Policy Authoring and Analysis Processes for Federation Policies

Jason Barron, BSc

Department of Computing, Mathematics and Physics

Waterford Institute of Technology

Thesis submitted in partial fulfilment of the requirements for the award of

Doctor of Philosophy

Supervisors : Dr. Steven Davy and Dr. Brendan Jennings

November 2013

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: 00132787
Date: November 2013

Dedication

To my always supportive parents

Edward and June Barron

and

my ever loving wife

Rattiya Barron

and

my dearest children

Nathaphon, Davin, Saoirse, Emhain & Aoife

Acknowledgements

I would like to thank Dr. Brendan Jennings for giving me the opportunity to pursue this PhD and for his extreme patience and invaluable guidance at many stages along the arduous journey. I would also like to thank Dr. Steven Davy for all the hours of dedicated supervision, motivation, and supportive conversations which helped bring this PhD to successful fruition. To all my colleagues and friends at the TSSG, many of whom provided useful debates, advise and conjectures on various aspects related to my research, I would like to say thank you. In addition, I would like to thank Prof. Joan Serrat and Dr. Alan Davy for making my PhD viva such a rewarding experience and for their insightful comments on the research undertaken. I would also like to thank the broader research community within the TSSG, both post-doctoral and senior management for their much needed support whenever it was required along the way. A big thank you to my parents, Edward and June for being so supportive and providing me the opportunity to complete my education to the fullest, I will be eternally grateful to you both. I would like to thank my ever loving wife Rattiya, whom has been at my side for every step of this PhD journey; because if she had not been I would never have been able to complete it, my gratitude to you has no bounds. Finally, I would like to thank all my children for maintaining my sanity through the difficult times by providing me with hours of entertainment through your wacky antics, you are the light in my life.

Abstract

Federations are viewed as persistent agreements between organisations that enable them to share information or capabilities in a controlled manner. Policy based management techniques can be used to support federations of service providers in two ways: 1) *policies alleviate the need for expensive human attention in the federation set-up and maintenance processes*, and 2) *policies can operate and maintain the federation in a more automated, but guided fashion requiring less manual intervention by system administrators.* This thesis presents a federation policy authoring process that allows for specification and consistency analysis of federation policies that adhere to a federation model during refinement. Federation policies are refined into multiple lower-level device language implementations using model-driven language development techniques. During refinement, and as part of the policy authoring process, the consistency analysis process uses ontologies and semantic web rules to retrieve deployed policies for consistency analysis using a policy element match algorithm. The element match algorithm analyses groups of related policies to detect relationships between them; this contrasts with state-of-the-art pairwise policy analysis which is not capable of detecting all inconsistency cases. In addition, an approach to tailor policy evaluation for enterprise social networks is proposed to cater for different policy execution environments ranging from low to high risk security environments; this approach is shown to increase evaluation performance.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

In today's dynamic business world, enterprises are increasingly appreciating the business value obtained from federating their resources and services. Enterprise social networking is an example of a typical service that can be federated between enterprise domains to allow employees communicate and collaborate outside their internal enterprise network boundaries. Unfortunately, there is a lot of manual effort required by system administrators in the initial set-up and maintenance of such federations of services. Automated processes are required for efficient and consistent federation set-up and maintenance. To ease the amount of manual effort required in the set-up and maintenance phases of federations, policy based management techniques can be leveraged. However, current policy based management techniques have been devised to cater for use within single enterprise domains and not across enterprises as is the case with federations.

In fact, most service providers (i.e. federation members) use heterogeneous policy based management systems that leverage different policy models, policy languages and policy deployment techniques which makes it extremely difficult to specify high-level federation (business) policies to dictate the overall goals (i.e. behaviour) of the federation and then have these policies realised (i.e. refined) unaided as lower-level enforceable policies by each enterprise participating in the federation. This thesis will investigate if current policy based management processes and algorithms can be tailored to assist in authoring federation policies with particular emphasis on federation policy specification, consistency analysis, and policy evaluation processes. In particular the approach outlined in this thesis investigates policy authoring across federated management domains where in each domain policies are organised hierarchically in a continuum from

high-level business policies to low-level enforceable policies. The goal is to provide a framework in which policies negotiated at the federation-level or modified at the local system-level (i.e. lower-levels) can be kept consistent with each other and with the goals of the federation through the use of policy specification and policy consistency analysis techniques. A federated policy authoring process will allow service providers federate their resources and services in an efficient, more automated, and consistent manner.

Federation policies allow multiple service providers to control federated services under specific rules agreed by the service providers. Federation policies specify the high-level goals and operating context of the federation which must be refined into lower-level application specific enforceable policies by each service provider participating in the federation. Unfortunately, it is not just a case of specifying federation policies and have these policies refined and enforced (unchecked) by service providers participating in a federation. There are three problems that need to be considered: 1) *what is an acceptable representation of these federation policies* 2) *what type of information needs to be conveyed in the federation policy (bearing in mind security issues and service providers' willingness to share internal information with potential competitors)* and 3) *any modification (create, update, delete) of a policy (federation/local) can have unforeseen effects on other deployed policies within the local system if consistency checking is not performed prior to the modification.* Any solution to these problems will require a generic means of representing both the static characteristics of management domains (i.e resources, services, devices, etc.) and the dynamic relationships that exist between the domain managed entities to aid federation policy specification and consistency analysis processes. An information model provides various generic techniques for modelling both static and dynamic aspects of managed systems through the use of UML (Unified Modelling Language) models and ontological models. A suitable information model should be capable of modelling not only the management domain and its management policies, but also essential federation aspects such as the context/context data of the federation and governance structure. The DEN-ng (Strassner (2003)) is an object-oriented information model that is capable of modelling not only the static characteristics of domain managed entities, but also the policies specified over those managed entities. However, as originally formulated the DEN-ng is not capable of modelling certain federation aspects and would need to be extended to cater for modelling these essential federation concepts. For the work performed in this thesis the DEN-ng has been selected as a

suitable candidate to extend for complete representation of both the managed domain and policy models applicable to federation management.

A federation policy authoring process that encompasses both policy specification and consistency analysis processes is more complex with the policy continuum as policies are represented at different abstraction levels. Federations of services and resources are composed at arbitrary layers (e.g. business, system, device) of abstraction within managed domains and are responsible for performing management tasks (including, network management, security management, etc.) within a local system. Some federations may comprise of only a single layer (e.g. federation at the service layer, federations of devices, etc.) others may consist of multiple layers (e.g. federation at the service layer that also requires lower-level layers such as the network layer to be federated). The policy continuum (Strassner (2003)) can be leveraged to map local polices for different constituencies of policy authors within the domains of services providers. It has proven to be effective at abstracting policy representations to aid policy conflict analysis processes in a single-domain environment (Davy *et al.* (2008b)). However, in federated-domain environments the policy continuum model would need to be extended to cater for mapping between local continuum policies and federation policies. Federation policies can be mapped to local policies at arbitrary layers within the policy continuum created as a result of refinement of the federation policy. This federation to local policy mapping is a useful tool to aid policy authoring and in particular policy consistency checking whenever a federation/local policy is added/modified/removed in some manner, but any policy mappings between federation/local system policies must be kept consistent to reflect policy changes to both local and federation policies. Any changes within the federation or within the local system that causes changes to federation/local policies will need to be monitored and possibly acted upon to keep federation policies consistent. Policy consistency analysis processes are required to detect potential inconsistencies between deployed local service provider policies and candidate federation policies before refinement and enforcement of the candidate federation policies.

1.1 Federations

This section describes the motivation for federating networks and the use of policy based management techniques to help automate the processes required to set-up and maintain

federations. The section then reviews previous approaches to federations defined the the literature.

Current network domains are predominantly managed on an individual basis, the aim being to optimise the transfer of information within administrative boundaries. Co-ordination across domain boundaries does exist, however its scope is limited primarily to participation in end-to-end routing protocols, network peering arrangements and exchange of certain management information (in particular for charging and billing purposes). Furthermore, such coordination is static in nature, with changes requiring time consuming negotiations and coordinated deployment by service providers. However, due to de-regulation and consequent increases in marketplace competition there is a strong impetus for service providers to support more flexible business models—a good example is that of virtual operators owning little or no infrastructure of their own, so that service delivery is always based on crossing multiple administrative domains. Network management systems need to be evolved to support *federations*, which are viewed as persistent agreements between organisations, parts of organisations and/or individuals that enable them share information or capabilities in a controlled manner. To provide for creation and management of federations of communications networks their management systems will need to cooperate closely. This cooperation will encompass the exchange of monitoring and configuration data and possibly the delegation of authority to other federation members to manage their resources. This form of cooperation is most readily achievable where management systems apply PBM techniques.

Jennings *et al.* (2009) introduce the Federated Autonomic Management of end-to-End communication services (FAME) project, the overall goal of FAME seeks to provide network management systems that harmonise both local autonomous systems and federated systems so as to maximise the value received through the utilisation of federation resources and services by each federation participant and by consumers of the federation's resources and services. One specific challenge as outlined by Jennings *et al.* (2009), relates to business-driven network configuration in order to address the management of end-to-end service delivery and involves the development of processes that analyse monitoring information and use inferred knowledge to trigger policy management analysis and decision processes that result in the generation and deployment of a set of device configurations. These device configurations should best align a network's

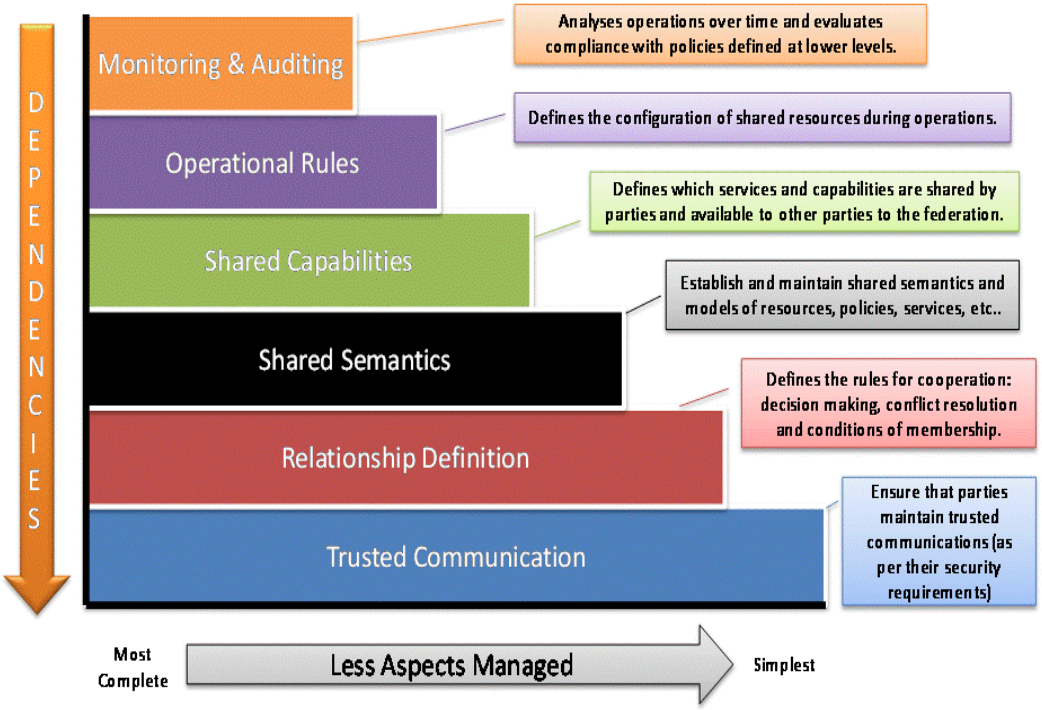

Figure 1.1: This figure illustrates the Layered Relationship Model that is a general-purpose conceptual model of the components of a domain relationship. The model is decomposed into layers, with each layer representing one aspect of the organisational arrangement.

behaviour with the business goals of its own organisation and those of the federation of which the organisation is a participant.

Feeney *et al.* (2004) propose an approach to managing communities (for example open source software project developer communities) using policy based management. They argue that such communities typically impose a management hierarchy which can be readily modelled in a policy management system. Sub-communities in a hierarchy can be delegated authority over a specified set of resources or services and within these constraints can specify and enforce their own "local" policies. When policies are created or modified they can be checked for conflicts with other local policies and, if no conflicts exist, can be passed to the parent community where potential conflicts can again be identified. Parent policies are given precedence in case of conflicts, thus the policy

hierarchy provides a well defined means of conflict resolution. The approach taken by Feeney *et al.* (2004) is close in spirit to the work outlined in this thesis since it provides a means for ensuring consistency between policies in a policy hierarchy. In an extension of the community-based management approach targeted towards federation management, the authors (Brennan *et al.* (2009, 2011); Jennings *et al.* (2010)) address management of federations directly, introducing the concept of a *Federal Relationship Manager (FRM)* component, illustrated in Figure 1.1 associated with individual federation members, that controls the secure delegation of capabilities to other federation members. The FRM is designed to mediate controlled sharing of capabilities between organisations wishing to participate in a federation. The goal of the FRM is to determine the complete set of technical requirements that an organisation must utilise in order to control and keep consistent federated relationships of varying complexity. It does not specify the use of any particular policy language, information model or management structure or process across a set of federated relationships. In essence, the FRM utilises semantic mapping management and authority delegation technologies to to aid in the negotiation of capabilities whenever organisations enter into resource sharing agreements ((Feeney *et al.*, 2010)). The FRM can be leveraged to facilitate federation policy authoring by providing secure capability sharing among federation participants. Jennings *et al.* (2010) focus on the negotiation and life-cycle aspects of a federation. The authors introduce a layered federal model, a conceptual model which is intended to represent the various facets of a federation agreement. The model is composed of six layers; namely, trusted communication, relationship definition, shared semantics, shared capabilities, operational rules and monitoring and auditing. These layers are required to define the relationships needed to federate resources and services across organisational boundaries. Although the approach is oriented towards autonomic network management and more specifically the life-cycle of a federation from its inception through to its culmination. The approach taken by Jennings *et al.* (2010) can assist in the high-level conceptual modelling of the components of a federation agreement where each layer of the federal model represents a particular aspect of the federation agreement.

1.2 Enterprise Social Networks

This section discusses enterprise social networks from the view point that they are a typical use case example of one such service that can be federated between enterprises where the federation techniques outlined in this thesis can be applied.

Enterprise Social Networks (ESN) (Cisco (2013), Microsoft (2013), IBM (2013)) are gaining in popularity as they are viewed as an effective tool by enterprises to leverage in a bid to increase communication, collaboration and productivity among employees and partners. Employees collaborate with their colleagues through communication (e.g. private messaging, etc.) and sharing/manipulating (i.e create, retrieve, update, delete) social network resources (i.e. documents, wikis, files, etc.) that require secure management using access control policies. Enterprise social networks try to mimic the openness (i.e social utility) of public social networks in facilitating easier information access, exchange and open communication flows, but operate within business processes that have strict security and privacy concerns. Public social networks operate under a relatively open ethos with regard to users privacy and security in order to more easily connect individuals and groups. However, in certain enterprise social network environments, users and resources operate under business processes that in some sectors such as health care, finance or government require very strict privacy and security regulation. Hence secure, but efficient management of enterprise social networks in these sectors is required. For example, in the financial sector, communication between a financial institution's officials and hedge fund investors should be monitored and policed to avoid misappropriation of the financial institution's funds. These security and privacy concerns as can be termed as safety concerns. These safety concerns can be alleviated through the consistent authoring (specification and conflict analysis) and enforcement of access control policies using the federation policy authoring process outlined in this thesis to control social network users communication/access to or manipulation of enterprise social network resources.

Typical use of enterprise social networks generates a large number of access requests by users/resources for permission to interact with other users/resources of the social network. Unfortunately, an enterprise's policy systems have limited resources to cope efficiently with large numbers of access requests and cannot efficiently evaluate all access requests received in a timely manner, as a consequence they either become a bottleneck

for traffic in the social network or worse they inconsistently evaluate access requests. The reason it is not possible to perform evaluation on all access requests received by a policy evaluation system in large applications with many users and resources (which is typical of enterprise social networks) in a realistic time frame is due to the complexity of rules nested within policies and the brute force method of request evaluation used by most request evaluation engines (Liu *et al.* (2011)). An additional point to note is when evaluating access requests in a social networking environment there is a delay constraint on communication as it should remain highly responsive and not hampered by adding excessive delays through policy evaluation of access requests (Marouf *et al.* (2011)). As a solution what is required is a method to optimise access request evaluation by tailoring expensive request evaluation towards specific users/groups of the social network deemed to pose a greater security risk. The approach should harness analyses of the social network to categorise the users/groups and access control policies applicable to those users/groups to increase access request evaluation performance while maintaining adequate security and privacy levels.

There have been a number of approaches to applying security controls in social networks based on the use of relations between users. Aleman-Meza *et al.* (2006) propose a framework to integrate two representative social networks and detect conflict of interest relationships between reviewers and authors of scientific papers. Central to this approach is the identification of relationship strength, based on the number of paper co-authorship between individuals to define a threshold value for conflict of interest. Carminati *et al.* (2006) propose an access control model that uses a combination of type of relationship, depth-level (i.e. distance between two social network users and trust level (manually specified value of how one users trusts another user) and trust-level to determine an aggregate relationship strength between two social network users. Kruk *et al.* (2006) analyse social networks to provide community driven access rights delegation. The authors use degree of separation combined with a computed friendship level metric to determine delegation of access control rights for social network resources. All of the approaches mentioned are not concerned with the actual evaluation of access requests, but how to specify access control rules over social network resources and/or detect inconsistencies over policies specified against social network users and resources.

Enterprises social networks can be federated between enterprises where federation policies are used to specify the overall behaviour of federated enterprise social networks

using federation policy authoring techniques such as federation policy specification and conflict analysis and access request evaluation. However, enterprise social network use cases need to be studied in more detail to understand the effects of applying the federation policy authoring techniques outlined in this thesis to set-up and maintain federations of enterprise social networks.

1.2.1 Enterprise Social Network Use Case

There are real security and privacy concerns with regards to an enterprise's private data being made publicly available as some of this data may be highly sensitive. There may even be a legal binding on the enterprise to keep its data private and financial penalties in failing to do so or problems permitting unintended communication between certain groups within collaborating (federated) enterprises as this may lead to conflicting situations. Typical social network structures provide open communication based around groups of individuals/groups (i.e. group task force, project group, etc.) that typify internal organisational structures often collaborating on projects or disseminating information with other groups both internal and external to the enterprise. Some typical examples of policy inconsistencies that may occur in an exemplar enterprise social networking scenario are given below:

- *Alice is part of a work group collaborating on a project between multiple enterprises, where policies applied to the working group permit access to project information to members both internal and external to the enterprise. A default enterprise policy is previously deployed strictly forbidding information dissemination outside of the enterprise boundary; these two policies are inconsistent with each other and will conflict if simultaneously deployed.*

- *Alice has specified a policy to only allow Bob (i.e. her direct friends) access to her on-line information. Bob is friends with Carol and has specified a policy to only allow Carol (i.e. his direct friends) access to his on-line information. Alice's work colleague is Carol. Alice has specified a policy not to share her on-line information with her work colleagues. Carol can still access some of Alice's on-line information through her friendship with Bob. These two access control policies are indirectly in contradiction with each other and will cause an inconsistency in the system if both policies are deployed simultaneously.*

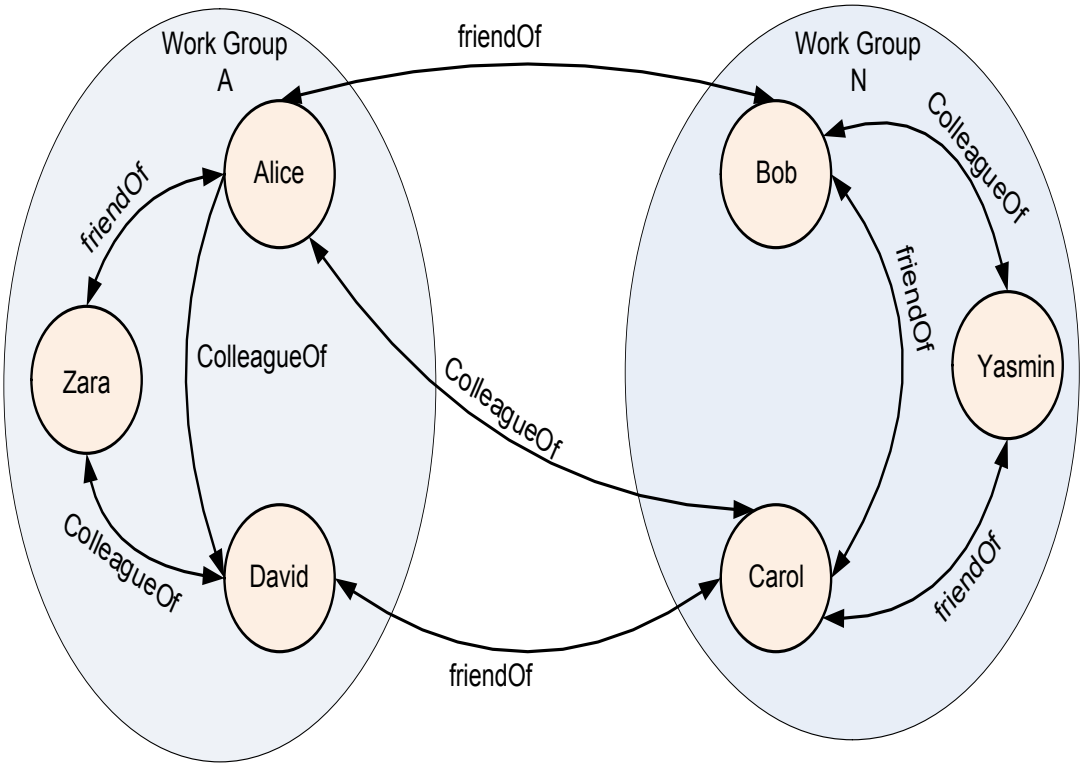

Figure 1.2: This figure depicts a typical enterprise social network that comprises multiple social network groups with many members that can span multiple enterprises. Group members have complex relationships of many types with members both internal and external to their group which makes the processes of policy specification, consistency analysis and evaluation much more difficult.

