model driven development on the Eclipse IDE platform (Eclipse, 2008*b*), it is the core component of the Eclipse Modelling Framework, EMF (Eclipse, 2008*a*). Once the model has been transformed from UML to Ecore, the many hundreds of plug-ins that are being developed in the open source community for Ecore can now be leveraged for this work. In particular, the Eclipse plug-ins developed by openArchitectureWare (oAW, 2008) were chosen to perform MDD as their implementations are mature and the most functional at the time of development of this test bed. Alternatives would have been to use AndroMDA (AndroMDA, 2008) or similar model driven tool-kits; however, the tool functionality of oAW was very attractive, as it supported the generation of code from models and the generation of DSLs from models.

The subset of the information model focused on for the generation of a structural DSL, was derived from DEN-ng. This DEN-ng subset defined logical resources and services. The logical resource portion is a superset of the functionality defined in the SID addendum

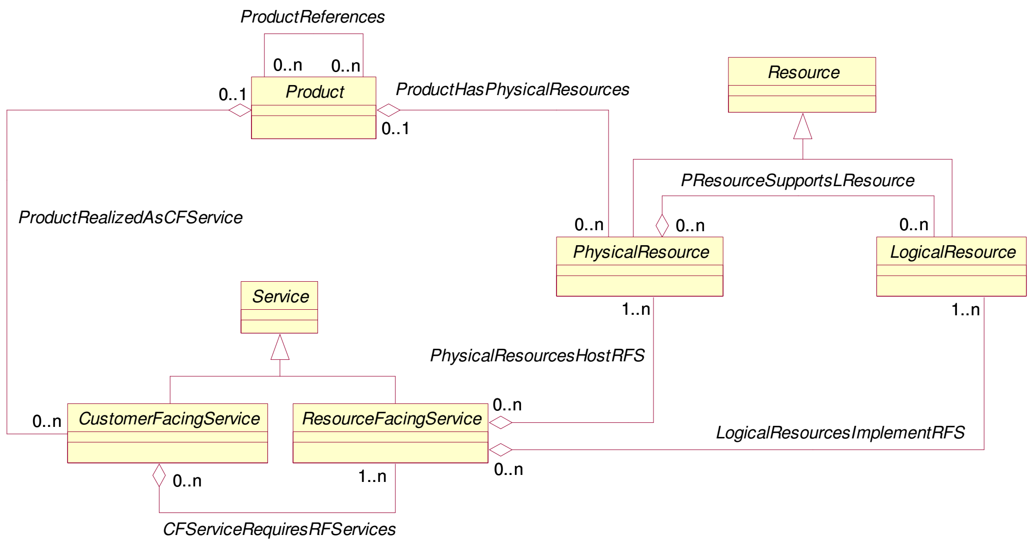
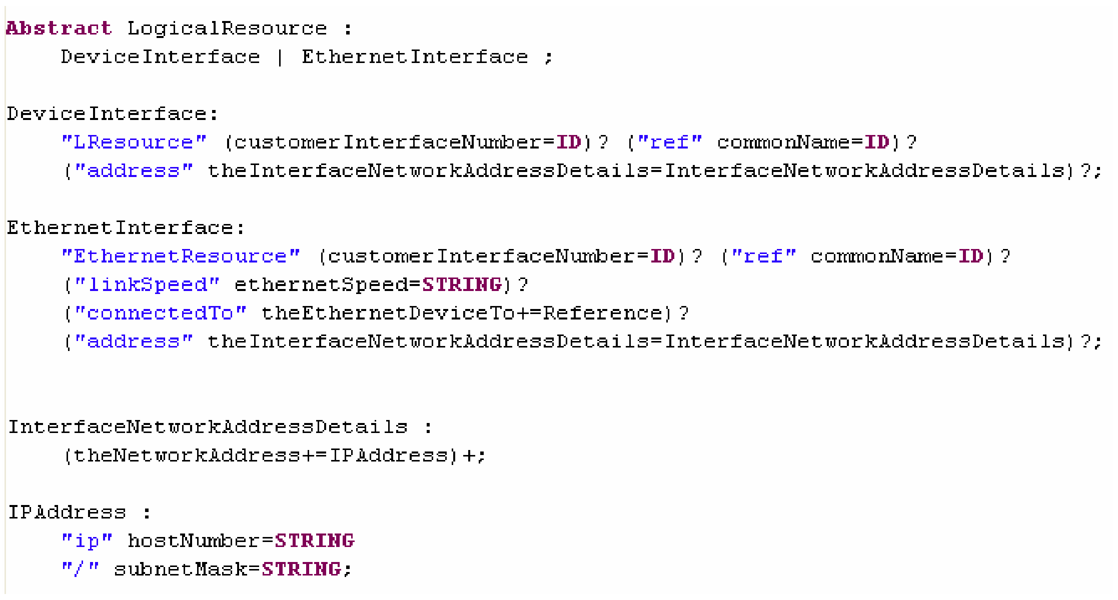


Figure 4.8: Product-Service-Resource DEN-ng

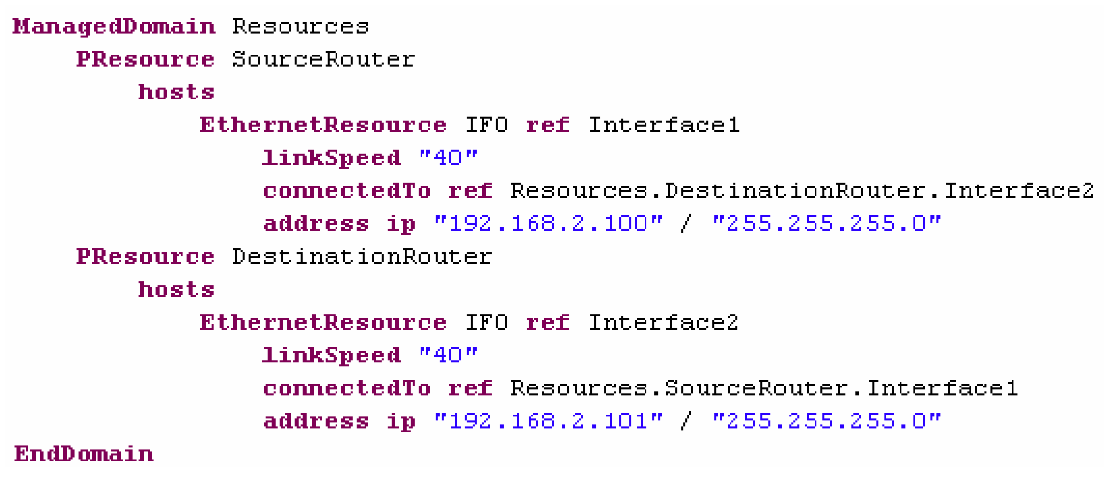
GB922 5LR (TMForum, 2008). This SID addendum defines a model of logical resources, such as routers and application servers. A picture illustrating a portion of the model is shown in figure 4.8. This figure shows the relationships between a product, service and resource as modelled in DEN-ng. The modelling of services that can be provided over the network to users was also needed; thus, a subset of the DEN-ng service model was defined, which is equivalent to a superset of the SID addendum GB922 4SO. The subset of the information model used to generate the policy DSL was as described in the chapter 3 section 3.1.1.

A snippet of the structural DSL is depicted in figure 4.9. Using oAW's Xpand model-to-text plug-in, a script was developed to generate the specification of the DSL from an Ecore model. This particular snippet is derived from a set of classes describing `LogicalResources`, and in particular `DeviceInterface` and `EthernetInterface`. A sample usage of this part of the structural DSL is given in figure 4.10. This usage snippet describes a collection of routing devices with manually configured IP addresses that are directly connected to each other. The structural DSL can be used to construct a complete communications network by describing sets of connected logical resources. Network services can also be similarly defined to run on the respective resources and this is catered for in DEN-ng in the associations between `Service` with `Resource`. However, as manually describing a large communications network may be time consuming, another plug-in was developed that can



```
Abstract LogicalResource :
    DeviceInterface | EthernetInterface ;

DeviceInterface:
    "LResource" (customerInterfaceNumber=ID)? ("ref" commonName=ID)?
    ("address" theInterfaceNetworkAddressDetails=InterfaceNetworkAddressDetails)?;

EthernetInterface:
    "EthernetResource" (customerInterfaceNumber=ID)? ("ref" commonName=ID)?
    ("linkSpeed" ethernetSpeed=STRING)?
    ("connectedTo" theEthernetDeviceTo+=Reference)?
    ("address" theInterfaceNetworkAddressDetails=InterfaceNetworkAddressDetails)?;


InterfaceNetworkAddressDetails :
    (theNetworkAddress+=IPAddress)+;

IPAddress :
    "ip" hostNumber=STRING
    "/" subnetMask=STRING;
```

Figure 4.9: Structural DSL snippet.



```
ManagedDomain Resources
    PResource SourceRouter
        hosts
            EthernetResource IF0 ref Interface1
                linkSpeed "40"
                connectedTo ref Resources.DestinationRouter.Interface2
                address ip "192.168.2.100" / "255.255.255.0"
    PResource DestinationRouter
        hosts
            EthernetResource IF0 ref Interface2
                linkSpeed "40"
                connectedTo ref Resources.SourceRouter.Interface1
                address ip "192.168.2.101" / "255.255.255.0"
EndDomain
```

Figure 4.10: Structural DSL usage snippet.

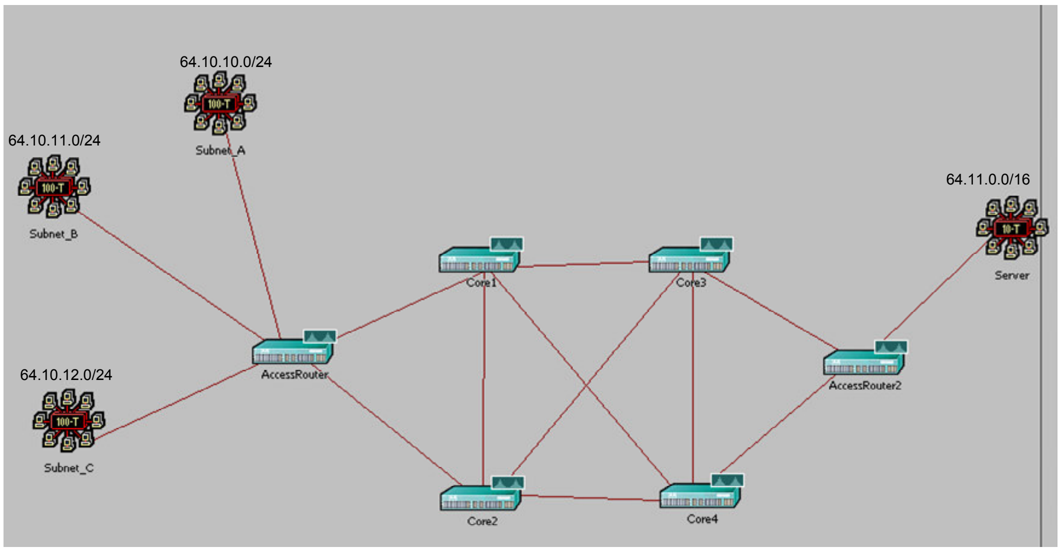
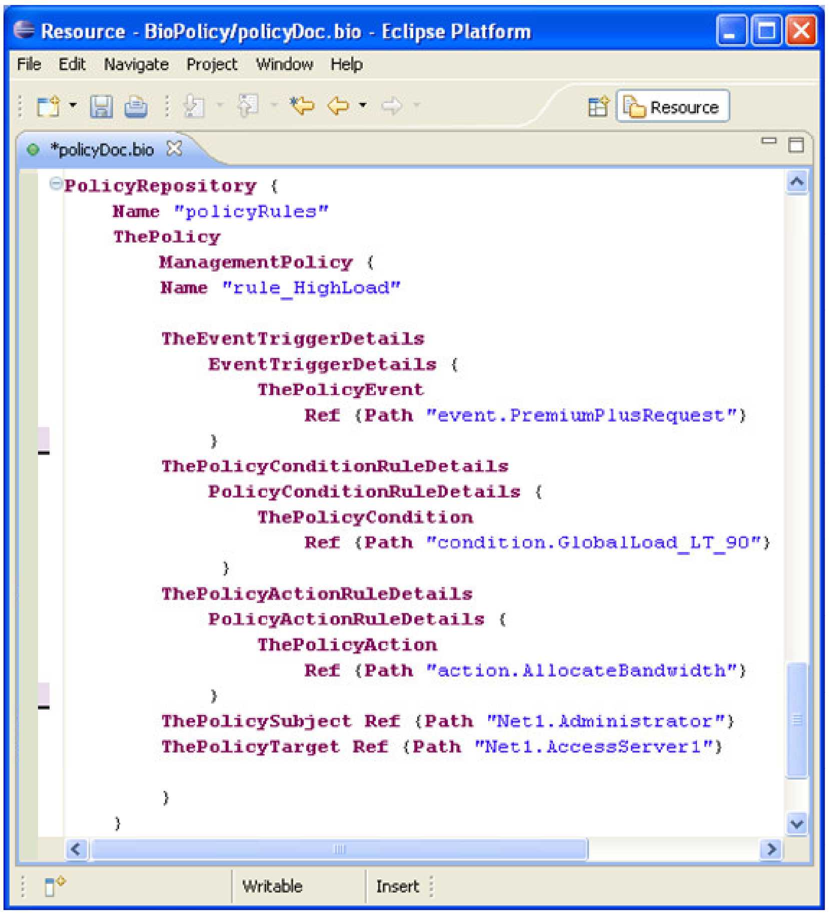


Figure 4.11: OPNET simulated network

parse the configuration file generated from OPNET, and construct the approximate DSL to reflect the simulated routers and services. A PDP and PEP were also developed to control the deployed routers and services. Therefore, when an object was altered in the object model, this modification would be propagated to the simulation environment.

The communications network that was designed in OPNET is an IP core network, with Differentiated Service enabled routers at the edge (shown in figure 4.11). The network is designed to support the provision of access to a set of servers from a set of client domains. The links across the core network are set at 100Mbps and the edge links are also 100 Mbps. The types of services that are considered are firewall filtering services, IPsec VPN services and DiffServ QoS services.

The policy DSL is generated in a similar manner to the structural DSL. The generated policy editor for the policy DSL provides very limited policy analysis, limited to syntactic analysis. The tool offers some functionality for more expressive analysis by providing the policy editor with an interface to the "Check" constraint checking language provided as a plug-in by oAW (openArchitectureWare, 2008). The "Check" language is a constraint language much like the Object Constraint Language (OCL) that enables the tool developer to construct OCL style statements to verify the consistency of the created policies. These statements, however, are primarily used only to check the values of parameters used to create the policies. For example, a typical Check statement may ensure that each policy has a unique identifier, or that only specific values can be used to set Differentiated Services


Resource - BioPolicy/policyDoc.bio - Eclipse Platform
File Edit Navigate Project Window Help
Resource
*policyDoc.bio
Writable
Insert

```
PolicyRepository {
    Name "policyRules"
    ThePolicy
        ManagementPolicy {
        Name "rule_HighLoad"

        TheEventTriggerDetails
            EventTriggerDetails {
                ThePolicyEvent
                    Ref {Path "event.PremiumPlusRequest"}
            }
        ThePolicyConditionRuleDetails
            PolicyConditionRuleDetails {
                ThePolicyCondition
                    Ref {Path "condition.GlobalLoad_LT_90"}
             }
        ThePolicyActionRuleDetails
            PolicyActionRuleDetails {
                ThePolicyAction
                    Ref {Path "action.AllocateBandwidth"}
            }
        ThePolicySubject Ref {Path "Net1.Administrator"}
        ThePolicyTarget Ref {Path "Net1.AccessServer1"}

        }
    }
```

Figure 4.12: Sample policy rule.

Code Point (DSCP) values. A sample policy rule is depicted in figure 4.12, this rule specifies that when a request for bandwidth is received from a PremiumPlus customer (the policy event), and the network is congested to less than 90% of its capacity (the condition), then the bandwidth is allocated (the action).

Once the policies have been fully analysed and checked for consistency against system constraints and against each other, they must be deployed. The JBoss Rule engine (Drools, 2008) was used to actively monitor the managed system and maintain it in accordance with currently deployed policies. The JBoss Rules engine is a forward chaining rule engine that is based on the RETE OO pattern matching algorithm (Forgy, 1982). It provides a highly efficient method of evaluating large volumes of rules simultaneously and carrying out safe and orderly execution.

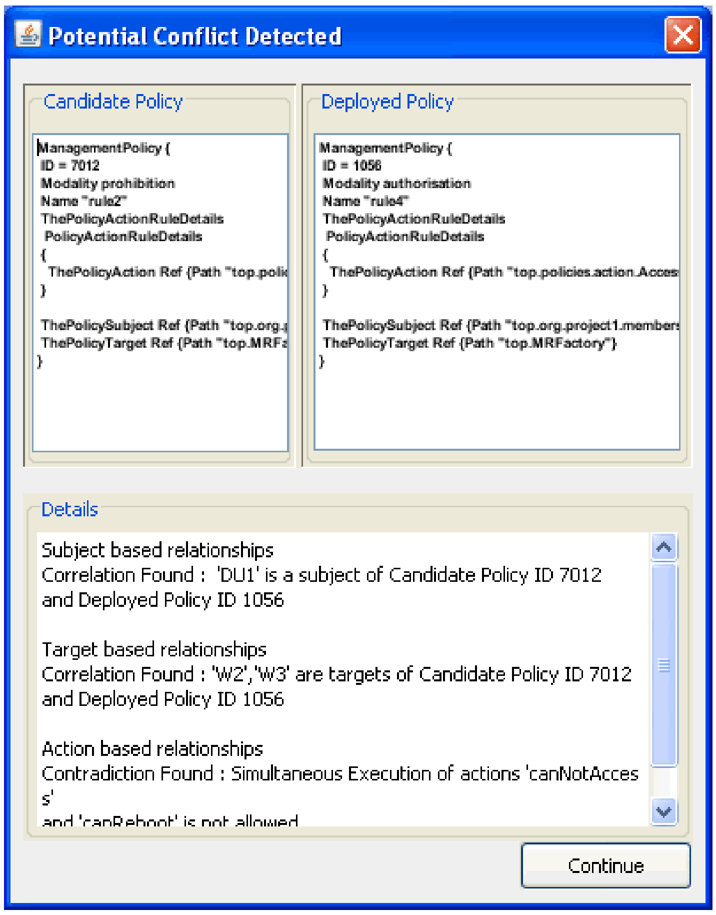
To monitor the simulated network, current configurations and status are exported to a file. This file is then parsed and used to construct the state of all routers and services. Any changes in system state (as compared to the previously monitored system state) are asserted into the JBoss Rule engine as facts. Asserted facts can be used to trigger the evaluation of installed rules in the rule base as the condition components of rules (policies) can be defined over the status of managed entities. Those rules that must be executed are added to the activation agenda and processed in accordance to rule priority. The actions taken by the rules are used to modify the current object model representing the state of the managed system. Any change in configurations are transformed into a corresponding OPNET configuration and imported back into OPNET, so that the effect of these changes can be simulated. The affect the operations have on the communications network is then modelled using OPNET.

To translate the policies into the JBoss Rule language, which is called Drools, oAWs Xpand model-to-text language was used. Depending on the type of policy, there is a mapping defined to translate it into a rule language. The translation process is outlined as follows:

1. Events are transformed into statements that appear at the head of the rule.
2. Conditions are transformed into statements that may reference system state, event attributes, or global attributes such as time. These statements are placed after the events.
3. Actions are placed in the consequent part of the rule, meaning that the actions will execute only if the events mentioned occur and the conditions are satisfied. Depending on the type of policy, the action may define the installation of a specific configuration or a change to a configuration, or it may return the answer to an access request.

### 4.2.3 Policy Analyser

The policy analyser uses the interfaces generated at step four in the process to access the instantiated objects of the managed system and the instances of policy defined by the policy author at the policy editing GUI. The policy conflict analysis algorithm is implemented directly in Java and makes extensive use of oAW plugins to access data. The algorithm has been extended to store objects in the policy relationship matrix as opposed to the



```
ManagementPolicy {
 ID = 7012
 Modality prohibition
 Name "rule2"
 ThePolicyActionRuleDetails
  PolicyActionRuleDetails
 {
   ThePolicyAction Ref {Path "top.poli
 }

 ThePolicySubject Ref {Path "top.org.
 ThePolicyTarget Ref {Path "top.MRFa
}
```

```
ManagementPolicy {
 ID = 1056
 Modality authorisation
 Name "rule4"
 ThePolicyActionRuleDetails
  PolicyActionRuleDetails
 {
   ThePolicyAction Ref {Path "top.policies.action.Acces
 }

 ThePolicySubject Ref {Path "top.org.project1.member
 ThePolicyTarget Ref {Path "top.MRFactory"}
}
```

```
Subject based relationships
Correlation Found :  'DU1' is a subject of Candidate Policy ID 7012
and Deployed Policy ID 1056

Target based relationships
Correlation Found : 'W2','W3' are targets of Candidate Policy ID 7012
and Deployed Policy ID 1056

Action based relationships
Contradiction Found : Simultaneous Execution of actions 'canNotAcces
s'
and 'canReboot' is not allowed
```

Figure 4.13: Policy conflict dialog box.

ones and zeros as outlined in the algorithm specification. Therefore, when a relationship is established between two policy component types, an object describing the nature of the relationship is stored in the matrix. More extensive information can then be provided back to the policy author concerning the nature of any policy conflicts that may occur. For example, once the policy author hits the analyse button after they have modified or created a policy, a dialog popup is displayed similar to that depicted in figure 4.13, should a potential policy conflict be discovered. The conflict matrices are stored alongside the policies and other object data and can be search and queried. The particular conflict matrices retrieved depend on the types of policies being currently analysed. The policy continuum is implemented as a set of policy langauges and operations that can modify the relationships explicitly linking policies in the policy continuum together. The refinement process is trivial for the moment as it is not the primary focus of the testbed.

## 4.3 Case Studies

This section describes the operation of policy conflict analysis for a policy continuum via a set of case studies. The case studies collectively relate to an ISP defining policy to manage the provisioning of Internet service products across its communications network. Policies are defined from the perspective of two management domains that have different concerns about the organisation and its services and resources. The management domains of users and services are as depicted in figure 4.14. The ISP's network administrators (DU1-3) define policy to effect management of network services and resources. In contrast, the ISP's business oriented sales users define policy to provision products based on Internet service grades. The management domains of users and services are as depicted in figure 4.14.

Three cases are examined. The first case analyses the interaction of firewall filtering policies defined at the system level with service provisioning policies defined at the business level. This case demonstrates the interaction between policies authored at different levels and by different users of the policy continuum. The second case examines the interaction of access control policies defined at the business level, but uses different types of conflict matrices; this demonstrates the versatility of the conflict analysis algorithm. The third case illustrates that if the policy model being used is extended, that the conflict analysis algorithm can also be extended to account for the new functionality provided and hence new forms of conflict that may arise. The policy model makes use of deontic concepts

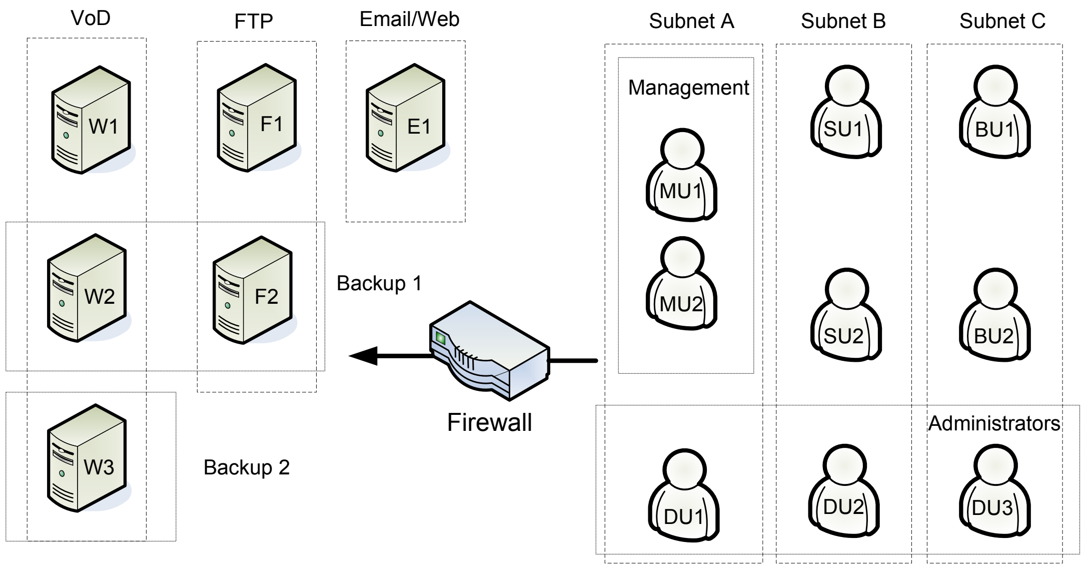


Figure 4.14: Policy authoring scenario domain hierarchy.

and illustrates how the conflict analysis algorithm can be extended to detect new forms of policy conflict based on deontic policies.

### 4.3.1 Business Level / System Level – Filtering Policy Conflict

At the business level, a policy author defines a set of policies that relate to the grades of Internet service package that a customer is assigned to, and which services it can utilise. Table 4.2 depicts a set of current business policies (in a pseudo policy language for the sake of brevity and clarity).

These policies are assumed to have already been added to the policy repository and that there were no potential conflicts detected. A refinement process is assumed to have derived a set of related system level policies that can fulfil the network resource access and packet filtering requirements that were previously defined by the business level policies. These are also depicted in Table 4.2. The network administrator needs to define a policy that will restrict video on demand traffic, so that further policies can be installed to perform essential file backup during off-peak hours. The candidate system level policy is defined at the bottom of Table 4.2 and should be analysed against all appropriate system level policies (i.e., those that are at the same level in the policy continuum).