Enterprise social network structures can often be overly complex and as a consequence authoring access control policies to protect privacy and control communication is often not possible using standard policy authoring techniques. This is especially the case when social networks are federated among enterprises as depicted in Figure 1.2.

The reasons for this are that traditional policy authoring processes were not designed to cater for federated social networking environments as they take a pair-wise approach to policy specification and consistency analysis which is at odds with the way social networks are naturally structured. Also authoring access control policies in federated enterprise social network environment is more dynamic and complex than in traditional network domains in that policies are modified more frequently and can involve several

policy authors (i.e. users/group members). An enterprise social network's users and resources are typically hierarchically structured, but dispersed across the entire social network; this causes more policy requests relating to interactions across different parts of the hierarchy as such analysing a candidate policy against a large group of dispersed deployed policies can cause efficiency issues in the evaluation process.

1.3 Research Questions

The research questions addressed by this thesis are listed below. They highlight the challenges encountered when designing processes and algorithms for the specification and consistency analysis of federation-level policies within the context of multiple policy continuum instances.

1. *How can a policy authoring process be defined to support the consistent specification of federation-level policies by multiple constituencies of policy authors at arbitrary abstraction levels aimed at controlling federations of services and resources?*

   In order to control federated network resources and services, service providers will need to specify candidate federation-level policies that need to be refined by each service provider into (possibly multiple) system-specific policy language policies for enforcement. Techniques are required to ensure the consistent specification and refinement of candidate federation-level policies into local system-specific policies. Mechanisms are also needed to indicate that the refinement of a candidate federation-level policy has been successful or not. If not, re-negotiation of federation-level policies will be required during the refinement process to resolve any possible policy specification inconsistencies.

2. *What conflict analysis algorithms need to be developed to assess the consistency of candidate federation-level and local policies when policies are created, modified or withdrawn?*

   Policy conflicts can occur between deployed local policies and candidate federation-level policies during the refinement and enforcement processes. Techniques are required to resolve these potential policy conflicts before they have a negative

impact on local systems. Semantic information regarding a managed domain's entities and the relationships that exist between those entities can be leveraged by powerful semantic web reasoning techniques to detect potential occurrences of policy inconsistencies. Unfortunately, current policy conflict analysis techniques have been designed to cater for conflict analysis within single domain environments and hence new conflict analysis techniques need to be developed to cater for conflict analysis within federated domain environments.

3. *What strategies can federating enterprises adopt to increase access request evaluation performance while maintaining an acceptable level of risk?*

   When managed domains federate their enterprise social networks, communication between users/groups or access to enterprise services/resources needs to be securely managed using access control policies. An enterprise's policy systems have limited resources to cope efficiently with large numbers of access requests generated through social network use. An approach to optimise access request evaluation performance is required by tailoring request evaluation towards specific users/groups of the social network deemed to pose a greater security threat. This will require processes that can harness analyses of social networks to categorise the users/groups and the access control policies applicable to those users/groups in a bid to increase access request evaluation performance while maintaining adequate security and privacy levels.

1.4 Main Contributions

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

- A federation policy authoring process, outlining the steps to be taken when local or federation-level policies are created, modified or withdrawn.

- As this process depends on the presence of a rich system model for policy analysis, extensions to the DEN-ng information model that models governance, and context/context data aspects of federated domains are provided along with an associated federation policy authoring process (Barron *et al.* (2010)).
- Polices are organised in a hierarchy from high-level goal policies to low-level enforceable policies within each managed domain. The policy continuum models policies at arbitrary abstraction levels and facilitates policy specification and conflict analysis processes. New extensions to the policy continuum have been provided to allow for mapping between federation and local policies (Barron *et al.* (2010)).
- A model driven approach that abstracts access control policies into a clear and structured set of rules defined using terms familiar to a non-systems expert, which may then be realised (refined) into multiple (lower) levels of abstraction. The proof of concept system uses Model-Driven Development (MDD) techniques to transform high level business policies into device specific policies that can be enforced by multiple access control system types. (Davy *et al.* (2013)).

- A policy consistency analysis process as a central component of the overall federation policy authoring framework for federation policies.
  - The primary requirement on any policy consistency analysis process is that it remain application-independent so as to be applicable to multiple policy-based management environments. Barron *et al.* (2011) outlined application-independent policy selection and consistency analysis processes required for authoring policies in a federated environment.
  - The policy consistency analysis processes outlined by Barron & Davy (2013) were used to analyse policies for inconsistencies and as an example policy redundancy was detected by firstly retrieving a minimal set of deployed policies that potentially matched a candidate federation policy and secondly identifying relationships over policy elements. Identification of relationships over

policy elements may be either complete or partial through the combination of policy elements.

- An approach to probabilistically optimise access request evaluation by tailoring request evaluation towards specific users/groups of an enterprise's social network deemed to pose a greater security threat.
  - An enterprise's policy systems have limited resources to cope efficiently with large numbers of access requests generated through social network use. The approach carried out by Barron *et al.* (2013) harnesses analyses of a social network to categorise the users/groups and access control policies applicable to those users/groups in a bid to increase access request evaluation performance while maintaining adequate security and privacy levels.
  - An access request router was developed using the Java programming language. It consists of a number of classes for processing access requests to parse subject details, query a database for social network user tie strength, determining the overall tie strength and routing the request to a particular Policy Decision Point (PDP). The access request router's function is to determine which PDP a request should be forwarded to and forward the request accordingly.
- A federated end-to-end eXtensible Messaging and Presence Protocol (XMPP) communication service test-bed to evaluate newly developed processes and algorithms outlined in this thesis
  - Experimentation and evaluation of algorithms and processes designed for federation policies required development and implementation of a federated end-to-end XMPP communication service test-bed. The test-bed enabled experimentation and verification of any newly developed federated policy specification and consistency analysis algorithms and processes. The federated XMPP test-bed was implemented and deployed within multiple managed domains and was comprised of the following components, an open-source XMPP server (i.e. Openfire (Realtime (2011))), an Interceptor and an eXtensible Access Control Markup Language (XACML) policy server (i.e. SunXACML PDP (Proctor (2006))).

– Interceptor - a software agent acting as a Policy Enforcement Point (PEP) in an XMPP network, used to intercept XMPP communication from a user, generate XACML access requests, send access requests to a XACML policy server (PDP) and enforce responses received from the XACML policy server (i.e. either permit or deny the XMPP communication). An Interceptor was implemented in the Java programming language to intercept XMPP packets travelling between the XMPP client and server with the aim of applying access control policy rules to the type of communication being sought. The Interceptor receives XMPP communications leaving and entering each organisation participating in a federation and can query and enforce XACML based policies on these XMPP messages.

1.5 Main Conclusions

This thesis presents processes and algorithms to assist in authoring federation policies that includes in particular policy specification, consistency analysis, and policy evaluation processes across federated domain environments. The approach provides a framework in which policies negotiated at the federation-level or modified at the service-provider-level can be kept consistent with each other and with the goals of the federation through the use of policy specification, policy conflict analysis, and policy evaluation techniques. The main conclusions of the thesis are:

- The DEN-ng information model was extended to cater for modelling governance, context/context data aspects pertaining to the federated and local domain environments. This context/context data is crucial as input to any policy specification or policy consistency analysis processes. Context data can be obtained from each federated domain participant to give an overall view of context for the federation. The context data related to federation policies can be harnessed to assist policy consistency analysis processes to determine situations under which deployed policies may conflict with a candidate federation policy.

- As part of the policy authoring process for federation policies, the formal policy continuum model was extended to cater for the re-negotiation of federation

agreements should policy conflict arise, or a change in the policies internal to an organisation. The policy continuum model enables experts responsible for authoring federation policies, a way to identify if their new or modified policies potentially conflict with current deployed policies, or if existing federation agreements need to be re-negotiated.

- Policy conflict analysis processes are required to detect potential inconsistencies that may arise between deployed local service provider policies and candidate federation policies before the federation policies are refined and enforced. Due to the heterogeneous nature of federations, these policy conflict analysis processes need to leverage some form of ontological modelling and associated reasoning processes to aid in policy specification and detection of inconsistencies between federation and local deployed policies. Ontological models represent concepts and rich semantic relationships in logical form to describe a target managed system that can be reasoned over to discover implicit knowledge from a domain model.

- In today's network environment users are typically members of, and participate in, many different groups for various reasons (both professional and personal). Policies can be specified over both individuals (i.e users) and/or groups, so it is imperative that any policy consistency analysis processes possess the ability to analyse not only individual policies (on a pair-wise basis), but also groups of deployed policies in order to detect certain policy inconsistencies that can only be detected when analysing individual and groups of deployed policies (as opposed to just comparing all deployed policies on a pair-wise basis).

- An enterprise's policy systems have limited resources (i.e, RAM, CPU, etc.) to cope efficiently with large numbers of access requests generated through social network use. By taking a probabilistic approach to optimise access request evaluation by tailoring request evaluation towards specific users/groups of the social network deemed to pose a greater security threat. The approach harnesses analyses of a social network to categorise the users/groups and access control policies applicable to those users/groups to increase access request evaluation performance while maintaining adequate security and privacy levels.

1.6 Thesis Outline

Chapter 2 presents a state of the art review of research in the area of policy based management, with particular emphasis on policy authoring specifically, policy specification, consistency analysis, and refinement aspects to support federation policy authoring processes. The chapter then discusses technologies such as information models, ontological models, and semantic web technologies to provide support for policy specification and consistency analysis processes. This chapter concludes with an identification of the related open research issues and challenges concerning federation policy specification and conflict analysis, together with a set of requirements for the work described in the remainder of the thesis.

Chapter 3 presents extensions to the DEN-ng information model and policy continuum model to cater for federation policy specification and consistency analysis. The extended DEN-ng federated domain model is specified in an Web Ontology Language-Description Logics (OWL-DL)representation that allows for additional domain specific semantics (entity relationships, etc.) to be added to the model. The OWL-DL model can then be queried and reasoned over using semantic web technologies to detect implicit relationships that hold between domain managed entities and potential inconsistencies among groups of deployed policies. As an initial step toward developing such federation policy specification and consistency analysis tools, this chapter introduces a federation policy authoring process whose steps ensure that local policies are kept consistent with federation policies as individual policies are created, modified or withdrawn. In particular, a federation policy specification process is described that uses model driven development techniques to specify multiple low-level enforceable policies from a single high-level candidate federation policy that adheres to the semantics of a federation model. Finally, an exemplar federated policy based management test-bed that was developed and implemented in order to evaluate newly developed processes and algorithms outlined in this thesis is described.

Chapter 4 frames policy consistency analysis as an optimisation problem and then describes how inconsistencies between candidate federation policies and previously deployed local policies can be detected when new federation policies are specified, modified and/or removed. The federation policy consistency analysis process takes a two phase

approach, a policy element selection phase and a policy element match phase. The consistency analysis processes firstly, uses semantic web queries to retrieve a minimal set of deployed policies that may potentially match a candidate federation policy over its elements and secondly, identifies matches over policy elements that may indicate cases of potential policy inconsistency. Finally, a number of federation use cases are then provided to evaluate the effectiveness of the policy selection and policy element match approach in federation scenarios.

Chapter 5 presents a novel access request router framework that harnesses social network analyses to categorise a social network's users and policies in order to evaluate specific categories of policies against specific access requests from the social network's users/groups in a bid to increase overall access request evaluation performance while maintaining an acceptable level of risk mandated by the business processes of an enterprise social network. The probabilistic approach taken ensures that access requests received from specific social network users (e.g. most active users generate more access requests) are processed more efficiently by performing less (repeated) policy evaluations on these types of requests while access requests from other social network users (e.g. whom do not communicate frequently) are evaluated against more policies as these social network users may pose a greater safety risk overall. This chapter concludes with analysis of extensive experiments designed to evaluate the performance and safety aspects of taking a probabilistic approach to access request evaluation.

Finally, **Chapter 6** summarises the contributions of this thesis and discusses future work. This chapter appraises the work carried out in this thesis under the constraints of extensibility and efficiency. The chapter discusses future challenges firstly with respect to extensions to the authoring process for federation policies and a federation policy negotiation framework. Secondly, an application of the policy consistency analysis approach in an enterprise social network environment is proposed. Thirdly, an optimal policy distribution approach is proposed that could complement the probabilistic approach to access requestion evaluation performance described in this thesis.

Chapter 2

State of the Art

This chapter provides a state of the art review of research related to policy authoring and in particular support of the processes of policy specification, consistency analysis and deployment for authoring and deploying federation policies. Firstly, this chapter overviews background work related to policy based management where the concepts and components of a typical policy based management system are described. Secondly, policy authoring techniques are described and reviewed from the perspective of reducing the complexity in setting up and maintaining federations. This includes a critical assessment of the weakness/gaps in policy specification, consistency analysis, refinement, and evaluation by current state of the art processes in support of federation policy authoring. Thirdly, support technologies such as information models are described from the perspective of harnessing ontological models together with semantic web rules to model and reason over domain managed entities and more importantly the semantic relationships that exist between those domain managed entities with the aim of making implicit knowledge contained within the models explicit to assist the processes of policy specification and consistency analysis.

2.1 Policy Based Management

Policy Based Management (PBM) is a management paradigm that separates the rules governing the behaviour of a system from its functionality. PBM systems have been used in the management of a wide range of application domains to reduce the complexity and associated running costs in the set-up and maintenance functions of managed systems

within these application domains. Two most notable application domains for PBM are access control security (Bell & La Padula (1976)) and network management (Moffett & Sloman (1991, 1993)). Policy based management (Boutaba & Aib (2007)) can be leveraged as an effective means to reduce to reduce the complexity associated with network management processes by minimising the manual effort required to control the behaviour of a managed domain's resources and services. Most PBM systems use high-level policies to specify the business goals of an organisation that are refined to (sometimes several) lower-level executable policies to be enforced by devices to control a managed domain's resources/services. However, there are open issues related to the consistent specification, refinement and enforcement of these policies.

Most service providers already leverage PBM systems to effectively manage resources and services within their domains. However, a major issue with PBM systems that has acted as a barrier to their widespread deployment is the fact that PBM systems use specific policy languages that have been designed for a particular application and are not easily adaptable to other applications. The main problem being the specification processes for policy languages is usually decoupled from any context information that can be yielded from system information models or ontologies. This has made working with and understanding policies unduly complicated and has also turned out to be a major barrier to policy specification processes. In order to allow PBM systems control the behaviour of federated resources and services, federation policies need to be agreed upon and specified to outline the high-level (business) goals of the federation that are then realised using a refinement process that produces local policies consistent with the local policy environment. However in the context of federations, an outstanding problem remains that service providers use PBM systems that are comprised of heterogeneous information models, policy languages and refinement processes to achieve this task. As a consequence of this heterogeneity, the goal of federating PBM systems becomes a complex task.

Another problem with PBM systems is that they use policies specified at arbitrary levels of abstraction to suit a different constituency of policy author (i.e. business, system, device, etc.) to make the complex task of network management more manageable. However, when participating in a federation, this makes it difficult to define a single process for policy specification due to the fact that policies are defined at different

abstraction levels, using disparate policy languages and related to different processes/-managed entities. It also becomes much more difficult to maintain the consistency of policies at any one level. In particular, inconsistencies may occur at any stage during the refinement process between a candidate policy and previously deployed policies that can have varying degrees of impact on a system if they are not detected and resolved. These policy inconsistencies may even prohibit the enforcement of the policy completely. What is required is a means to abstract and reason over this heterogeneous information in a generic representation to ensure the consistent specification and enforcement of federation policies with local deployed policies. Fortunately, the DEN-ng information model (Strassner (2003)) provides support for representing heterogeneous network resources/services and the context data relating to their operating environment. This information is of the utmost importance to the specification of federation policies by abstracting away the differences between these heterogeneous managed entities.

2.1.1 Description of a PBM System

A policy as outlined in RFC (RFC 3198) (Westerinen *et al.* (2001)) and later elaborated on by Strassner (2003) can be defined from two perspectives (1) A definite goal, course or method of action to guide and determine present and future decisions. "Policies" are implemented or executed within a particular context (such as policies defined within a business unit) (2) Policies as a set of rules to administer, manage, and control access to network resources. This implies that policies are specified at arbitrary abstraction levels within a network. A high-level business policy (also known as a goal policy) has the potential to map to several lower-level system or network policies (also known as action policies) in order to achieve its goal in directing the overall behaviour of the network or system. A PBM system is specifically concerned with the installation, deletion and monitoring of policy rules, as well as ensuring the system operates in accordance to those policy rules. According to RFC3198, a PBM system controls the running state of a system and also the managed objects within a system through the use of policies where the control paradigm used is based on finite state automata. When PBM techniques are applied to network management, the system is used in the provision of services across the network in a predictable manner. In the case of security, PBM techniques are used to provide secure access of or maintain the privacy of users, services and resources within the network. Terms in general associated with policy based management are described

in RFC 3198. In general policy terms are split into two groups, those terms that describe the characteristics of a policy and those terms that describe the management of policies. The following policy terms are derived from 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 defines 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 policy based management systems are described 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 Repository: 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. The next section discusses policy architectures that accommodate the different abstraction levels applicable to policy based management.

2.1.2 Policy Architectures

Analysis of policy architectures has been carried out by Rajan *et al.* (1999) where the authors examine issues that arise in the definition, deployment and management of policies related to QoS in an IP network. This includes an overview of requirements for QoS policies, alternative policy architectures that can be deployed in a network, different protocols that can be used to exchange policy information and exchange of policy information among different administrative domains. The Common Open Policy Service (COPS) (Durham *et al.* (2000)), depicted in Figure 2.1, and outlined in RFC2748 is a protocol standardised by the IETF for the administration and enforcement of policies. COPS is designed to be used with models based on the PCIM or other models extended from the PCIM to implement the decisions of the policies specified using the PCIM. The COPS models incorporates a PDP and a Policy Enforcement Point (PEP) and includes a standardised simple query and response protocol for the exchange of policy information. The COPS protocol is used to communicate policy decisions between the PDP and the PEP using one of two modes of operation for the COPS model, either an outsourcing model or a provisioning model.

Policy decisions are obtained as a result of evaluating policy conditions with the enforcement of a decision basically the execution of an action associated with a particular policy. Unfortunately, some elements of the policy based management architecture are missing from the COPS model as the it only describes the communication between

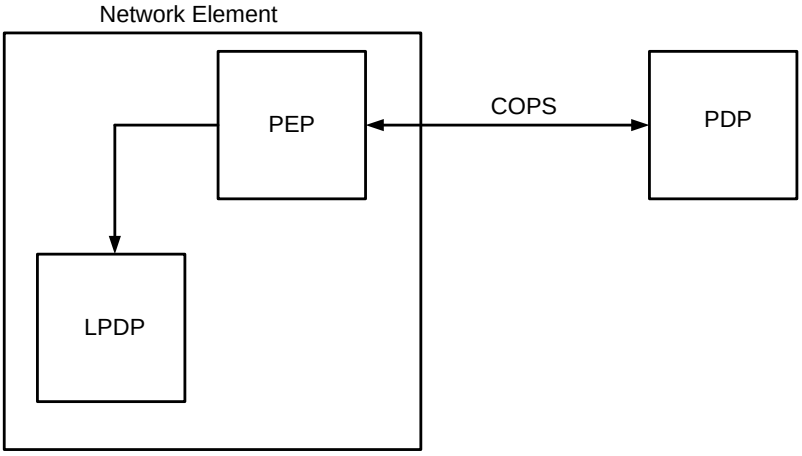

Figure 2.1: This figure illustrates the layout of various policy components in a typical COPS Architecture. COPS is used to communicate policy information between a PEP and a remote PDP within the context of a particular type of client. The optional Local Policy Decision Point (LPDP) can be used by the device to make local policy decisions in the absence of a PDP.

the PDP and PEP and does not include the processes of creating, modifying or deleting policies. However, the IETF/Distributed Management Task Force (DMTF) policy framework depicted in Figure 2.2, consists of four policy based management elements: policy management tool, policy repository, PDP and PEP. The policy repository can be either a database or a Lightweight Directory Access Protocol (LDAP) system used to store policies generated by the policy management tool. The PEP applies and executes various types of policies and uses an intermediary known as the PDP to communicate with the policy repository. The PDP is responsible for interpreting the policies stored in the repository and communicating them to the PEP. The PEP and PDP may be hosted in a single device or both components may be host on different physical devices.

For example, in an access control scenario that uses the XACML language, the PEP may be hosted on a separate device from the PDP. In situations like this the PDP acts as a XACML policy server with the PEP incorporated into a software component that is capable of communicating XACML requests to the XACML policy server and subsequently enforcing the decision of the access request on the managed resource. However, in a network filtering scenario, the PEP and PDP may be hosted on the same physical device with all decision logic contained within the filtering process and in situations like this the filtering process carries out the policy decisions (i.e. forward, mark, drop) on the packet. Both scenarios adhere to the policy based management system described in RFC2748, but use very different execution semantics which demonstrates the flexibility of the architecture.

With the introduction of standards to provide Quality of Service (QoS) guarantees across networks using integrated services with Resource Reservation Protocol (RSVP) and differentiated services. Policy based management techniques were viewed as a perfect technology to implement such a solution. However, each network used vendor specific policy based management solutions to implement QoS that were not interoperable across large scale communications networks. This heterogeneity and lack of interoperability between policy implementations occurred because there was no standard representation of policy or the semantics of what policy was trying to achieve. In order to achieve interoperability across domains Moore *et al.* (2001) defined an object-oriented information model; the Policy Core Information Model (PCIM), depicted in Figure 2.3, for representing policy information as an extension to the Common Information Model (CIM) (Lamers *et al.* (2010)). The PCIM was designed for the specification of policy and not for the enforcement of policy. The PCIM model describes the structure of policies, but does not specify the contents of policies.

The semantics of a PCIM policy is defined as *"If a set of PolicyConditions are satisfied, then execute the appropriate set of PolicyActions"*. The condition element of the policy is evaluated and if the condition element is evaluated to true then a set of policy actions defined in the policy are executed. This is a generic approach to specifying policy that can be applied to many applications or extended as necessary. It was further extended by the Internet Engineering Task Force (IETF) to represent policy to manage quality of service in RFC3644 Policy QoS Information Model (QPIM) and security in RFC3585 IP Security (IPsec) Configuration Policy Information Model

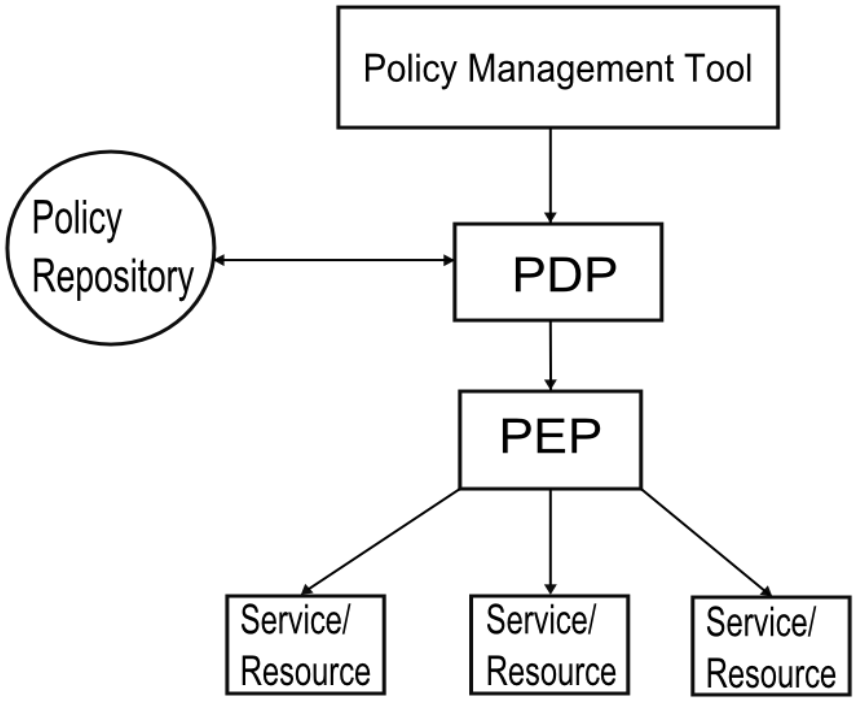

Figure 2.2: This figure illustrates the IETF Policy Framework that consists of four policy based management elements: a policy management tool used for policy administration tasks (e.g. policy specification, consistency analysis, etc.), a policy repository for storing pre-specified policies generated by the policy management tool, a PDP for evaluating requests against policies and a PEP for creating policy requests and enforcing policy decisions. The PEP can be used to control a single or multiple services/resources.

that dictates how policy can describe the enforcement of IPsec security services. In such policy applications as IPsec, policy conditions are specified over IP header information from received IP traffic from a router interface with the matching policy actions either modifying the information in the packet header and/or altering the processing of the IP packet.

2.2 Policy Authoring

Policy authoring is the process of taking natural language business objectives (or goals) and realising them as enforceable action policies through the sub-processes of policy

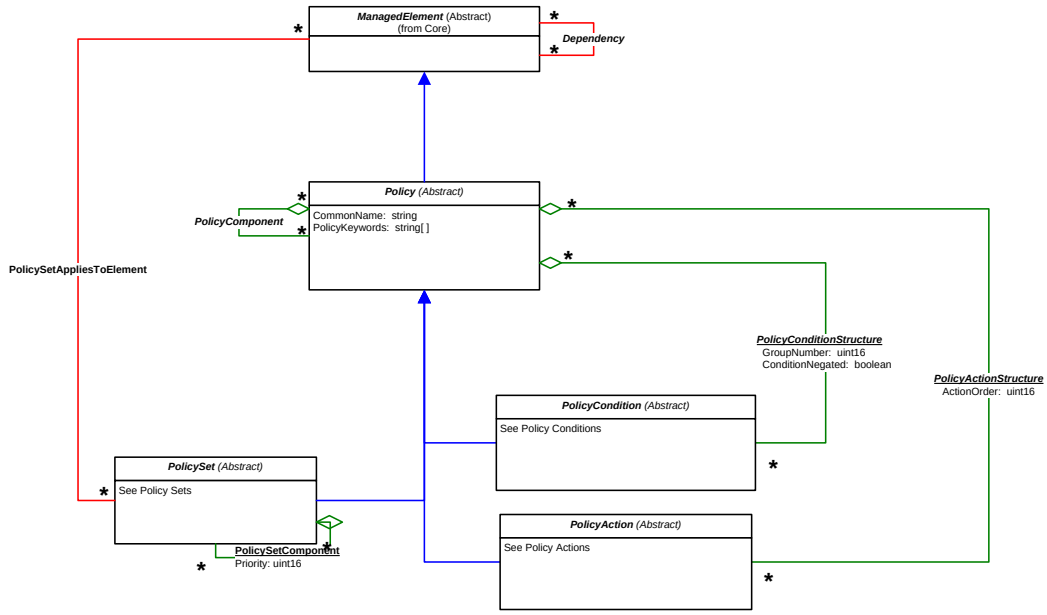

Figure 2.3: This figure illustrates the CIM policy model. A policy based on the CIM policy model is subclassed from ManagedElement and contains a PolicyCondition element and a PolicyAction element. A policy based on the CIM policy model follows the "condition-action" rule paradigm. It is also possible for a policy in the CIM policy model to be part of a policy set.

specification, consistency analysis and refinement. Business policies themselves are abstract policy specifications that not directly implementable and need to be refined to lower-level implementable policies that can be enforced by devices. As part of an overall policy authoring process, policy specification, consistency analysis and refinement processes are required to ensure that candidate policies are kept consistent with local polices during the refinement process. The need for a policy authoring process was motivated by Davy *et al.* (2008b). The authors content that there are typically many policy systems deployed in an organisation that require synchronisation to a guiding set of business policies. The presented authoring process was defined against a formalised structuring of policy sets termed the *Policy Continuum*. First mentioned by Strassner

(2003), the *Policy Continuum* abstracts business policies that are defined at a high level of abstraction from device specific policies that may be in many different policy language formats. The policy continuum model (Davy *et al.* (2008b)) has been defined as a framework for the development of stratified sets of policy languages, that utilise a common information model and allow for policy authoring (CRUD) across different constituencies. The policy continuum has been proven to effectively model polices at arbitrary levels of abstraction to assist policy specification and policy conflict analysis processes. Davy *et al.* (2008b) describe how the *policy continuum* model provides the basis of an approach for policy authoring in the domain of autonomic network management. In such scenarios, policies are specified in terms of an information model that is hierarchical in nature. The resulting policies can be arranged in a continuum, ranging from semi-abstract business policies to policies that can be implemented on devices with specific configuration parameters. Between the two extremes are policies with differing levels of detail and refinement, supporting different perspectives. The authoring process controls the flow of policy authoring in the policy continuum so that policy editors at any level of abstraction can be notified of indirect changes to their policies typically made due to changes in higher level policies. Also policy analysis processes are implemented to maintain the consistency and correctness of deployed policies at each level.

The work outlined in this thesis makes use of a policy authoring process and policy continuum model (across federated domains) to transform multiple device policy languages from a set of federation-level (business) policies. A single-domain policy authoring process is outlined by Davy *et al.* (2007); this process enables experts responsible for authoring policies that are deployed to manage various applications, a way to identify if their new or modified policies potentially conflict with currently deployed policies. This process is split into three steps: the first step traces the relationships between associated types of policies to verify that the modification of the selected policy does not invalidate currently deployed sets of policies. The second step analyses the policies associated to similar applications that may potentially conflict with the selected policy. The third step invokes a refinement process to derive a set of lower-level policies from the selected policy that must be recursively verified and tested for conflict and validity. This single domain policy authoring process would need to be extended to support the

consistent analysis and deployment of policies for multiple organisations, where the authoring process would need to be carried out independently by each enterprise involved in an inter-organisational (federation) agreement. Barrett *et al.* (2007b); Davy *et al.* (2008b) examined how policy refinement and conflict analysis can be carried out in the context of a policy continuum and in particular if the policy continuum spans multiple organisations (Barron *et al.* (2011)). During the authoring process, policy editors are notified while they are editing the policies, if there is a potential conflict in the current policy set. These notifications are facilitated by a conflict analysis process that takes into consideration policies deployed across all the device-specific policy systems. This ensures that consistent behaviour is observed in accordance with the initial business policies. A simple example would be a business policy granting access to a source code repository to a particular group of users. It is expected that the firewall policies would allow network traffic for the group of users to access the file server, and that sufficient access rights are also enabled for the group of users to read the files. By analysing the policies before they are deployed, and taking into consideration previously deployed policies, the authoring process can maintain consistency between multiple device-specific policies.

2.2.1 Policy Languages

Policy languages are used to specify policies for various management tasks (e.g. routing, security, etc.) to enforce the required behaviour of a managed system. Policy languages are specified at different levels of abstraction, but generally two levels of abstraction are accepted, high-level policy languages (for specification of goal-based policies) and low-level policy languages (for specification of action policies). For example, low-level security policy languages apply policy rules on a per packet basis, whereas, higher-level security policy languages specify the overall security goals for the organisation that must be mapped to low level action policies (device specific) and enforced. This has implications for policy authoring processes and in particular policy specification and policy consistency analysis processes. There are numerous policy languages available for specifying policy and each is usually suited to a particular policy application (i.e network management, security management, etc.). This section is dedicated to discussing some of the more popular network management and security management policy languages.

Policy languages follow either the "if-condition-then-action" (CA) or "on-event-if-condition-then-action" (ECA) execution semantics to triggering a policy. With regard to the "if-condition-then-action" approach; system information is obtained to use in the condition part of the rule. Two popular policy languages that follow the CA policy semantics include Rei ((Kagal & Rei, 2002)), and KAoS (Uszok *et al.* (2003)). The "on-event-if-condition-then-action" approach uses an event object which contains information regarding the event that has just occurred and the managed system to trigger the policy. This method is more useful as extra information can be obtained about the status of the system. Ponder ((Damianou *et al.*, 2001)), and the Policy Description Language (PDL)(Lobo *et al.* (1999)) follow the ECA policy semantics. Goal-based policies are high-level policies that specify the overall objectives to be achieved by the underlying policy system. Action-based policies are low-level polices that can be executed by devices.

2.2.1.1 Network Management Policy Languages

PDL (Lobo *et al.* (1999)) is a policy language based on the event-condition-action paradigm. The semantics of PDL are founded on action theories that have been mostly applied to active databases. In their approach, a policy description defines a transition function that maps a series of events into a set of actions to be executed by a policy enforcer. This work is based on the use of a service that polls its local environment for any alterations and relays any such changes detected to a policy server to check if a modification of policy is required. PDL does not utilise an information model or finite state machine to model the operating environment. Although PDL is a descriptive language it does not offer support for access control policies or more importantly the composition of policy rules into roles, or other grouping structures.

Ponder (Damianou *et al.* (2001)) is a declarative object-oriented language for specifying security and management policies and includes constructs for specifying the following basic policy types: authorisation policies, event-triggered obligation policies, refrain policies and delegation policies. Policy groups are used to scope related policies before common constraints can be applied to such groups; this allows Ponder to be used to apply policy to large organisational domains. Ponder uses domains to define groups (relating to organisational (role) hierarchy, geographical location, object type, etc.) from which to apply policies. Relationships are used to define the interactions

that can take place between a set of domains. Ponder also uses management structures to define a configuration that consists of role instances, relationships and nested management structures; these are typically applied to organisational units that have similar role configurations (i.e. bank branch, university department) and are in essence a means of specifying composite policies. In Ponder meta-policies are used to specify constraints that can be applied to a single policy or a group of policies. Ponder utilises a subset of the Object Constraint Language (OCL) to specify constraints.

The Ponder toolkit (Damianou *et al.* (2001)) comprises of a policy specification language roughly based on the IETF PCIM, but includes concepts for access control and a set of management processes that can perform some analysis of policies and enforce them. The ponder toolkit was specifically designed towards the network management and security aspects of services in distributed systems. Ponder combines two different aspects of policy that enable policy administrators to specify various types of policies for both network and security management. With Ponder, policies could be directly related to groups of managed entities that were either 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. Authorisation policies in ponder are semantically equivalent to access control policies and obligation policies dictate the execution of operations (i.e. policy actions), which is similar to the original definition of policy defined by the IETF.

Twidle *et al.* (2008, 2009) present Ponder2 which is basically a re-design of Ponder. It is based on a general purpose object management system that supports passing messages between objects. It utilises events and policies and employs a policy execution engine. The policy language associated with Ponder2 is PonderTalk, a high-level configuration and control language. Ponder2 provides support for managed objects that have been programmed in Java. In further work with Ponder2, Zhao *et al.* (2008) propose using algebra to specify both authorization and obligation policies for Ponder2 policies. The algebra employs two policy constants; namely, positive authorisation and negative authorisation and six basic operations; addition, intersection, subtraction, projection

and two negation operations, one for authorisations and another for obligations. Policy conflict analysis is only considered at the language level and only considers policy dominance checking and coverage checking. Conflict resolution is achieved by applying precedence to policies. Ponder2 environments can be federated giving a consistent view of the name spaces within the environments and the ability to propagate events in a transparent manner. Even though Ponder2 is considerably tied to Java, it still has the potential to be utilised in applying policy to federations.

The CIM Simplified Policy Language (CIM-SPL) (Lobo *et al.* (2009)) was developed by the DMTF. The objective of CIM-SPL is to provide a means for specifying if-condition-then-action style policy rules to manage computing resources using constructs defined by CIM. The design of CIM-SPL is inspired by existing policy languages and models including PDL, the Ponder policy language and Autonomic Computing Policy Language (ACPL). One of the main design points of CIM-SPL was to provide a policy language compatible with the CIM policy model and the CIM Schema.

KAoS (Uszok *et al.* (2003)) is a collection of componentised services that rely on a DARPA Agent Markup Language (DAML) description logic-based ontology of the computational environment, application context, and the policies themselves to enable runtime extensibility and adaptability of the system, as well as the ability to analyse policies relating to entities described at different levels of abstraction. Services supported by KAoS include policy specification, management, conflict resolution, and enforcement of policies within the context of an organisation's operating environment. KAoS policies are represented in Web Ontology Language Description Language (OWL-DL) which support formal system specifications that are based on the use of temporal logic. The use of temporal logic allows refinement patterns to be defined to derive low-level domain-specific polices that are formed through the conjunction or disjunction of sub-goals from high-level goals. By using a goal-regression process, a set of enforceable low-level policy actions can be selected for deployment.

2.2.1.2 Security Management Policy Languages

Rei (Kagal & Rei (2002)) is a policy language based on first order logic that provides constructs based on deontic concepts to support domain-independent policy specification. Rei provides support for policy specification, analysis, and reasoning. Rei is a

semantic policy language that specifies its policies in OWL-Lite. Rei uses domain-independent ontologies to represent rights, prohibitions, obligations, dispensations, and policy rules. Rei provides specifications for representing domain-independent information (constraints, actions, etc.) allowing the policy makers to use specific information that Rei has no prior knowledge of, but can still reason over while making decisions. However, Rei also requires domain-specific ontologies for reasoning purposes. The policy engine associated with Rei accepts policies in first order logic and Resource Description Framework (RDF) and provides run-time conflict resolution, but cannot predetermine policy conflicts. On the positive side, Rei's policy engine provides support for delegation management that allows for controlling and propagating access rights using delegation. Unfortunately, Rei's policy engine has only been designed to reason over policies and not enforce them.

XACML (Rissanen (2013)) is an OASIS ratified general purpose access control language that is based on XML and is founded on the use of attributes of both subjects and targets. It was originally designed to model access control policies which means that it provides a syntax suitable for managing authorisation decisions and was originally targeted towards distributed systems. It provides standardised formats for both access control policies and request/response formats. It attempts to match attribute values from a given request with the attribute values of a policy target and if a match is found between an access request and a policy target then that access control policy is deemed to be applicable to the access request with the corresponding policy action enforced against the request. XACML also supports several rule combining and policy combining algorithms for use when more than one rule/policy is found to be applicable to a given access request. XACML is a flexible and extensible access control policy language which may be suitable for specifying and enforcing access control policies in a federated domain environments. However, due to its extensibility; policy conflict analysis becomes arbitrarily complex and hence, necessitates the use of a common information model to leverage pertinent information regarding the target managed systems to assist policy specification and conflict analysis processes. Lorch *et al.* (2003b) discuss XACML in authorisation systems where the authors illustrate different operating scenarios that make use of XACML to incorporate varying authorisation approaches and show various ways to leverage XACML. The authors make a comparison between Shibboleth ((Erdos & Cantor, 2002)) (a web-based authentication and authorisation system), Cardea

((Lepro, 2004)) (a distributed authorisation system), developed as part of the NASA Information Power Grid, and PRIMA ((Lorch *et al.*, 2003a)) (a system for distributed access control in grid computing environments). In the scenarios illustrated XACML was shown to be a flexible access control language capable of supporting various authorisation requirements. However, XACML does not support formal representation and reasoning over it language constructs as provided by other policy languages. Nor does it provide support for policy conflict analysis or resolution.