Table 4.2: Business level policies and system level policies.

| Business Level (1) | System Level (2) | |
|---|---|---|
| **[ID 1]** Subnet_A *isAssignedForward*{VoD,FTP,Web} | **[ID 1.1]** On IPPacketRecieved at AccessRouter.IF0<br>Condition: SourceIPin64.10.10.0/24<br>DestIP in 64.11.1.0/24<br>DestPort equals 2000<br>Action: **Forward** | **[ID 1.3]** On IPPacketRecieved at AccessRouter.IF0<br>Condition: SourceIP in 64.10.10.0/24<br>DestIP in 64.11.3.0/24<br>DestPort equals 80<br>Action: **Forward** |
| | **[ID 1.3]** On IPPacketRecieved at AccessRouter.IF0<br>Condition: SourceIP in 64.10.10.0/24<br>DestIP in 64.11.2.0/24<br>DestPort equals 21<br>Action: **Forward** | |
| **[ID 2]** Subnet_B *isAssignedForward* {FTP,Web} | **[ID 2.1]** On IPPacketRecieved at AccessRouter.IF0<br>Condition: SourceIP in 64.10.11.0/24<br>DestIP in 64.11.2.0/24<br>DestPort equals 21<br>Action: **Forward** | **[ID 2.2]** On IPPacketRecieved at AccessRouter.IF0<br>Condition: SourceIP in 64.10.11.0/24<br>DestIP in 64.11.3.0/24<br>DestPort equals 80<br>Action: **Forward** |
| **[ID 3]** Subnet_B *isAssignedDrop* {VoD} | **[ID 3.1]** On IPPacketRecieved at AccessRouter.IF0<br>Condition: SourceIP in 64.10.11.0/24<br>DestIP in 64.11.1.0/24<br>DestPort equals 2000<br>Action: **Drop** | |
| **[ID 4]** Subnet_A *isAssignedProduct* GoldInternet | **[ID 4.1]** On IPPacketRecieved at AccessRouter.IF0<br>Condition: SourceIP in 64.10.10.0/24<br>DestIP in 64.11.1.0/24<br>DestPort equals 2000<br>Action: **Mark AF21** | |
| **[ID 5]** Subnet_B *isAssignedProduct* SilverInternet | **[ID 5.1]** On IPPacketRecieved at AccessRouter.IF0<br>Condition: SourceIP in 64.10.11.0/24<br>DestIP in 64.11.1.0/24<br>DestPort equals 2000<br>Action: **Mark AF31** | |
| | **[ID 0.1]** On IPPacketRecieved at AccessRouter.IF0<br>Condition:SourceIP in 64.10.0.0/16<br>DestIP in 64.11.1.0/16<br>DestPort in 2000<br>Action: **Mark AF21** | |

The new candidate policy (at the bottom of the table) defines that the video on demand service access (port 2000) must be remarked with a DSCP of AF21, thus reducing its priority and its ability to congest the core links of the communications network. The conflict matrix that is tested relate to detecting conflict relationships specifically among network filtering polices; this means that particular attention must be paid to the overlapping condition in the IP Header match criteria of the deployed policies. From examining the policies, the candidate policy installed by the network administrator may potentially

$$\begin{bmatrix} 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \end{bmatrix} \circledast \begin{bmatrix} 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ \mathbf{1} & 0 & \mathbf{1} & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \end{bmatrix} = \mathbf{1}$$

Figure 4.15: Phase 2 relationship matrix and conflict matrix for business to system level conflict

conflict with several deployed policies, including ID3.1. The policy conflict analysis processes the candidate policy with each of the deployed policies, and for each pair of policies analysed, a set of relationship matrix is built. The relationship matrix established between the candidate policy and

The policy relationship matrix between the candidate policy (ID 0.1) and the deployed policy (ID3.1) is depicted in figure 4.15. The conflict matrix used to determine a case for conflict is chosen from a set of conflict matrices that represent well known conflict types that can occur among network filtering policies. In this example, the conflict matrices represent (1) policy subjects that are the same, (2) policy targets that are the same, (3), identical events, (4) subset, equal or correlated condition, and (5) contradicting actions. When a potential conflict is detected, the encompassing policy authoring notifies the current policy author with information concerning the discovered potentially conflicting policies..

In this case, the policy with ID 3 at the business level is a parent policy of the problematic deployed policy (ID3.1). Subsequently, an alert is presented to the system level policy author describing the potential conflict detected; in parallel, a list of higher level policies that the deployed policy is related to is also presented to the policy author. By providing a list of higher level policies the policy author is given more context as to why the deployed policy exists and who to coordinate with in order to find out more information about the policies in order to resolve the conflict.

This case study demonstrates that policy conflict can be discovered that originates from the authoring of policies at different levels of the policy continuum, by

1. Analysing policies at the same level and then,

2. Tracing up the policy continuum to ascertain the parent policies associated to the conflicting policies.

The next case studies illustrate and discuss the flexibility of using the conflict matrix to

Table 4.3: Business level access control policies.

| Business Level (1) |
|---|
| [ID 1] **Subnet_A** ***isAssignedForward*** **{VoD,FTP,Web} during interval (09:00 – 18:00)** |
| [ID 2] **Subnet_B** ***isAssignedForward*** **{FTP,Web} during interval (09:00 – 18:00)** |
| [ID 3] **Subnet_B** ***isAssignedDrop*****{VoD} during interval (09:00 – 18:00)** |
| *[ID 4] **Administrators** ***isAssignedForward*** **{VoD,FTP,Web} during interval (17:00 – 23:00)** |

dictate the definition of conflict by utilising a different conflict matrix for a different pair of policies.

### 4.3.2 Business Level Policies – Access Control Policy Conflict

This case details a policy conflict that exists solely at the business level. Therefore, the candidate policy does not need to be refined and the policy author can be immediately notified. This is illustrated in the policy authoring process as presented in chapter 3 where the process exits if a conflict is detected.

The business level policy author defines policies that control access to a set of services available to a set of customers. The currently deployed policies are as outlined in table 4.3, and a new candidate policy is defined at the bottom of table 4.3, marked with an asterisk. The candidate policy defines that users within the Administrators managed domain are granted access ($isAssignedForward$ allows traffic to be forwarded) to all services during the interval 17:00 to 23:00. The first phase of the algorithm compares this candidate policy with all deployed policies to define all appropriate relationships that exist among the policies. The second phase then examines the derived relationships matrices and discovers that a potential conflict exists among the candidate policy and the deployed policy with ID 3. A conflict exists because there is an overlap between the subject components of

$$\begin{bmatrix} 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \end{bmatrix} \circledast \begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \end{bmatrix} = \mathbf{1}$$

Figure 4.16: Phase 2 Relationship Matrix (LHS) and Conflict Matrix (RHS) for Business to Business Level Conflict.

the policies (i.e. user DU2 is shared between Subdomain B and Administrators) there is a target component overlap (since both policies reference the VoD service), and there is a correlation among the condition components (meaning that they temporally overlap). The existence of these relationships indicates a potential conflict as specified in the conflict matrix depicted in figure 4.16. Once the potential conflict is detected, details describing the conflict are relayed back to the candidate policy author.

### 4.3.3 Business Level Policies - Deontic Policy Conflict

Deontic concepts add extra expressiveness to policies; consequently, there are also more chances for policies to conflict with each other. In DEN-ng, deontic policies are classified by type, as they are subclasses of `ManagementPolicy`. Deontic policy extensions to DEN-ng are covered in section 3.1.1 on page 60. This new component of policy, namely its type, introduces a new dimension which can be used to relate policies. Therefore, in order to take advantage of a new policy relationship, the existing algorithm must be extended by introducing a new row into the policy relationship matrix, and introducing new `PolicyRelationship` classes in the DEN-ng information model. The new row functions as follows, there are five new entries in the matrix; therefore, the policies can be compared in five new ways.

1. Exemption vs Authorisation (dea)
2. Obligation vs Prohibition (dop)
3. Exemption vs Obligation (deo)
4. Authorisation vs Prohibition (dap)
5. Similar Positive Types (dspt)

$$\begin{bmatrix} ssb & ssp & seq & scor & 0 \\ tsb & tsp & teq & tcor & 0 \\ esb & esp & eeq & ecor & emux \\ csb & csp & ceq & ccor & cmux \\ asb & asp & aeq & acor & actd \\ dea & dop & deo & dap & dspt \end{bmatrix}$$

Figure 4.17: Policy relationship matrix with Deontic relationships.

The first type of relationship signifies that one of the policies (the candidate or the deployed) is an exemption policy and the other is an authorisation policy. This type of relationship between two policies can be trivially established by comparing the types of policies together ("dea" in the relationship matrix). This logic also applies for cases 2, 3 and 4 ("dop","deo" and "dap" in the relationship matrix). The fifth relationship type means that there are two policies of the same type (i.e., authorisation or obligation) or that one policy is of type authorisation while the other policy is of type obligation (dspt in the relationship matrix). In this case, the policy type does not stop the associated action from being performed.

The relationships of Exemption vs Prohibition are not recorded, as both policy types deter the actions defined in the policy from being performed. The new policy relationship matrix is now depicted in figure 4.17. The algorithm for computing the values for the matrix is also extended with a new function that is called to compute the relative relationships between two policies as a function of their policy type. The new entries are prefixed with a *'d'* to represent their deontic nature.

To demonstrate the operation of the extended policy relationship matrix, the policies depicted in figure 4.4 are considered. Note that the keywords used in the pseudo policy languages dictate the type of policies being considered. The keyword **must** indicates an obligation policy, **isPermitted** indicates an authorisation policy, **isNotPermitted** indicates a prohibition policy and **mayNot** indicates an exemption policy. The business policy author describes a set of policies to permit and restrict particular operations of user groups and individuals.

Policy 1 is an obligation policy that states that user MU1 must upload a specific directory to the FTP set of servers. This policy may be used to ensure regular backups are made of sensitive documents that user MU1 is currently working on. Policy 2 is an authorisation policy and states that users from the Subnet A domain can upload files to the FTP servers from 16:00 to 18:00. This policy ensures that the obligation policy

Table 4.4: Deontic Policies.

| Business Level (1) |
|---|
| [ID 1] **MU1** must **uploadDirectory "X" to FTP at 17:00** |
| [ID 2] **Subnet A** isPermitted **to upload to FTP from 16:00 to 18:00** |
| *[ID 3] **Management** mayNot **upload to Backup1 from 16:00 to 18:00** |

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \circledast \begin{bmatrix} 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Figure 4.18: Relationship matrix between policy 3 (candidate) and policy 1 (deployed).

defined in Policy 1 will be permitted. Now, assume that after these two policies have been defined, another business policy author defines Policy 3 and hits the analysis button in the authoring GUI. Policy 3 is a prohibition policy stating that users in the Management domain may not upload to the machine Backup 1 between 16:00 and 18:00. Analysis of the policy produces the following policy relationship matrix with Policy 1 (figure 4.18) and Policy 2 (figure 4.19).

The relationship matrix produced between Policy 3 and Policy 1 states that Policy 3 is a super set of Policy 1 via the subject component, and a subset via target component. The

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} \circledast \begin{bmatrix} 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Figure 4.19: Relationship matrix between policy 3 (candidate) and policy 2 (deployed).

policies are associated by having equal events (i.e., none, which in DEN-ng is translated as "always true", since all DEN-ng policy rules must have event, condition and action clauses). The conditions associated to Policy 3 are a super set of the conditions of Policy 1. Policy 3 contains a superset of the actions specific in Policy 1, this is because Policy 3 specifies "upload", whereas Policy 1 specifies "upload a directory". The last relationship type is the deontic comparison, and the appropriate relationship discovered is Exemption vs Obligation. A conflict is established, since all relationship requirements set out in the conflict matrix are satisfied. A similar relationship can be established between Policy 3 and Policy 2, where a potential policy conflict also exists. The action components being compared for policy 2 and 3 are equal in that they both specify "upload", but the targets of the are related via subset.

## 4.4 Summary and Discussion

A generic and extensible algorithm was presented that can analyse for policy conflict at multiple levels of a policy continuum and for multiple policy applications. Current policy conflict analysis algorithms are primarily focused on detecting policy conflicts within specific applications and make assumptions about the policy model in use. By designing a conflict analysis algorithm for specific applications, then a lot of different algorithms are needed when there are multiple applications. For a large communications network many conflict analysis algorithms are not feasible. Therefore, an extensible policy conflict analysis algorithm that is application independent is developed as required. Research question 2 as specified in chapter 1 asks how can the information model be leveraged to aid in policy conflict analysis. This question is addressed in this chapter as the conflict detection algorithm uses the information model to abstract from the type of policy conflicts that can be detected and the types of policy relationships that can be established. Research question 3 as specified in chapter 1 asks how a policy conflict analysis process be defined independent of the nature of the policies the must be analysed. The solution presented in this chapter addresses this question by harnessing the knowledge embodied in an information model.

Extensibility is achieved by separating the information specific to an application from the conflict analysis algorithm. As the conflict analysis algorithm makes use of a generic interface to an information model, the model can be changed depending on the application. The definition of a potential policy conflict is also decoupled from the conflict analysis algorithm, as the description of a conflict is encoded in a relationship matrix and can be stored in the information model. This was demonstrated by extending DEN-ng to model policy relationships. The extensibility of the algorithm is illustrated and discussed through three diverse case studies, where different conflict matrices and different types of policy applications are used.

The detection of a policy conflict is directly related to the ability to relate components of policy together in ways that satisfy the needs of an application. However, there are some current limitations to this approach. The expressiveness of the information model is somewhat limited and currently, there is only limited support in UML to verify the semantics of the model. As the conflict analysis algorithm is extensible, new ways of representing the semantics associated to the information model can be leveraged. An extension to the presented conflict analysis algorithm is the topic of the next chapter where the information model is augmented with more extensive semantics represented by accompanying ontologies. Such extensive semantic information can lead to the representation and discovery of a wider range of policy conflicts. Another very important feature of the policy authoring process that needs to be addressed is the policy selection algorithm that controls which policies are input into the conflict detection algorithm. Ontologies can be used to aid the selection algorithm to select those policies that require further conflict analysis. This is the focus of the next chapter.

# Chapter 5

# Enhancing Policy Conflict Analysis using Ontologies

Policy conflict analysis within the policy authoring process comprises two main steps as described in the chapter 4. They are policy selection and policy conflict analysis. The deployed policies supplied to the conflict analysis algorithm are determined by the policy selection algorithm. In order to select a subset of deployed policies (at the same policy continuum level only) that may conflict with the candidate policy currently being analysed, the selection algorithm requires the ability to differentiate the types and purpose of different deployed policies.

A policy selection algorithm is presented that is capable of determining a subset of deployed policies that must be selected and passed to the conflict analysis algorithm without requiring the thorough analysis of all deployed policies. This is done by incorporating an ontology associated with an information model, to represent the semantic relationships among various types of policies. Specially designed selection rules are added to the ontology that can determine which policies should be analysed for conflict. The selection rules provide a form of lightweight analysis that can aid in reducing the need to perform a potentially computationally complex analysis of a large number of possibly irrelevant policies. This reduction in complexity is achieved by using policy relationships that are pre-defined in the ontology. This step then pushes that responsibility to a knowledge engineering expert that can coordinate with the application domain expert to define the appropraite selection rules. This type of selection is particularly useful when dealing with policies that apply to different sets of resources, where complex forms of policy conflicts can exist due to the implicit relationships between resources. In this case, the conflict analysis algorithm is extended to be able to ascertain a case for potential policy conflict when policies apply to different resources. Information models are not designed to represent rich semantic information; hence, ontologies are used to augment the information model by explicitly indicating the policy component types that are incompatible with each other. This semantic information is defined in the ontologies and is used by the the conflict analysis algorithm, which uses this information to establish relationships among the candidate policy and deployed policies.

In section 5.1, the approach is motivated by describing a typical example where the policy conflict analysis algorithm is used. The example illustrates that determining a case for conflict in a scenario where the resources are physically distributed can be computationally complex and some special forms of conflict may go undetected if the conflict analysis algorithm is not enhanced. Section 5.2 discusses how the ontology can be leveraged to create a policy selection algorithm driven by selection rules. It also discusses how the policy conflict analysis algorithm is extended to relate policy components via ontological relationships. Section 5.3 describes the ontology construction process to accommodate the use of ontologies. A usage scenario is outlined and discussed in section 5.4 to illustrate the problems that this approach can now handle and how the use of ontologies aids in the analysis of policies for conflict. Discussions and a summary are presented in section 5.5.

## 5.1 Motivating Example

The different ways that entities can be related to each other depends on the relationships expressed within the information model. Establishing relationships such as equality, subset, superset, and correlation of groups or individual managed entities is possible; however, in certain situations that typically arise in large communications networks, policies may need to be analysed against each other even if they are not being applied to the same entities. This situation raises three important issues:

1. When analysing policies defined to manage different resources having different functionality, the set of relationships will be diverse; this increases the runtime complexity of the policy conflict analysis algorithm, as there is a increase in the number of policies that must be analysed.

2. When analysing policies defined to manage physically distributed resources, it is difficult to determine those policies the need to be analysed without retrieving all

deployed policies, this increases the runtime complexity associated with the policy conflict detection algorithm.

3. Policies applying to distributed resources may implicitly rely on other policies in the network, depending on the type of policy the reliance may be non trivial, and thus pose significant problems when ascertaining cases of potential policy conflict.

An example case study is presented relating to distributed firewall policy analysis in order to motivate the need to extend the approach, outlined in chapter 4.

Three specific cases are examined: the first case examines the situation where no extra information in the information model is warranted, in this case the existing approach in which ontologies (selection rules) are not needed is completely justified; the second case exposes a difficult policy conflict detection problem that is subtly different from the first case, which warrants adaptation of the existing approach; the third case illustrates an example of a policy conflict involving different security services, illustrating an implicit interdependency among policies that cannot be easily represented in the information model; hence, one or more ontologies are required.

The network topology depicted in figure 5.1 represents a small communications network. The first case depicted is where two disjoint firewalls are used to police traffic originating from the two networks to a common destination network. The firewall policies on Firewall A police the IP traffic originating from network Source A, and similarly for Firewall B. No matter what effect the firewall policies of Firewall A have on its traffic, they will never affect the traffic being policed by Firewall B.

The conflict pattern presented for firewall policy conflict (see chapter 4 figure 4.15 on page 133) requires that the target entities be identical; therefore, as the firewalls are distinct, a conflict is impossible. Consequently, a candidate policy deployed to Firewall A must be analysed against all policies in Firewall B since the process does not distinguish between policy location in the current selection process, thus making the associated policy analysis redundant. In actual fact, all of the policies on Firewall B will return the same answer to the target relationship query. A more informed selection process can aid in reducing the need for redundant policy analysis.

The case depicted in figure 5.2 shows Firewall A and Firewall C along a specific path in the network. A firewall policy still exists on Firewall A specifying that VoIP traffic should be allowed; however, there exists a firewall policy on Firewall C that specifies that VoIP traffic destined for the Destination network should be dropped. Obviously the

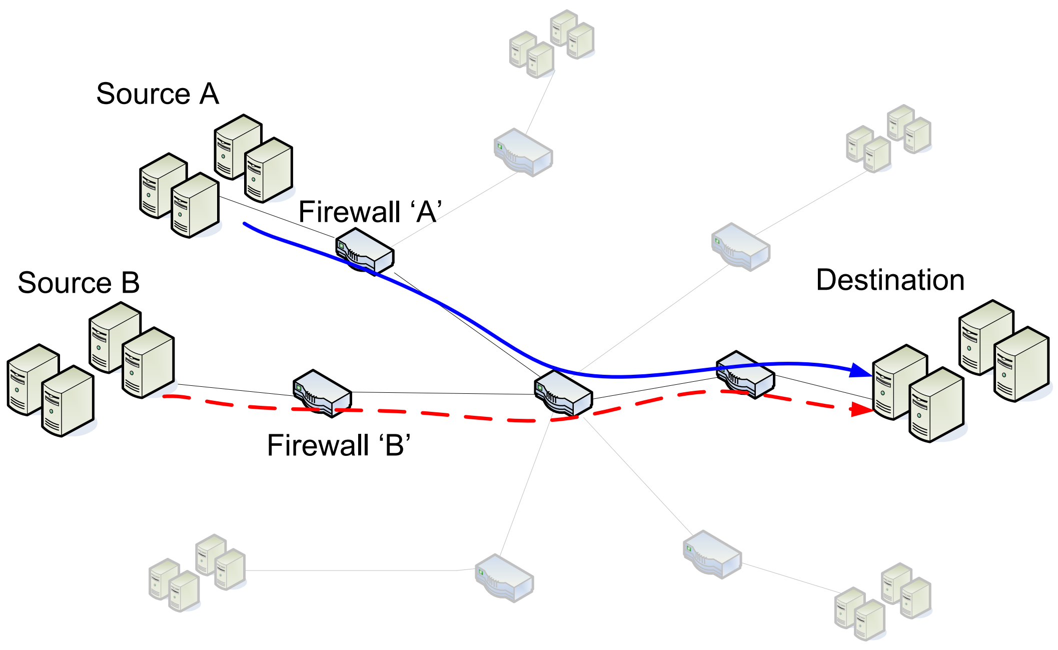


Figure 5.1: Sample network topology for distributed firewall configurations: Case 1, firewalls for different networks.

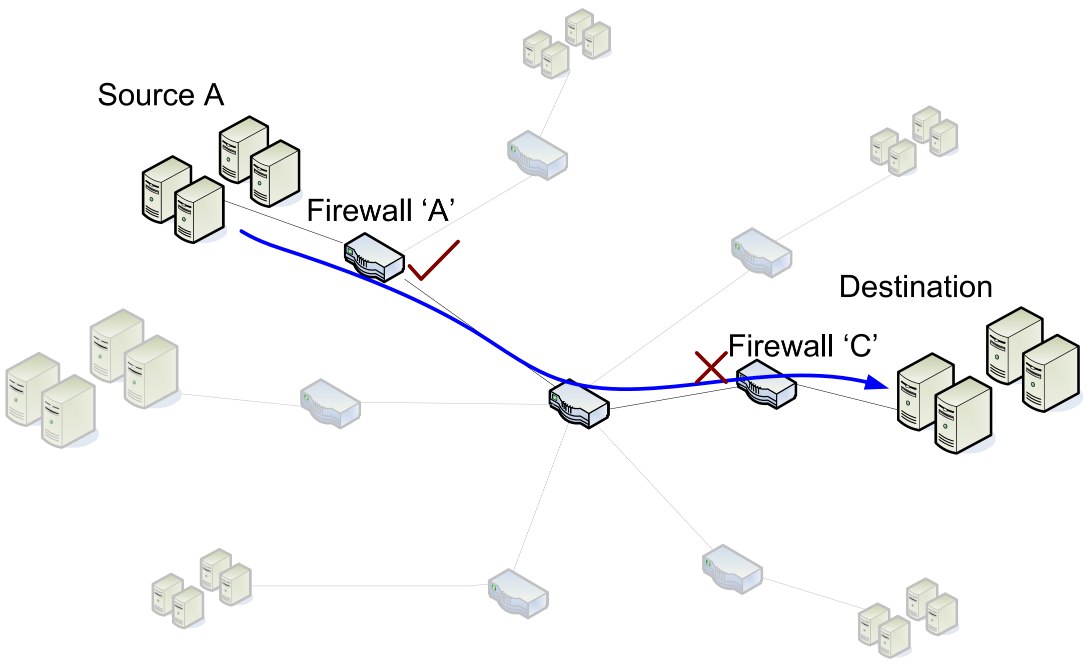


Figure 5.2: Sample network topology for distributed firewall configurations: Case 2, firewalls on the same path.

intended behaviour of Firewall A is that voice communication should be possible between source and destination; however, its policies can only affect the IP traffic on its own router interfaces and not those belonging to other IP interfaces of another device. There has been a lot of documented research describing the issues that may arise in this scenario. The misbehaviour observed is a distributed firewall policy conflict (i.e. physically distributed devices) where downstream devices treat traffic differently from upstream devices. Clearly, there should be an implicit reliance between the two routing devices, since they are policing traffic for a common path in the network.

The source of the problem is that the firewall policies are defined for distributed devices acting autonomously. In fact, the problem can arise for many different policy applications where some implicit coordination of behaviour is required. This example specifically looks at the case for firewall policies, but the problem may also arise for routing policies and network security IPsec policies (Hamed and Al-Shaer, 2006). A flexible and extensible method of incorporating semantic information into the information model is required so that such cases as described here do not cause a problem for policy conflict analysis. At present, it is cumbersome to represent such information in the model using the tools provided by UML. Dedicated associations could be added to resolve the issue; however, this is not scalable since the information model would need to be continuously modified with such cumbersome associations as new service types are modelled, which adversely impacts the effort required.

In the third case, as depicted in figure 5.3, the security services that policy is managing on the associated routers are different, one being a firewall service and the other being an IPsec VPN service. The problem here is that firewall policy authors may not be fully aware of the nature and behaviour of IPsec VPN policies; similarly, IPsec VPN policy authors may not be fully aware of the nature and behaviour of firewall policies. Whilst the two security services are very similar at a high level, their functionality and implementation are different and can cause conflicts in the network if not handled carefully. The solution presented in chapter 4 would not be able to discern the semantics associated with different security policies even if they were on the same router without extra information. The case study presented in section 5.4 at the end of this chapter goes into more details concerning this problem.