The WS-Policy language (Bajaj *et al.* (2006)) is a general purpose policy language for specifying the capabilities, requirements, and general characteristics of entities in XML web services-based systems. WS-Policy defines a policy to be a collection of policy alternatives (a policy alternative defines the required behaviour of a policy subject), where each policy alternative is a collection of policy assertions. Assertions and alternatives are not ordered; this means it may prove difficult to predict the overall behaviour of the policy and hence specify federation-level policies. Policy assertions are not mutually exclusive which may make policy conflict analysis more difficult in a federated domain environment. However, XACML and WS-Policy were developed for access control; the specification of these policies is based on the extensible XML format, enforcement of policies follows defined standards so that multiple service providers can implement their own enforcement processes.

2.2.2 Policy Specification

The *Xtext* Framework (Efftinge & Völter (2006)) can be used to automate policy specification. State transitions are defined by rules that are specified using a grammar. Transforming from a high-level abstract language to a low-level concrete language (*refinement*) can be performed by inspecting both grammars and by defining the transformation rules that map one grammar element into one or more grammar element(s) in the less abstract language. Closer integration with static attributes reduces the abstraction level and leads to more concrete rules. Language driven approaches, where a structured language is defined for use by a specific constituency of policy authors, follows on from model driven approaches, where an information model is used to generate tools for use by a specific constituency of policy author. Barrett *et al.* (2007b); Van der Meer *et al.* (2006) examined how model driven approaches could be useful for aiding the automated deployment and analysis of policies for an organisation. The leap to

languages has been driven primarily through feedback on approaches and the maturity of toolkits available, namely the *Xtext* Framework (Efftinge & Völter (2006)). Barrett *et al.* (2007a) outlined a generic process to policy authoring that uses a model driven development approach and included the following steps.

- Step 1: Information model tagging - Most models, be it an information model such as DEN-ng, the TMF Shared Information/Data model (SID) or the DMTF Common Information Model (CIM), are heavy in the detail they provide. However, often only a subset of a model is relevant for the purpose of tool and language generation for a particular domain. Hence, it is necessary to tag and extract only the relevant aspects of the model for the generation of DSL(s) and tools.

- Step 2: Generate structural DSL - A DSL generated from an information model can be used to build an object model of the managed system. Such a structural DSL, and accompanying parser and editor, is cognisant of the types of entities that can be linked to one another and in precisely what manner, as such information is specified in the information model. A structural DSL can therefore prevent the user from describing configurations of the managed system that are inconsistent with the information model. Structural DSLs represent the structure, but not the behaviour of a managed system.

- Step 3: Generate policy DSL - The policy model subset of an information model usually includes entities to represent various components of policy rules that are used to specify part of the behaviour of a system. The policies defined using the Policy DSL orchestrates the behaviour of managed entities described within the object model, 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 object model and that are inconsistent with the information model.

Nejdl *et al.* (2005) propose the use of ontologies to simplify the tasks of policy specification and administration specifically tailored towards trust management on the web. The authors content that to make controlled sharing of resources more simplified in such an environment, parties will need software that automates the process of iteratively establishing bilateral trust based on the parties' access control policies, i.e., trust

negotiation software that will more that likely be in heterogeneous policy formats. The authors also content that ontologies can provide important supplemental information to trust negotiation agents both at compile time to simplify policy management and composition. For compile time usage, ontologies with their possibility of sharing policies for common attributes provide an important way for structuring available policies. In this context two strategies are proposed to compose and override these policies, building upon the notions of mandatory and default policies. The approach of using ontologies for the controlled sharing of resources and attributes would be beneficial for federation policy authoring where policy element attributes would need to be negotiated among federation members.

Verlaenen *et al.* (2007b) propose the use of an ontology to define a generic policy model that can be employed to specify various types of management and security policies. Specific policy languages for specific domains can be created as extensions of this generic policy model providing concepts relevant to that particular application. By representing and mapping the domain the policies are applied to in an explicit manner, domain concepts can be used directly inside policies. As a consequence of using a generic policy model, policies can be more targeted towards different areas and domains. However, this requires the definition of concepts specific to a particular area to be specified in a domain specific language (DSL). Also, the policy model used in this work is based on the PCIM, but extends the PCIM by further defining how policy conditions are structured. Nevertheless, it lacks an event component to specify when a policy should be triggered. Moreover, important policy authoring processes required for authoring federation policies such as policy refinement and conflict analysis are not supported. Even though the approach taken in this work of cascading ontologies is very promising and has been applied to various application domains, it does not tackle the issue of policy specification across management domains which is of the utmost importance for federations.

2.2.3 Policy Conflict Analysis

This section reviews the current state of the art related to policy analysis processes in general and in particular policy conflict analysis processes. The section will investigate the dependencies between the various policy analysis processes. The full potential of policy based management systems can only be realised if certain supportive processes

are implemented to ensure the consistent specification and enforcement of policies. Policy analysis processes are concerned with the examination of candidate and deployed policies to ensure that the deployment of a candidate policy does not leave the system in an inconsistent state. Policies are defined over shared managed resources to control the behaviour of a resource in some manner. However, certain policies when deployed simultaneously may illicit some behaviour for the shared resource that is in opposition to the behaviour specified by other policies. As highlighted by Davy (2008) a single definition of policy conflict is difficult to define as the definition of a policy conflict depends on the policy application domain (e.g. firewall policy conflicts, access control policy conflicts, management policy conflicts, etc) and so a policy conflict in one policy application may not indicate a policy conflict in another policy application. The definition of a policy conflict dictates the type of policy conflict analysis processes required to detect and possibly resolve conflicts among policies in specific applications. Strassner (2003) defines a 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 of a policy conflict dictates that when more than one policy is applicable to a set of overlapping managed entities and each applicable policy specifies different behaviour for those managed entities then a conflict is possible. For example, if the overlapping set of policies are triggered simultaneously by each policy having its conditions evaluated to true and if the action element of each policy specifies a contradictory action then a policy conflict is deemed to occur. Each policy based management approach dictates its own definition of how a policy conflict occurs, but generally they all are related to the enforcement of incompatible policy actions that lead the managed system to act in an inconsistent manner.

Initial research into policy conflicts (Moffett & Sloman (1994)) focused on application-independent conflicts in the areas of security and network management. The authors 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. Management policies according to the authors are either positive or negative authorisation or obligation policies. Moffett & Sloman (1994) contend that a 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. Conflicts

can be associated to the modality of a policy (authorisation/obligation) and the goals of the policy. Modality conflicts occur when the policy action is both permitted and prohibited, or obligated and refrained (not obliged to). Goal conflicts are associated with the effect permitted policy actions have on a managed system when both policy actions are enforced simultaneously. An example of this type of goal conflict is, if two policies require access to a resource (e.g. file), but due to resource restrictions only one policy can access the resource at a time. A conflict of duty occurs when the actions of two policies are executed simultaneously and the actions of both policies are defined by the application as conflicting. An example of this type of conflict occurs when an accounts clerk is both authorised to enter payment details for a cheque and is authorised to sign for payment of the cheque. This may be in breach of company regulations and would ultimately lead to a conflict of duty. A conflict of duty is similar to the conflict of interest in that the same policy subject can perform some type of management task on two different sets of targets and the outcome of both policy actions could be beneficial to the policy subject. For example, when a financial adviser is acting as an adviser for two different organisations, e.g. on a takeover bid for one client while advising other clients on investment decisions which would be influenced by knowledge of the takeover.

The definitions of policy conflict provided by Moffett & Sloman (1994) have been accepted as typical policy conflicts by the policy based management community and have been further researched by Lupu & Sloman (1997, 1999) with resolution strategies provided to resolve these types of conflicts. Dunlop *et al.* (2001, 2002, 2003) define similar policy conflicts, but further classify policy conflicts into dynamic and static cases. The authors contend that dynamic policy conflicts can only be detected at runtime, whereas static policy conflicts can be detected at compile time. Network management policies specified over the network are defined at the system-level which is a higher-level view of policy as opposed to policies specified for enforcement on network devices at the network-level. Chomicki *et al.* (2000, 2003) specify policies as sets of event-condition-action (ECA) rules. Policies are translated into logic programs where the subjects and targets of the policy are implicitly specified in the action component. Policy conflict is defined as the overlap of events, conditions and actions, where the actions must contradict for a conflict to occur which is similar to Strassner's definition of a policy conflict and can be applied to the network or system view of policy. These types of policy conflicts are detected dynamically as violations of action constraints.

Al-Shaer & Hamed (2003, 2004a,b) investigate policy conflicts from the network management perspective and define policy conflict specifically from the view of network filtering policies where a policy conflict is deemed to occur based on the specific ordering of policies that leads to anomalous behaviour. Conflicts also known as anomalies in network filtering policies come in the form of redundancy, contradictory, generalisation and correlation. A redundancy conflict is deemed to occur when a filtering policy is made redundant due to the specific ordering of the rules. In cases of redundancy the rule is not required and can be removed, whereas a contradictory conflict requires that the rule is not removed, but that the firewall policies should be re-ordered to resolve the inconsistency. The consistent enforcement of access rights for requesting entities through the use of access control policies is a complex topic. However, the definition of a policy conflict in the area of access control is unambiguous and occurs when contradicting access rights are granted to an individual requesting entity. Jajodia *et al.* (2001) and Wijesekera & Jajodia (2003) investigate policy conflicts in the area of access control. The authors designed formal models of access control and devised algorithms to search for various inconsistencies in deployed policies.

More recently, Alcaraz Calero *et al.* (2010) extends the definition of policy conflict types to include semantic conflict types. The authors provide a taxonomy of semantic conflicts that can occur and outline some realistic scenarios based on the main types of semantic conflict. The types of semantic conflicts defined include: *conflict of authority, redundancy conflict, conflict of priorities, multiple-managers conflict and self-management conflict.* The definition of the circumstances under which a policy conflict is likely to occur is dependent on the policy application area, wherein each policy application has its own requirements that define the necessary conditions to produce policy conflicts. Alcaraz Calero *et al.* (2010) used the CIM model (specified in OWL representation) to model policies and SWRL rules used to define the behaviour of the system.

Bandara *et al.* (2003) use a formal logic notation known as Event Calculus (EC) to represent the structure of a target managed system in an information model that can then be leveraged to assist in policy conflict analysis. The EC is used to model the dynamic aspects of a managed system where the EC object model describes the service descriptions and systems constraints. The EC is used to firstly represent the structural aspects of the system and secondly the behaviour of the system by associating pre-

and post-conditions to the actions that are permissible within the system. The authors considered two types of policy conflict, domain independent and application specific policy conflicts. A domain independent policy conflict can manifest itself across multiple policy applications as the semantics of the policy types do not change across policy applications and is defined to occur when two opposing policy actions that are triggered simultaneously, i.e. when one policy permits an action and another policy prohibits the same action. Whereas, application specific conflicts are specific to certain policy applications and are defined to occur due to conflicting actions associated specifically with the policy application, i.e two traffic management policies that both increase bandwidth and if both are executed simultaneously may increase bandwidth above a permissible threshold. With application specific conflicts, the semantics of the policy actions do not conflict in the traditional sense as is the case with domain independent conflicts and so these types of conflicts need to be defined by a system expert so they can be specified in the model to aid in their detection and possible resolution. In a similar approach to Bandara *et al.* (2003), Charalambides *et al.* (2005, 2006) focused on application specific policy conflicts that occur for static resource management aspects of QoS provisioning, known as network dimensioning for managing DiffServ aware networks. The authors defined a structural and behavioural model using EC that was based on the TEQUILA framework and proposed the use of abductive reasoning to explain the sequence of actions that must occur in a runtime execution sequence in order for application specific conflicts to occur. The authors outline a set of application specific conflicts that are specifically applicable to their application such as when a particular sequence of actions are executed which ultimately leads to an over provisioning of bandwidth. Unfortunately, the structural and behavioural EC models are static in nature and cannot be easily modified at runtime or extended to cover more policy applications without extensive re-modifications which indicates that this approach is not suitable for use in typical federation environments.

Chomicki *et al.* (2003) propose the use of event monitors and actions monitors to model application specific information that should be considered when examining situations in which policy conflicts may occur. The model used by the authors is based on the ECA rule paradigm coupled with the use of the PDL policy language as presented by Lobo *et al.* (1999). The authors approach to policy conflict detection is based on determining if a specific sequence of actions are being executed simultaneously. The

sequence of actions are defined in an action monitor list and represents a set of actions that should not be enforced together. The event monitor list is similar to the action monitor list, but takes a preventative approach in that when a set of events defined in the event monitor list occurs, then the likelihood of a policy conflict occurring is greatly increased. If an event on the event monitor list occurs then preventative action can be taken to avert a possible policy conflict from occurring. This approach to policy conflict detection requires a system expert to define the correct sequence of actions or events necessary for a policy conflict to occur and how to take decisive action to prevent a conflict from occurring which may prove difficult to extend to other policy applications or if the system itself is extended quite regularly as is the case with federations.

KAoS (Uszok *et al.* (2003)) was originally designed to manage agent based software systems, but was extended to include support for grid and web based services. KAoS uses the DARPA Agent Markup Language (DAML) (Connolly *et al.* (2001)), a description-logic-based ontology language to represent the structural aspects and policies applicable to a target managed system. KAoS provides support for authorisation and obligation polices and can easily be extended to include support for other application areas due to the fact that KAoS is based on an extensible description-logic-based ontology language. The policy analysis services for KAoS are capable of detecting application independent conflicts. The policy conflict detection algorithm employed by KAoS relies on the Java Theorem Prover (JTP) (Frank *et al.* (2008)) developed by Stanford to provide inferencing and subsumption capabilities during the evaluation and enforcement of policies. This has the added advantage of allowing reasoning over policies at different abstraction levels to infer implicit relationships between deployed policies which is required for federation policy authoring.

Kagal *et al.* (2003) present the Rei policy language as a policy language specifically targeted towards pervasive computing environments. Unlike KAoS (Uszok *et al.* (2003)) which is based on description logic principles, Rei is specified using OWL-Lite concepts and is based on first order logic programming principles, so as a result does not take advantage of the inherent reasoning capabilities available with OWL. Instead Rei's policies are transformed into Prolog where reasoning is performed using a logic programming system. Rei dictates that the domain descriptions over which a policy is being specified are also defined using OWL concepts which allows the approach to be easily extended to different application areas. Rei uses deontic policy concepts similar to that proposed

in Ponder (Damianou *et al.* (2001)) and as such is susceptible to similar policy conflicts that can occur in Ponder. If a policy conflict occurs, Rei uses meta polices to outline how the conflict should be resolved. Rei does not address application-specific policy conflicts specifically, but does include support for representing policy action preconditions and postconditions which means Rei could easily be extended to offer support for application-specific policy conflicts, but would be susceptible to scalability issues in large federation scenarios due to its requirement for first-order logic specification and reasoning to detect policy conflicts.

Baliosian & Serrat (2004) 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 specific 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 and not suitable for application across multiple domain environments.

Feeney *et al.* (2004) propose an approach to modelling communities for PBM systems. This approach focuses on the concept of communities within a hierarchy of authority as the fundamental unit of organisational analysis. The authors focus mostly on a single domain organisation that spans multiple countries. Where, each country's domain is responsible for controlling local resources, while the overall development and management of the community is handled on a global scale. As a consequence of this, there are global and local policies enforced in each domain. Communities can be extended through specialisation; where, sub-communities are restricted to having just one parent and can author policies for the same resource as their parent (i.e delegation). To check for conflict between such policies; the sub-community checks for policy conflict, if

none exists the policy is passed to the parent to check for conflicts against its policies, so as a result policies defined higher up the hierarchy take precedence over lower-level policies. However, this means that lower-level policies can only be more restrictive than higher-level policies due to the fact that communities can only communicate directly with parent communities or sub-communities. The policy conflict resolution process proposes to negotiate a conflicting policy between the level at which the policy was specified and the level at which the conflict occurs. Unfortunately, this may be several layers up or down the hierarchy and so, may prove complex to resolve possible policy conflicts if communities can only communicate directly with parents or sub-communities of themselves and not directly across peers as commonly occurs in federations.

Kempter & Danciu (2005) investigate the use of information models to aid in policy conflict analysis and specifically used the CIM models of a managed system to dynamically detect application-specific policy conflicts. The approach uses UML to represent implicit knowledge regarding a managed system and in particular the OCL is used to specify invariants of the managed system which are mapped to policy actions based on the ECA model. Any change in an invariant as a result of a policy action being executed is used to indicate a potential policy conflict. Resolution strategies are suggested to try and resolve policy conflicts with the main theme being to eliminate conflicting policies. However, the approach does not imply a consistent application of a set of policies and was targeted at lower-level system policies that only considered conflicts occurring at that level of the managed system, whereas to support consistent federation multiple levels of a managed system need to be represented where the models can be harnessed by policy specification and consistency analysis processes as required. In similar work to Kempter & Danciu (2005), Shankar *et al.* (2005) investigate the use of OCL concepts to aid policy analysis in the area of pervasive computing environments. The authors propose that the specification of policies should be based on an ECA-P (postconditions) model where the postcondition indicates the value an objects' attributes should have after a policy has been executed. The authors content that a policy conflict occurs when two or more policies define contradicting postconditions. They propose an algorithm to detect policy conflicts by analysing postconditions to determine if they are in contradiction to each other. The approach taken does not require implicit knowledge regarding the sequence of policy actions required to identify potential policy conflicts. Unfortunately, the approach does not consider constraints on the actual state of the

managed system which could be represented using other OCL concepts such as invariants and action pre-conditions and as such limits the functionality of the approach and its ability to support federation policy authoring processes.

Martin *et al.* (2006) propose an approach to determine policy coverage over policy test suites that are sets of requests and responses targeted towards access control policies. The coverage measurement tool measures the cover provided by a particular XACML policy against a randomly generated set of policy requests to determine if the XACML policy covers all requests made against it. In later work, Hwang *et al.* (2008) applied the same policy coverage techniques as Martin *et al.* (2006) to firewall policies. Both these authors take the most common view of policy coverage defined in the literature, in the sense that for a particular policy will the most typical requests made to a security system be met by that policy or are there cases where the policy cannot evaluate a response to the request. However, dominance detection performs a different kind of analysis, instead of detection to determine if a set of requests are covered by a particular policy, the check in this case is to see if a set of deployed policies cover or dominate a new candidate policy. The difference is subtle, but the problem is more difficult to solve given, the requirement to assess combinations of deployed policies, as opposed to sequential streams of requests.

Davy *et al.* (2006) outline an approach that uses a policy analyser to query an information model representing a managed system to which policies are being applied to retrieve the constraints over the operation of the system managed entities. Moreover these constraints represent the action pre-conditions, invariants, or post-conditions that must not be violated at any stage of the refinement process. The retrieved system constraints are then used during the refinement of high-level policies into lower-level polices by conflict analysis processes in an attempt to prevent conflicts with previously deployed policies. Later, Davy *et al.* (2008c) demonstrate the use an information model and associated ontology to assist in policy selection and conflict analyses. The selection algorithm uses selection rules based on a newly specified policy to select deployed policies for further analysis. The selected policies are then compared against the newly specified policy by a conflict analysis algorithm using a conflict signature matrix leveraged from an information model. In an extension to this work, Davy *et al.* (2008a) outline a policy selection process that uses a history of previous policy comparisons in a tree-based data structure to select policies for analysis. This selection process is part of

a two phase conflict analysis algorithm where the first phase identifies relationships between policies using a policy comparison matrix, while the second phase examines relationships in context of application-specific conflicts from an information model in a second relationship matrix and then performs a comparison operation on both matrices to detect occurrences of conflicts. The approach taken is concerned with querying a UML model in an attempt to prevent conflicts from occurring at lower levels in a system as a high level policy is refined and does not consider analysing policies in federated domain scenarios.