The common theme in the second and third cases is the lack of semantic information that cannot be represented and implemented using only information models. The rest of

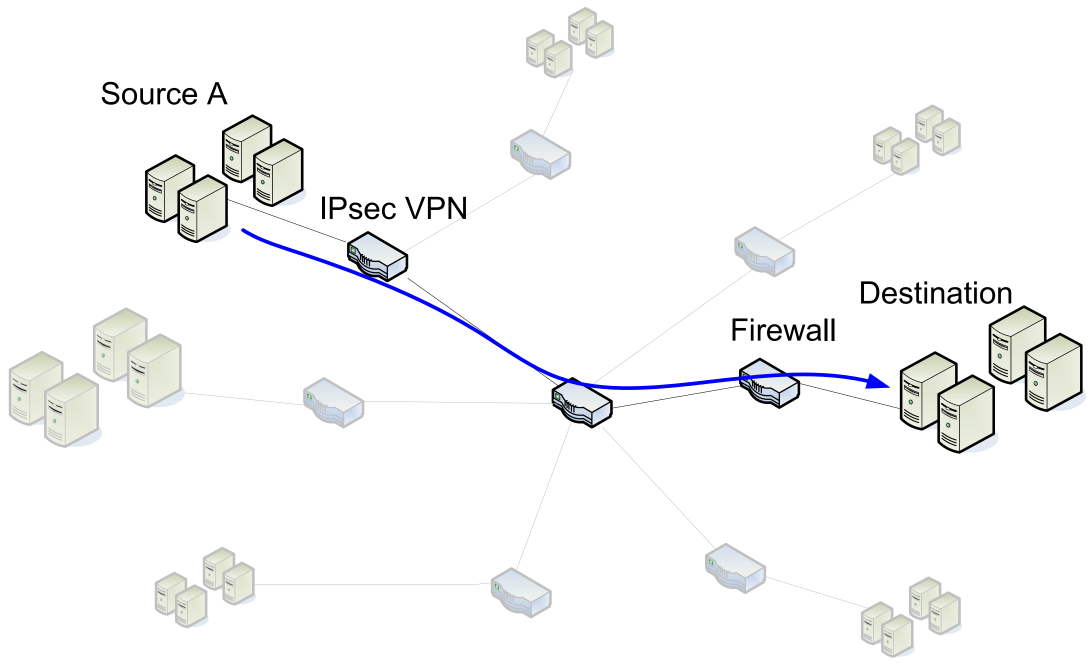


Figure 5.3: Sample network topology for distributed firewall configurations: Case 3, different devices and policy languages.

this chapter shows how ontologies can be used to address this issue. In addition, the use of ontologies enables the overall computational complexity of the solution to be reduced by ensuring that redundant and non-applicable policies are eliminated in a light weight analysis step.

## 5.2 Ontology Enhanced Policy Conflict Analysis Process

An ontology constructed from an information model can represent the structural information of an information model in a manner that can be reasoned with. More detail describing the way ontologies are constructed and used in presented in section 5.3. Given an information model associated ontology, a description of how the policy conflict analysis process is enhanced to use this ontology is presented. As shown in algorithm 4 on page 96, the policy selection algorithm is used to identify those policies that require analysis for conflict. The selected policies are then input into the conflict analysis algorithm. The policy selection algorithm is designed to take advantage of the increased specificity in defining semantic relationships that ontologies bring and the conflict analysis algorithm is enhanced to consider relationships established with the help of the ontology. The enhancements presented here establish the potential impact that incorporating ontologies into the analysis process have. The enhanced policy selection algorithm and the flexible analysis of conflict among policies yields a more effective and more powerful policy conflict analysis process in comparison to existing approaches that do not leverage ontologies in the way presented in this chapter.

### 5.2.1 Enhancing the Policy Selection Algorithm

For large (physically) distributed policy repositories, such as multiple routers in a communications network, or multiple distributed application servers coordinating to fulfil a work flow, a method of retrieving and analysing these distributed sets of policies must be devised. Such a method should also facilitate the detection of a broader set of policy conflicts that exist only when physically distributed policies are considered. The approach presented in chapter 4 cannot consider conflicts based on relationships such as, for example, IPsec security policies compared against firewall filtering policies, because the relationship between the policy types and their actions in particular are not trivial and are difficult to model. In the new approach two enhancements are incorporated: (1) semantically enriched search queries; and (2), more expressive relationship analysis. The approach is depicted in figure 5.4 and outlined below.

1. A policy is initially modified or created by a policy author and is sent to the selection algorithm.
2. The policy “type” (e.g. firewall or IPsec) is used to discover the selection rules that are used to determine the deployed policies that should be retrieved and analysed for conflict. Selection rules are added to the associated ontology and are specific rule patterns that must assert to true for a policy to be considered for conflict analysis.
3. Once the rules are returned they must be iterated over by the policy selection process. Each rule is used to retrieve a list of policies from the policy repository.
4. The ontology can be queried in order to satisfy the requirements of a selection rule.
5. The information model may also be queried in order to satisfy the requirements of a selection rule.
6. Once the policies have all been selected the appropriate policies are returned and forwarded from the policy selection process to the policy conflict analysis process.

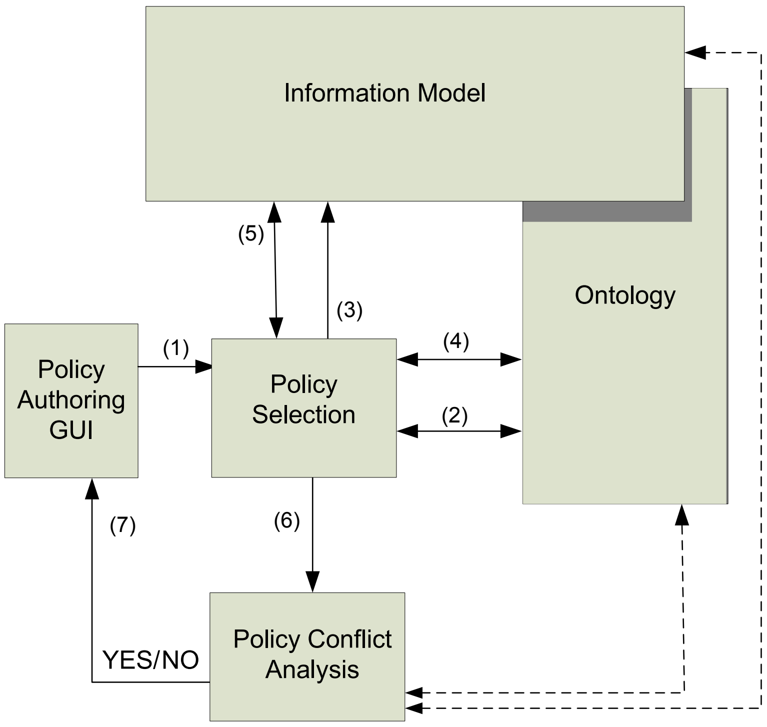


Figure 5.4: Policy selection.

**Algorithm 32** Policy selection algorithm.

```
$\mathbf{selectPolicies} : (\mathit{PolicyRule} \times \mathbb{P}\mathit{PolicyRule}) \rightarrow$
    $(\mathit{Ontology} \times \mathit{InfoModel}) \rightarrow \mathbb{P}\mathit{PolicyRule}$
$\mathbf{selectPolicies}\,(p_{cnd}, ps)\; ot, \mathrm{in} \triangleq$
    $\mathit{let\ rules} = \mathbf{getOntoRules}\,(p_{cnd}, ot)$
    $\mathit{let\ list} = \emptyset$
    $\forall \mathrm{r} \in \mathit{rules} :$
        $\mathit{list} = \mathit{list} \cup \mathbf{evalRule}\,(r, ot, in)$
```

**Algorithm 33** GetOntoRules.

```
$\mathbf{getOntoRules} : (\mathit{PolicyRule} \times \mathit{Ontology}) \rightarrow \mathbb{P}\mathit{SelectRule}$
$\mathbf{getOntoRules}\,(p_{cnd}, ot) \triangleq$
$\mathit{let\ type} = \mathit{objCl}\,(p_{cnd})\; ot$
$\mathit{let\ list} = \emptyset$
$\forall \mathrm{t} \in \mathit{type} :$
    $\mathit{list} = \mathit{list} \cup \mathit{ot}\,(t)$
```

7. If a conflict is detected, then the policy author is notified and any information available is relayed.

The algorithm for policy selection is shown in algorithm 32, which illustrates the use of the information model and associated ontology. The **getOntoRules** function, shown in algorithm 33, retrieves all associated policy selection rules defined in the ontology. This function takes as input the candidate policy and the ontology, and returns a list of selection rules. The selection rules are rules specified in the ontology that indicate which policies need to be retrieved for further conflict analysis. The rules are designed as a pre-analysis step that can be readily evaluated against currently deployed policies. Pre-analysis eliminates the requirement for potentially expensive policy conflict analysis to be preformed on all deployed policies. The rules are stored in the ontology by way of policy type. As shown in algorithm 33 the policy rules are queried for their class name in the information model by using the map function *objCl*, specified in chapter 3 in section 3.2.3 on page 70. As the name of ontology concepts are taken from the information model, the class name of the policy is used to lookup the rules in the ontology. Once retrieved, the rules are iterated over so that they can select those appropriate policies that should be analysed. The *evalRule* function calls the relevent rule engine to evaluate the specific rules discovered. Each deployed policy is linked to the ontology as a instance and is accessible to the rules.

$$\begin{bmatrix} ssb & ssp & seq & scor & 0 & sot \\ tsb & tsp & teq & tcor & 0 & tot \\ esb & esp & eeq & ecor & emux & eot \\ csb & csp & ceq & ccor & cmux & cot \\ asb & asp & aeq & acor & actd & aot \end{bmatrix}$$

Figure 5.5: New conflict signature matrix.

The description of the policy selection rules allow the author to ignore the different types of policies that can be defined in the system. For example, a search rule may require analysis of all security type policies; as the ontology offers a flexible way of performing classification that is not based strictly on inheritence, reasoning may be used to help the search query discover all security related policies even those not originally envisioned by the creator of the search rule. The rules also enable the policy author to not have to be concerned with the size of the communications network as only those policies that are deemed applicable by search rules need be analysed futher for conflict. An example of search rules is presented in section 5.3.3 on page 155.

### 5.2.2 Enhancing the Policy Conflict Analysis Algorithm

The extensions to the conflict analysis algorithm that leverage the ontology are now discussed. As described in chapter 4, there are two phases to the conflict analyse algorithm that are embodied in the *analyseConflict* function depicted in algorithm 5 on page 97. The proposed extensions enable the algorithm to compare different types of policies to establish new relationships in the policy relationship matrix.

The policy relationship matrix is extended to incorporate an extra column to represent ontological relationships discovered among policies; the new matrix layout for policy relationships is depicted in figure 5.5. The new column adds an entry to the end of each row; this new entry represents a potential relationship that can exist between policy components. It is established if said components are related via an ontological relationship. For example, the subject components of the candidate policy and the deployed policy can now be related by some ontological relationship. When the algorithm is establishing relationships among policy components, it will do so by querying the associated ontology. There are advantages to extending the matrix in this fashion:

1. Policy components no longer need to rely on the restricted semantics of the information model to represent all relationships, as more extensive semantic relationships

can now be represented and discovered in the ontology.

2. The algorithm can now benefit from relationships that have been automatically generated due to ontology classification rules or subsumption.

A single new column is used to instruct the conflict analysis algorithm that a relationship of significance was detected between the two policies being analysed. Consequently, there may be many relationships discovered between two policy component types via the ontology, and therefore many reasons why a "1" is inserted into the matrix in the ontology column. By only allowing a single new column, the meaning of the ontological relationship discovered is left up to the policy author who makes the final decision as to whether a conflict has been discovered. The alternative is to introduce a new entry into the matrix representing all the types of relationships that can be established in the ontology between ontology types. The consequence of this approach is that the matrix may become too complex and difficult to understand. However, the final decision as to which approach to take is up to the information model architect who designs the relationship matrix in the information model.

## 5.3 Testbed Implementation

The testbed implementation was previously presented in section 4.2. The testbed is extended to cater for information model associated ontologies and the querying of ontologies within the policy conflict analysis algorithm. Section 5.3.1 discusses the need for ontologies to augment the knowledge embodied in the information model and how they can be harnessed. Section 5.3.2 describes how ontologies were constructed from the DEN-ng information model and section 5.3.3 illustrates a example of an ontology being constructed.

### 5.3.1 Information Model Associated Ontology

Previous approaches to augmenting a UML based information model with ontologies were targeted at extending UML to support the description of ontologies (Cranefield and Purvis, 1998). The method presented here is based on constructing an ontology that represents the structure of entities defined in the information model (Lehtihet et al., 2006). Some classification and constraint rules can then be added to the resulting ontology to aid in the discovery of policy conflicts.

The constructed ontology can be augmented with extensive semantics. Information describing the managed system can be more accurately represented in the ontology and reduces the requirements of representing rich relationships in UML which is cumbersome. The most useful aspect of an associated ontology is the ability to indicate relationships that would be either cumbersome or impossible to represent in the information model. For example, associating a router interface to a customer's billing event may not make sense in the information model, as the relationship is not direct and discovering the link between the entities will involve tracing through the associations of many classes. However, the relationships may be easily defined or inferred in the ontology and may exist to determine that a failed interface can be linked to a breach in service associated to a specific billed customer. Such a relationship can be readily represented in an ontology as a rule.

Distinct IP services defined in the information model, such as an IPsec VPN and a firewall filtering service, may not be directly related (in that there may not be an association that relates them to each other or to another common managed object). The motivating example in section 5.1 illustrates that the two security services are similar to each other and may actually overlap in functionality. This implicit relationship can be made explicit in the ontology by defining a relationship between the two services. This relationship can be later used to aid in the analysis of potential policy conflict.

### 5.3.2 Ontology Construction

The extended implementation of the testbed is depicted in figure 5.6. Pellet 1.5.1 (Pellet, 2008) was used as the inference engine to host the query interface to the ontology. The rules defined over the ontology were described in SWRL (W3C, 2008*b*), where inferencing over the rules was supported by the JESS (Friedman-Hill, 2008) rules bridge to the Protégé ontology editor (Protégé, 2008). The objects instantiated in the by the DSLs can be mirrored in the ontology inference engine by instantiating individuals (i.e. instances) in the ontology.

Most of the information required to construct the ontology is already defined in the information model. This enables the ontology construction process to concentrate on adding semantics to the facts defined in the information model. When this ontology construction process is performed against a specific information model, it can generate a baseline system ontology representing the structure of the information model. The baseline ontology is enhanced with application specific semantics to enhance reasoning over the structure

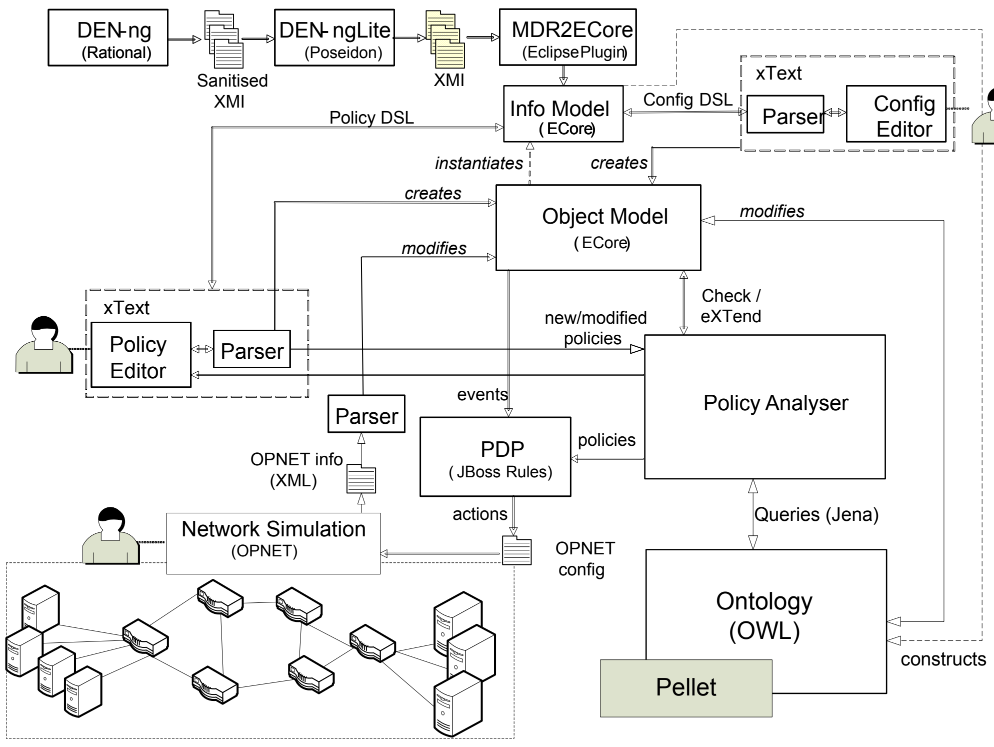


Figure 5.6: Architecture of the testbed incorporating ontologies.

of the information model. The construction of the ontology is based on a tagged portion of the information model. Therefore, multiple ontologies can be generated from a given information model depending on the needs of the management processes and the different types of semantics that each application requires.

The MDD process is extended to incorporate the semi-automated construction of the ontology. The steps of the ontology generation process are depicted in figure 5.7. To review, step 1 of the process requires the developer to identify the pertinent subsets of the information model for the development of the Domain Specific Languages (DSLs); the information model tags are also used for ontology construction. Step 2 generates DSLs that can be used to define object models that represent the structure of the system being managed. These objects are also instantiated in the ontology to aid in reasoning. Step 3 generates an associated policy DSL to aid in the definition of system behaviour. Step 4 automatically constructs an ontology that can be further extended to provide reasoning capabilities for the policies under consideration. This step is also concerned with enhancing the basline ontology with semantic information concerning the applications to be managed. Step 5 generates interfaces for policy analysis to can take advantage of the information model and ontology. These steps have previously been discussed; therefore the new step (Step 4) is now discussed.

**Step 4: Constructing a Baseline Ontology** The tagged portion of the information model is used to construct a baseline ontology. The baseline ontology holds semantic information currently available within the UML information model but in a form that can be reasoned over. This baseline ontology should be further enhanced to incorporate semantic concepts that could not readily be represented within a UML-based information model to allow for automated reasoning. The process followed for transforming a UML model into an Ontology is as follows:

1. Each class in the information model is mapped to a concept in the ontology, with associated inheritance hierarchies.

2. Properties of the classes in the information model are translated into properties in the ontologies.

3. Associations of classes in the information model are translated into properties in the ontologies with specific restrictions. These restrictions are used to enforce the association ends multiplicity at a minimum. Properties in ontologies can relate concepts together, much like the use of associations in UML.

4. All sub classes defined in the information model are translated into disjoint concepts in the ontology. Disjoint concepts in ontologies refers to the fact that an instance in the ontology cannot belong to two or more disjoint concepts.

5. The baseline ontology is enhanced with relationships, constraints and rules that are used to aid in policy selection and policy conflict analysis.

**Step 5: Generated Ontology Interfaces for Policy Analysis** To make the extensive semantic information available to policy based management processes, there must be an interface to query and search the ontology. A query based interface is chosen, where the processes build queries in an ontology query language such as SWRL (W3C, 2008*b*) or SPARQL (W3C, 2008*a*). The use of SWRL as a method of extending the semantics of the ontology is demonstrated in section 5.3.3.

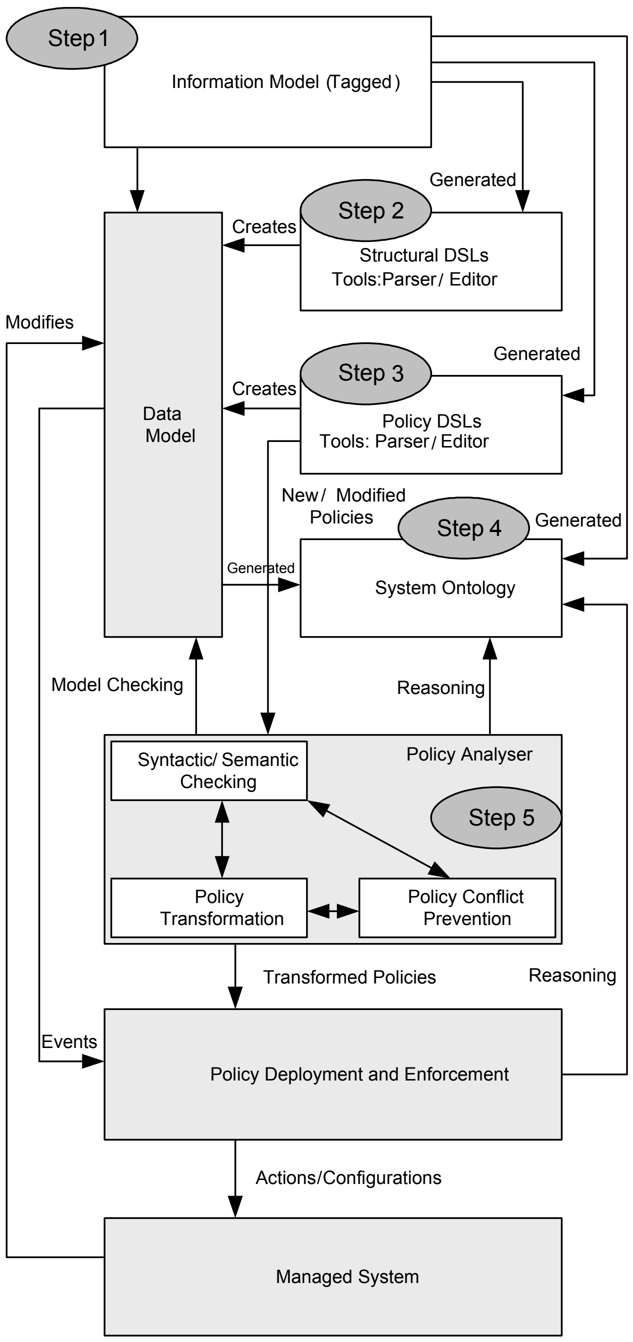


Figure 5.7: Ontology construction steps.

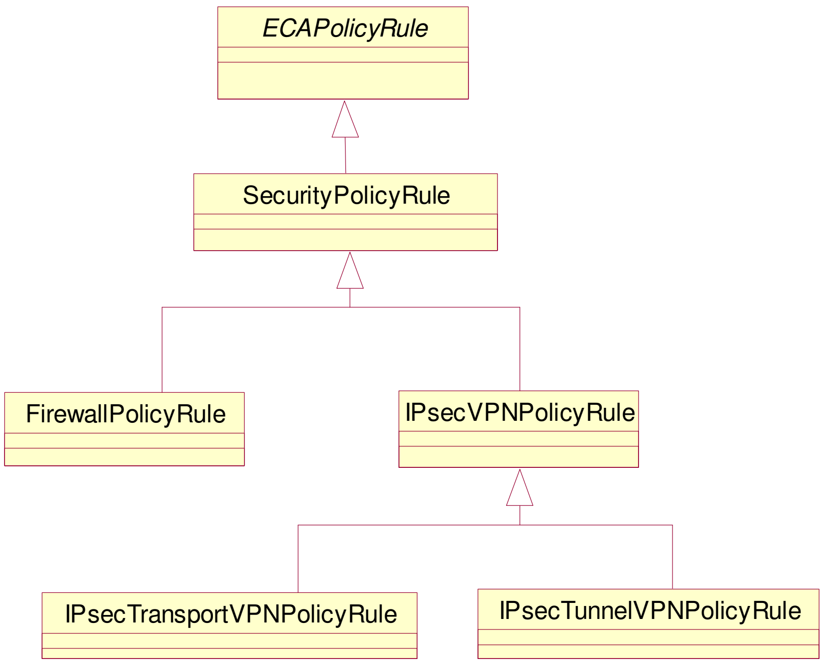


Figure 5.8: Security policies hierarchy information model.

### 5.3.3 Ontology Construction Example

The ontology can be queried for semantic information pertaining to an information model defined entity. A demonstration of how a simple ontology can be constructed and augmented to aid in policy analysis is now presented. Figure 5.8 illustrates a UML diagram of different types of policies, as defined in DEN-ng, and in figure 5.9 associated actions are illustrated.

$$\begin{gathered}
ECAPolicyRule \sqsubseteq SecurityPolicyRule \\
SecurityPolicyRule \sqsubseteq FirewallPolicyRule \sqcup IPsecVPNPolicyRule \\
IPsecVPNPolicyRule \sqsubseteq IPsecTransportVPNPolicyRule \sqcup IPsecTunnelVPNPolicyRule \\
\vdots \\
PolicyAction \sqsubseteq FirewallAction \sqcup IPsecVPNAction \\
FirewallAction \sqsubseteq FirewallDropAction \sqcup FirewallAllowAction
\end{gathered} \tag{5.1}$$

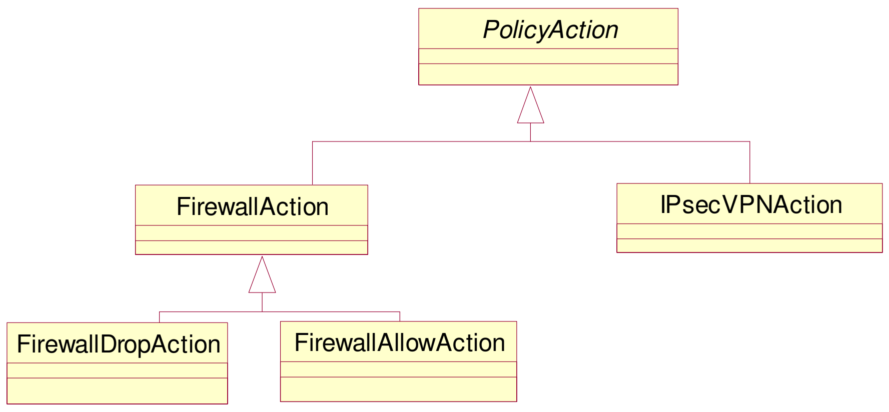


Figure 5.9: Security related policy actions.

$$FirewallPolicyRule \equiv ECAPolicyRule \sqcap \exists hasAction.FirewallAction \tag{5.2}$$

The UML is translated into the ontological concepts as represented by description logic in equation (5.1). The description logic depicted states that there is a concept `ECAPolicyRule` that subsumes the concept of `SecurityPolicyRule`. This type of relationship is similar to a super class. Similarly, `FirewallPolicyRule` and `IPsecVPNPolicyRule` are subsumed by `SecurityPolicy`. An advantage of using ontologies is that a new policy concept can be created that is associated to a the `FirewallAction` class. A rule in the ontology can specify that any `ECAPolicyRule` that has a `FirewallAction` is classed as a `FirewallPolicy`. The policy actions depicted can be used to infer a policy type; any policy that references a `FirewallAction` is a `FirewallPolicy`. This can be carried out by adding the rule as depicted in equation (5.2). If however, the same policy can also be classed as an `IPsecVPNPolicyRule` (because it also contains an `IPsecVPNAction`); then an inconsistency in the model is detected because these two policy types are disjoint.

A disjoint relationship can also be associated between action concepts in the ontology. By explicitly mentioning that certain actions are disjoint, then the information model architecture is explicitly specifying that those actions contradict. The approach taken here is similar to that presented by Chomicki et al. (2003) and by Campbell and Turner (2007), where that supply extra infromation to the conflict analysis process indicating those actions

1. $PolicyRule(?cand) \wedge$
2. $PolicyRule(?dep) \wedge$
3. $hasProperty(?cand, ?t1) \wedge$
4. $hasProperty(?dep, ?t2) \wedge$
5. $hasIP(?cand, ?sip) \wedge$
6. $hasIP(?cand, ?dip) \wedge$
7. $differentFrom(?t1, ?t2) \wedge$
8. $\rightarrow select(?dep)$

Figure 5.10: Template search rule.

that conflict with each other. However, they rely exclusively on this approach to conflict analysis, whereas the approach presented in this chapter explicitly marks contradicting actions only when automatic detection is not feasible. For example, it is not feasible to indicate a-priori all possible combinations of contradicting actions in the ontology. Therefore, a balance is struck between automatically discovering contradicting actions as presented in chapter 4 section 4.1.2.2 and explicitily marking "disjoint" actions in the ontology.

The more interesting rules added are that of search rules defined for various types of policies. Depending on the type of policy, a search rule can be defined that can be matched against different policies and the classification and restriction rules defined in the ontology can then be leveraged to indicate whether a particular types of policy should be retrieved for analysis. For example, a search query can be specified to return all policies that contain an IP address field in their condition component or all policies that are of type `SecurityPolicyRule`. Taking advantage of subsumption in ontologies allows us to specify more expressive search requirements to select a variety of policy types with simple or complex restrictions than is otherwise possible using information models. A template of a search rule is depicted in figure 5.10. The search rule begins by selecting a type of policy (1), depending on the type of policy the candidate policy is, this rule may or may not be applicable. The rule retrieval function must examine this first line to ascertain whether the rule should be retrieved or not. The next part of the rule (2) indicates the type of candidate policies that must be examined. In this case, a default PolicyRule type is declared to search on. The search criteria are then specified in (3) through (7) where attributes and properties of the policies can be checked. The last part of the query (8) is entailed if a policy is found to match the search criteria. The select operation will add the deployed policy to a list for later analysis.

Coming up with a useful search rule inevitably comes down to knowledge of the implicit constraints and relationships in the applications of policy. In designing a search rule using

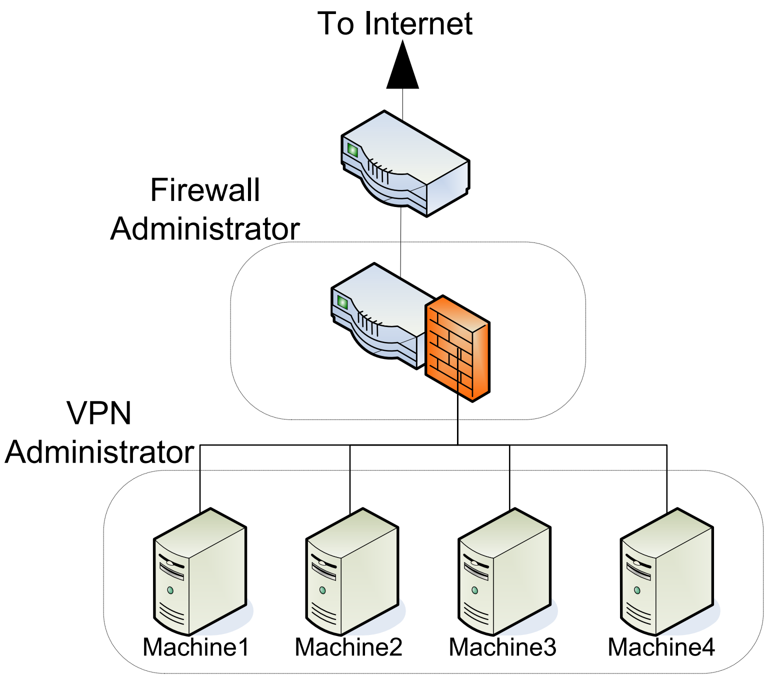


Figure 5.11: Case study administrative domains.

ontologies, the designer needs to figure out the types of distributed policies that need to be returned in order to detect for conflict. The use of an ontology based search rule is a very flexible method of doing this. However, the true power of the solution very much depends on the accuracy in definition of the search rules to retrieve only those relevent policies.

## 5.4 Case Study

The case study illustrates the new capabilities of the updated conflict analysis process and outlines a specific conflict that can arise among security related IP traffic engineering policies across multiple routers. Multiple policy authors are assumed, where each policy author is an expert in a single policy application. The associated management domains are that of firewall filtering policies and IPsec VPN policies. The network topology as depicted in figure 5.11 is owned by a single organisation having at least two management domains, one under control of the firewall administrator and one under control of the VPN administrator. Each administrator is responsible for authoring policies for their respective domains.

Different security services that have an overlap in functionality are chosen. Firewall filtering policies at a basic level instruct a routing device to allow or drop IP packets based

Table 5.1: Firewall policies.

| ID | Type | Policy |
|---|---|---|
| 1 | Firewall | $From[LocalNetwork]To[Internet]$ $Protocol[VideoOnDemand]$ $Action = DROP$ |
| 2 | Firewall | $From[LocalNetwork]To[Internet]$ $Protocol[HTTP]$ $Action = ALLOW$ |
| 3 | Firewall | $From[LocalNetwork]To[Internet]$ $Protocol[IPSEC]$ $Action = ALLOW$ |

on a pattern (e.g. the source address) in the IP packet header. This service can be used to block traffic from specific hosts on the Internet or traffic of a specific type. IPsec is a security protocol that can provide confidentiality, authenticity and integrity to IP traffic (Kent et al., 1998). The service can function in two modes: transport mode or tunnel mode. Transport mode secures the payload of a packet, whereas tunnel mode secures the full IP packet, header and payload and places it inside a new packet. The anomaly that is explored in this case study is that IPsec in tunnel mode masks the original IP header thus making it impossible for egress firewall routers to recognise the true nature of the IP traffic.

The firewall administrator has the responsibility of defining policies that control the type of traffic that can enter the network. This administrator has access to only a single access router as depicted in figure 5.11. A sample of the policies defined by the administrator are depicted in table 5.1. The three policies defined are used to drop video on demand (VoD) traffic, allow HTTP traffic and allow IPsec VPN traffic respectively. These typical firewall policies may have been installed because the network connection to the Internet may not have enough bandwidth to support the strict bandwidth requirements for VoD; therefore, the decision was made to drop all VoD exiting the local network. Allowing HTTP and IPsec traffic is a typical requirement of edge routers, since HTTP is a fundamental Internet service and IPsec traffic supports the transmission of secure IP traffic.

The VPN administrator has the responsibility of defining policy that controls the setting up and securing of VPNs by configuring IPsec services on end-user machines. Therefore, depending on installed policies, traffic may or may not be secured. The policies defined on the end-user machines under control of the VPN administrator are depicted in table 5.2. The policies are distributed across the appropriate end-user machines in the VPN

Table 5.2: VPN policies.

| ID | Type | Policy |
|---|---|---|
| 1 | IPsecTun | $From[Machine1]To[InternetIP1]$ $Action = ESP_{tun}$ |
| 2 | IPsecTun | $From[Machine2]To[InternetIP2]$ $Action = AH_{tun}$ |
| 3 | IPsecTra | $From[Machine3]To[InternetIP2]$ $Action = AH_{Tra}$ |

administrator's domain. The policies describe what actions should be performed on the IP packets as they leave the machine. Policy 1 defines that IP traffic leaving Machine1 whose destination is a machine on the Internet with IP address InternetIP1 must be encapsulated and encrypted in an IPsec tunnel. Policies 2 and 3 are similar, except that policy 2 ensures authentication in tunnel mode and policy 3 ensures authentication in transport mode.

A problem arises when a user on Machine1 sends VoD traffic to InternetIP1. The VPN policy 1 ensures that the packet is encrypted. Note that this also includes the header information, thus masking the fact that the traffic is of type VoD. When the IP traffic passes through the firewall router, it is classified as IPsec traffic and not VoD traffic and is thus allowed; as shown in figure 5.12. This is a conflict, as the passing of VoD traffic through the firewall router goes against the rules of the organisation.

The first step is to generate the associated ontology from the information model of the system and augment it with search rules and classifications to aid in policy conflict analysis. Some important classifications in the ontology are policy types as depicted previously in equation (5.1). A PolicyRule class can be specialised into a SecurityPolicyRule class, which in turn can be either an IPsecVPNPolicyRule or a FirewallPolicyRule. Another addition to the ontology is depicted in equation (5.3) which states that FirewallPolicyRule is disjointFrom IPsecTunPolicyRule, and that any tunnel encryption action class is also *disjointFrom* the FirewallDropAction action class. These classifications are harnessed later in the conflict analysis process.

$$\begin{aligned} &disjointFrom\,(FirewallPolicyRule, IPsecTunPolicyRule)\\ &disjointFrom\,(IPsecTunAction_{ESP}, FirewallDropAction)\\ &disjointFrom\,(IPsecTunAction_{AHESP}, FirewallDropAction)\end{aligned} \tag{5.3}$$

One of the specific search rules that are added to the ontology is outlined in figure 5.13. This search rule is designed to look for FirewallPolicyRule types (1) and deployed policies

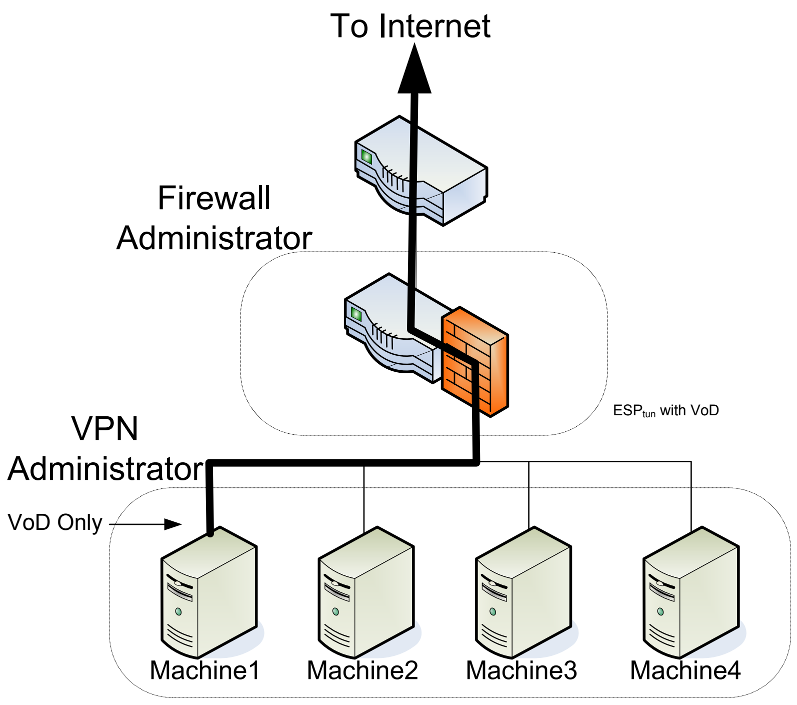
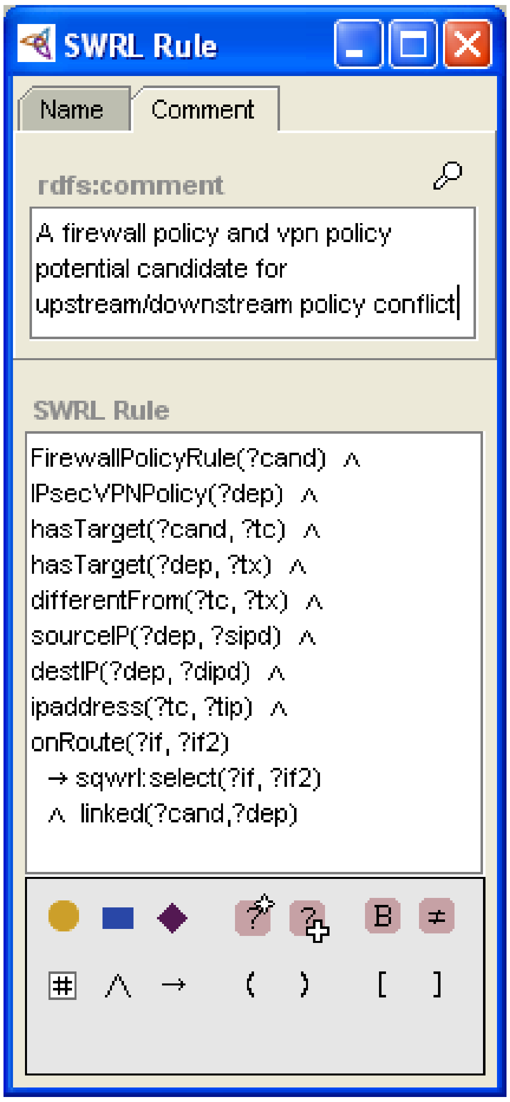


Figure 5.12: VoD traffic passes through firewall.

1. $FirewallPolicyRule\,(?cand) \wedge$
2. $IPSecTunPolicyRule\,(?dep) \wedge$
3. $hasTarget\,(?cand, ?tc) \wedge$
4. $hasTarget\,(?dep, ?tx) \wedge$
5. $differentFrom\,(?tc, ?tx) \wedge$
6. $sourceIP\,(?dep, ?sipd) \wedge$
7. $destIP\,(?dep, ?dipd) \wedge$
8. $ipddress\,(?tc, ?tip) \wedge$
9. $onRouter\,(?sipd, ?dipd, ?tip) \wedge$
10. $\rightarrow select(?dep) \wedge$
11. $linked(?cand, ?dep)$

Figure 5.13: Search rules.



A firewall policy and vpn policy potential candidate for upstream/downstream policy conflict

```
FirewallPolicyRule(?cand) ∧
IPsecVPNPolicy(?dep) ∧
hasTarget(?cand, ?tc) ∧
hasTarget(?dep, ?tx) ∧
differentFrom(?tc, ?tx) ∧
sourceIP(?dep, ?sipd) ∧
destIP(?dep, ?dipd) ∧
ipaddress(?tc, ?tip) ∧
onRoute(?if, ?if2)
  → sqwrl:select(?if, ?if2)
  ∧ linked(?cand,?dep)
```

Figure 5.14: Search rule editor GUI and comment.

of type IPSecTunPolicyRule (2). Lines (3) and (4) retrieve the associated targets of the policies referring to the router interfaces that they are deployed on. Line (5) ensures that the deployed policies retrieved are on different routers; therefore, only deployed policies on other routers can be considered to satisfy this rule. Lines (6) and (7) retrieve the source and destination IP address of the potential deployed policy, and line (8) retrieves the IP address of the target device of the candidate firewall policy. In line (9), the information model is queried to ascertain whether the traffic flow referenced in the source / destination IP addresses on the IPsecTunnelVPNPolicyRule passes through the interface that contains the candidate policy. If line (9) returns true, then there may be a tunneling conflict and more analysis is required. By (10) the policy has been selected and in (11) a special relationship is created to establish that the two policies are linked. This special relationship is instantiated in the ontology and reflects that an ontological relationship has been established between the two policies.

The policy authoring process followed by the firewall administrator is now described. The process begins with the firewall administrator designing a policy, in this case policy 1 as shown in table 5.1, that is to be added to the policy continuum. According to the process as outlined for creating a new policy in the policy continuum, the policy is first analysed to see if any higher-level policies are invalidated by the insertion of the new policy. This process is outlined in algorithm 2 on page 91. The higher-level policies are satisfied and the process continues to analyse for policy conflict between the policies at the same policy continuum level.

All other mentioned policies are assumed to have already been added and analysed for conflict. The next step in the process is policy selection as shown in algorithm 4 on page 96 and algorithm 32 on page 148. The appropriate search rule for FirewallPolicyRule types is retrieved from the ontology (i.e., that depicted in figure 5.13). This rule is retrieved because there is a search rule stored in the ontology that references the class name of the policy currently being analysed. Next, the rule is evaluated and a list of potential policies is enumerated. The policies returned are policies 1 and 2 from table 5.2, which they both satisfy all the criteria outlined in the search rule. Notice that policy 3 from the same figure was not considered because it is of type IPsecTransportVPNPolicyRule. Now that the policies have been selected they must be sent to the conflict analysis algorithm for further processing.

The resulting policy relationship matrix is shown in figure 5.15. The first comparison

$$\begin{bmatrix} ssb(0) & ssp(0) & seq(0) & scor(1) & 0 & sot(0) \\ tsb(0) & tsp(0) & teq(0) & tcor(0) & 0 & tot(1) \\ esb(0) & esp(0) & eeq(1) & ecor(0) & emux(0) & eot(0) \\ csb(0) & csp(1) & ceq(0) & ccor(0) & cmux(0) & cot(0) \\ asb(0) & asp(0) & aeq(0) & acor(0) & actd(0) & aot(1) \end{bmatrix} \circledast \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} = 1$$

Figure 5.15: Phase 2 conflict matrix for the case study.

is between the candidate firewall policy and the deployed policy, VPN policy 1. The subjects of the policies are correlated, or overlap as they as members of a shared domain (i.e. the organisation). According to the policy conflict analysis algorithm, the policies should not be related via target. However, because the search rule returned true for the selected policy, there now exists a relationship between the policy targets (i.e., the previously established "linked" relationship in figure 5.13 ). The existence of this ontological relationship establishes a one in the "tot" (target ontology relationship) field of the policy relationship matrix. The events are equal (i.e., IP packet received). The conditions are related, in that both policies are applied to overlapping IP header information. More specifically, the firewall policy is a superset of the VPN policy via conditions. The action components of the two policies cannot be directly compared because, the candidate policy action is *Drop* and the deployed policy action is $ESP_{TUN}$. Hence, the ontology must be queried. An ontology query reveals that the actions belong to classes *FirewallDropAction* and $IPsecTunAction_{ESP}$ respectively. The ontology specifies that these two classes of actions are *"disjointFrom"* each other; hence, the fact that they conflict with each other can be inferred. This conflict information represents a potential policy conflict between the two policies and is relayed back to the policy author.

## 5.5 Summary and Discussion

This chapter presented an updated policy conflict analysis algorithm with an enhanced policy selection algorithm. The enhancements come from the integration of an information model with an associated ontology that is capable of representing more extensive semantics than that possible in the information model. This combination can be used by algorithms and processes to improve the selection of deployed policies for conflict analysis and reduces the number of policies that need to be retrieved for more thorough conflict analysis. The primary objective of the enhancements was to reduce the complexity associated with analysing large sets of policies. Using an ontology that is associated with an information model, increases the ability of the policy selection algorithm to select only appropriate policies for conflict analysis. The selection algorithm is made more efficient by introducing a pre-analysis step that can determine whether a deployed policy should be further analysed for conflict or disregarded. The selection rules defined within the ontology are the key to making the selection process more efficient, as they are based on policy type, which can be established using subsumption or classification algorithms. Research question 2 is addressed in this chapter, it asks *"What processes and algorithms need to be developed so that existing knowledge bases can be harnessed to aid in policy conflict analysis?"*. In this chapter, ontologies are leveraged and their ability to express more extensive semantic information than UML is used to improve the functionality of the policy conflict analysis process.

Research questions 3 and 4, ask "*How can a policy conflict analysis process be developed that is independent of the nature of the policies?*" and "*How can processes and algorithms developed for policy authoring and policy conflict analysis be developed so that they are made efficient when large numbers of policies are being considered?*". This chapter presents a solution that is relevant to both questions, as the conflict analysis algorithm is now independent of the location of the policies and is more efficient due to the inclusion of the policy selection algorithm.

The case study presented demonstrated that it is possible to select a subset of deployed policies for further analysis, and that a rich set of policy conflicts can be ascertained that are based on the information supplied by the ontology and not solely in the information model. The next chapter focuses on a slightly different problem associated to policy selection, which is the re-use of historical information on previous policy comparisons to reduce computational complexity.

# Chapter 6

# Efficient Policy Selection for Policy Conflict Analysis

In large communication networks, there may be a large number of policies that exist at multiple levels of the policy continuum. This puts a focus on the performance of the policy based management processes. An enhanced selection process based on selection rules was presented in chapter 6 that can reduce the number of policies returned by the selection process. Although this process significantly improves the situation, there is no way of prioritising policies for analysis or for eliminating policies from being analysed should new information become known.

In this chapter, an efficient policy selection process is presented, that maintains a history of previous policy comparisons in a lopsided (or unbalanced) tree data structure that is used to reduce the number of comparisons required in subsequent iterations of the selection process. The ability to incorporate historical information into the selection process stems from the two phase approach taken in the conflict analysis algorithm.

The outline of the chapter is as follows. Section 6.1 describes and discusses the policy selection process and the improvements made. Next, section 6.3 presents the experimental analysis of the implementation of the policy selection process where properties of the process are tested against policy repositories of a specific nature. Experimental results presented here show that significant performance improvements can be made using this approach; however, the degree of this improvement is dependent on the nature of the relationships between deployed policies. A non-trivial case study is presented and discussed in section 6.4. The results of the case study illustrate the potential of the approach defined in the thesis in reducing the computational complexity associated with policy conflict

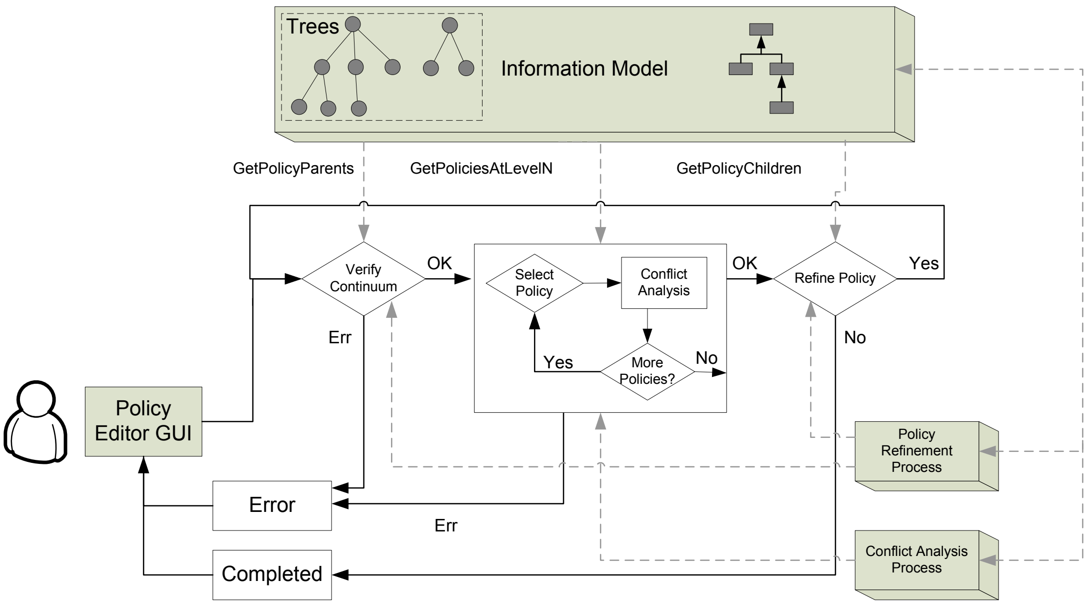


Figure 6.1: Policy authoring process, including policy selection and conflict analysis.

analysis within a network. This process can be used with the selection process outlined in chapter 6. A summary and discussion is presented in section 6.5.

## 6.1 History based Policy Selection using Lopsided Trees

The policy authoring process as presented in chapter 3 controls the selection of which deployed policies a candidate policy should be compared to. This is reflected in the updated policy authoring process depicted in figure 6.1. The figure illustrates the relative sequence of steps the authoring process follows. The policy selection process as presented in chapter 3 is based on a pair-wise comparison of the candidate policy against each deployed policy (at the same policy continuum level) to test for potential conflict. For policy repositories with a high number of policies, this approach may cause potentially significant scalability issues as it is $\mathcal{O}(n)$, $n$ being the number of currently deployed policies. This section discusses how to improve the efficiency of the selection process by reusing the results of previous iterations to reduce the number of comparisons required in future iterations.