Verlaenen *et al.* (2007a) 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 authors suggest a number of policy analysis processes to support the consistent enforcement of policy which are defined as follows:

1. Policy conflict analysis: does policy A conflict with policy B.
2. Policy refinement: can policy A be realised at a lower level by policy set B.
3. Dominance checking: if policy B were removed will the system behaviour be altered.
4. Policy optimisation: the alteration of policies to reduce computational complexity of the policies analysis or enforcement.
5. Coverage checking: if a policy is defined for a particular condition, or specified over a set of managed entities.
6. Policy combination: can two or more policies be replaced with less policies, where the resulting behaviour is the same.
7. Policy deduction: can a policy be deduced from a set of policies.
8. Policy transformation: if policy A is transformed to policy B does it still meets its objectives.

Campbell & Turner (2007) developed a policy language for a call control system called APPEL that is based on the use of ontologies. The APPEL policy language is based on the ECA rule paradigm that can be extended to include support for any

application domain, provided a domain-specific ontology that is compatible with the APPEL language is defined for the domain. The conflict detection approach for APPEL is similar to the approach proposed by Chomicki *et al.* (2000, 2003) in that special rules are defined to execute when particular sequences of policy actions are enforced simultaneously. An optional resolution strategy is proposed that can be enforced to prevent a possible policy conflict from negatively affecting the managed system. By basing the APPEL policy language on ontologies, the APPEL language and its associated conflict analysis techniques can easily be extended to cater for other application areas by extending the base ontology to include concepts from a particular application area. A later approach by Campbell & Turner (2008b), specify policies in the APPEL language which are transformed into OWL using POPPET before being reasoned over using PELLET. The technique was applied to call control policies, where static and dynamic conflicts can be detected. However, the approach only focuses on detection and resolution of dynamic conflicts. Conflict detection is based on comparing the actions of a pair of policies to detect application-specific conflicts. Conflict resolution is achieved through the use of resolution policies that specify the action to be taken when two policies conflict. With this approach conflict resolution policies require domain-specific knowledge and the use of applying policy priorities to resolve possible policy conflicts. The approach is only applicable to a single-domain environment as it is grounded in the use of the APPEL language. It would be unrealistic to assume that each participant of a federation would employ the same policy language. In a federation of service providers, each federation participant will more than likely use different policy models and languages. A goal-directed proactive approach to monitoring the conditions of sensor networks from a wind farm is presented by Campbell & Turner (2008a). The approach makes use of ontologies to model the domain and policies to control the sensors of the network. The work presented here is based on the use of the ACCENT system architecture where there are three conceptual levels in the architecture, a user interface layer, policy system layer and a communications system layer. The APPEL policy language consists of a core language that can be extended for any application domain used to specify both the high-level goals of the system and the pre-defined system layer policies that are triggered by the high-level goals and used to control the communications layer. The analysis of policies in this work is only considered at a single level (i.e the system layer) which still leaves the potential for inconsistencies to occur at other

levels during the refinement stage and as a consequence the approach is not suitable for federation scenarios where multiple levels of policies need to be analysed.

Lin *et al.* (2007) demonstrate an approach to aid policy conflict analysis by determining the similarity of policies. The authors propose the use of policy similarity algorithms as an efficient means of determining the similarity between two polices and thus quickly eliminate such policies from any further extensive policy analysis processes. The policy similarity algorithms rely on ontological representations of the policies to assist in determining a value that indicates the level of similarity between both polices. Even though the approach taken highlights the need to reduce the level of complexity associated with policy analysis techniques that require the use of computationally expensive policy analysis algorithms it does not detect policy inconsistencies directly and is therefore not suitable for federation environments.

Kolovski & Hendler (2007); Kolovski *et al.* (2007) present a logic-based approach to XACML policy engineering that provides a comprehensive set of automated analysis services (policy verification, policy comparison, redundancy checking) for XACML. This work combines the expressive extensibility of XACML with the reasoning capabilities of DL. The authors approach maps a large fragment of XACML (including core XACML, XMLSchema datatypes and the administrative policy profile) to a subset of First Order Logic called Description Logics (DL), and uses DL reasoners to provide analysis services. This mapping of XACML to DL allows generic DL reasoners to be used for analysis tasks such as policy comparison, verification and querying. The authors demonstrated that logic based algorithms are a powerful method of policy analysis for authorisation policies and would be quite suitable for policy specification and conflict analysis of federation policies in homogeneous environments using only a single policy language, but not in federated environments where multiple distinct policy languages are used.

Wu *et al.* (2009) modelled and reasoned over trust policies through the use of EC with WS-Policy used for policy specification. Their approach was applied to the web service environment and in particular to security and dynamic trust aspects of a federation and focused specifically on detecting modality conflicts. The approach taken offers support for dynamic policy conflict analysis, with resolution to policy conflicts achieved through the use of resolution strategies. However, the approach may prove difficult to implement in federated domain scenarios as the authors only consider modality conflicts and do not specify a way of gathering application-specific information for the analysis

of application-specific conflicts. Wang *et al.* (2010) propose a conflict free access control model. The model maps every subject to a group and every object to a type. Access requests are based on privileges granted to the group and the requesting subjects role within the group. The authors outline situations in which redundancy can occur and propose the use of priorities to resolve redundancy conflicts, but do not provide an implementation of an algorithm to detect such redundancy conflicts and so this work could not be easily applied in federation scenarios.

Fitzgerald *et al.* (2009) propose a description logic approach to abstract deployed firewall policies and reason over the instantiations to ensure that no conflicts arise and that high-level business policies are consistently enforced. In later work, Fitzgerald & Foley (2009) demonstrate the use of ontologies to analyse firewall configurations associated with semantic web applications. The approach attempts to define an appropriate firewall configuration that is aligned with the type of semantic web application being hosted by the system. An ontology is leveraged to represent domain knowledge relating to the semantic web application. The authors develop ontologies to represent Linux Netfilter for filtering capabilities and TCP-Wrapper which is used to represent and reason over network access control configurations. The technique is based on harnessing ontologies to provide secure access control (not higher-level role-based access control) for lower-level firewall filter rules used by applications such as Netfilter. The approach outlined only provides support for a single level of policy authoring and does not offer support for multiple levels of policy authoring as is required for federation policy authoring.

Hu *et al.* (2011) present an anomaly management framework for representing and analysing different types of access control policies. The approach centres on the creation of a generic policy ontology based on the most common characteristics from each of the policy models. The generic policy ontology is then instantiated with instances representing policies from each of the policy models. A segmentation technique is used to aid in the check of anomalies between policies. A binary decision diagram (BDD) is leveraged as the underlying data structure for representing the atomic elements of a policy rule. The use of a BDD allows certain set operations such as unions, intersections, and set differences to be performed to segment the policies. The policies can then be segmented into non-overlapping and overlapping segments. The overlapping segments can be further subdivided into conflicting overlapping and non-conflicting overlapping.

Non-overlapping segments indicate a unique rule, while overlapping segments indicates a set of related rules that may potentially conflict with each other. It is feasible that the approach could be extended to highlight redundancy as well as other conflicts; however, the authors do not present such results in their work. The authors analyse several access control policy structures and build a generic ontology model based on the most common policy elements which is in-line with approach taken in this thesis. However, the approach suffers from scalability issues as the authors do not base their model on any standard policy information model and as such it may prove difficult and cumbersome to create a generic policy model when a federation contains a large number of members.

2.2.4 Policy Refinement

There have been a number of approaches taken to refinement and analysis of high-level goals into implementable low-level policies applied to multiple domains.

Moffett & Sloman (1993) contend that policy based management of very large (i.e. distributed) systems requires the generation of low-level and system-specific policies from general high-level business (i.e. goal) oriented policies. The authors propose the generation of policy hierarchies, where low-level policies are generated to represent the goals and objectives of high-level policies. The process of building the policy hierarchy is described from the perspective of policy refinement where policies are continuously modified as are the associated policy hierarchies. The authors highlight some of the main research challenges associated with automated policy refinement and contend that the refined policies must be analysed to ensure they can be enforced consistently onto the target managed system and that the refined policies meet the goals and objectives of their higher-level parent policies.

Verma *et al.* (2002) define two levels of policy; business-level policies and technology-level policies and assume that business-level policies can be mapped directly to technology-specific services (i.e QoS, IPsec, etc.). The approach uses a policy translation logic to translate from high-level policies to device configurations. Algorithms used for policy conflict analysis make use of a policy schema to check limit bounds of attributes and policy relationships for dominance against policies stored in a database. The authors assume that policies do not conflict with any other deployed polices or there is only one generic policy being applied to all devices.

A goal-based approach to policy refinement is presented by Bandara *et al.* (2004) that is based on a formal representation of the management system to assist in the realisation of low-level polices. EC as presented by Russo *et al.* (2002) is a formal language designed to model and reason over dynamic systems that is based on a logic formalism to represent the managed system. Bandara *et al.* (2004) modelled the managed system in EC and then used the EC to describe how low-level actions affect the managed system. Goal elaboration techniques are leveraged to transform a policy into a set of low-level actions that meet the objectives of the high-level policy. A system administrator is then responsible for ensuring that the correct sub-goals (or system actions) are selected to enforce the high-level goals (policies). The approach taken harnesses system-specific information that may be extracted from a UML model and its associated finite state machine and then incorporated into the refinement process. In later work, Bandara *et al.* (2006) describe an implementation of the goal-based policy refinement approach applied to an IP Differentiated Services application. In that work the authors show that the EC can be used to model the managed system and the high-level QoS policies defined for its operation. The high-level policies are refined into low-level enforceable policies used to provision the DiffServ classes of service. The approach taken demonstrates the beneficial use of harnessing pertinent system information to support policy analysis processes. Rubio-Loyola *et al.* (2005, 2006a,b) extended the logic based approach to policy refinement that included a managed system model proposed by Bandara *et al.* (2004) by incorporating formal verification techniques into the refinement process. Formal verification was achieved by using model checking techniques based on Linear Temporal Logic (LTL). The temporal relationships that exist between actions of a system can be examined using LTL. The authors demonstrated that by using LTL techniques new policies could be produced that were consistent with the system specifications. Unfortunately, existing deployed policies were not considered in the policy analysis approach.

In a similar approach to Bandara *et al.* (2004), Craven *et al.* (2009) present a logical framework for policy specification. The framework provides support for both authorisation and obligation policies and includes a model of the changing system states. The analysis algorithms are based on abductive, constraint logic programming systems with EC used for describing how events and actions that occur in the system can lead

to the triggering of a policy rule. Craven *et al.* (2010), Craven *et al.* (2011) later extended this work to include policy decomposition, a process where policies specified at a high-level of abstraction are mapped into lower-level implementable policies using pre-defined refinement mapping rules based on an application-specific system model. The refinement mappings between policies at different levels of abstraction are statically based on the use of pre-defined refinement rules that need to be specified or modified by a system expert and that are only applicable to a particular application domain. A similar approach was taken by Zhao *et al.* (2011) where the authors propose a framework to automatically transform security policies into implementable and enforceable rules. The proposed framework consists of a centralised policy server that includes three elements: a knowledge database that maintains information on the policy domain, a refinement rule set that defines rules for policy transformation and composition, and a policy repository for storing policies specified at different abstraction levels. The approach taken is similar to Craven *et al.* (2010), Craven *et al.* (2011) in that it makes use of pre-defined refinement rules that must be specified by a domain expert and again are application-specific.

2.2.5 Policy Evaluation

Policy evaluation is the process of determining whether the attributes of a particular request (e.g. access request) matches the attributes specified in a particular policy. If a request matches a policy then it can be said that the policy is applicable to that request and that the action element of the policy should be applied to the request (e.g. either permit or deny the request). The most common method of policy evaluation is a *"brute force"* method where a request is received and sequentially compared against each policy/policy set in a policy repository to determine if a match exists. This section outlines related work in the area of re-formulating policy sets and increasing the performance of policy decision points and evaluation of policies in general.

In Marouf *et al.* (2009, 2011) the authors apply a clustering technique to policy sets based on the K-means algorithm. The approach re-orders policies within policy sets and rules within policies based on target subjects to find the optimal policy execution environment. The technique is based on two assumptions, users who share similar properties also share similar request types (in other words the same subset of rules are evaluated against those types of users) and optimal rule ordering is conceptually

linked to actual user requests. The approach categorises users' access requests at two levels. Firstly, an access request is categorised by subject to determine which policy is applicable to it and secondly, a match is found between the request and so-called execution vectors for that policy (execution vectors are the order in which the rules in a policy are applicable to a request. The approach is limited in that it can only guarantee the consistency of request decisions when evaluating policies or policy sets that use the permit-overrides or deny-overrides algorithms. In a similar approach to Marouf *et al.* (2009, 2011), Miseldine (2008) propose an approach to minimise the cost of finding at match at the rule-level, target-level, and policy-level through policy configurations. No changes are made to the XACML policy specifications themselves, but the improvement is achieved by applying optimisation techniques to newly specified XACML policies where policy authors can choose to generate an optimised version of the policy by using the optimisation techniques outlined.

Liu *et al.* (2011, 2008) propose an approach to convert XACML policies and requests from their verbose native textual (XML) format to a concise numerical format in a bid to increase policy evaluation performance. The authors contend that numerical policy comparison is more efficient than textual policy comparison and hence the conversion of policies alone leads to efficiency gains in the comparison process. The numerical policies are normalised and converted into a tree data structure for efficient evaluation by an associated PDP, called XEngine that contained the algorithms developed to specifically evaluate numerical policies. In a similar approach to Liu *et al.* (2011, 2008), Butler *et al.* (2011) analysed the performance characteristics of two PDP implementations and based on the analysis of that work proposed that the following features ultimately lead to efficiency gains in PDP evaluation performance. These features are 1.) policies and requests should be encoded more efficiently and 2.) PDP implementations should be more efficient. Leading on from these findings Griffin *et al.* (2012) propose a PDP prototype implementation named njsrPDP that uses non-blocking I/O techniques to efficiently evaluate polices (and requests) that have been transformed from native XML format into a new policy representation format called JSONPL that is a less verbose representation of policies which ultimately leads to increased performance in policy evaluation.

Marouf *et al.* (2011),Miseldine (2008),Liu *et al.* (2011),Griffin *et al.* (2012), all outline methods to increase the performance of XACML policy evaluation engines by mod-

ifying either the policies themselves or the PDPs. The approach taken in this thesis to increase policy evaluation performance does not require modification to either the policies or PDPs, but proposes to categorise the policies based on the application environment to increase performance. In particular, the authors in Liu *et al.* (2011) designed the XEngine that includes algorithms for the efficient evaluation of XACML policies. The authors provided a comparison against the Sun PDP implementation that demonstrated impressive efficiency gains, but the approach uses a numerical representation of polices which makes policy analysis tasks more difficult. Unfortunately, all request evaluation approaches previously mentioned sequentially evaluate all incoming access requests against deployed policies and therefore still attempt to brute force evaluate all access requests received.

2.2.6 PBM Application Areas

Policy based management techniques have been successfully applied in a number of application areas including both network and security management. Policies are specified for access control where the policy defines the actions to be taken when a request is made by a managed entity to access a resource or service in the network. In access control policies, the policy conditions indicate the circumstances under which a requesting entity is either permitted/denied access to a protected resource/service. Policy conditions are normally specified as user/environment attributes that are contained within an access request and matched against deployed access control policies and if the requesting entity meets the conditions, the corresponding actions are performed. Policy based management techniques provide secure access control over web services using languages such as XACML (Rissanen (2013)) and WS-Policy ((Bajaj *et al.*, 2006)). These languages are based on the Extensible Markup Language (XML) and follow well-defined standardised enforcement semantics that allow multiple vendors to implement their own enforcement processes. Policies are specified at different abstraction levels within a domain. For example, higher-level access control policies (e.g. XACML policies) are enforced by network attached servers dedicated to processing access control requests, whereas low-level filtering policies (e.g. iptables rules) are enforced by router interfaces usually deployed at various points throughout the network. These two policy models have different execution semantics, but may be related by protecting the same underlying resources/services of the network. With such a wide range of policy models

available that are tailored towards particular application domains coupled with the different execution semantic of each of the policy models, the consistent specification and enforcement of policies becomes a much more difficult task.

The majority of the published state of the art literature concerning management of federated domains concentrates on aspects of maximising the performance of network domains controlled by a single organisation. However in the broader literature on computing systems, co-operation between autonomous systems has been addressed. An early example is the work of Hull *et al.* (2003), where the authors describe a framework to support federated policy management within a single administrative domain—the authors consider "federations" of multiple policy engines and argue that network administrators should be able to deploy a single policy rule set and have appropriate rules deployed on each policy engine in the federation. The authors present an algorithm whereby users can specify a single coherent rule set expressing their preferences with this rule set mapped to multiple rule sets, one for each PEP in the application. The approach is not concerned with federation of resources and services across multiple domains, but uses a single policy engine that is federated within a single domain to control a single application. This view of federation does not correspond to the definition of federation in this thesis; it is closer in conception to the policy continuum in which policies specified at a high level of abstraction can be refined for deployment on groups of policy engines. Machiraju *et al.* (2003) present an architecture, object model, components, and protocols of a management overlay for federated service management, called Web Services Management Network (WSMN). WSMN is concerned with managing the Service level Agreement (SLA) for web services. In this management overlay proxies for web services coordinate to manage service-level agreements that must be adhered to. However, the approach does not address interaction of the management overlays with the management systems of the hosting environments for individual web services. It uses a proxy between a service and the Internet to control the SLA of that service using agreed upon protocols. The approach taken is not necessarily concerned with federating the actual services it controls between service providers. Bhatti *et al.* (2006) describe X-FEDERATE, a framework for access control management within federations. The approach taken is to specify an XML-based policy specification language and an enforcement architecture that constitutes an extension of the well-known RBAC model.

The authors work concentrates on security management aspects and again does not consider the issue of maintaining consistency between federation-level and local policies.

Enterprise Social Networks (ESN) (Cisco (2013), Microsoft (2013), IBM (2013)) are gaining in popularity as they are viewed as an effective tool by enterprises to leverage in a bid to increase communication, collaboration and productivity among employees and partners. Employees collaborate with their colleagues through communication (e.g. private messaging, etc.) and sharing/manipulating (i.e create, retrieve, update, delete) social network resources (i.e. documents, wikis, files, etc.) that require secure management using access control policies. Enterprise social networks try to mimic the openness (i.e social utility) of public social networks in facilitating easier information access, exchange and open communication flows, but operate within business processes that have strict security and privacy concerns. Public social networks operate under a relatively open ethos with regard to users privacy and security in order to more easily connect individuals and groups. However, in certain enterprise social network environments, users and resources operate under business processes that in some sectors such as health care, finance or government require very strict privacy and security regulation. Hence secure, but efficient management of enterprise social networks in these sectors is required. For example, in the financial sector, communication between a financial institution's officials and hedge fund investors should be monitored and policed to avoid misappropriation of the financial institution's funds. These security and privacy concerns as can be termed as safety concerns. These safety concerns can be alleviated through the consistent authoring (specification and conflict analysis) and enforcement of access control policies using the federation policy authoring process outlined in this thesis to control social network users communication/access to or manipulation of enterprise social network resources.

Typical use of enterprise social networks generates a large number of access requests by users/resources for permission to interact with other users/resources of the social network. Unfortunately, an enterprise's policy systems have limited resources to cope efficiently with large numbers of access requests and cannot efficiently evaluate all access requests received in a timely manner, as a consequence they either become a bottleneck for traffic in the social network or worse they inconsistently evaluate access requests. The reason it is not possible to perform evaluation on all access requests received by a policy evaluation system in large applications with many users and resources (which is

typical of enterprise social networks) in a realistic time frame is due to the complexity of rules nested within policies and the brute force method of request evaluation used by most request evaluation engines (Liu *et al.* (2011)). An additional point to note is when evaluating access requests in a social networking environment there is a delay constraint on communication as it should remain highly responsive and not hampered by adding excessive delays through policy evaluation of access requests (Marouf *et al.* (2011)). As a solution what is required is a method to optimise access request evaluation by tailoring expensive request evaluation towards specific users/groups of the social network deemed to pose a greater security risk. The approach should harness analyses of the social network to categorise the users/groups and access control policies applicable to those users/groups to increase access request evaluation performance while maintaining adequate security and privacy levels.