The results of previous iterations of the policy conflict analysis algorithm can be reused by maintaining the results in groups of tree data structures. A tree data structure enables simple updating by adding policies based on their relationship (via policy component) to existing deployed policies. Policies that form similar relationships with other policies are grouped together. Therefore, if a candidate policy is deemed similar (by way of a policy component relationship) to a group of existing policies, the policy conflict analysis algorithm can re-use the set of existing relationships with other deployed policies, instead of comparing the policy to all deployed policies. This approach reduces the number of computations required to ascertain if two policies input into the policy conflict analysis algorithm, that are not related via a specific component, potentially conflict with each other. A tree data structure is used becuase is can encode multiple relationships between its nodes in a efficient way, because not every node of the tree needs to be related explicitly. Given any two nodes in the tree, a relationship can be discovered between then by tracing the structure of the tree. The natural structure of the tree is leveraged to improve the efficiency of storing relationships between policies.

$$\begin{aligned} & t \in Tree = Node \rightarrow \mathbb{P}Node \\ & Node \subset Tree \\ & nd \in NodeDetails = Node \rightarrow \mathbb{P}PolicyRule \end{aligned} \tag{6.1}$$

A tree data structure is defined for each row in the policy relationship matrix. That is, there is a separate tree for policy events, conditions, actions, subjects and targets. Indeed a tree should be defined for any new policy relationship row introduced into the relationship matrix. For example, a new tree may be introduced to handle "deontic policy type", since a deontic policy is a special organisation of policy components with specific semantics. A new row in the policy relationship matrix was introduced to represent relationships between deontic policies in chapter 4 section 4.3.3 on page 135. Each tree data structure as defined in equation (6.1) illustrates that a tree is made up of a node map, where each node is mapped to a set of nodes. In the context of this work a node is a subset of a tree, meaning that a node can be viewed as a tree. This is useful when defining recursive search algorithms to add and delete nodes from a tree. A node can be mapped to a set of policies. Each tree data structure aids in establishing relationship patterns among policy components for phase 1 of the algorithm, for each row of the relationship matrix respectively. The relationships ascertained in phase 1 of the algorithm are currently equality, superset, subset and correlation for each component type. A tree can be defined to efficiently store equality, subset and superset. Equality, subset and superset relationships are reflexive; therefore, when a candidate policy is compared to a deployed policy and establishes that a relationship holds for one of their components, the conflict analysis algorithm can re-use the candidate

**Algorithm 34** Add a Policy to the tree.

```
$\mathbf{AddPolicyToTree} : (PolicyRule \rightarrow Tree \times NodeDetails) \rightarrow Tree$
$\mathbf{AddPolicyToTree}\,(p)\,t, nd \overset{\wedge}{=}$
$\forall n \in \left(I \rightarrow \pi^1\right) t :$
    if $(p = nd\,(n)\,[0])$
        $nd = nd \sqcup (n \rightarrow \{p\})$
    elseif $(p \subset nd\,(n)\,[0])$
        $\mathbf{AddPolicyToTree}\,(p)\,n, nd$
    elseif $(p \supset nd\,(n)\,[0])$
        $nd = nd \sqcup (n_e \rightarrow \{p\})$
        $t{=}t \sqcup (n_e \rightarrow \{n\})$
        $t = t \backslash n$
else
    $nd = nd \sqcup (n_e \rightarrow \{p\})$
```

**Algorithm 35** Delete a Policy from the tree.

```
$\mathbf{DeletePolicyFromTree} : (PolicyRule \rightarrow Tree \times NodeDetails) \rightarrow Tree$
$\mathbf{DeletePolicyFromTree}\,(p)\,t \overset{\wedge}{=}$
$\forall n \in \left(I \rightarrow \pi^1\right) t :$
    if $(p = nd\,(n)\,[0])$
        $nd = nd \sqcup (n \rightarrow (nd(n) \backslash p))$
        if $nd\,(n) = \emptyset$ then
            $t = t \backslash n$
    elseif $(p \subset nd\,(n)\,[0])$
        $\mathbf{DeletePolicyFromTree}\,(p)\,n, nd$
```

policy to infer that is it also related to other deployed policies. Correlation relationships can exist between nodes too, but the tree data structure does not store them elegantly. Therefore, it is assumed that only the relationships of equality, subset and superset are stored in the tree data structure.

As more policies are added to the policy trees they begin to form groups around common managed entities in regards to subject and target trees, as well as common event types, conditions and actions that are used for the other trees, respectively. Each tree is kept ordered, with the nodes sorted in order of size (i.e. the number of related policies). Therefore, policies are compared to the most popular policies first. By comparing a candidate policy to the most popular policies first, the probablity of discovering a position for the policy in the tree is increased. This is because the new policy has a higher probability of being related to more general policies then very specific ones. Trees that are maintained unbalanced are commonly referred to as lopsided trees.

Algorithm 34 shows the policy add function and algorithm 35 shows the policy delete function, that modify a policy tree and maintains its structure appropriately. The *AddPolicyToTree* algorithm takes as input an existing tree and a candidate policy and outputs a the new tree. The candidate policy is compared against the top level nodes of the tree, where each node contains a list of policies related via equality. If the candidate policy matches the node via equality, it is merged into the tree at that node and the algorithm finishes. If the candidate policy is a subset of the current node, then the policy is recursively added to the child nodes. If the candidate policy is a superset of the current node, then a new node is created for the candidate policy, and the current node is added to the new node as a branch. If the candidate policy is not related to any existing nodes, then a new node is created in the tree and the candidate policy is added to it. Depending on the type of tree, the equality, subset and superset comparators determine by which component the policies are being related. A counter is incremented for each node as a policy is either added to it or to one of its branches. Therefore node and branch size can be calculated. The *DeletePolicyFromTree* is used to remove a policy from the policy tree. Firstly, the top level branches are searched, if the policy is equal to any of the nodes via the specific policy component type, then the policy is assumed to exist in that node, the policy is removed and appropriate counters are decremented. If the function can establish a subset relationship between the policy and any of the nodes, then the function is called recursively on the appropriate branch node of the tree.

Each tree, per component type, grows as policies are added to it. Subsequently, the least specific policies are placed at the top of the trees, and the most specific are placed at the leaf edges of the trees. A count is maintained for each node that represents the total number of policies associated to that node including the count of its child nodes. The nodes at the same level are sorted so that the node with the highest count is compared against first, as it is associated with the largest number of policies. To illustrate the concept, figure 6.2 depicts a simple insertion operation. The contents of a policy component are represented as a set of letters for the ease of illustration. Therefore, policies that reference the same set of letters can be grouped together. By organising the policies into a tree structure where the top node is a superset of all policies, the policies can be grouped into nodes of the tree.

From this tree, relationships among policies can be derived. For example, policies located in nodes in the same branch that are closer to the root of the tree are a superset

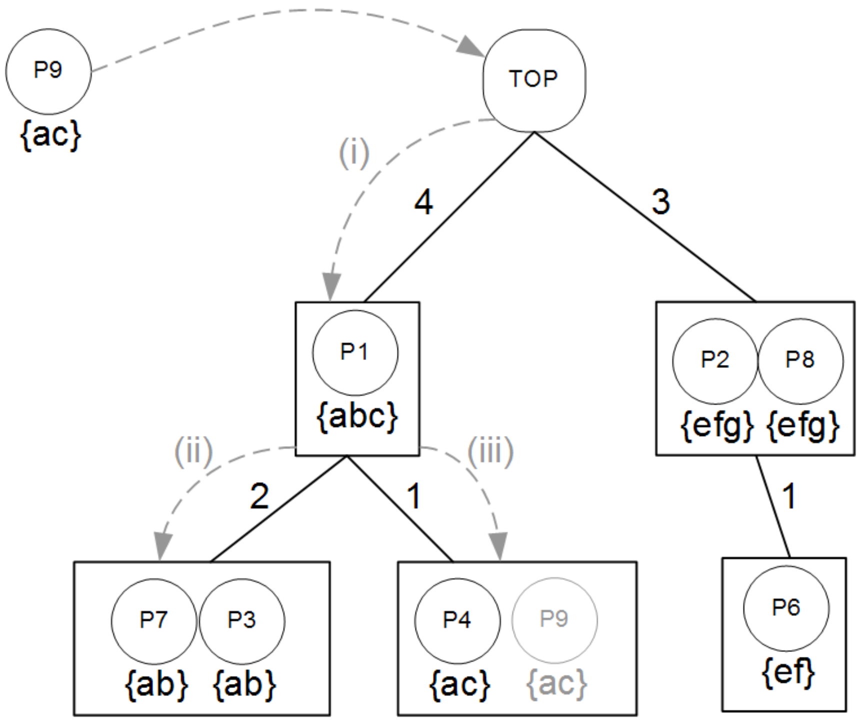


Figure 6.2: Example policy insertion.

of the policies contained in lower nodes, while policies located at the same node position reference equivalent entities. Establishing the order of policies can be carried out per component, based on the specific policy components they may be related differently to other policies. Therefore, a policy "A" may be a subset of another policy "B" via one component type, but via another component type, policy "A" by may be a superset of policy "B".

The steps depicted in figure 6.2 are now described. At step (i), the contents of the candidate policy are compared against the contents of the node with the highest count at the top level (i.e., the node containing P1). The count (which is 4) is simply the number of child nodes plus that particular node, and is used to efficiently select, in order of decreasing magnitude, which nodes of the tree should be compared against the candidate policy. The candidate policy is not related to any other branch of the tree at the top level, since its contents {ac} are not contained in the contents of the other nodes {efg} or {ef}. Indeed, only one comparison is needed, since policy P6 is a subset of the node containing policies P2 and P8, and the contents of P2 and P8 are equal. Since the contents of the candidate policy are not related to the contents of the other node (the one containing policies P2, P8, and P6), this eliminates the need to compare the candidate policy against this node. At step (ii), the candidate policy is compared against the largest node, which contains P7

**Algorithm 36** Updated AnalysePolicyConflict.

```
$\textbf{analysePolicyConflict} : (PolicyRule) \rightarrow$
    $(PolicyContinuum \times Ontology \times InformationModel) \rightarrow \mathbf{B}$
$\textbf{analysePolicyConflict}\,(p_{cnd})\,pc, ot, in \triangleq$
$let\ p_{list} = \textbf{selectPolicies}\left(p_{cnd}, GetPoliciesAtLevelN\left(\pi^1 \circ pc\,(p_{cnd})\right) pc\right) ot, \mathrm{in} \triangleq$
$\forall t \in \textbf{GetPolicyTree}\,(p_{cnd}) :$
  $\textbf{AddPolicyToTree}\,(p_{cnd})\,t, nd$
  $\forall\, cf \in \textbf{RetrieveConflictPattern}\,(p_{cnd}) :$
    $\forall row \in cf :$
     $\forall rel \in row \wedge rel = 1$
     $p_{list} = p_{list} \cap \textbf{getPoliciesViaRelationship}(p_{cnd}, rel)$

$\forall p_{dep} \in p_{list} :$
$FlagForConflict\,(p_{cnd}, p_{dep})$
```

and P3. However, as policies P7 and P3 are equivalent, only a single policy in the group needs to be compared against the candidate policy, thus eliminating another comparison. The candidate policy is not related to that node, so the search continues. The next node compared against, step (iii), is of size one and contains P4. The candidate policy is equivalent to this policy; hence, it is inserted into this node. Policies that exist in the tree within other branches are thus pruned from the search.

## 6.2 Integration with Policy Authoring Process

The use of policy trees to store the relationships between policy components has to be integrated into the policy authoring process. The updated process is is depicted in figure 6.1 on page 167. Algorithm 36 shows the modification to the previous analysePolicyConflict algorithm which was presented in chapter 4 section 4.1.1 in algorithm 4. In that algorithm the selected policies were iterated through the conflict analysis algorithm where the relationships in the policy relationship matrix were established, and these relationships were evaluated against the conflict matrix. In algorithm 36, the process is slightly altered to take advantage of the efficient policy trees. The process is now described as follows:

1. **Create/Modify Policy:** The policy author identifies a policy that must be modified in, or creates a new policy that must be added to, the policy continuum (the candidate policy).

2. **Verification:** The candidate policy is used by the policy verification process to

**Algorithm 37** GetPoliciesViaRelationship.

```
$\mathbf{getPoliciesViaRelationship} : (PolicyRule \times Relationship \rightarrow Tree \times NodeDetails)$
                                      $\rightarrow (\mathbb{P} PolicyRule)$
$\mathbf{getPoliciesViaRelationship}\,(p, r)\, t, nd \triangleq$
$if\, r = Equal\, then$
 $let\, n = nd^{-1}\,(p)$
 $return \quad nd\,(n)$
$elseif\, r = Superset\, then$
 $let\, ns = nd \circ t^{-1}\,(n)$
   $return \quad ns \cup \mathbf{getPolicyViaRelationship}\,(ns[0], r)$
$elseif\, r = Subset\, then$
 $let\, ns = nd \circ t\,(n)$
   $return \quad ns \cup \mathbf{getPolicyViaRelationship}\,(ns[0], r)$
```

investigate if the goals of higher-level policies are affected, if not the process continues.

3. **Conflict Analysis:** The candidate policy now has to be analysed for conflict between the policies at the same policy continuum level.

   (a) **Select Policies:** The selection algorithm as outlined in chapter 5 on page 140 section 5.2.1 is called. This algorithm makes use of the policy type (class) information to retrieve the appropriate selection rules represented in the ontology. These rules are then used to select a subset of deployed policies that should be further analysed for policy conflict with the candidate policy. This step is performed before the policy trees are considered in order to restrict the number of policies that need to be considered for conflict analysis, and to leverage semantic information available in the ontology.

   (b) **Select Policy Trees:** The selected policies have pre-existing relationship information stored in the policy trees that relates those policies to other deployed policies. The policy trees are retrieved based on the type of relationships that can be established in the policy relationship matrix. Essentially, there is a policy tree for each row of the policy relationship matrix. The retrieval of the policy trees is carried out by calling the *GetPolicyTree* function which should access and retrieve the set of trees being maintained.

   (c) **Add Policy to Trees:** Before the conflict matrices are considered, the candidate policy is efficiently added to each tree, per relationship type. This step effectively evaluates how the candidate policy is related to all other deployed policies with a reduced number of policy comparisons. After the policy has been added to all policy trees, it has established relationships between the policies selected from step (a). The conflict matrix is retrieved from the information model and is used to investigate the nature of the relationships just established.

(d) **Considering the Conflict Matrix:** The conflict matrix dictates the relationships that should exist for a potential conflict to occur. Each row of the conflict matrix is associated with a particular policy component type. For each row of the conflict matrix, the associated policy tree is considered. The policy trees are **reduced** to contain only those policies selected from step (a). If a "1" is in the conflict matrix, then the associated policy tree is queried to enumerate all deployed policies that are connected to the candidate policy with respect to the highlighted relationship. For example, if a "1" existed in the subject row indicating that equality contributed to conflict, then all equal policies are retrieved from the subject component tree. This can be done efficiently by examining the policy tree as described in section 6.1. All other trees are sucessively **reduced** to contain only selected policies.

(e) **Iterate for Each Tree:** The process is **repeated** for each row of the conflict matrix. Algorithm 37 is used to search the policy trees and retrieve sets of policies that are associated with the input policy via a specific relationship. The process stops if all conflict matrix rows have been explored, or if no policies remain selected as a result of a reduction step. If there are policies remaining after each row of the conflict matrix is considered, then these policies may potentially conflict with the candidate policy. The candidate policy then should be removed from all affected policy trees. If there are no policies remaining, then there are no potentially policies that can conflict with the candidate policy.

4. **Refinement:** If there are no potential policy conflicts detected, then the candidate policy is refined by the policy refinement process. For each new policy at the lower level of the policy continuum, the process is repeated.

The updated policy authoring process is effected in how it is used to analyse for policy conflict only and the other processes such as verification and refinement are not effected. The use of policy trees introduces an efficient way of eliminating policies from the analysis step depending on the nature of the policy being added to the policy continuum.

## 6.3 Theoretical Analysis

To test the performance of the enhancement to policy selection for the conflict analysis algorithm, a set of experiments were devised that test the sensitivity of the algorithm to the distribution of node sizes, and its sensitivity to the ratio of policies to nodes. Note that actually detecting a policy conflict between two policies is not important here, only the number of comparisons required to ensure all policies were analysed for conflict. In fact, no policy conflicts may exist among the set of policies. To reduce the number of variables measured, the experiments were carried out to only compare the equality of policy subjects between the candidate policy and deployed policies. Therefore, only a single tree was generated for this relationship. The efficiency of the process depends on the number of nodes in the resulting policy tree and also on the distribution of policies across the nodes; therefore, it is independent of how a node is created and added to a policy tree. These assumptions do not affect the accuracy of the results that are outlined throughout the rest of this section, because in an real deployment of policies, only the number of created trees will increase and not the behaviour of the trees.

For the theoretical analysis, sets of policies have to be generated to fit a specific predetermined nature. For example, the nature of the policies generated must follow a specific probability distribution function via policy subject components. Typically, it is the job of the policy author to define policies, but as the theoretical analysis experiments are not concerned with the behaviour that the policies are defining, random policies can be generated. Random policies are generated by populating the policy components of a policy with random entities (i.e., subjects). By controlling the probability distribution functions being used to build the set of policies, the nature of these policy sets can be designed a priori.

The first experiment is related to measuring the sensitivity of the approach to node size (i.e. how many policies are stored in a node). The second experiment is related to measuring the sensitivity of the approach to the number of nodes in the tree.

Table 6.1: Distributions of policies per subject.

| Trial | Distribution |
|---|---|
| 1.1 | 1, 1, 1, 1, 2, 5, 30, 215 |
| 1.2 | 2, 3, 5, 8, 16, 31, 63, 128 |
| 1.3 | 4, 6, 9, 14, 23, 38, 61, 101 |
| 1.4 | 12, 15, 19, 24, 31, 39, 50, 66 |
| 1.5 | 32, 32, 32, 32, 32, 32, 32, 32 |

### 6.3.1 Node Size Distribution Sensitivity

Each level of the policy continuum can have very different characteristics when it comes to how much the existing relationships among deployed policies can be leveraged. An experiment was devised to demonstrate the applicability of the process to sets of policies of varying characteristics. For this experiment, it is assumed that there are a fixed number of policies and a fixed number of nodes within the tree data structure. The distribution of policies among the nodes is varied to simulate the different popularity of some policy subjects over others. A node is established if a policy references a particular subject; the policies are grouped via subject equality. The experiments assumes there are 256 policies in total and 8 different policy subjects; therefore, there is a maximum of 8 nodes in the tree. The distribution of policies per node was generated using an exponential distribution with varying weights to give distributions of different gradients. Five distributions were generated to represent the varying nature of policy repositories. The distribution of policies per subject is outlined in table 6.1.

The experiment consisted of generating a random sequence of policies, based on a predetermined random seed, to be added to the tree. When the final policy (the candidate policy) is added to the tree (i.e., policy numbered 256), the total number of comparisons required to add this last policy to the tree were counted. A pair wise comparison would take 255 comparisons (i.e. the candidate policy is compared against all deployed policies). The test for each set of policies was performed 1000 times where each iteration had a different random seed. The average number of comparisons for the approach along with the reduction in computational complexity, is outlined in table 6.2, the distribution of the number of comparisons is depicted in figure 6.3. The experiment assumes that the addition of new candidate policies follow the same probability distribution as that used to populate the other 255 nodes.

Figure 6.3 shows that the distribution of policies per tree node affects the performance of the policy selection process. The behaviour observed is directly related to the nature of

Table 6.2: Average number of comparisons for policy add.

| Trial | Average No. Comparisons | Percent Reduction |
|---|---|---|
| 1.1 | 1.295 | 99.49% |
| 1.2 | 2.032 | 99.23% |
| 1.3 | 2.456 | 99.04% |
| 1.4 | 3.254 | 98.72% |
| 1.5 | 4.666 | 98.17% |

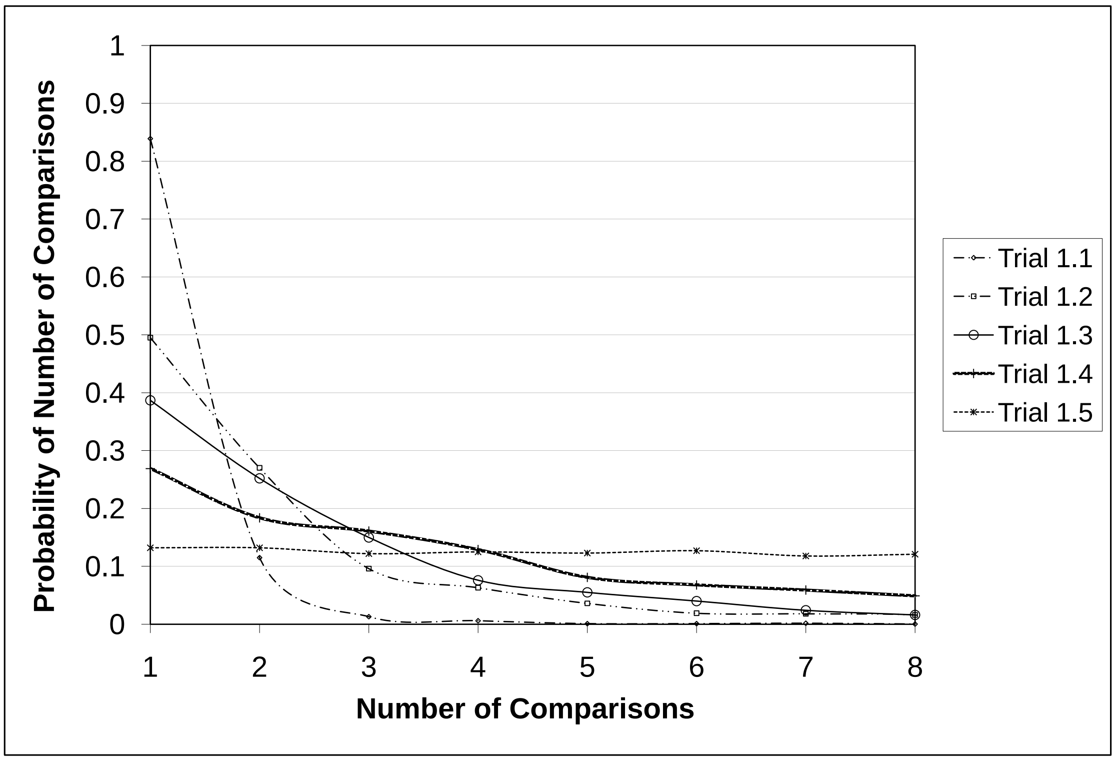


Figure 6.3: Probability of number of comparisons required per trial.

relationships between the sets of policies. This demonstrates why the tree is maintained sorted in order of node count (i.e., node size) after each policy insertion. The probability of the candidate policy being compared against a group that is most related to it is thus increased; this in turn subsequently increases the probability of it being compared against fewer nodes. The distribution with the least amount of variability (trial 1.5) has roughly an equal probability of matching against any node when the final policy is added. However, notice that the number of comparisons is now bounded by the number of established nodes as opposed to the number of policies. The number of comparisons is of the order of the number of nodes in the tree, never rising above eight.

### 6.3.2 Number of Nodes Sensitivity

This experiment demonstrates the sensitivity of the policy selection process to the number of nodes produced by the tree to reflect the equality of policy subjects. For this experiment, it is assumed that there are 100 policies and 100 subjects that these policies can reference. For each trial, the number of subjects that are referenced by the set of policies is varied. The number of subjects referenced determines the number of nodes in the tree, because this experiment only considers equality relationships. There were 13 trials in total, where the distribution of policies per node reduced as the number of nodes increased. The number of nodes per trial are as follows: 1, 2, 3, 4, 8, 15, 20, 30, 40, 50, 70, 90, 99. The number of nodes was chosen to illustrate the performance of the process as the nature of the policy repository varied. Form each trial, policies were distributed uniformly and exponentially among the numbers of nodes that are created. For example, when 20 nodes are created, each node can have 5 policies or the distribution of policies can be exponential in which case one of the nodes will have the majority of policies. Similar to the previous experiment, the policies were put into a random sequence and added to the policy tree. The number of comparisons required for the last policy to be added to the tree is recorded for each trial. Each trial is repeated 1000 times with a different random sequence for each trial. The results for the experiment are presented in table 6.3.

The number of nodes produced per set of 100 policies directly affects the performance of the policy selection process, as shown graphically in figure 6.4. The difference that the distribution of node size has on the process can be observed, as it is designed to leverage any commonalities among policies. Therefore, the performance of the exponential distributions sustain better performance than the uniform distributions. The performance gain can be

Table 6.3: Number of nodes and distribution of policies.

| Trial | Nodes | Average Comparisons (Exp) | Average Comparisons (Uni) | Exp % Reduction | Uni % Reduction |
|---|---|---|---|---|---|
| 2.1 | 1 | 1 | 1 | 99.0% | 99.0% |
| 2.2 | 2 | 1.296 | 2 | 98.7% | 98.0% |
| 2.3 | 3 | 1.466 | 2.623 | 98.5% | 97.4% |
| 2.4 | 4 | 1.586 | 4 | 98.4 | 96.0% |
| 2.5 | 8 | 1.914 | 8 | 98.0 | 93.0% |
| 2.6 | 15 | 3.216 | 15 | 96.8 | 85.0% |
| 2.7 | 20 | 4.407 | 20 | 95.6 | 80% |
| 2.8 | 30 | 9.464 | 25.796 | 90.5 | 74.2% |
| 2.9 | 40 | 15.688 | 33.93 | 84.3 | 66.0% |
| 2.10 | 50 | 22.927 | 50 | 77.1 | 50% |
| 2.11 | 70 | 47.241 | 57.128 | 52.8 | 42.9% |
| 2.12 | 90 | 79.756 | 79.839 | 20.2 | 20.1% |
| 2.13 | 99 | 99 | 99 | 0.0% | 0.0% |

put down to the fact that more policies exist in the earlier nodes of the tree, and thus there is a higher probability that the candidate policy can be matched and inserted.

Most notably, for trial 2.1, one subject was referenced in common to all policies. When the last policy was added, it too referenced the same subject as all currently deployed policies and so it required only a single comparison to establish its association to all deployed policies. This case represents the best performance case for the algorithm, however this performance will not typically be achieved in all cases as it requires that all policies via a specific component type (e.g. subject) be equal.

In contrast, for trial 2.13, each deployed policy referenced a distinct subject. When the last policy was added, it too referenced a distinct subject; therefore, it would have to be compared against each deployed policy to establish a relationship pattern. This case represents the worst case performance of the algorithm, specifically when the number of nodes generated approaches the total number of policies in the repository.