There have been a number of approaches to applying security controls in social networks based on the use of relations between users. Aleman-Meza *et al.* (2006) propose a framework to integrate two representative social networks and detect conflict of interest relationships between reviewers and authors of scientific papers. Central to this approach is the identification of relationship strength, based on the number of paper co-authorship between individuals to define a threshold value for conflict of interest. Carminati *et al.* (2006) propose an access control model that uses a combination of type of relationship, depth-level (i.e. distance between two social network users and trust level (manually specified value of how one users trusts another user) and trust-level to determine an aggregate relationship strength between two social network users. According to Carminati *et al.* (2009) there are five categories of social network data that can be modelled using semantic web technologies. These are: (1) personal information; (2) personal relationships; (3) social network resources; (4) relationships between users and resources; (5) actions that can be performed in a social network. The FOAF ontology (Brickley & Miller (2010)) for example includes the categories of knowledge referred to by Carminati *et al.* (2009) and can easily be modelled and extended (if required) to model additional social network semantics pertaining to users, groups, resources and their complex relationships representative in social networks. Kruk *et al.* (2006) analyse social networks to provide community driven access rights delegation. The authors use degree of separation combined with a computed friendship level metric to determine delegation of access control rights for social network resources. All of the approaches

mentioned are not concerned with the actual evaluation of access requests, but how to specify access control rules over social network resources and/or detect inconsistencies over policies specified against social network users and resources.

Enterprises social networks can be federated between enterprises where federation policies are used to specify the overall behaviour of federated enterprise social networks using federation policy authoring techniques such as federation policy specification and conflict analysis and access request evaluation. However, enterprise social network use cases need to be studied in more detail to understand the effects of applying the federation policy authoring techniques outlined in this thesis to set-up and maintain federations of enterprise social networks.

2.3 Supporting Technologies

This section discusses technologies such as information models for modelling aspects of managed domains that can be harnessed by federation policy specification processes and augmented with semantic web technologies (i.e ontological models, semantic web queries, etc) to provide support for consistency analysis processes.

2.3.1 Information Models

According to Westerinen *et al.* (2001) "an information model is an abstraction and representation of the entities in a managed environment, their properties, attributes and operations, and the way that they relate to each other. It is independent of any specific repository, software usage, protocol, or platform". Information models are used to provide a high-level graphical representation of the managed entities (i.e. devices, services, etc.) that constitute a managed network, but information model representations are limited in how they represent relationships between managed entities.

The CIM was developed by the DMTF (Lamers *et al.* (2010)), as an information model to aid in the integrated management of distributed systems. The CIM uses a graphical and textual notation to describe the information model. The graphical notation is based on the UML, but is more correctly referred to as the Managed Object Format(MOF) and is used to illustrate managed objects and relationships between those managed objects. CIM is structured into three distinct layers: core model, common model, extension schemas. The core model is an information model that applies to

all areas of management. The core model is a small set of classes, associations, and properties for analysing and describing managed systems. The common model is a basic set of classes that define various technology-independent areas, such as systems, applications, networks, and devices. The extension schemas are technology-specific extensions to the common model.

The Directory Enabled Networks (DEN) initiative (Strassner (1999)), enhanced the 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. In order to reduce the complexity associated with OSS management systems, the New Generation Operations Systems and Software (NGOSS) was developed by the TeleManagement Forum (TMF) (Strassner *et al.* (2004)). 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 TMF focused on creating a conceptual or analysis model based on a standardised version of the DEN-ng information model. The DEN-ng was developed by Strassner (2003) and was submitted and adopted by the TMF and renamed the SID. The Shared Information and Data (SID) model (Faurer *et al.* (2004)) is an analysis model which is focused on representing real world objects which are of interest to Business. An analysis model includes things in which the business is interested (domain entities), how they are related to one another (associations), and key details about those things which help to define them unambiguously (domain-level-attributes). Using analysis techniques can provide a more detailed understanding of business concepts and aid in defining business processes more precisely.

The SID and DEN-ng have diverged in recent years with the DEN-ng focusing more in the area of communications network management, and the SID has focused on the business oriented aspects of management. The DEN-ng is an object-oriented information model that can be used to represent the business and system entities of a network. Unlike the CIM model, the DEN-ng model can represent the different layers of management from the business layer down to implementation layer. The main advantage DEN-ng has over other information models is that it defines a management methodology

that dictates how the management of a large scale communications network should be performed and did not just model the structure of management information. DEN-ng representations are independent of any specific type of repository, software usage, or access protocol which makes DEN-ng suitable for describing how different management information is related to each other and could be used to build management representations and solutions. More specifically, because the business and system entities are represented in generic form in the information model; they can be translated quite easily to platform-specific implementations. The DEN-ng is fully UML compliant and so can be used to automate the generation of management interfaces (via Model Driven Architecture(MDA)/MDD)) to the services and resources of the network, and therefore can be made compatible with all existing management technologies.

The DEN-ng information model was designed with a policy information model built in. The justification for this is that policy could be related to arbitrary layers of management from business managed entities right down to system managed entities. The DEN-ng information model can be easily extended to model a wide range of application domains including network management and security management. One of the main differences between the DEN-ng policy model and the IETF PCIM policy model is that DEN-ng specifies the use of policy events. With the inclusion of a policy event, the semantics of a policy can now be read as "On the occurrence of the Policy Events, if the Policy Conditions are satisfied, then execute the associated Policy Actions." The addition of a policy event means that the enforcement of policy is now more extensible to other policy applications as the policy conditions are only evaluated after the occurrence of a specific event and not continuously evaluated by using a condition-action policy semantics as specified in the PCIM model.

Another distinguishing feature between the DEN-ng policy model and the IETF PCIM is that DEN-ng includes a policy management application hierarchy. The DEN-ng policy model differentiates between using policy to manage applications and management applications used to manage and monitor the execution of policy. Not only does the DEN-ng policy model represent the structure of policy, but it also specifies how policy should be integrated into various policy application domains. The DEN-ng model specifies 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 a PDP defined within the IETF PCIM.

The DEN-ng information model is designed to model the structure of managed entities from the business layer down to lower layers (system layer, network layer) within a managed system. The added advantage is that it is possible to realise policy derived from business goals and objectives to configure the behaviour of the communications network and services. Strassner (2003) addresses the concept of a *Policy Continuum*, in which policies at different levels of abstraction are linked in order to allow high-level goals expressed using business concepts (specified in terms of entities such as products, services and customers, etc.) to be translated through a number of levels of abstraction into corresponding device instance configurations in a single domain environment (e.g. CLI-based configuration of router traffic shaping and queuing). Davy *et al.* (2007) formalise the policy continuum in a model that can be utilised to assist policy consistency analysis and policy refinement processes. Implementation of the policy continuum enables different constituencies of policy authors, who understand different concepts and use different terminologies, to manipulate sets of policy representations at a view appropriate to them, and to have those view-specific representations mapped to equivalent representations at views appropriate for manipulation by other constituencies of policy authors.

Davy & Jennings (2007) make use of a shared information model in order to link the constructs of a policy language to the entities of an information model. Knowledge such as relationships, associations, constraint information and behavioural specifications are obtained from finite state machines. Semantic information is expressed via ontologies to augment policy analysis. This enables the definition of an enhanced context model that, while constructed as an information model, has been specifically designed to be able to generate ontologies for governing behaviour. Latré *et al.* (2010) introduce extensions to the DEN-ng information model that models how distributed entities consume remote contextual data to optimise the Quality of Experience (QoE) of services. The authors' work is similar to the approach taken in this thesis in that they use an algorithm to semi-automatically generate the required filter rules from information contained in the DEN-ng information model. The algorithm derives a baseline ontology from the DEN-ng

information model, and defines semantic relationships that achieve a higher expressiveness. However, the approach differs in that the authors introduce extensions to the DEN-ng information model to model how distributed entities use remote contextual data to optimise QoE services.

Information models provide a common means of representing different types of network entities. Most enterprises use heterogeneous management applications to manage their resources and services. These applications quite often have their own representation of data which could be shared with other applications if there was a common manner of representing it. However, while policy applications are representing data in formats that only they can interpret they will not be able to share the data with other policy applications. As an added consequence of this, each policy application will need to specify the same (redundant) data in a different format which can lead to redundancy and consistency issues. Information models can provide a solution to this problem as a single unified information model can relate the differences in data model implementations to each other. Unfortunately, the most inherent problem with information models is their lack of mechanisms for representing semantic information. This semantic information is of the utmost importance in aiding consistent policy specification and policy conflict analysis processes. Fortunately, 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. Ontological models can be harnessed to provide semantic information from various policy application domains to assist policy specification and conflict analysis processes (Wong *et al.* (2005)).

2.3.2 Semantic Web Technologies

An ontological model as defined by Uschold & Gruninger (1996) is a "shared understanding of some domain of interest and can be thought of as a conceptualisation of some world view". This conceptualisation consists of a set of concepts (e.g. entities, attributes, processes), their definitions and their inter-relationships. Strassner (2007) extends the definition of ontologies for network and system administration as a particular type of ontology whose subject domain is constrained to the administration of networks and systems. Administration is defined as the set of management functions required to create, set up, monitor, adjust, tear down, and keep the network or system

operational. The use of ontologies allow different parties to share their representation of how they perceive certain information describing concepts from their domains and also allow interpretation of related concepts from external parties in a bid to semantically match related concepts. For example, certain aspects of a router can be described by a particular party by the number of available interfaces while another party may describe the router from its specifications. Each party can describe the router using an ontology and then by sharing their ontologies each party can reason over the other's ontology and in essence they will be describing the same router just from different aspects. This type of semantic reasoning essential when trying to relate different data models or programming models together. DLs can be used to formally represent candidate federation and deployed local policies in order for semantic web queries to be executed over instances of the ontological models to identify possible occurrences of policy inconsistencies.

Ontologies can be specified in many different formalisms such as predicate calculus, frame logic (Gruber *et al.* (1993)); however, a more widely accepted method for specifying ontologies is using description logics. Description Logic (DL) as defined by Baader *et al.* (2003) is the name for a family of knowledge representation formalisms that represent the knowledge of an application domain (the "world") by firstly defining the relevant concepts of the domain (its terminology), and then using these concepts to specify properties of objects and individuals occurring in the domain (the world description). DLs are based on a formal decidable fragment of First Order Logic (FOL) where basic descriptions are atomic concepts and atomic roles and more complex descriptions can be built from them inductively using concept constructors and role constructors. This allows automated reasoning to be performed over the formalisms with logic-based semantics that focus on reasoning as a central service. Reasoning is essential to ontological models as it can infer implicitly represented knowledge from knowledge that is explicitly contained in a knowledge base.

Nardi & Brachman (2003), focus on the relationship between DL, semantic and frame systems. The authors identify problems with older semantic and frame systems and introduce basic features of DL languages and related reasoning techniques. The authors content that DL can automatically deduce implicit knowledge from the explicitly represented knowledge, and always yield a correct answer in finite time. The TBox, see Figure 2.4, introduces the terminology, i.e., the vocabulary of an application domain, while the ABox contains assertions about named individuals in terms of this

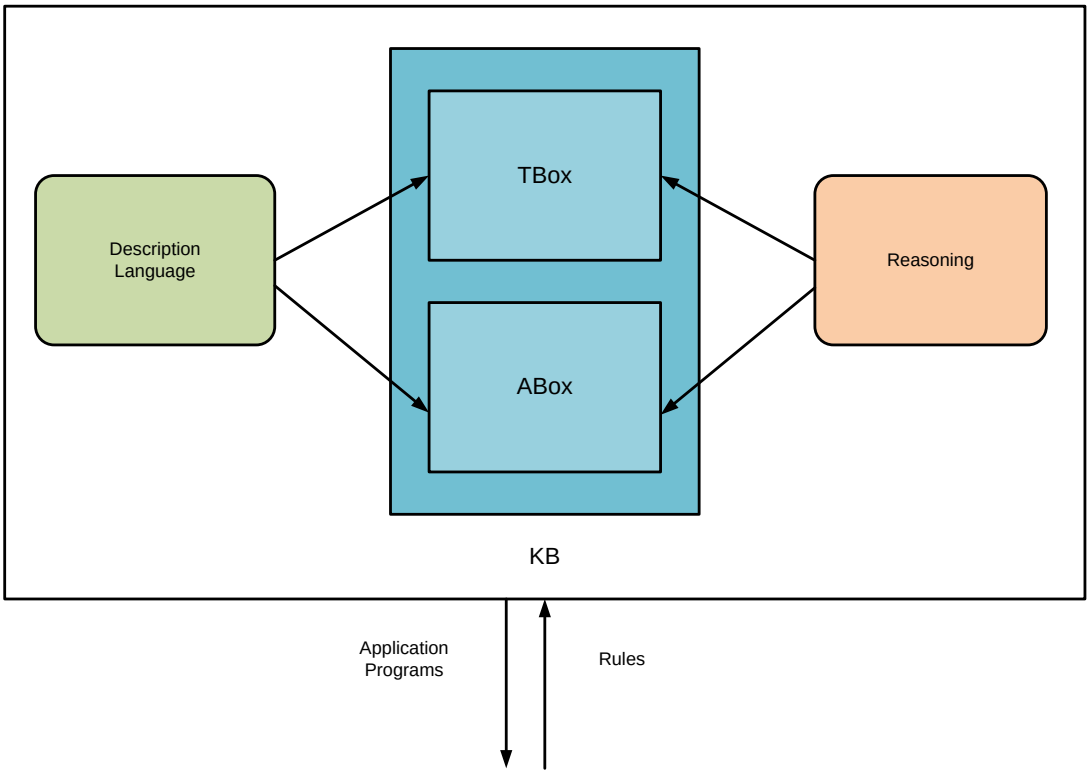

Figure 2.4: This figure illustrates a typical KR Architecture. A knowledge base (KB) comprises two components, the TBox and the ABox specified using a logical description language.

vocabulary. DLs provide a semantic modelling approach to defining federation concepts (hierarchically), their relationships and the policies specified over them in the TBox. This enables looping tree automata reasoning to be used for policy consistency analysis over the ABox. With regards to the closed-world verses open-world semantics of semantic web technologies, DL knowledge bases are often compared to database systems where the schema of the database is compared to the TBox and the database instance is compared to the ABox. However, the semantics of ABoxes differs from the semantics of databases in that databases follow the closed-world semantics where absence of information in a database instance is interpreted as negative information (i.e it does not exist). While the ABox follows the open-world semantics where absence of information in an ABox only indicates lack of knowledge in that it has not yet been proven to exist (i.e unless it is explicitly stated in the KB not to exist).

There has been a rapid increase in the use of semantic web technologies to annotate web pages, documents and web services on the web with semantic information in a bid to help machines and people better understand the contents of these valuable information sources. There is a family of well-known related technologies being developed by the World Wide Web Consortium (W3C) namely: RDF, RDF Schema (RDFS), Web Ontology Language (OWL), SPARQL Protocol and RDF Query Language (SPARQL) 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 (Beckett & McBride (2004)) is a language standard for representing information about resources in the web. An RDF triple t contains three elements, a subject $\{s\}$, a predicate $\{p\}$, and an object $\{o\}$, these elements are used to form a tuple $t\{s,p,o\}$. The tuple represents a property that holds between the subject and object where the property is either a data type or object property. Data type properties describe the attributes (or characteristics) that form a concept while object properties refer to relationships that hold between concepts (object properties basically tie concepts together through semantic type relationship connections) in a knowledge model. The semantic web queries are executed over the properties of concepts specified in the knowledge model and more specifically over sets of triples to form a basic graph pattern. RDF is good at representing simple relationships associated to documents, but is too limited for certain applications (Baader *et al.* (2003)). 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).

OWL (Smith *et al.* (2004)) developed by the W3C Web Ontology Working Group has been designed to provide a language that can be used to describe the classes and relations between them that are inherent in web documents and applications. OWL is by far the most popular technology for representing semantic information in ontological format. OWL can be used to formalise a domain by defining classes and the properties of those classes. It is also used to define individuals and assert properties about the individuals. OWL can be employed to reason over classes and individuals, where the level of reasoning depends on the flavour of the language chosen. A version of OWL, named OWL-DL is focused on using the notions from description logic to describe semantic information. OWL exploits and extends the capabilities of RDF and RDFS (Horrocks *et al.* (2005)). The OWL language refers to three sub-languages (OWL Lite,

OWL DL, and OWL Full) with differing level of expressivity and reasoning. OWL Lite is the least expressive, offering support for classification hierarchy and simple constraint features. OWL DL is more expressive than OWL Lite supporting type separation where a class cannot be an individual or property and vice versa. OWL DL offers computational completeness and decidability with regard to reasoning and is guaranteed to finish in finite time. OWL Full provides maximum expressiveness, but does not guarantee to complete reasoning in finite time. OWL Lite and OWL-DL correspond to SHIF(D) and SHOIN(D) respectively (Horrocks (2005)). SHIF(D) and SHOIN(D) are DL languages and, hence, OWL-Lite and OWL-DL may be conceptualised as mark-up language based syntax for DL languages. Aligning OWL-Lite and OWL-DL with DL languages gives their developers, the developers of OWL tools and users of OWL a greater understanding of their computability and complexity.

OWL 2 is an extension of OWL and has two different forms of semantics, namely RDF-Based semantics and Direct Semantics (Motik *et al.* (2009)). OWL 2 Full which is based on the OWL2 RDF-Based Semantics. The RDF-Based semantics is an extension of the semantics for RDFS and is based on viewing OWL2 ontologies as RDF graphs. OWL 2 Full is undecidable, there are no reasoners available for OWL 2 Full under the RDF-Based semantics. OWL 2 DL is based on the OWL 2 Direct Semantics. The direct model-theoretic semantics allows for the representation of OWL 2 in Description Logic style. OWL 2 DL can be viewed as a syntactically restricted version of OWL 2 Full that remains decidable. There are several production quality reasoners that cover the entire OWL 2 DL language under the direct model-theoretic semantics. OWL 2 defines three sub-languages or profiles. These profiles are OWL 2 EL, OWL 2 QL, and OWL 2 RL (Motik *et al.* (2009)). The OWL 2 EL profile enables polynomial time computability complexity for reasoning tasks, whereas the OWL 2 QL profile increases the efficiency of answering queries to relational databases, while the OWL 2 RL profile enables polynomial time computability complexity for reasoning tasks using rule-extended database technologies. Due to the expressiveness of the OWL 2 language each of these subsets can be seen as providing further restrictions on the expressiveness of the OWL 2 language which is sufficient for a variety of applications. OWL 2 DL can be seen as a syntactic fragment of OWL 2 Full and OWL 2 QL is a syntactic fragment of OWL 2 DL. None of the profiles is a subset of another.

OWL tools designed specifically for the specification and instantiation of ontological models include Protege, Swoop, OilEd, and TopBraid Composer DL. OWL concepts can be translated into an expressive DL where an inference engine (i.e Pellet, HermiT) can be used to reason over the asserted concepts in the Abox. However, OWL is restricted to using XML-based datatypes, where the constructors and axioms of OWL can be translated into SHIQ (Horrocks *et al.* (2003)). DL Reasoners include: RacerPro, FaCT++, Jena, KAON2, Pellet, HermiT. A DL reasoner provides a set of DL inference services, such as:

1. Consistency checking: ensuring that a knowledge model does not contain any contradictory facts.

2. Classification: computing the classes' subclass relations to create a complete class hierarchy.

3. Realisation: finding the most specific classes that an individual is a member of.

SWRL (Horrocks *et al.* (2004)) is based on a combination of the OWL DL and OWL Lite sublanguages of the rule markup language (RML) along with the Unary/Binary Datalog RuleML sublanguages of the RML. It extends the set of OWL axioms to include Hornlike rules. This allows Horn-like rules to be combined with an OWL knowledge base. SWRL includes a high-level abstract syntax that extends the OWL abstract syntax. SWRL rules are of the form of an implication between an antecedent (body) and consequent (head) with the intended meaning of whenever the conditions specified in the antecedent hold, then the conditions specified in the consequent must also hold. SPARQL is essentially a graph-matching query language that can be used to query RDF graphs and return the queried data (Prud *et al.* (2008)). Given a particular data source D, a query is comprised of a pattern which is matched against D, and the resulting values obtained from this pattern matching are processed to give the answer. The queried data source D can consist of multiple sources. A SPARQL query is comprised of three parts. The pattern matching part, which includes several interesting features of pattern matching of graphs, like optional parts, union of patterns, nesting, filtering (or restricting) values of possible matchings, and the possibility of choosing the data source to be matched by a pattern. The solution modifiers, which once the output of the pattern is ready (in the form of a table of values of variables), allows to

modify these values applying classical operators like projection, distinct, order, limit, and offset. Finally, the result of a SPARQL query can be of different types, namely: yes/no queries, selections of values of the variables which match the patterns, construction of new triples from these values, and descriptions about resources queries (Pérez *et al.* (2006)). Semantic web technologies provide powerful modelling and reasoning tools such as ontology languages (OWL-DL, OWL-Lite, etc.) and semantic rule languages (SWRL, SPARQL, etc.) that the approach outlined in this thesis can harness to represent and reason over an enterprise's complete social network to augment policy consistency analysis processes.