## 6.4 Case Study Analysis

In order to illustrate the impact that the performance enhancements have on a realistic deployment of policies, a case study based on a set of network level firewall filtering policies is described. The reduction in comparisons required to associate a new policy to currently deployed policies is investigated. This is compared to a pair-wise (i.e O(n)) approach, which

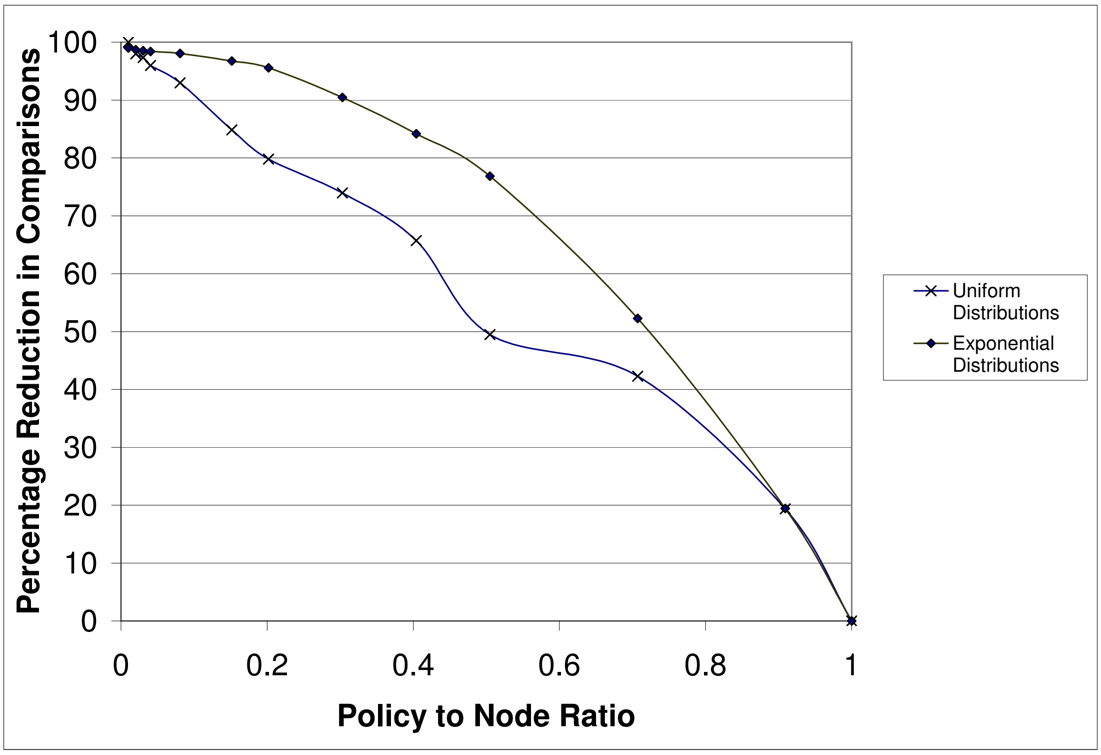


Figure 6.4: Node to policy ration.

is required for inter- and intra-device policy conflict analysis. The process is repeated for each policy to produce a set of data points indicating the average reduction in comparisons for this scenario. For the case study analysis, the policies were very specific and could not simply be randomly generated. In this case each policy was defined at the policy authoring GUI.

The network topology is depicted in figure 6.5. The network is made up of an internet service provider (ISP) that is connected to the wider Internet. This ISP is contracted to serve four corporate customers. It is the task of the ISP to configure edge routers denoted by the letters A, B, C and D, which are located on the customers premises. Therefore, the ISP has total administrative control over the network. Each customer has different connectivity requirements based on their SLA and these requirements are realised by different firewall filtering policies deployed on the respective edge routers. The router denoted by E is an edge router on the core of the ISP network. A complete list of the firewall policies are detailed in the Appendix chapter. There are 57 policies defined in total across all the routers considered.

The business agreements represented by each customer's SLA with the ISP dictate

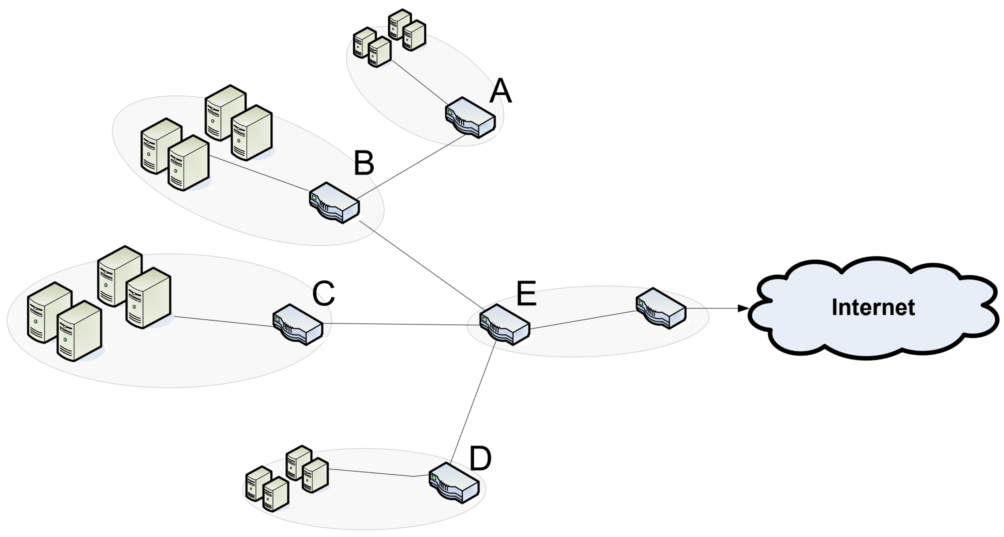


Figure 6.5: Case study network topology.

the behaviour of the firewall configured on the customer premise equipment (CPE). The network administrator uses the policy conflict analysis algorithm as presented in chapter 4 to analyse the policies for a set of conflict types. The information model of the network is extended so that inter-router policy conflicts can be discovered. The policy relationship matrix is designed to consider the following relationships that can be ascertained between two firewall policies for this application:

- Between the *targets* of the policies (i.e., the router interfaces that policies are deployed to)
  - Equality is only considered, as each policy is defined to be deployed on a single router interface
- Between the *conditions* of the policies (i.e., the rules over the IP packet header information)
  - Equality, subset, superset and correlation must all be considered
- Between the *actions* of policies (i.e., the action to be taken on the IP packet)
  - Equality, as there are only two types of actions considered - forward and drop

When a new candidate policy is being added to any of the router interfaces, it must be analysed for conflict against currently deployed policies. The rules and ontologies as defined

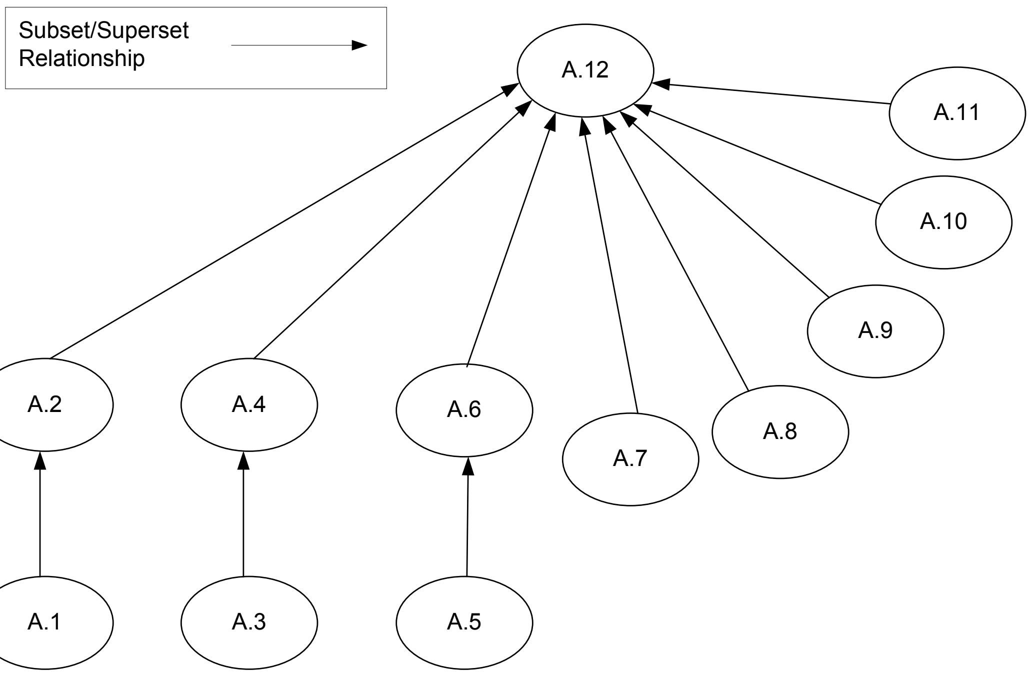


Figure 6.6: Condition tree for firewall policies for ISP A.

in chapter 5, can aid in guiding the policy analysis to first detect for conflicts on the same target device to which the policy is being added and then analyse the policies on other routers. For this application, no matter what order the policies are analysed, all policies still need to be processed through the policy selection algorithm. A candidate policy must carry out six distinct computations with a deployed policy to establish a relationship matrix. One relationship exists between policy targets, four between policy conditions and one between policy actions. There is no need to establish a relationship between the policy subject components, as they are assumed to be identical across all policies for this case study. There is also no need to establish a relationship between the policy event components as there is only a single type of policy event for this case study (i.e., IP packet received). The total number of relationship computations is thus 57 policies times 6, which is 342, assuming that no leveraging of previous relationships is used. Therefore, to reduce the number of comparisons the administrator invokes the use of the tree based approach presented in this chapter to leverage historical information about previous policy conflict analysis comparisons.

The tree formed to represent condition relationships established from all policies associ-

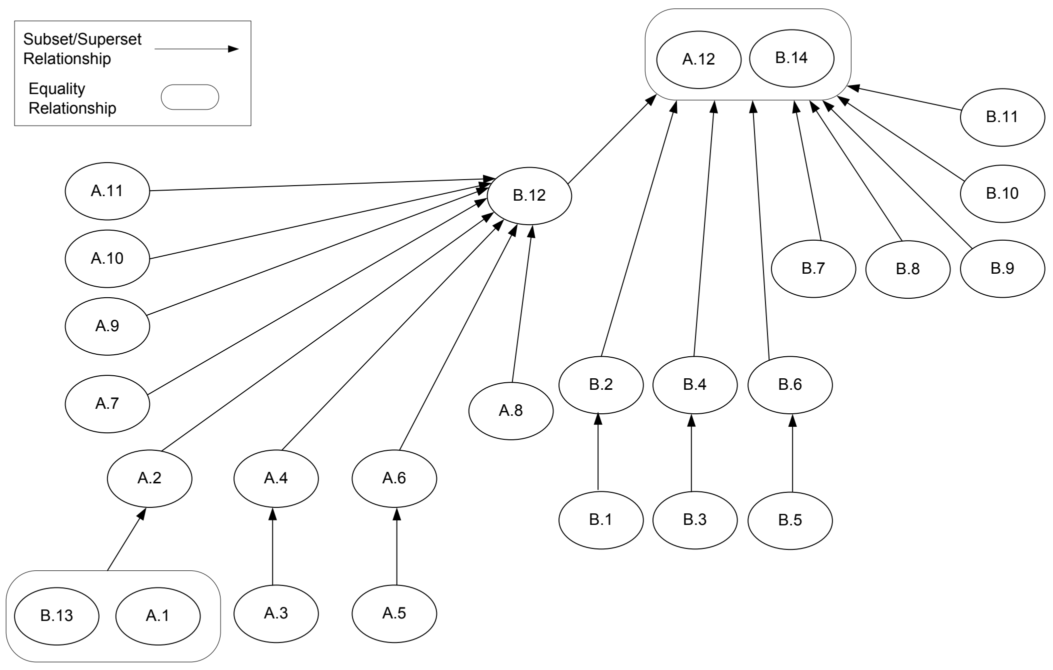


Figure 6.7: Tree created from ISP A and ISP B.

ated to the CPE of ISP A is depicted in figure 6.6. This tree can represent the relationships of subset, superset and equality. There are no equality relationships established for this policy component type, so none are depicted in the figure.

As there is a single common target for all the policies on the CPE for ISP A, there is only a single node produced in the "policy targets" tree, and each policy is associated with this node. The "policy actions" tree contains only two nodes; a node representing those policies that forward IP packets and those that drop IP packets. Similar trees are constructed from the addition of policies to the other CPE routers and the NP (network provider) edge router. Figure 6.7 depicts the combination of trees resulting from the analysis of the condition components of the firewall policies associated to the CPE for ISP B.

After all policies have been analysed and the relevant trees for each component type have been created, the number of comparisons required to add a new candidate policy to one of the routers is examined. Note that as the deployed policies were added, the network administrator followed the policy authoring process, which took advantage of the efficiencies introduced in this chapter. To illustrate the savings in computation in comparison to a fully search, a new candidate policy is analysed; however, the savings in computational complexity would have been observed as each of the existing deployed policies were added.

As calculated earlier, the total number of comparisons required for establishing a relationship between a candidate policy and all existing deployed policies is 342. The network administrator wishes to add a new policy to the CPE of B which will allow VoD traffic on the network; the policy added is:

| | | | | | | |
|---|---|---|---|---|---|---|
| B.x | 64.10.32.0/19 | * | * | 2000 | TCP | Forward |

To establish a relationship concerning the policy target, the policy is inserted into the "policy targets" tree. As there are only five potential policy targets, there are at most five comparisons. In this case, there is actually only one comparison. This is because the trees' nodes are sorted in order of decreasing size, and the node referring to the CPE of ISP B has the most policies. Next, the policy is inserted into the "policy conditions" tree. The first node contains a reference to five equal policies, namely A.12, B.14, C.12, D.12 and E.7. The condition component of the candidate policy needs only to be related to one of these policies condition components in order to establish a relationship with all five policies. The comparison establishes that a subset/superset relationship exists. The next largest child node is then queried, which is the node that contains policy B.12, as it is has a superset relationship with twelve policies. The computation establishes that no condition relationship exists between these policies and so eliminates these thirteen policies from consideration. The next node references B.2, which has a subset/superset relationship with one other policy. The computation establishes that the condition of the candidate policy is not related by a subset/superset relationship to the deployed policy B.2. The candidate policy is correlated with B.2 but not correlated with B.1. The computations continue; subsequently, six further policies are pruned form the tree. The pruned policies are C.1, C.3, C.5, D.1, D.3, D.5, because the analysis process was able to establish that no relationships existed between those policies "superset" members and inferred that no relationship could exist with the "subset" policies.

To establish relationships between the action component of the candidate policy and the deployed policies, the candidate policy is inserted into the "policy actions" tree. As there are only two types of distinct actions, there are only two nodes in the tree. There are 32 policies in the first node of the tree, all these nodes are equal with respect to their actions, since all of these policies enforce the same traffic forwarding action. The

Table 6.4: Reduction in comparisons for firewall case study.

| Relationship Type | Comp. Before | Comp. After | Reduction |
|---|---|---|---|
| Target Equality | 57 | 1 | 98.2% |
| Condition Equality | 57 | 33 | 42.1% |
| Condition Subset | 57 | 33 | 42.1% |
| Condition Superset | 57 | 32 | 48% |
| Condition Correlation | 57 | 32 | 48% |
| Action Equality | 57 | 1 | 98.2% |
| Total No. Comparisons | 342 | 132 | 61.4% |

other node in the tree has 27 policies that are all related, since they all enforce the same traffic dropping action. The candidate policy only needs to perform a single comparison to determine if it is related to all existing policies via the "policy actions" tree.

The total number of comparisons required by the candidate policy concerning establishing condition relationships was 130. The total number of comparisons required was 132, illustrating that establishing relationships between conditions is relatively complex. The total reduction in the number of comparisons required overall was 61.4%, which is a significant reduction in computational complexity.

The next step is to detect policy conflicts among the policies. This step involved tracing through the established trees for each policy component and discovering if a specific set of relationship types exist between the new candidate policy and the deployed policies. The specific set of relationship is provided in the conflict matrix as discussed in chapter 4. To further validate the approach, another experiment was carried out, where each policy played the role of candidate and was compared against all the other deployed policies. In total there were 57 runs. For each run of the experiment a different policy was selected to be the candidate policy. This experiment produced the following results as depicted in figure 6.8. The results illustrate that the average reduction in comparisons is a 59.5 %, with a standard deviation of 8.8%. This is depicted in the figure by the solid vertical line (average) and the dashed lines (standard deviation), and again is a significant reduction in computational complexity.

## 6.5 Summary and Discussion

An efficient policy selection process that uses a lopsided tree to maintain the history of policy relationships for each policy component type was presented. The policy authoring process can leverage the tree data structure to introduce significant performance enhance-

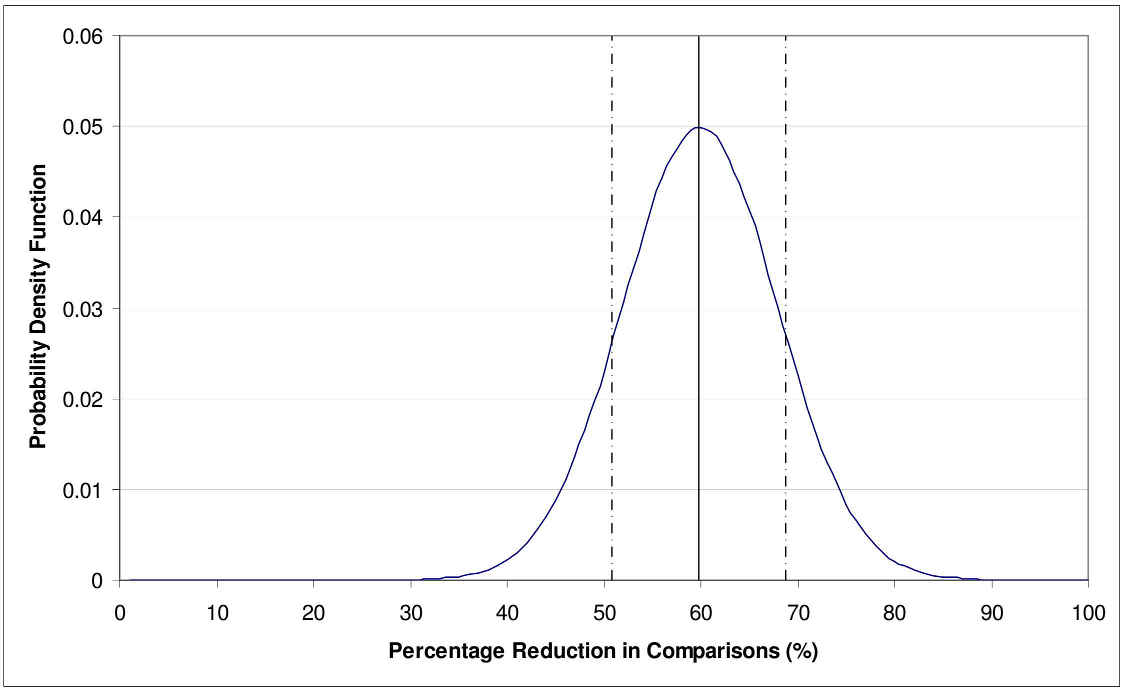


Figure 6.8: Probability density function of percentage reductions in comparisons.

ments by reducing the required number of policies that are need to analysed for potential conflict by the conflict analysis algorithm.

The potential benefits of using the approach in comparison to traditional pair-wise comparison was demonstrated. Sensitivity analysis of node count, and node size distribution yielded a set of results illustrating the potential reduction in computational complexity that this enhancement offers against policy sets with varying characteristics. It is assumed that the range of characteristics of policy sets tested can be aligned with the varying characteristics of different levels of the policy continuum. In this light, the experiments presented are designed to demonstrate the applicability of the approach to a range of policy continuum level types, where specific performance gains for particular policy continuum level types can be expected to be between the extremes presented. The results presented in this chapter are equally applicable to the extended algorithm as presented in chapter 5 when an extra column is used to represent ontological relationships. The integration of the selection process described in chpater 5 and the enhancement presented in this chapter are discussed. Selection rules are used to limit the number of policies that need to be considered for policy conflict and the policy trees are used to efficiently establish the relationships as required in the policy relationship matrix, for equality, subset and superset for each policy component type.

Research question 4, as presented in chapter 1, asks "*How can processes and algorithms developed for policy authoring and policy conflict analysis be developed so that they are made efficient when large numbers of policies are being considered?*". The work presented in this chapter addressed this question by incorporating the use of lopsided trees to aid in the re-use of historical information towards reducing the runtime complexity of the policy selection algorithm.

# Chapter 7

# Conclusion and Future Work

This thesis presented a contribution to the research area of policy based network management, specifically to progress upon the methods currently used for policy authoring and conflict analysis for the policy continuum. Chapter 2 discussed work related to network management approaches, along with the use of information modelling in network and distributed systems management. This motivated the use of the policy continuum in this work. From examining the shortcommings of the current state of the art in policy based management, the requirements of the research carried out in this thesis were then presented. Chapter 3 presented a formal model of the policy continuum as well as an associated policy authoring process, required to definitively outline how policy modifications should be dealt with within the policy continuum. This thesis focused primarily on policy conflict analysis algorithm presented in chapter 4. The conflict analysis algorithm is designed to harness the knowledge embodied in an information model. Harnessing the information model allows for the development of a flexible policy conflict analysis algorithm, that is designed to be independent of the application which is being managed by policies. Chapter 5 describes how the conflict analysis algorithm is enhanced through the use of ontologies to create a policy selection process that can be used to determine a subset of deployed policies that need to be analysed against a new candidate policy for conflict. Chapter 6 presented a further enhancement to the policy selection process making it more efficient. This enhancement makes use of a specialised data structure to leverage historical policy analysis information.

The hypothesis of this thesis is that by leveraging semantic models (information models and ontologies) that have been enriched with application specific information, then policy conflict analysis and policy selection processes can be developed for use specifically with policies deployed as a policy continuum, agnostic to the application of the policies. The hypothesis lead to the development and examination of the new algorithms for policy conflict analysis and policy selection for the policy continuum.

## 7.1 Appraisal of the Thesis

To place this research in context, the requirements for the research are now reviewed as presented in chapter 2. In that chapter, current policy based management processes were discussed. This highlighted a lack in the ability of current policy based management processes to cope with policies defined as a policy continuum. At a minimum, the policy continuum requires the integration of policy verification, policy conflict analysis and policy refinement processes in order to be effectively utilised. The proposed policy authoring process focused on defining a way of integrating the above listed processes. However, the main contribution of the thesis is the policy conflict analysis algorithm and its enhancement to leverage existing semantic models to aid in policy selection and defining criteria for conflict in an application independent way.

The advantages of the presented approach to policy analysis are as follows:

- **Flexiblity**

  The conflict analysis approach presented in this thesis is flexible, as the policy conflict matrix used in this approach is separate from the conflict analysis algorithm. The conflict analysis algorithm can be extended to define the requirements for defining conflicts for different applications being managed with policies. The policy relationship matrix is also defined per application; therefore, relationships that can exist between specific policy component types can be defined per application.

  Information concerning the entities and relationships that exist in a communications network are separated into an information model and associated sets of ontologies. This enables the same conflict analysis algorithm to be re-used to maintain the consistency of policies for arbitrary applications, as application information is defined by the information model and associated ontologies. To support new applications, the information model and ontologies can be readily extended.

- **Efficient**

  The approach presented in this thesis is efficient, as the policy selection algorithm can leverage: 1) ontologies that represent selection rules, which can be used to restrict the number of deployed policies that need to be analysed for conflict; 2) historical information that is stored from the result of the analysis of previously added (or modified) policies is re-used to aid in the elimination of remaining deployed policies that need to be considered for conflict analysis.

The above described advantages have been highlighted as essential from the detailed discussion of current approaches from chapter 2. The presented work meets the requirements set out to improve PBNM solutions designed to manage large scale communications networks. The research questions as presented in chapter 1 are now reviewed.

1. *How can a policy authoring process be defined that incorporates policy conflict analysis, which is specifically targeted at multiple constituencies of policy authors?*

   This question is addressed in chapter 3, where a formal policy authoring process is presented that describes in detail the interaction of multiple policy authors, each authoring policies at different levels of the policy continuum.

2. *What processes and algorithms need to be developed so that existing knowledge bases can be harnessed to aid the policy authoring and policy conflict analysis processes?*

   The algorithms developed in this thesis for policy conflict analysis, and policy selection using selection rules and policy trees, make use of query processes to access the knowledge embodied in the information models and ontologies. The algorithms assume a well defined interface to the semantic models is available and this interface is integrated into the specification of the algorithms. The interface to the information model is presented in chapter 3, and the use of this interface is part of the algorithms and processes specified in chapters 4, 5, and 6.

3. *How can a policy conflict analysis process be developed that is independent of the nature of the policies?*

   This question is addressed in chapter 4, where a policy conflict analysis algorithm is presented that makes extensive use of information models as a method of substantially reducing the reliance on knowing the nature of policies prior to conflict analysis. This is achieved by separating the analysis of policies into two phases. The first phase establishes a policy relationship matrix between two policies, and the second phase compares the relationship matrix to a conflict signature matrix that is defined specifically for an application which describes the criteria for a potential conflict in the context of that application. Multiple conflict signature matrices can be described and a process of extending the information model to realise these conflict signature matrices is presented.

4. *How can processes and algorithms developed for policy authoring and policy conflict analysis be developed so that they are made efficient when large numbers of policies are being considered?*

   This question is addressed in chapters 5 and 6, where policy selection algorithms are integrated with the policy conflict analysis algorithm. In chapter 5, ontologies are used to enhance the semantic information available to the conflict analysis algorithm so that informed decisions can be made when determining those deployed policies that require more detailed conflict analysis. Selection rules are encoded into the ontologies and an algorithm is defined to query and make use of these selection rules in the retrieval of deployed policies for conflict analysis. In chapter 6, the relationships that are established between two policies as part of the conflict analysis algorithm are analysed and maintained in lopsided policy trees. These policy trees are then used to significantly improve the efficiency of the determining those policies that must be analysed for potential policy conflict.

There are shortcomings to the approach taken in this thesis that will be addressed in future research, specifically:

**Complex component relationships.** The relationships that can be ascertained between two policies via a specific policy component type for use in the policy relationship matrix are limited, from the perspective of this thesis. For example, when relating the subject components of two policies, the relationships that can be ascertained are limited to subset, superset, equality and correlation when only the information model is used. There are more complex relationship types that are not considered; for example, subjects can also be related by roles and responsibilities. Ascertaining such relationships would required the information model to model the associated relationships and devise the computations required to ascertain those relationships. There are a number of policy conflict types that require role overlap for a conflict to occur. Note, however, that the approach of this thesis is not compromised, as incorporating this feature results in additional columns to be added to the matrix.