2.4 Summary and Conclusions

This chapter reviewed state of the art work related to policy based management and in particular policy authoring which includes policy specification, conflict analysis, policy refinement, and policy deployment. As a means to realise federations, service providers can leverage PBM techniques to assist in the authoring and deployment of federation policies. Unfortunately, PBM systems are heterogeneous by nature and as a result use policies that may be specified in different policy languages which is problematic for managed domains participating in a federation as this heterogeneity makes it even more difficult to specify consistent policies as they may be related to different processes/managed entities across independent domains. It may also be the case that policies are enforced at various abstraction levels within each managed domain, (i.e. system, network, device, etc.) and as a consequence the specification of policies at arbitrary levels has an adverse affect on the enforcement of policies as it becomes much more difficult to maintain the consistency of policies at any one level. The above challenges motivate the need to develop policy based management techniques such as policy authoring that includes policy specification and consistency analysis processes tailored specifically towards federation policy authoring spanning multiple independently managed domains. The state of the art in the area have been designed to operate within a single domain environment and would prove difficult to extend to multiple domain environments. Also a lot of these works are only targeted towards to single layer within an managed domain (system, network, etc.). However, these processes would need to

operate over multiple arbitrary layers to cater for federations. There is strong motivation for development of federation policy specification processes specifically targeted towards consistent specification and deployment of polices across a federation, by each federation participant. Federation policy consistency analysis processes will also need to be developed to operate across multiple domains to aid in the consistent analysis and refinement of federation policies realised as local policies which will ultimately lead to less human control in the system.

Question 1 states *"How can a policy authoring process be defined to support the consistent specification of federation-level policies by multiple constituencies of policy authors at arbitrary abstraction levels aimed at controlling federations of services and resources?"*. The requirements for this question indicate that any federation policy authoring process be able to represent and leverage federation-specific information (i.e federation context, federation context data, etc.) pertaining to the federation agreement to assist in consistent federation policy specification. The policy authoring approach outlined in this thesis follows on from the concepts of *Federation* as presented by Jennings *et al.* (2009). The authors of that work discuss how federation typically occurs at many levels between organisations, where legal agreements can describe specific obligations on each member of a federation. This may result in policies being deployed throughout each organisation. The approach outlined in this thesis is useful in this context in that the federation agreements can represent business policies at an abstract level which can be automatically deployed within an organisation using the framework. The use of ontologies has also been highlighted as a useful approach to aid in the automation of federations between organisations (O'Sullivan *et al.* (2009)). With regards to policy specification, related work in this area has been concerned with policy specification within single domain environments. As part of the federation authoring process, policy specification will need to take place across multiple domains. A fundamental problem with specifying policies for multiple domains is using a language that is generic enough to be used with a large number of heterogeneous applications. Two technologies that have been presented to solve this situation include using UML models augmented with ontologies to specify policies. UML models allow for the generic representation of general policy concepts that can be extended with application-specific information as required and then transformed into an ontological format that can be shared with other federation members thereby facilitating policy information exchange. UML models can

be used to represent both the managed domain entities and the policies specified over those managed entities to achieve the desired behaviour of a managed system. This information can then be leveraged to augment policy specification and conflict analysis processes which results in more powerful policy authoring processes. Unfortunately, the UML models presented in this chapter lack certain modelling concepts required to represent federation aspects that cross organisational boundaries (i.e. federated domains, federated context, federated contextdata, etc.). This UML concepts are crucial for modelling federations and as a result any UML model selected to represent federations will need to be extended with these federation concepts accordingly. One disadvantage of UML models is that when multiple administrative domains need to distribute the management (i.e policies) of the communications network each administrative domain must agree on a common information model to leverage. Ontological models on the other hand offer support for the integration of policies where concepts specified in one ontological model can be mapped to concepts in another ontological model thereby facilitating policy exchange. Ontological models represent concepts and rich semantic relationships represented in logical form to describe a target managed system that can be reasoned over to discover implicit knowledge from a domain model.

Question 2 states *"What conflict analysis algorithms need to be developed to assess the consistency of candidate federation-level and local policies when policies are created, modified or withdrawn?"*. The requirement on this question indicates that during refinement of federation policies consistency checks between deployed local policies and the candidate federation policy will need to be performed. Candidate federation policies have the potential to conflict with previously deployed local policies within the domains of service providers participating in a federation. Some of the challenges encountered with applying policy consistency analysis techniques within federations come from the heterogeneous nature of the PBM systems that comprise the federation. More specifically, potential inconsistencies can occur at arbitrary levels of abstraction within managed domains. As a consequence the detection of potential inconsistencies requires retrieval of information specific to the federation/local operating environment and dependencies that hold between policies at arbitrary abstraction levels within a network management domain. Policy conflict analysis processes will need to leverage some form of UML and ontological modelling and reasoning processes to aid federation policy specification and detection of inconsistencies between federation and local deployed

policies. Unfortunately, the policy conflict analysis approaches outlined in the chapter are specifically tailored to operate within single managed domain environments and are not suitable for operation across organisational boundaries which is a fundamental requirement for supporting federations. Also the policy consistency analysis approaches mentioned in this chapter only operate at specific levels of a managed domain, whereas for applicability in federations policy consistency analysis approaches would need to operate over multiple layers of a managed domain. The proposed federation policy authoring process encompasses a policy specification process for the consistent refinement of low-level implementable policies from a high-level candidate federation policy based on model driven development techniques to cater for the multiple levels of policy authoring within the managed domains of federation members. The consistency analysis processes leverage semantic web technologies such as ontological models to model both the managed domain and its associated policies and semantic web queries to reason over these ontological representations to detect possibly policy inconsistencies.

Question 3 states *"What strategies can federating enterprises adopt to increase access request evaluation performance while maintaining an acceptable level of risk?"*. The requirement on this question indicates that federated enterprise social networks have limited policy processing capabilities (CPU, RAM, etc.), so they can become a bottleneck for network traffic when the access request evaluation rate is high which is typical of standard social network behaviour. When evaluating access control policies in a social network environment there is a delay constraint on all communication because near real-time communication should be highly responsive and not hampered by the excessive overhead caused by the policy evaluation system. Marouf *et al.* (2011),Miseldine (2008),Liu *et al.* (2011),Griffin *et al.* (2012), all outline methods to increase the performance of XACML policy evaluation engines by modifying either the policies themselves or the PDPs which is not a feasible or extensible solution when federating enterprise social networks. The extensible approach taken in this thesis to increase policy evaluation performance does not require modification to either the policies or PDPs, but proposes to categorise the social network and its associated policies based on the application environment to increase request evaluation performance. The approach taken can be easily applied by each federation member by analysing its own social network and policies.

Chapter 3

Federation Policy Authoring

Market pressures are pushing service providers toward greater federation of their networks with each other and with their customers' networks. Whilst in the past such federations were limited in scope, statically defined and long lasting the current trend is towards a much more dynamic environment in which federations will be formed for different purposes, will be much more numerous and may be relatively short lived and have changing membership profiles. To support this service providers require flexible tools such as policy based management techniques to help them manage their networks in a more automated manner cognisant of their commitments to federations in which they participate. As an example, resource access control policies for an organisation are often derived from best practice standards or from high level business policies. To ensure that access control is enforced effectively, these business policies need to be translated into deployable system configurations or lower level policies for multiple diverse systems, for example file based access controllers or firewalls. These target policy representations require experts to coordinate and collaborate so that business policies are fully supported. It is difficult and cumbersome to effectively ensure that all access control policies are enforced with the desired effect and in a consistent way, particularly given that there may be many people editing policies and that the business policies can change over time.

This chapter presents a model driven approach that abstracts access control policies into a clear and structured set of rules defined using terms familiar to a non-systems expert, which may then be realised into multiple levels of abstraction. The proof of concept system provided uses MDA techniques to transform high level business policies

into device specific policies that can be enforced by multiple access control system types. The presented scenario examines the application of access control to instant messaging communications and network server access, two systems with different access control configuration languages. A policy authoring process is described in this chapter specifically tailored towards the consistent specification of low-level enforceable policy specifications from high-level federation policies. To achieve this, the DEN-ng domain model has been extended to include support for modelling federated domains. The extended DEN-ng federated domain model can represent context data pertaining to federated and local domain environments. This context data is crucial as input to any policy specification or policy consistency analysis processes. Extensions to the policy continuum model have also been provided to cater for federation policies, that enables experts responsible for authoring federation policies, a way to identify if their new or modified policies potentially conflict with current deployed policies, or if existing federation agreements need to be re-negotiated.

To aid policy consistency analysis the DEN-ng federated domain model has been transformed into an OWL-DL representation that allows managed domain entities and the semantic relationships that hold between them to be modelled and reasoned over using semantic web technologies. Specifically, ontological models have been generated based on the DEN-ng federated domain model and DEN-ng policy model to model aspects of federated domains and the policies specified over the federated domains. This allows semantic reasoning to be performed over instances of the ontological models to detect implicit relationships that could indicate possible occurrences of policy inconsistencies. The federation policy authoring process described in this chapter is comprised of three phases. The first phase involves the transformation of relevant aspects of the DEN-ng information model into the required ontological models. The second phase involves the execution of semantic web queries for the retrieval of currently deployed local policies that are related over a policy's elements to the candidate federation policy. The third phase invokes a refinement process to transform the policies into lower-level enforceable policy specifications. This chapter also describes the development and implementation of a federation test-bed specifically designed to evaluate the newly developed processes and algorithms outlined in this thesis.

The rest of this chapter is structured as follows, Section 3.1 describes extensions to the DEN-ng domain model to support the modelling of federated domains. Section 3.2

outlines extensions to the policy continuum model to support mapping between policies for policy specification and conflict analysis of policies at multiple abstraction levels in a federated environment. In Section 3.3 the DEN-ng federated domain model has been transformed into an OWL-DL representation that allows for additional domain specific semantics (managed entities, semantic relationships, etc.) to be added to the model. The OWL-DL model can then be queried and reasoned over to detect semantic relationships and potential inconsistencies among groups of deployed policies that can indicate cases of policy conflict. Section 3.4 describes the federation policy authoring process that includes verifying the policies in the policy continuum, consistency analysis, and refinement. Section 3.5 describes the implementation of a federation test-bed designed specifically to evaluate the algorithms and processes outlined in this thesis. Finally in Section 3.6, the contributions of this chapter are summarised and analysed.

3.1 DEN-ng Federated Domain Model

The original DEN-ng domain model as illustrated in Figure 3.1 addresses the representation of management domains; where a domain is defined as a group of entities (resources, services, devices, etc.) that share a common goal. They can be organised hierarchically or linearly and are uniquely addressable within their respective domains. A management domain adds three important behavioural features to a domain: (a) it defines a common set of administrators (user, group of users, and/or organisation(s)) to control its entities, (b) it defines a set of applications for various administration tasks (monitoring, configuration, etc.), (c) it defines a common set of management mechanisms, such as policy rules, that are used by the administration tasks. A domain has context data (i.e. time of day, environment settings, etc.) that can be harnessed when specifying policy rules. Note, that a more complete version of the DEN-ng information model can be found in Appendix B.

DEN-ng domains are containers whose elements are `ManagedEntities`. A domain is a collection of entities, sometimes hierarchically organised, that share a common purpose. In addition, each constituent entity in a domain is both uniquely addressable within that domain. A special type of domain, called a `ManagementDomain`, is defined which uses a set of `PolicyRules` to enforce the governance aspects defined for it and its constituent entities. A `ManagementDomain` refines the notion of a domain by adding

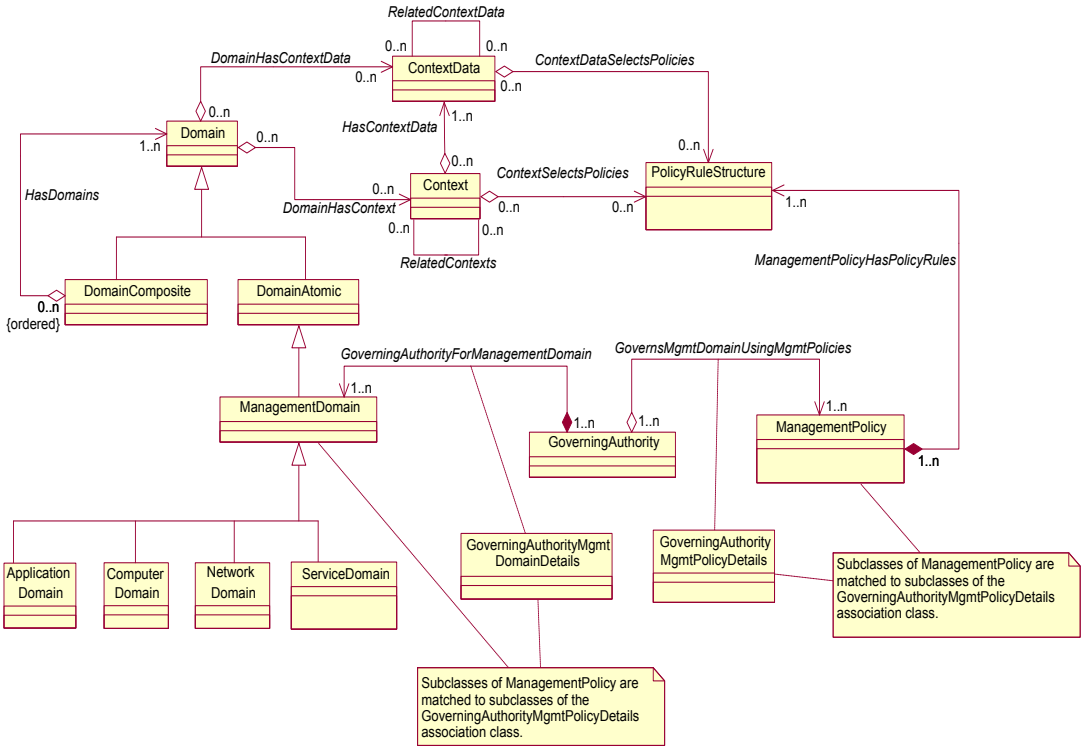

Figure 3.1: This figure illustrates a simplified view of the original DEN-ng domain model. The original model provides support for modelling various types of domains with their associated context and management policies.

important behavioural features. First, it defines a common set of administrators that govern the managed entities that it contains. In other words, all constituent managed entities in this `ManagementDomain` are administered by the same user, group of users, and/or organisation(s). Second, it defines a set of applications that are responsible for different governance operations, such as monitoring, configuration, and so forth. Third, it defines a common set of management mechanisms, such as policy rules, that are used by the management mechanisms. A DEN-ng domain can have zero or more contexts. In DEN-ng, `Context` is defined as an aggregate object containing different aspects (such as time, location, communication method, etc.), where each aspect is defined as a `ContextData` object. This enables `Context` and `ContextData` to be richly described.

Each `Context` selects a set of `PolicyRules` that are used to govern behaviour ap-

propriate for that `Context`. Hence, as the `Context` of a domain changes, the set of `PolicyRules` change in order to maintain the set of goals, business objectives, regulatory rules, and/or other governance constructs that is shared by each entity contained in the domain. A `GoverningAuthority` in the model of Figure 3.1 represents an individual or collection of `ManagedEntities` that are responsible for performing governance operations. A `GoverningAuthority` can be either human or non-human. The `GoverningAuthority` uses appropriate `ManagementPolicies` to govern both the `ManagedEntities` in the domain as well as the management of the domain itself. The `Context` of an entity is an aggregation of `ContextData` objects that describes the set of all interrelated conditions in which an entity exists. Events point out changing conditions that may affect that entity; an appropriate governance mechanism, such as policy rules, then defines a set of actions in response to the event(s) to change or maintain the state of the entity according to these conditions and actions. `Context` can have multiple distinct sets of related data and knowledge that are used to adjust its state in accordance with the changes in the environment that it exists in.

3.1.1 Extensions to DEN-ng Domain Model for Federations

The existing DEN-ng domain model needs to be extended to include concepts inherent in federations, so to this end, the DEN-ng domain model has been extended to include support for modelling federated domains as illustrated in Figure 3.2. `ContextData` can be obtained from each service provider domain to give an overall view of `ContextData` for a federation of service providers. This type of data can be used to assist in federation policy specification and consistency analysis processes. This DEN-ng extension defines four types of federated domains which are characterised according to the federation's governance structure and continuum of governance mechanisms. The continuum is constrained by autonomy on the one side and local and/or global rules which must be adhered to on the other side. This leads to a range of possible governance structures with the most fundamental ones being:

1. A single central authority that governs all other federation participants through the use of policy (i.e., single central authority, subservient domains, only global policy rules).

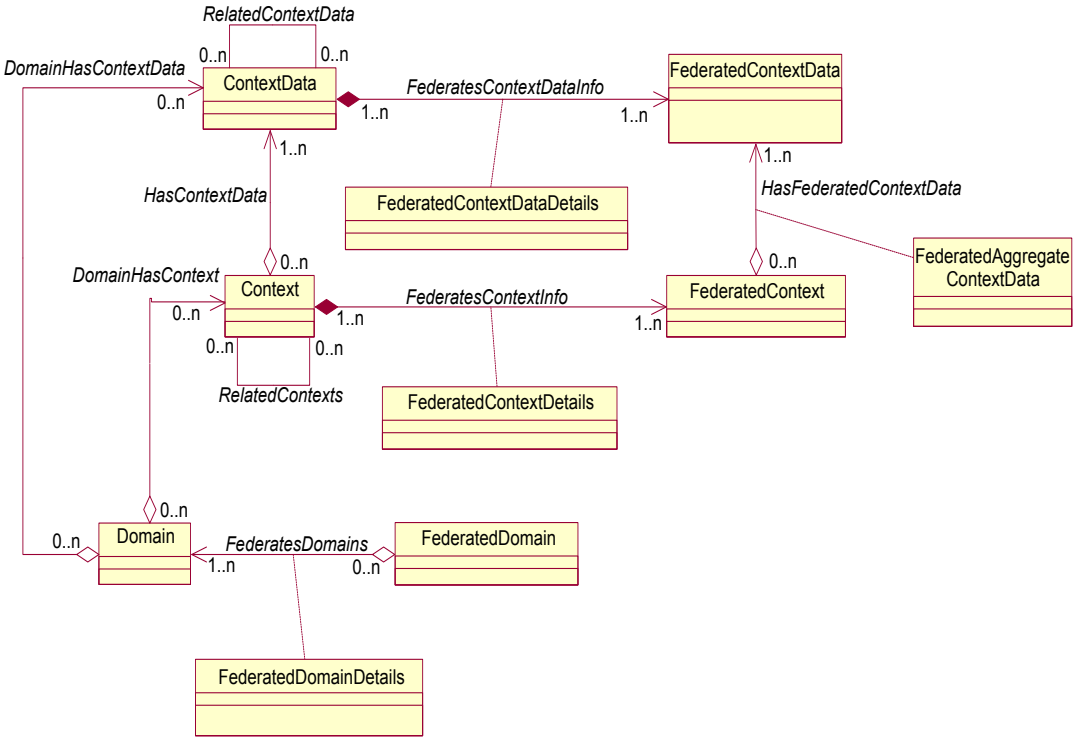

Figure 3.2: This figure illustrates the extensions made to the original DEN-ng domain model to provide support for modelling federations. The illustration shows the conceptual relationship between Context, ContextData, FederatedDomain, and Domain.