Another limitation of the policy relationship matrix is that the condition components of policies are related in a simplified manner. In reality, relating the condition components of policies is a complex task (Agrawal et al., 2005). Due to the associated complexity, a simple condition model was used that was able to represent ranges of values with a limited number of attributes. Although the simplified condition model can represent temporal conditions and IP address classification conditions, it can not represent more complex conditional expressions, with multiple or nested OR statements. Such condition types are used widely in rule based systems and research on relating rules together in rule-based systems may be leveraged. For example, the rete algorithm developed by (Forgy, 1982) can analyse a set of rules and determine relationships between the rules so that an efficient data structure can be developed to aid in evaluating the rules at runtime.

**Dependency on policy verification and policy refinement.** The authoring process defined in this thesis brings together policy verification, policy conflict analysis and policy refinement. Policy conflict analysis is the primary focus of this thesis; however, there is an implied assumption that policy verification and policy refinement processes are available. In fact, there are policy verification and policy refinement processes published in related works, but they do not follow the model based approach as prescribed in this thesis (Bandara et al., 2004), (Rubio-Loyola et al., 2005, 2006*a*,*b*). The approach presented in this thesis assumes the availability of fully functioning verification and refinement processes; however, in the developed testbed, simplified implementations of policy refinement and policy verification were used.

## 7.2 Future Work

The future research directions are split into three sections. Section 7.2.1 discuss further possible extensions to the policy conflict analysis algorithm. Section 7.2.2 suggests extensions to the policy continuum and associated authoring process, while section 7.2.3 discusses some issues for future related work focusing on other processes for policy based management.

### 7.2.1 Extensions to the Policy Conflict Detection Algorithm

**More expressive policy relationships.** The policy conflict analysis algorithm presented makes some assumptions as to how which types of relationships can be ascertained between two policies. The assumptions are concerned with reducing the complexity associated to ascertain those relationships. Indeed, research presented by Agrawal et al. (2005) concerning policy ratification highlights the associated complexity in establishing relationships between the conditional components of policies. Although their research is not totally concerned with policy conflict analysis, the types of relationships their approach can establish between policies should be investigated for adaptation into the algorithm for conflict analysis presented in this thesis.

Other research, conducted by Lin et al. (2007), discusses the complexity associated to establishing a similarity factor between two policies. They present an algorithm that can readily establish how similar two policies are, by examining the respective policy components of each policy. This similarity measure can then be used to guide which policies require further policy analysis. The similarity measure between policies can be integrated into the policy relationship matrix and can be described by a new row in the matrix. This extension to the policy relationship matrix can enable the definition of conflicts based on a similarity measure, making the definition of conflict more flexible especially when a direct comparison may be difficult if the policies are defined in different policy languages.

**Higher order analysis.** The presented conflict analysis algorithm compares policies two at a time. However, an individual policy may conflict with no other individual policies, but instead conflict with the combined effect of a set of deployed policies. For example, a firewall policy A may be correlated with a preceeding policy B and another preceeding policy C. Conflict analysis between policies A and B may not yield a conflict, and analysis between A and C may also not yield a conflict. However, it is possible that the combined affect of policies B and C may completely cover policy A, this shadowing the policy and causing a conflict. The underlying complexity in this case is that in order to detect such a conflict, the candidate policy must be analysed against each deployed policy as well as analysed against sets of deployed policies. A possible approach may be to extend the dimensions of the policy matrix to consider relationships between sets of deployed policies.

### 7.2.2 Extensions to the Policy Continuum Authoring Process

**Runtime analysis.** The policy analysis processes are currently intended for use in an offline environment, where all conflicts and refinements are processed and analysed before they are fully deployed. An obvious extension to the policy analysis processes is to adapt them for runtime environments. The added flexibility of runtime analysis is that policies can be modified while the system is being managed at runtime.

**Other policy authoring related processes.** This thesis focused specifically on the algorithms for policy conflict analysis and policy selection. Other related processes that need to be further researched to realise policy based management using a policy continuum include policy refinement, policy verification/validation, policy transformation and policy optimisation. These processes are depicted in figure 7.1. Policy refinement and policy verification/validation have been integrated into the policy authoring process in this thesis, but more research is required to realise these processes for use with the formalised policy continuum model. Policy transformation is concerned with translating a policy from one policy language into an equivalent policy or set of policies in another policy language. Realising policy transformation will require the use of ontologies to aid in translating between the terms and concepts used within each policy language. There has been works published in this area, most notably by Kaviani et al. (2007*a*).

### 7.2.3 Related Fields of Research

**Maximising the use of policy relationships.** The policy relationship matrix presented in this thesis provides a lot more information about how two policies are related and can be used for more than just indicating if the policies potentially conflict. For example, the relationship matrix may be used as part of algorithms to ascertain if one policy can be replaced by another or if two policies can be combined. Algorithms can be developed to aid in the analysis of policies for reduncancy and coverage checking which is an emerging area in policy based management as presented by Verlaenen et al. (2007*a*), who propose that the relationships between policies can indicate more than just policy conflicts. They use the results of analysing policies to perform consistency checks to ensure that the policies can operate effectively. In the context of the policy continuum, the type of relationships between policies is more complex as a policy at one level can have multiple links to policies at a higher or lower level. Thus the policy relationship matrix can be used to examine the complexity of the relationships between levels of the policy continuum. Complex relationships between the levels of the policy continuum may make policy authoring difficult and evaluating such complexity can lead to the development of

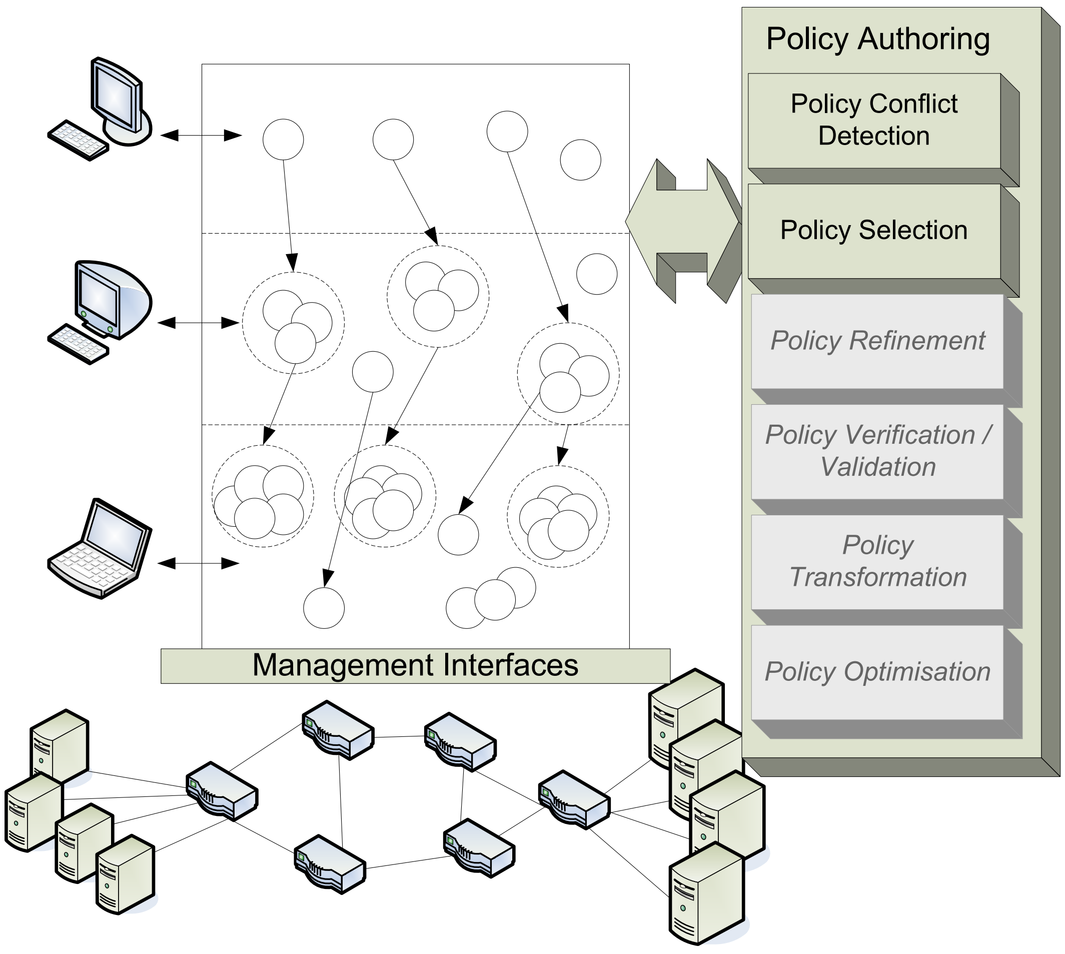


Figure 7.1: Singly administered domain challenges.

algorithms to alleviate this.

**Extending the use of ontologies.** The use of ontologies as presented in this thesis is limited to describing semantic relationships between policies to aid in policy conflict analysis and policy selection algorithms. However, there are additional uses of ontologies that may aid in the delivery of policy based management solutions. Ontologies can aid in the deployment of policies, the refinement of policies and the negotiation of policies. In these approaches, a process known as ontology mapping can be leveraged to integrate between two different ontologies. Augmenting the information model with associated ontologies provides a starting point for the potential semantic integration of disparate information models, which will aid in establishing distributed policy based management solutions. Van der Meer et al. (2005) investigated the use of ontologies to aid in policy mobility, where ontologies aid in using policies between management domains. As presented by Verlaenen et al. (2007*a*) and Lin et al. (2007), ontologies can be used extensively in other policy based management processes, such as similarity matching between policies and the integration of policy languages.

**Use of data-mining techniques.** Data-mining techniques look for hidden relationships within large volumes of data (Han and Kamber, 2001). Policies for large scale communications networks certainly occur in high numbers. Data-mining of large policy repositories may offer more substantial analysis capabilities over and above pair-wise policy analysis. For example, data-mining of policy repositories may yield information pertaining to implicitly chained policies, where one policy may be triggered by another policy in a chain. A potential anomaly is that the cumulative effect of the chained policies may lead to conflict, whereas each separate policy, when analysed, will not reveal a conflict. These so called policy chains are extremely difficult to detect, but data-mining techniques may be part of the solution. Golnabi et al. (2006) investigates the use data mining techniques to analysis sets of firewall rules for anomalies; their results illustrate that data mining is a viable approach to policy conflict analysis of firewall policies. Further research can expand the use of data mining techniques to aid in the analysis of policies in more general policy applications.

**Policy based management for multiple administrative domains.** An interesting challenge, that has yet to be resolved concerning the policy continuum, is the challenges associated to the use of policies to orchestrate management across multiple domain boundaries. This may also be referred to as inter-domain PBM. Currently inter domain management of policies is limited to areas of low level network management such as routing and quality of service management. Negotiation of policies across management domains should be incorporated into related processes for the policy continuum. This is required because policies can be related to each other across administrative domain boundaries and this relationship should be propagated from higher level policies.

When two distinct policy domains, for example two ISPs, need to coordinate the deployment of policies, the negotiation of policy is required. Such a coordination would be required to enable ISPs to managed their SLAs with partner ISPs. One method would be to make agreements offline and establish the policies before they are deployed. An alternative approach is the dynamic creation and negotiation of policies, which reduces management overhead and speeds up the deployment of services. Policy conflict analysis from this perspective is increasingly important, automated negotiation of policies is required. Some important issues are raised, including, how to interpret policies defined in different policy languages (inter domain policy transformation) and with different information models in mind. Another issue is how to detect conflict between distributed policy bases, a part of the problem is the selection of policies across domain boundaries where access may be restricted, or there may be security issues to consider. Future research will need to solve these problems to enable inter-domain policy based management, as depicted in figure 7.2.

**Emerging application areas.** Cognitive radio networks are radio networks that opportunistically manage spectrum in a way that maximises the utilisation of the various spectrum bands available for radio communications (Ghaseml and Sousa, 2008). The requirements of cognitive radio networks demand systems that can dynamically adapt the behaviour of spectrum management. Policy based management may be well suited to this application (Strassner, 2007*c*). The policy continuum can aid in representing the behaviour of the cognitive radio networks and the associated processes can be used to provide effective solutions to the problem.

The goal of autonomic network management as described by Jennings et al. (2007) is to realise a management system that can abstract the complexity associated with low level network management tasks. A potential avenue of research in this regard is that of policy based bio-inspired algorithms as presented by Balasubramaniam et al. (2008), who inves-

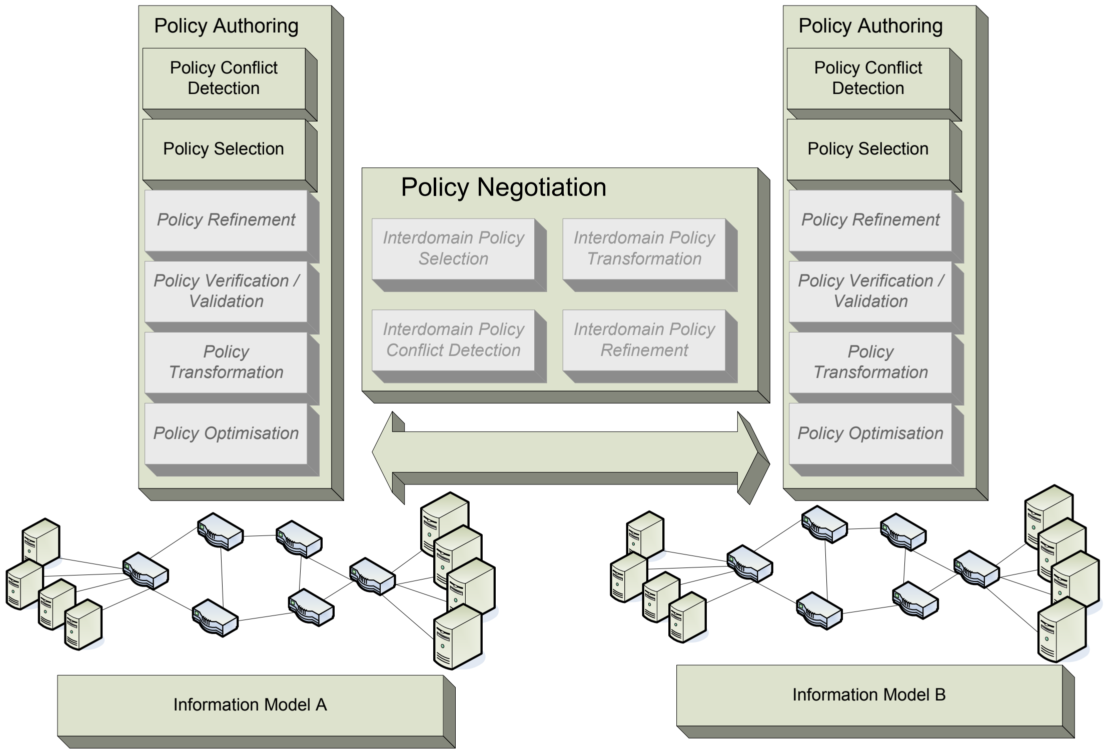

Policy Authoring
Policy Conflict Detection
Policy Selection
Policy Refinement
Policy Verification / Validation
Policy Transformation
Policy Optimisation
Policy Negotiation
Interdomain Policy Selection
Interdomain Policy Transformation
Interdomain Policy Conflict Detection
Interdomain Policy Refinement
Policy Authoring
Policy Conflict Detection
Policy Selection
Policy Refinement
Policy Verification / Validation
Policy Transformation
Policy Optimisation
Information Model A
Information Model B

Figure 7.2: Multi-administered domain challenges.

tigate the use of policy to managed the behaviour of bio-inspired routing and bandwidth allocation algorithms. The analysis of conflicts between the different configurations of these algorithms has yet to be investigated. The DEN-ng policy information model holds significant potential to be used in the above emerging application areas. Research into extending the DEN-ng information model has been carried out by Strassner et al. (2008). In that paper a new policy information model is presented that can be used specifically for use in cognitive radio networks and in autonomic communications networks.

# Bibliography

Abedin, M., Nessa, S., Khan, L. and Thuraisingham, B. (2006), 'Detection and Resolution of Anomalies in Firewall Policy Rules', *in Proc. of the 20th Annual IFIP WG 11.3 Working Conference on Data and Applications Security* , pp. 15–29.

Agrawal, D., Giles, J., Lee, K.-W. and Lobo, J. (2005), 'Policy Ratification', *in Proc. of the Sixth IEEE International Workshop on Policies for Distributed Systems and Networks, (Policy 2005)* , pp. 223–232.

Al-Shaer, E. and Hamed, H. (2003), 'Firewall policy advisor for anomaly detection and rule editing', *in Proc. of the Eight IEEE/IFIP International Symposium on Integrated Network Management, (IM 2003)* , pp. 17–30.

Al-Shaer, E. and Hamed, H. (2004*a*), 'Discovery of Policy anomalies in Distributed Firewalls', *in Proc. of the 23rd Conf. IEEE Communications Soc. (INFOCOM 2004)* , pp. 2605–2616.

Al-Shaer, E. and Hamed, H. (2004*b*), 'Modeling and Management of Firewall Policies', *IEEE Transactions on Network and Service Management* 1(1), pp. 2–10.

Al-Shaer, E., Hamed, H., Boutaba, R. and Hasan, M. (2005), 'Conflict classification and analysis of distributed firewall policies', *IEEE Journal on Selected Areas in Communications, JSAC* 23(10), pp 2069–2084.

AndroMDA (2008), 'Andromda.org - home'. http://www.andromda.org/ available 12/9/2008.

Baader, F., Calvanese, D., McGuinness, D., Nardi, D. and Patel-Schneider, P. (2007), 'The description logic handbook: theory, implementation, and applications', *Cambridge University Press* .

Bajaj et al. (2006), 'Web Service Policy Framework (WS-Policy)' http://specs.xmlsoap.org/ws/2004/09/policy/ws-policy.pdf available 12/09/2008

Balasubramaniam, S., Botvich, D., Jennings, B., Davy, S., Donnelly, W. and Strassner, J. (2008), 'Policy-constrained bio-inspired processes for autonomic route management', *Elsevier Computer Networks Special Issue on Autonomic and Self-organising Systems, (COMNET, 2008)* .

Baliosian, J. and Serrat, J. (2004), 'Finite State Transducers for Policy Evaluation and Conflict Resolution', *in Proc. of the Fifth IEEE International Workshop on Policies for Distributed Systems and Networks, (Policy 2004)* , pp. 250–259.

Bandara, A. K., Lupu, E. C. and Russo, A. (2003), 'Using Event Calculus to formalize policy specification and analysis', *in Proc. of the 4th IEEE Workshop on Policies for Distributed Systems and Networks, (Policy 2004)* , pp. 1–14.

Bandara et al. (2004), 'A Goal-based Approach to Policy Refinement', *in Proc. of the Fifth IEEE International Workshop on Policies for Distributed Systems and Networks, (Policy 2004)* , pp. 229–239.

Bandara et al. (2006*a*), 'Policy Refinement for IP Differentiated Services Quality of Service Management', *IEEE eTransactions on Network and Service Management, TNSM* **3**(2), pp 2–13.

Bandara et al. (2006*b*), 'Using Argumentation Logic for Firewall Policy Specification and Analysis', *in Proc. of the 17th IFIP/IEEE Distributed Systems: Operations and Management, (DSOM, 2006)* , pp. 185–196.

Barrett et al. (2007), 'A Model Based Approach for Policy Tool Generation and Policy Analysis', *in Proc. IEEE Global Information Infrastructure Symposium, (GIIS 2007)* , pp. 99–105.

Bell, D. and LaPadula, L. (1973), 'Secure computer systems: Mathematical foundations', *Technical Report esd-tr-278, MITRE Corporation.*

Bjorner, D. and Jones, C. (1978), *The Vienna Development Method: The Meta-Language*, Springer-Verlag London, UK.

Boyle, J., Cohen, R., Herzog, S., Rajan, R., Sastry, A. and Durham, D. (2000), RFC2748: The COPS (Common Open Policy Service) Protocol, Technical report.

Campbell, G. A. and Turner, K. J. (2007), 'Ontologies to Support Call Control Policies', *in Proc. Third Advanced International Conference on Telecommunications, (AICT 2007)* , pp. 18–28.

Chadha, R. (2006), 'Beyond the Hype: Policies for Military Network Operations', *in Proc. International Conference on Systems and Networks Communication, (ICSN 2006)* , pp. 38–42.

Charalambides, M., Flegkas, P., Pavlou, G., Bandara, A., Lupu, E., Russo, A., Dulay, N., Sloman, M. and Rubio-Loyola, J. (2005), 'Policy conflict analysis for quality of service management', *in Proc. of the Sixth IEEE International Workshop on Policies for Distributed Systems and Networks, (Policy 2005)* , pp. 99–108.

Charalambides, M., Flegkas, P., Pavlou, G., Rubio-Loyola, J., Bandara, A., Lupu, E., Russo, A., Sloman, M. and Dulay, N. (2006), 'Dynamic Policy Analysis and Conflict Resolution for DiffServ Quality of Service Management', *in Proc. of the 10th IEEE/IFIP Network Operations and Management Symposium, (NOMS 2006)* , pp. 294–304.

Chomicki, J., Lobo, J. and Naqvi, S. (2000), 'A Logic Programming Approach to Conflict Resolution in Policy Management', *in Proc. Ninth International Conference of the Principles of Knowledge Representation and Reasoning, (KR 2000)* , pp. 121–132.

Chomicki, J., Lobo, J. and Naqvi, S. (2003), 'Conflict Resolution Using Logic Programming', *IEEE Transactions on Knowledge and Data Engineering, (TKDE 2003)*, pp. 244–249.

Cranefield, S. and Purvis, M. (1998), 'UML as an Ontology Modelling Language', *in Proc. of the 16th International Joint Conference on Artificial Intelligence Workshop on Intelligent Information Integration, (IJCAI 99)* .

Cridlig, V., State, R. and Festor, O. (2007), 'A model for checking consistency in access control policies for network management', *in Proc. of the 10th IFIP/IEEE International Symposium on Integrated Network Management, (IM 2007)* , pp. 11–19.

Damianou, N., Dulay, N., Lupu, E. and Sloman, M. (2001), 'The Ponder Policy Specification Language', *in Proc. of the International Workshop on Policies for Distributed Systems and Networks, (Policy 2001)* , pp. 18–38.

Davy, S., Jennings, B. and Strassner, J. (2007), 'The Policy Continuum - A Formal Model', *in Proc. of the 2nd IEEE International Workshop on Modelling Autonomic Communications Environments, (MACE 2007)* , pp. 65–79.

Davy, S., Jennings, B. and Strassner, J. (2008*a*), 'Application Domain Independent Policy Conflict Analysis Using Information Models', *in Proc. IEEE/IFIP Network Operations and Management Symposium, (NOMS 2008)* , pp. 17–24.

Davy, S., Jennings, B. and Strassner, J. (2008*b*), 'Efficient Policy Conflict Analysis for Autonomic Network Management', *in Proc. 5th IEEE Workshop on Engineering of Autonomic and Autonomous Systems, (EASe 2008)* , pp. 16–24.

Davy, S., Jennings, B. and Strassner, J. (2008*c*), 'The Policy Continuum - Policy Authoring and Conflict Analysis', *in Computer Communications (31), (COMCOM 2008)* pp. 2981–2995.

Davy, S., Jennings, B. and Strassner, J. (2008*d*), 'Using an Information Model and Associated Ontology for Selection of Policies for Conflict Analysis', *in Proc. of the Ninth IEEE International Workshop on Policies for Distributed Systems and Networks, (Policy 2008)* .

de Albuquerque, J., Krumm, H. and de Geus, P. (2005), 'Policy modeling and refinement for network security systems', *in Proc. of the Sixth IEEE International Workshop on Policies for Distributed Systems and Networks, (Policy 2005)* , pp. 24–33.

de Vergara, J., Villagre, V. and Berrocal, J. (2004), 'Applying the Web ontology language to management information definitions', *IEEE Communications Magazine* **42**(7), pp. 68–74.

Distributed Management Task Force, (2008)'Common Information Model Schema version 2.19'

Drools (2008), 'Drools - jboss rules'. http://www.jboss.org/drools/ available 12/09/2008.

Dunlop, N., Indulska, J. and Raymond, K. (2001), 'Dynamic Policy Model for Large Evolving Enterprises', *in Proc. of the Fifth IEEE International Enterprise Distributed Object Computing Conference, (EDOC 2001)* , pp. 193–197.

Dunlop, N., Indulska, J. and Raymond, K. (2002), 'Dynamic conflict detection in policy-based management systems', *in Proc. of the Sixth IEEE International Enterprise Distributed Object Computing Conference, (EDOC 2002)* , pp. 15–26.

Dunlop, N., Indulska, J. and Raymond, K. (2003), 'Methods for conflict resolution in policy-based management systems', *in Proc. of the Seventh IEEE International Enterprise Distributed Object Computing Conference, (EDOC 2003)* , pp. 98–109.

Eclipse (2008*a*), 'The eclipse modelling framework'. http://www.eclipse.org/modeling/emf/ available 12/09/2008.

Eclipse (2008*b*), 'The eclipse platform'. http://www.eclipse.org available 12/09/2008.

Fact++ (2008), 'OWL : Fact++'. http://owl.man.ac.uk/factplusplus/ available 12/09/2008.

Ferraiolo, D., Sandhu, R., Gavrila, S., Kuhn, D. and Chandramouli, R. (2001), 'Proposed NIST standard for role-based access control', *ACM Transactions on Information and System Security, (TISSEC 2001)* **4**(3), pp. 224–274.

Forgy, C. (1982), 'Rete: A fast algorithm for the many pattern/many object pattern match problem', *Artificial Intelligence* **19**(1), pp. 17–37.

FP6 (2008), 'Fp6 - research - european commission'. http://ec.europa.eu/research/fp6/ available 12/09/2008.

Friedman-Hill, E. (2008), 'Jess, the rule engine for the java platform'. http://herzberg.ca.sandia.gov/ available 12/09/2008.

Fu, Z. and Strassner, J. (2006), 'Access Control and Authentication for Converged Wireless Networks', *in Proc. Third Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, (MOBIQUITOUS 2006)* , pp. 1–8.

Fu, Z., Wu, S., Huang, H., Loh, K., Gong, F., Baldine, I. and Xu, C. (2001), 'IPSec/VPN Security Policy: Correctness, Conflict Detection and Resolution', *in Proc. of the IEEE International Workshop on Policies for Distributed Systems and Networks, (Policy 2001)* , pp. 39-56.

Ghaseml, A. and Sousa, E. (2008), 'Spectrum Sensing in Cognitive Radio Networks: Requirements, Challenges and Design Trade-offs', *IEEE Communications Magizine* **26**(4), pp. 32–39.

Godik, S., Moses, T. et al. (2003), 'eXtensible Access Control Markup Language (XACML) Version 1.0', *OASIS Standard, February* http://www.oasis-open.org/committees/download.php/2406/oasis-xacml-1.0.pdf available 12/09/2008 .

Golnabi, K., Min, R., Khan, L. and Al-Shaer, E. (2006), 'Analysis of Firewall Policy Rules Using Data Mining Techniques', *in Proc. of the 10th IEEE/IFIP Network Operations and Management Symposium, (NOMS, 2006)* , pp. 305–315.

Gruber, T. (1993), 'A translation approach to portable ontology specifications', *Knowledge Acquisition* **5**(2), pp. 199–220.

Hamed, H., Al-Shaer, E. and Marrero, W. (2005), 'Modeling and Verification of IPSec and VPN Security Policies', *in Proc. of the 13th IEEE International Conference on Network Protocols, (ICNP, 2005)* , pp. 259–278.

Hamed, H. and Al-Shaer, E. (2006), 'Taxonomy of conflicts in network security policies', *IEEE Communications Magazine* **44**(3), pp. 134–141.

Han and Kamber Jiawei Han and Micheline Kamber (2001), 'Data Mining', *Morgan Kaufann* ISBN 1558604898.

Hari, A., Suri, S. and Parulkar, G. (2000), 'Detecting and resolving packet filter conflicts', *in Proc of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies, (INFOCOM, 2000)*, pp. 1203–1212.

Hegering, H., Abeck, S. and Neumair, B. (1999), *Integrated Management of Networked Systems: Concepts, Architectures, and Their Operational Application*, Morgan Kaufmann.

Howes, T. and Smith, M. (1995), 'The LDAP Application Program Interface', RFC 1823 .

Lougheed et al. (1991), 'A border gateway protocol 3 (bgp-3)' RFC 1267.

Jason et al. (2003), 'IPsec Configuration Policy Information Model', RFC 3585 .

Jajodia, S., Samarati, P., Sapino, M. and Subrahmanian, V. (2001), 'Flexible support for multiple access control policies', *ACM Transactions on Database Systems, (TODS, 2001)* **26**(2), pp. 214–260.

Jennings, B., Van Der Meer, S., Balasubramaniam, S., Botvich, D., óFoghlu, M., Donnelly, W. and Strassner, J. (2007), 'Towards Autonomic Management of Communications Networks', *IEEE Communications Magazine* **45**(10), pp. 112–121.

JTP (2008), 'Java Theorem Prover: An object oriented modular reasoning system'. http://www.ksl.stanford.edu/software/JTP/ available 12/09/2008.

Jude, M. (2001), 'Policy-based Management: Beyond The Hype', *Business Communication Review* , pp. 52–56.

Kagal, L., Finin, T. and Joshi, A. (2003), 'A Policy Language for a Pervasive Computing Environment', *in Proc. of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks, (Policy 2003)* , pp. 63–74.

Kaviani et al. (2007*a*), 'Exchanging Policies between Web Service Entities using Rule Languages', *in Proc. IEEE Congress in Services, (Services, 2007)* , pp. 57–64.

Kaviani et al. (2007*b*), 'Web Rule Languages to Carry Policies', *in Proc. of the Eighth IEEE International Workshop on Policies for Distributed Systems and Networks, (Policy 2007)* , pp. 188–192.

Kempter, B. and Danciu, V. (2005), 'Generic Policy Conflict Handling Using a priori Models', *in Proc. of the 16th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, (DSOM, 2005)* , pp. 84–96.

Kent et al. (1998), 'Security architecture for the internet protocol', *RFC 2401* .

Kikuchi, S., Tsuchiya, S., Adachi, M. and Katsuyama, T. (2007), 'Policy Verification and Validation Framework Based on Model Checking Approach', *in Proc. of the Fourth International Conference on Autonomic Computing, (ICAC 2007)*, pp. 1-10 .

Lehtihet, E., Strassner, J., Agoulmine, N. and Fóghlú, M.Ó. (2006), 'Ontology-Based Knowledge Representation for Self-Governing Systems', *in Proc. of the 17th IFIP/IEEE Distributed Systems: Operations and Management, (DSOM 2006)* pp. 74–85.

Lin, C., Xue, C. and Zhitang, L. (2006), 'Analysis And Classification of IPSec Security Policy Conflicts', *in Proc. Japan-China Joint Workshop on Frontier of Computer Science and Technology, (FCST 2006)* , pp. 83–88.

Lin, D., Rao, P., Bertino, E. and Lobo, J. (2007), 'An approach to evaluate policy similarity', *in Proc. of the 12th ACM symposium on Access control models and technologies, (SACMAT 2007)* , pp. 1–10.

Lobo, J., Bhatia, R. and Naqvi, S. (1999), 'A policy description language', *in Proc. of the Sixteenth National Conference on Artificial Intelligence, (AAAI 1999)* , pp. 291–298.

Lopez de Vergara, J., Villagra, V., Asensio, J. and Berrocal, J. (2003), 'Ontologies: giving semantics to network management models', *IEEE Network* **17**(3), pp. 15–21.

Luck, I., Vogel, S. and Krumm, H. (2002), 'Model-based configuration of VPNs', *in Proc. of the IEEE/IFIP Network Operations and Management Symposium, (NOMS, 2002)* , pp. 589–602.

Lupu, E. and Sloman, M. (1997), 'Conflict Analysis for Management Policies', *in Proc. of the 5th International Symposium on Integrated Network Management, (IM, 1997)* , pp. 430–443.

Lupu, E. and Sloman, M. (1999), 'Conflicts in Policy-Based Distributed Systems Management', *IEEE Transactions on Software Engineering* **25**(6), pp. 852–869.

Moffett, J. and Sloman, M. (1991), 'The Representation of Policies as System Objects', *in Proc. of the Conference on Organisational Computer Systems, (COCS, 1991)* **12**(2-3), pp. 171–184.

Moffett, J. D. and Sloman, M. S. (1993), 'Policy Hierarchies for Distributed Systems Management', *IEEE Journal on Selected Areas in Communications, JSAC* 11,**(9)**, pp. 1404–1414.

Moffett, J. and Sloman, M. (1994), 'Policy Conflict Analysis in Distributed System Management', *Journal of Organizational Computing* **4**(1), pp. 1–22.

oAW (2008), 'openarchitectureware'. www.openarchitectureware.org avaliable 12/09/2008.

OMG (2008), ‘Uml 2.0 ocl specification’. www.omg.org/docs/ptc/03-10-14.pdf available 12/09/2008.

openArchitectureWare (2008), ‘Openarchitectureware 4.1 check validation language’. http://mail.eclipse.org/gmt/oaw/doc/4.1/ available 12/09/2008.

OPNET (2008), ‘Opnet modeller $^{TM}$’. http://www.opnet.com available 12/09/2008.

Pellet (2008), ‘Pellet: The open source owl dl reasoner’. http://pellet.owldl.com available 12/09/2008.

Poseidon (2008), ‘Poseidon for uml’. http://www.gentleware.com/ available 12/09/2008.

Protégé (2008), ‘The protégé ontology editor and knowledge acquisition system’. http://protege.stanford.edu available 12/09/2008.

Racer (2008), ‘Racer System’. http://www.racer-systems.com available 12/09/2008.

Rational Rose (2003), ‘Rose enterprise edition’. http://www-306.ibm.com/software/rational/ available 12/09/2008.

REWERSE (2008), ‘Rewerse - reasoning on the web’. http://rewerse.net/ available 12/09/2008.

Rubio-Loyola, J., Serrat, J., Charalambides, M., Flegkas, P., Pavlou, G. and Lafuente, A. (2005), ‘Using linear temporal model checking for goal-oriented policy refinement frameworks’, *in Proc. of the Sixth IEEE International Workshop on Policies for Distributed Systems and Networks, (Policy 2005)* , pp. 181–190.

Rubio-Loyola, J., Serrat, J., Charalambides, M., Flegkas, P. and Pavlou, G. (2006*a*), ‘A Functional Solution for Goal-oriented Policy Refinement’, *in Proc. of the Seventh IEEE International Workshop on Policies for Distributed Systems and Networks, (Policy 2006)* , pp. 133–144.

Rubio-Loyola, J., Serrat, J., Charalambides, M., Flegkas, P. and Pavlou, G. (2006*b*), ‘A Methodological Approach towards the Refinement Problem in Policy-based Management Systems’, *IEEE Communications Magazine* 44(10) , pp. 60–68.

Russo, A., Miller, R., Nuseibeh, B. and Kramer, J. (2002), ‘An Abductive Approach for Analysing Event-Based Requirements Specifications’, *in Proc. of the 18th International Conference on Logic Programming, (ICLP, 2002)* , pp. 22–37.

Shankar, C., Ranganathan, A. and Campbell, R. (2005*a*), 'An ECA-P Policy-based Framework for Managing Ubiquitous Computing Environments', *in Proc. of the 2nd Annual International Conference on Mobile and Ubiquitous Systems: Networks and Services, (MOBIQUITOUS, 1999)* , pp. 33–44.

Shanahan, M. (1999)'The Event Calculus Explained', Lecture Notes in Computer Science, LNCS 1600

Snir et al. (2003), 'Policy Quality of Service (QoS) Information Model', *RFC 3644* .

Strassner, J. (1999), *Directory Enabled Networks*, Macmillan Technical Publishing. ISBN 1-57870-140-6.

Strassner, J. (2003), *Policy-Based Network Management*, Morgan Kaufmann. ISBN 1-55860-859-1.

Strassner, J., Agoulmine, A. and Lehtihet, E. (2006), 'FOCALE–A Novel Autonomic Networking Architecture', *in Proc. Latin American Autonomic Computing Symposium, (LAACS, 2006)* .

Strassner, J., Neuman de Souza, J., Raymer, D., Samudrala, S., Davy, S. and Barrett, K. (2007), 'The Design of a New Policy Model to Support Ontology-Driven Reasoning for Autonomic Networking', *in in Proc. 5th Latin American Network Operations and Management Symposium, (LANOMS, 2007)* , pp. 114–125.

Strassner, J. (2007*a*), 'Policy Management and Autonomic Mechanisms for Seamless Mobility Networks and Applications', *in Proc. IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, (WoWMoM, 2007)* , pp. 1–6.

Strassner, J. (2007*b*), 'The Role of Autonomic Networking in Cognitive Networks', Cognitive Networks : Towards Self-Aware Networks, Chapter 2, John Wiley and Sons, Ltd., ISNB: 9780470061961 pp. 23–52.

Strassner, J. (2007*c*), 'Using Autonomic Principles to Manage Converged Services in Next Generation Networks', *in Proc. of the Fourth IEEE International Workshop on Engineering of Autonomic and Autonomous Systems, (EASe, 2007)* , pp. 176–186.

Strassner, J. and Ó Fóghlú, M. and Donnelly, W. and Agoulmine, N. (2007), 'Beyond the Knowledge Plane: An Inference Plane to Support the Next Generation Internet', *in Proc.*

*First International of the Global Information Infrastructure Symposium, (GIIS, 2007)* , pp. 112–119.

Strassner, J. and Menich, B. and Johnson, W. (2007), 'Providing Seamless Mobility in Wireless Networks Using Autonomic Mechanisms', *in Proc. First International Conference on Autonomous Infrastructure, Management and Security, (AIMS, 2007)* pp. 121–130.

Strassner, J. and Raymer, D. and Samudrala, S. (2007), 'Providing Seamless Mobility Using the FOCALE Autonomic Architecture', *in Proc. 7th International Next Generation Teletraffic and Wired/Wireless Advanced Networking Conference, (NEW2AN, 2007)* , pp. 330–341.

Strassner et al. (2008), 'The design of a novel context-aware policy model to support machine-based learning and reasoning', *accepted for publication in the Special Issue of LANOMS 2007 in the Journal of Network and Systems Management, JNSM* .

Tequila (2002), 'TEQUILA : Traffic Engineering for Quality of Service in the Internet, at Large Scale'. http://www.ist-tequila.org/ available 12/09/2008.

TMForum (2003), 'An Overview of the NGOSS Arhcitecture', *TeleManagement Forum Whitepaper* http://www.tmforum.org/browse.aspx?catID=2009linkID=29204docID=2535 available 12/09/2008

TMForum (2004), 'Enhanced Telecom Operations Map (eTOM) The Business Process Framework GB921' http://www.tmforum.org/page35597.aspx available 12/09/2008.

TMForum (2008), 'Shared Information / Data (SID) Model', *GB922 Addendum, TeleManagement Forum). http://www.tmforum.org available 12/09/2008*

Uschold, M. and Gruninger, M. (1996), 'Ontologies: principles, methods and applications', *Knowledge Engineering Review* **11**(2), 93–136.

Uszok, A., Bradshaw, J. M., Jeffers, R., Suri, N., Hayes, P., Breedy, M. R., Bunch, L., Johnson, M., Kulkarni, S. and Lot, J. (2003), 'KAoS policy and domain services: Toward a description-logic approach to policy representation, deconfliction, and enforcement', *in Proc. of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks, (Policy 2003)* , pp. 93–96.

Van der Meer, S., Jennings, B., O Sullivan, D., Lewis, D. and Agoulmine, N. (2005), 'Ontology based policy mobility for pervasive computing', *in Proc. of 12th Workshop of the HP Open University Association, HP-OVUA* , pp. 211–224.

Verlaenen, K., De Win, B. and Joosen, W. (2007), 'Policy Analysis Using a Hybrid Semantic Reasoning Engine', *in Proc. Eight IEEE International Workshop on Policies for Distributed Systems and Networks, (Policy 2007)* , pp. 193–200.

Verlaenen, K., Win, B. D. and Joosen, W. (2007), 'Towards simplified specification of policies in different domains', *in Proc. 10th IFIP/IEEE International Symposium on Integrated Network Management, (IM, 2007)* , pp. 20–29.

Vidales, P., Baliosian, J., Serrat, J., Mapp, G., Stajano, F. and Hopper, A. (2005), 'Autonomic System for Mobility Support in 4G Networks', *IEEE Journal on Selected Areas in Communications, JSAC* **23**(12), 2288–2304.

W3C (2001), 'Daml+oil (march 2001) reference description'. http://www.w3.org/TR/daml+oil-reference available 12/09/2008.

W3C (2004), 'Web ontology language OWL / W3C semantic web activity'. http://www.w3.org/2004/OWL/ available 12/09/2008.

W3C (2008*a*), 'Sparql query language for RDF'. http://www.w3.org/TR/rdf-sparql-query/ available 12/09/2008.

W3C (2008*b*), 'SWRL: A semantic web rule language combining OWL and RuleML'. http://www.w3.org/Submission/SWRL/ available 12/09/2008.

Westerinen et al. (2001), 'Terminology for Policy-Based Management', *RFC 3198* .

Wies, R. (1995), 'Using a Classification of Management Policies for Policy Specification and Policy Transformation', *in Proc. of the 4th Integrated Network Management, (IM, 1995)* **4**, pp. 44–56.

Wijesekera, D. and Jajodia, S. (2003), 'A Propositional Policy Algebra for Access Control', *ACM Transactions on Information and System Security, (TISSEC, 2003)* **6**(2), pp. 286–325.

Wong, A., Ray, P., Parameswaran, N. and Strassner, J. (2005), 'Ontology mapping for the interoperability problem in network management', *IEEE Journal on Selected Areas in Communications, JSAC* **23**(10), pp. 2058–2068.

Yang, Y., Martel, C. U. and Wu, S. F. (2007), 'CLID: A General Approach to validate security policies in a dynamic network', *in Proc. of the 10th IFIP/IEEE International Symposium on Integrated Network Management, (IM 2007)*, pp. 1–10.

Yang, Y., Martel, C. and Wu, S. (2004), 'On building the minimum number of tunnels: an ordered-split approach to manage IPSec/VPN policies', *in Proc. of the IEEE/IFIP Network Operations and Management Symposium, (NOMS 2004)*, pp. 277–290.

Yavatkar et al. (2000), 'A Framework for Policy-based Admission Control', *RFC 2753* .

Zhang, C. C., Winslett, M. and Gunter, C. A. (2007), 'On the Safety and Efficiency of Firewall Policy Deployment', *in Proc. IEEE Symposium on Security and Privacy, (SP 2007)* pp. 33–50.

# List of Acronyms

| | |
|---|---|
| ABox | Assertion Box |
| AF | Assured Forwarding |
| AH | Authentication Header |
| AS | Autonomous System |
| BGP | Border Gateway Protocol |
| CIM | Common Information Model |
| COPS | Common Open Policy Service |
| CPE | Customer Premise Equipment |
| DHCP | Dynamic Host Configuration Protocol |
| DL | Description Logic |
| DMTF | Distributed Management Task Force |
| DSCP | Differentiated Services Code Point |
| DSL | Domain Specific Language |
| EC | Event Calculus |
| ECA | Event Condition Action |
| ESP | Encapsulated Security Payload |
| eTom | enhanced Telecom Operations Map |
| FCAPS | Fault, Configuration, Accounting, Performance, Security |

| | |
|---|---|
| FOCALE | Foundation, Observation, Comparison, Action, Learning and rEasoning |
| FTP | File Transfer Protocol |
| GUI | Graphical User Interface |
| HTTP | Hyper Text Transfer Protocol |
| IETF | Internet Engineering Task Force |
| IP | Internet Protocol |
| ISO | International Standards Organisation |
| ISP | Internet Service Provider |
| ITU-T | International Telecommunication Union, Telecommunication Standardisation Sector |
| LDAP | Lightweight Directory Access Protocol |
| LTL | Linear Temporal Logic |
| Mbps | Mega Bits Per Second |
| MDA | Model-Driven Architecture |
| MDD | Model-Driven Development |
| MIB | Management Information Base |
| MOF | Meta Object Facility |
| NGOSS | New Generation Operations Systems and Software |
| NIST | National Institute of Standards and Technology |
| NP | Network Provider |
| oAW | Open Architecture Ware |
| OCL | Object Constraint Language |
| OMG | Object Management Group |
| OO | Object Oriented |

| | |
|---|---|
| OSI | Open Systems Interconnection |
| OSS | Operation Support System |
| OWL | Web Ontology Language |
| PBNM | Policy-Based Network Management |
| PCIM | Policy Core Information Model |
| PDL | Policy Description Language |
| PDP | Policy Decision Point |
| PEP | Policy Enforcement Point |
| PVP | Policy Verification Point |
| PXP | Policy eXecution Point |
| QoS | Quality of Service |
| RBAC | Role Based Access Control |
| RDF | Resource Description Framework |
| RDFS | Resource Description Framework Schema |
| RFC | Request For Comment |
| RFC | Request For Comments |
| SID | Shared Information and Data Model |
| SMI | Structure of Management Information |
| SNMP | Simple Network Management Protocol |
| STA | Subject-Target-Action |
| SWRL | Semantic Web Rule Language |
| TBox | Terminology Box |
| TMF | TMForum |
| TMN | Telecommunications Management Network |

| | |
|---|---|
| UML | Unified Modelling Language |
| VDM | Vienna Development Methodology |
| VoD | Video On Demand |
| VoIP | Voice Over Internet Protocol |
| VPN | Virtual Private Network |
| W3C | World Wide Web Consortium |
| WS-Policy | Web Service Policy |
| XACML | eXtensible Access Control Markup Language |
| XML | Extensible Markup Language |

# Appendix: Firewall Policies

The following policies are associated to the case study as presented in chapter 6, section 6.4 on page 179.

## Policies deployed on firewall device A

| No. | SourceIP | SourcePort | DestIP | DestPort | Proto | Action |
|---|---|---|---|---|---|---|
| A.1 | 64.10.0.0/19 | * | 64.10.32.0/19 | 80 | TCP | Forward |
| A.2 | 64.10.0.0/19 | * | 64.10.32.0/19 | * | * | Drop |
| A.3 | 64.10.0.0/19 | * | 64.10.64.0/18 | 80 | TCP | Forward |
| A.4 | 64.10.0.0/19 | * | 60.10.64.0/18 | * | * | Drop |
| A.5 | 64.10.0.0/19 | * | 64.10.128.0/18 | 80 | TCP | Forward |
| A.6 | 64.10.0.0/19 | * | 60.10.128.0/18 | * | * | Drop |
| A.7 | 64.10.0.0/19 | * | * | 80 | TCP | Forward |
| A.8 | 64.10.0.0/19 | * | * | * | ESP | Forward |
| A.9 | 64.10.0.0/19 | * | * | 5060 | TCP | Forward |
| A.10 | 64.10.0.0/19 | * | * | 21 | TCP | Drop |
| A.11 | 64.10.0.0/19 | * | * | 2000 | TCP | Drop |
| A.12 | * | * | * | * | * | Drop |

## Policies deployed on firewall device B

| No. | SourceIP | SourcePort | DestIP | DestPort | Proto | Action |
|---|---|---|---|---|---|---|
| B.1 | 64.10.32.0/19 | * | 64.10.0.0/19 | 80 | TCP | Forward |
| B.2 | 64.10.32.0/19 | * | 64.10.0.0/19 | * | * | Drop |
| B.3 | 64.10.32.0/19 | * | 64.10.64.0/18 | 80 | TCP | Forward |
| B.4 | 64.10.32.0/19 | * | 64.10.64.0/18 | * | * | Drop |
| B.5 | 64.10.32.0/19 | * | 64.10.128.0/18 | 80 | TCP | Forward |
| B.6 | 64.10.32.0/19 | * | 64.10.128.0/18 | * | * | Drop |
| B.7 | 64.10.32.0/19 | * | * | 80 | TCP | Forward |
| B.8 | 64.10.32.0/19 | * | * | * | ESP | Drop |
| B.9 | 64.10.32.0/19 | * | * | 5060 | TCP | Forward |
| B.10 | 64.10.32.0/19 | * | * | 21 | TCP | Forward |
| B.11 | 64.10.32.0/19 | * | * | 2000 | TCP | Drop |
| B.12 | 64.10.0.0/19 | * | * | * | * | Forward |
| B.13 | 64.10.0.0/19 | * | 64.10.32.0/19 | 80 | TCP | Forward |
| B.14 | * | * | * | * | * | Drop |

## Policies deployed on firewall device C

| No. | SourceIP | SourcePort | DestIP | DestPort | Proto | Action |
|---|---|---|---|---|---|---|
| C.1 | 64.10.64.0/18 | * | 64.10.0.0/19 | 80 | TCP | Forward |
| C.2 | 64.10.64.0/18 | * | 64.10.0.0/19 | * | * | Drop |
| C.3 | 60.10.64.0/18 | * | 64.10.32.0/19 | 80 | TCP | Forward |
| C.4 | 64.10.64.0/18 | * | 60.10.32.0/19 | * | * | Drop |
| C.5 | 60.10.64.0/18 | * | 64.10.128.0/18 | 80 | TCP | Forward |
| C.6 | 64.10.64.0/18 | * | 60.10.128.0/18 | * | * | Drop |
| C.7 | 64.10.64.0/18 | * | * | 80 | TCP | Forward |
| C.8 | 64.10.64.0/18 | * | * | * | ESP | Drop |
| C.9 | 60.10.64.0/18 | * | * | 5060 | TCP | Forward |
| C.10 | 64.10.64.0/18 | * | * | 21 | TCP | Forward |
| C.11 | 60.10.64.0/18 | * | * | 2000 | TCP | Drop |
| C.12 | * | * | * | * | * | Drop |

## Policies deployed on firewall device D

| No. | SourceIP | SourcePort | DestIP | DestPort | Proto | Action |
|---|---|---|---|---|---|---|
| D.1 | 64.10.128.0/18 | * | 64.10.0.0/19 | 80 | TCP | Forward |
| D.2 | 64.10.128.0/18 | * | 64.10.0.0/19 | * | * | Drop |
| D.3 | 60.10.128.0/18 | * | 64.10.32.0/19 | 80 | TCP | Forward |
| D.4 | 64.10.128.0/18 | * | 60.10.32.0/19 | * | * | Drop |
| D.5 | 64.10.128.0/18 | * | 64.10.64.0/18 | 80 | TCP | Forward |
| D.6 | 60.10.128.0/18 | * | 60.10.64.0/18 | * | * | Drop |
| D.7 | 64.10.128.0/18 | * | * | 80 | TCP | Forward |
| D.8 | 64.10.128.0/18 | * | * | * | ESP | Drop |
| D.9 | 64.10.128.0/18 | * | * | 5060 | TCP | Forward |
| D.10 | 64.10.128.0/18 | * | * | 21 | TCP | Forward |
| D.11 | 60.10.128.0/18 | * | * | 2000 | TCP | Drop |
| D.12 | * | * | * | * | * | Drop |

## Policies deployed on firewall device E

| No. | SourceIP | SourcePort | DestIP | DestPort | Proto | Action |
|---|---|---|---|---|---|---|
| E.1 | 64.10.0.0/18 | * | 64.10.64.0/18 | 80 | TCP | Forward |
| E.2 | 64.10.0.0/18 | * | 64.10.128.0/18 | 80 | TCP | Forward |
| E.3 | 64.10.64.0/18 | * | 64.10.0.0/18 | 80 | TCP | Forward |
| E.4 | 64.10.128.0/18 | * | 64.10.0.0/18 | 80 | TCP | Forward |
| E.5 | 64.10.0.0/18 | * | 64.10.64.0/18 | 80 | TCP | Forward |
| E.6 | 64.10.0.0/17 | * | * | * | * | Forward |
| E.7 | * | * | * | * | * | Drop |