2. A single central authority that organises the actions of the other federation participants (i.e., single central authority, autonomous domains, local and global policy rules).

3. A distributed set of authorities that use agreed-upon mechanisms of governance to specify rules that each federation participant must abide by while remaining somewhat autonomous (i.e., distributed authority, autonomous domains, local and global policy rules).

4. A distributed set of authorities that dictate policy to be adhered to by subservient federation participants (i.e., distributed authority, subservient domains, global policy rules only).

The federation is itself a `ManagedEntity`, and is typically logically centralised, but physically distributed. However, DEN-ng allows for logical distribution as well. In a federation, if the governance model allows for autonomous or semi-autonomous constituent domains, then the self-governing status of those domains cannot be altered by the `FederatedDomain` that contains them. The basis of the federation may include social, political, geographical, and/or governance mechanisms that must be applied to all constituent domains in order to govern behaviour that is of mutual interest. This is represented by appropriate policy rules (along with state automata to orchestrate the behaviour represented by the policy rules). Note, however, that each constituent domain can act autonomously in other matters that are outside the governance provisions of the federation. The extensions to the DEN-ng domain model include the introduction of the `FederatedDomain`, `FederatedContext` and `FederatedContextData` classes.

The first extension is the introduction of a new concept to group domains, the `FederatedDomain` class. A `FederatedDomain` is an aggregation of (DEN-ng) domains; which is defined as a collection of domains (the actual organisation of domains in the federation can be linear or hierarchical as necessary) in which each domain in the federation agrees to use zero or more local policy rules to govern the operation of the `ManagedEntities` that they contain. The overall governance principles for the `FederatedDomain` are defined through policy rules, but are related to the current `Context` of the federation. The `FederatesDomains` aggregation defines not only the set of individual domains that are federated into this particular `FederatedDomain`, but also the type of governance structure applied to this federation. The `FederatedDomainDetails` association class implements the semantics of the `FederatesDomain` aggregation. This class serves as a container, whose attributes can be populated to suit the needs of the application(s) using this part of the model in order to restrict the types of policies that can be used by particular `FederatedDomains` in order to determine which domains can participate in a given federation. A typical example usage of the `FederatedDomain` class from the specified use case is *"FederatedDomain contains the service provider domains:* ***Domain A*** *&* ***Domain B*** *that have been federated to provide a service* ***S****"*.

A `FederatedDomain` would have a particular `Context`, and each of the constituent domains in the `FederatedDomain` would have corresponding `ContextData`, which corre-

sponds to the (local) context of the domain. Hence, each of the `ContextData` objects are part of the overall `Context` object, which means that each of the local contexts of each domain contributes to the overall `Context` of the federation. This allows the different aspects of `Context` (the `ContextData` entities) to signal that they have a local change that may or may not affect the overall `Context` (and vice-versa). However, it is likely that not all contextual information from each domain will be available for federation, due to privacy and other considerations. Hence the second extension to the DEN-ng domain model involves the introduction of a new concept, called the `FederatedContext` class. This class represents the set of contextual data that is allowed to be seen and used by the federation. A `FederatedContext` represents the overall aggregate contextual information for a `FederatedDomain`. It collects local `ContextData` from each local domain in the federation and then filters the contextual information according to a set of context-aware policy rules. This is realised using the `FederatedContextDetails` association class of the `FederatesContextInfo` composition, and enables privacy and other rules governing the usage of contextual information to be enforced. The `FederatesContextInfo` composition defines the set of `Context` information that is available to be used by a federation. The semantics of choosing the subset of `Context` information that is made available to the federation is defined by the `FederatedContextDetails` association class. The `FederatedContextDetails` association class defines the semantics of the `FederatesContextInfo` composition. This class is designed to be a container, whose attributes and relationships are populated by a set of external applications enabling which `Context` information can be used to produce a `FederatedContext`.

The `HasFederatedContextData` aggregation defines the set of individual `FederatedContextData` elements that collectively determine the overall `FederatedContext` of a federation. The semantics of this aggregation are represented by the `FederatedAggregateContextDetails` association class. This class is similar to the `FederatedContextDetails` in that it is designed to be a container, whose attributes and relationships are populated by a set of external applications; in addition, these attributes and relationships are then used to control the semantics of the aggregation, enabling which `Context` information can be used to produce an aggregated `FederatedContext`. An example of the use of the `FederatedContext` class is *"Domain A and Domain B are federated to collaborate on*

project X"

The third extension to the DEN-ng model includes the introduction of a new concept called the `FederatedContextData` class. This class represents the overall aggregate contextual information for an individual domain in a federation. Since each `ContextData` object represents a different aspect of contextual information, each `ContextData` object may have different visibility, access, and other rules that govern its usage. Hence, a `FederatedContextData` object collects local `ContextData` information from the local domain in the federation and then filters the contextual information according to a set of context-aware policy rules. This is realised using the `FederatedContextDataDetails` association class of the `FederatesContextDataInfo` composition, and enables privacy and other rules governing the usage of contextual information to be enforced. The `FederatesContextDataInfo` composition defines the set of `ContextData` that is available to be used by a federation. The semantics of choosing the subset of `ContextData` that is made available to the federation is defined by the `FederatedContextDataDetails` association class. The `FederatedContextDataDetails` association class defines the semantics of the `FederatesContextDataInfo` composition. This class is designed to be a container, whose attributes and relationships are populated by a set of external applications; these attributes and relationships are then used to control the semantics of the aggregation, enabling which `ContextData` information can be used to produce a `FederatedContextData` object. An example of the use of the `FederatedContextData` class is *"Group Z are permitted to instant message (IM) Group Y"*

Figure 3.3 above shows a customisable template for applying DEN-ng context-aware policy rules to manage the federation of contextual information. `ManagementPolicy` provides deontic rules independent of the structure of the rule; hence, the actual policy rule content can be expressed as event-condition-action, goal, and/or utility function policy rules. The `ManagementPolicy` class realises deontic actions (e.g., obligations, permissions, etc.) independent of the actual structure of the `PolicyRule` being used. The `PolicyRuleStructure` class is used to represent the structure of a policy rule. Supported rule types include CA (condition-action for backwards compatibility), ECA (event-condition-action, preferred over CA), Goal, and Utility policies. More formally, the purpose of this class is to define different subclasses that each formalise the semantics of different types of policy rules using a subsumption relationship. This enables a

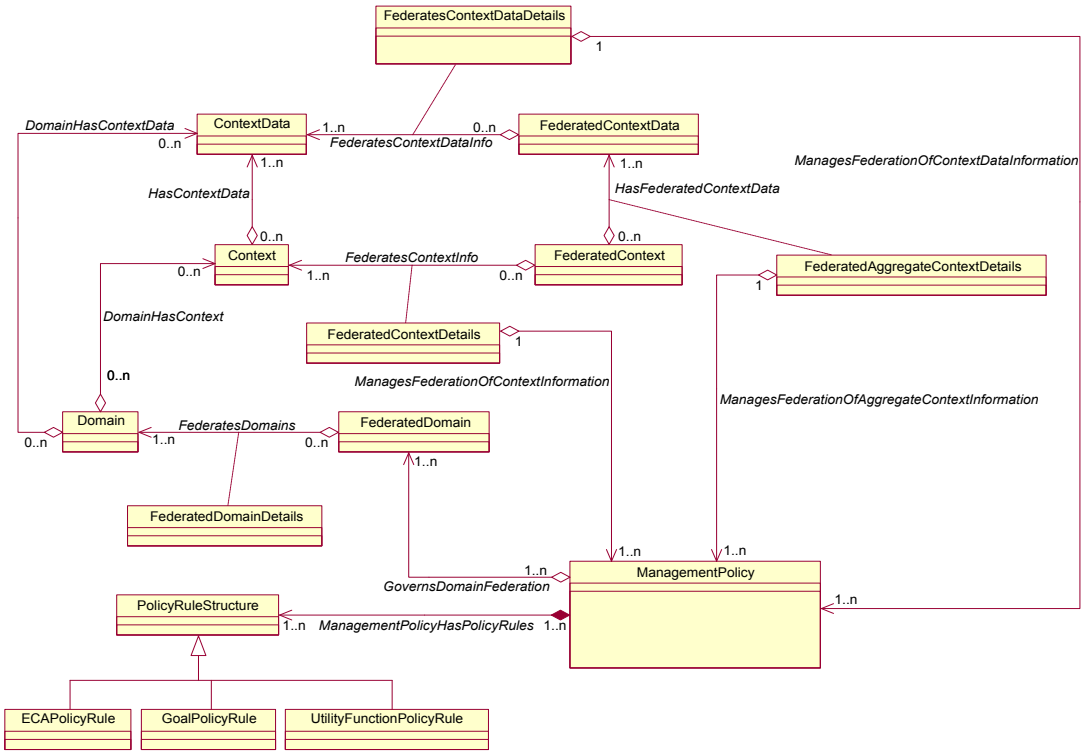

Figure 3.3: This figure illustrates a customizable template for applying DEN-ng context-aware policy rules to manage the federation of contextual information.

system (such as FOCALE (Strassner *et al.* (2006))) that uses DEN-ng to import different types of policy rules, each with their own specific structure, and represent how each is used. This provides extensibility, so that new policy rule types can be added without adversely affecting the overall design of the DEN-ng policy hierarchy. From an ontological perspective, it is important to separate the semantics of the structural representation of the policy rule from other concepts that are required to use the policy rule, such as policy target and policy subject. This enables particular policy types, for example `ManagementPolicy`, to add them as required. An example of a subclass that can be defined from `PolicyRuleStructure` is `ECAPolicyRule`, which formalises the semantics of a policy rule with an {Event, Condition, Action} structure. In essence, an `ECAPolicyRule` is a policy rule that has a policy event, a policy condition and a policy action.

3.2 Extensions to DEN-ng Policy Continuum Model for Federations

The policy continuum (Strassner (2003)) comprises a layered set of policy languages, each utilising terminology and syntax appropriate for a specific layer of policy author. The policy languages can be bound together using a common information model where the policies, the policy events, conditions, actions, subjects and targets are modelled and associated to each other. The use of a common information model outlines the scope of the views and the cross-layer relationships necessary to relate policies together defined by the different levels of policy authors. Each level of the policy continuum is maximised for the different layers of policy authors that require information of a specific type. A single policy within the policy continuum may reference a set of lower-level policies and/or may be associated to more than one higher-level policy. However, it is possible that a policy within the policy continuum may exist exclusively at a single level (e.g. device configuration or policy deployed onto a PDP at the bottom level of the continuum) or at arbitrary levels of the continuum without the need for being associated to policies at other policy continuum levels. The level at which a policy appears usually delineates how the policy will be ultimately enforced. Federation-level policies specified in high-level abstract terms would need to be refined into lower-level system policies before being enforced on devices to control network resources and services. The policy continuum is essentially a modelling tool that aides in the representation of policies at a high-level of abstraction and provides a way of relating these high-level policies to lower-level policies and/or device configurations. The policy continuum generates dependencies between the sets of policies, a change at one level directly affects the dependencies with policies specified at other levels.

This section describes extensions to a policy authoring process for federation policies in a continuum, that enables experts responsible for authoring federation policies, a way to identify if their new or modified policies potentially conflict with current deployed policies, or if existing federation agreements need to be re-negotiated. The extensions cater for three cases: (i) when an organisation seeks to modify an existing or new federation policy, (ii) when an organisation seeks to modify a system or device-level policy that may affect an existing federation agreement, (iii) when an external federated organisation seeks to modify its policies, affecting an organisation's policies. The cases

are dealt with by extending the policy authoring process specified by Davy *et al.* (2007) that was designed to cater for authoring policies in a single domain environment, towards authoring federation policies within federated domain environments.

These extensions are to cater for the re-negotiation of federation agreements should policy conflicts arise, or a change in the policies internal to an organisation. The modification of a policy in the policy continuum is similar to that presented by Davy *et al.* (2007); however, to cater for the case that a policy may be modified that affects a federation, an additional step has been added. This step involves retrieving affected federation agreements and examines if any federation agreements need to be re-negotiated if the modification proceeds. Also the re-negotiation may be processed and if no successful negotiation is possible, then the process returns to notify the policy author accordingly. Apart from this addition the process (i) verifies the hierarchy of the policy continuum, (ii) analyses for policy conflicts using the approach presented in Chapter 4, and (iii) refines the policies down the policy continuum using the federation policy specification approach described in this chapter in Section 3.4.3.

This is a recursive process; if any step fails the policy author is notified and negotiation of the federation policy will need to be commenced with the other participants of the federation. A negotiation process will need to be employed to indicate if a service provider is capable (i.e. possesses sufficient resources) of implementing a federation policy. If a service provider cannot realise a federation policy, then the service provider may indicate that it can implement a more constrained version of the policy. It may also be the case that a service provider has the resources available to implement a federation policy, but that the implementation of the federation policy will conflict with previously deployed local policies, if this is the case the service provider may request a change in the federation policy before attempting to implement the modified federation policy. It should be noted that this policy authoring process is independent of the manner in which federations themselves are governed. As described in Section 3.1.1 federations may be controlled by a single central manager or be collectively governed by peer managers. The authoring process addresses only the interactions between local and federation policies, it does not make any assumptions regarding how the federation policies are negotiated.

The formal notation used to describe the extensions to the policy continuum model is based on the VDM formal specification language of Bjorner & Jones (1978). 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
$(R \times S \times T)$	A tuple consisting of an entity from each set specified
π^i	Tuple index function, returns the element of a given tuple at index i
$\triangleq$	The function specification symbol
$\forall a \in S : f(a)$	For each entity a in set S, apply function f

Table 3.1: VDM Notation

is a set of techniques used as an independent formal representation for describing the structural characteristics and operations of software systems. It consists of a specification language called VDM-SL that is based on the mathematical principles of sets and maps. Sets are used in VDM-SL to model the various types of entities that exist in a system and the relationships that hold between those entities, this is similar to object oriented principles. The notation used in this section to describe the extensions to the policy continuum model is outlined in table 3.1.

Algorithm 1 Modify a Policy in a Federation.

ModifyFederationPolicy : $(PolicyRule \times PolicyRule \times PolicyContinuum) \rightarrow \mathbf{B}$

ModifyFederationPolicy $(p_{old}, p_{new}, pc) \triangleq$

$\forall ag_f \in$ **AffectedFederationAgreements** (p_{old})

if *not* **ReNegotiateAgreement** $\left(p_{old}, ag_f\right)$

then

NotifyCurrentAuthor $\left(p_{old,} ag_f\right) \rightarrow$ *return* **false**

ModifyPolicyContinuum (p_{old}, p_{new}, pc)

The first extension to the DEN-ng policy continuum model outlined in Algorithm 1 specifies the steps to be taken when a modification is required to a federation policy. In this extension, the process identifies any federation agreements that a modified policy (p_{old}, p_{new}) affects, and investigates if any of these federations (ag_f) need to be re-negotiated. The negotiation process is out of the scope of this work, but there are many possible schemes that could be followed. If a re-negotiation step fails, then the author is notified of the reason and information about the policies and agreements. If the re-negotiation is not required or successful, then the federation policy is integrated into the local policy continuum as described in Algorithm 2.

Algorithm 2 Modify a local Policy that affects a Federation.

ModifyPolicyContinuum : $(PolicyRule \times PolicyRule \times PolicyContinuum) \rightarrow \mathbf{B}$

ModifyPolicyContinuum $(p_{old}, p_{new}, pc) \triangleq$

$\forall p_{fed} \in$ **AffectedFederationLevelPolicies** (p_{old})

  if **VerifyFederationPolicies** (p_{new}, p_{fed})
  then
    if **RequiresReNegotiation** (p_{new}, p_{fed})
    then
      ReNegotiateAgreements $(p_{new,}p_{fed})$
  else
    NotifyCurrentAuthor $(p_{old,}p_{fed}) \rightarrow$ *return* **false**

$\forall p_{parent} \in$ **GetPolicyParents** $(p_{old})\, pc$:

  if *not* **VerifyPolicyContinuum** $(p_{parent}, p_{new}, pc)$
  then
    NotifyCurrentAuthor (p_{parent})
    return **false**

if $\mathbb{P}\mathbf{p_{cnf}}$ = **AnalysePolicyConflict** (p_{new}, pc)

then

  $\forall p_{cnf} \in$ **PotentialConflictList** $(p_{new})\, pc$:

    $\forall p_{parcnf} \in$ **GetPolicyParents** $(p_{cnf})\, pc$:

      NotifyCurrentAuthor (p_{parcnf})

    NotifyCurrentAuthor (p_{cnf})

    return **false**

else

  $\forall p_{\text{oldcld}} \in$ **GetPolicyChildren** (p_{old}, pc) :

    DeletePolicy (p_{oldcld}, pc)

  $\forall p_{ref} =$ **RefinePolicy** (p_{new}, pc) :

    AddPolicy $\left(\pi^1 \circ pc\,(p_{new})\,, p_{ref}\right) pc$

  VerifyPolicyContinuum (p_{new}, p_{ref}, pc)

CommitChange (p_{old}, p_{new}, pc)

The second extension to the policy continuum model is outlined in Algorithm 2 dictates the steps to be taken in the authoring process when a proposed modification is made to an operator's policies. Some of these steps can be automated via policy analysis processes that use system models to assess relationships between polices, whilst

others require decisions from (human) policy authors responsible for policies relating to different policy continuum levels and federations. In this extension, 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 process identifies any federation policies (P_{fed}) that are affected by the modification to a local policy. If required, the federation policy can be re-negotiated or possibly the affected federation agreement will need to be re-negotiated. If a re-negotiation step fails, then the author is notified of the reason and information about the policies and agreements. If the re-negotiation is not required or successful, then the federation policy is integrated into the local policy continuum. The next step traces up the policy continuum and retrieves the parent policies (P_{parent}) related to the local policy (P_{old}). 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 ($GetPolicyParents$ 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 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. If a conflict is detected, a ($PotentialConflictList$) is created which contains the set of associated parent policies that are indirectly involved in the conflict so that more information about the conflict can be relayed back to the current candidate policy author where the policy author can use the policy consistency analysis processes described in Chapter 4 to retrieve more information regarding the conflicting policies that may prove useful in establishing a resolution to the conflict. If no conflict is detected at the current policy continuum level, the process returns the child policies (p_{oldcld}) all the way down the policy continuum. These child policies are then removed from the policy continuum and (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, the appropriate 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.

Algorithm 3 Modify a Policy considering Federation Agreements.

ModifyPolicy : $(PolicyRule \times PolicyRule \times PolicyContinuum \times \mathbb{P}Federation) \rightarrow \mathbf{B}$

ModifyPolicyContinuum $(p_{old}, p_{new}, pc, fs) \triangleq$

if $p_{old} \in$ **GetFederationPolicies** (fs)

then

$\forall ag_f \in$ **AnalyseFederationAgreements** (p_{new}, pc, fs)

if $failed | rejected \rightarrow$ **ReNegotiateAgreement** (p_{old}, ag_f)

then

NotifyFederationPolicyAuthor $(p_{old,} ag_f) \rightarrow return$ **false**

else

$\forall p_{pc1} =$ **RefinePolicyToLevel$_1$** (p_{new}, pc) :

if $conflict \rightarrow$ **AnalyseForPolicyConflict** (p_{pc1}, pc)

then

NotifyPolicyContinuumLevelAuthor$_1$ $(p_{new}, p_{pc1}) \rightarrow return$ **false**

else

if $invalidated \rightarrow$ **ValidateFederationAgreements** (p_{pc1}, fs)

then

ReNegotiateAgreements (p_{pc1}, fs)

else if $p_{old} \in$ **GetLevelPolicies$_1$** (pc)

then

if $invalidated \rightarrow$ **ValidateFederationAgreements** (p_{new}, fs)

then

ReNegotiateAgreements (p_{new}, fs)

else–*Continue with original authoring process*

$\forall p_{parent} \in$ **GetPolicyParents** $(p_{old})\, pc$:

if *not* **VerifyPolicyContinuum** $(p_{parent}, p_{new}, pc)$

then

NotifyCurrentAuthor (p_{parent})

$return$ **false**

if $\mathbb{P}\mathbf{p_{cnf}} =$ **AnalysePolicyConflict** (p_{new}, pc)

then

$\forall p_{cnf} \in$ **PotentialConflictList** $(p_{new})\, pc$:

$\forall p_{parcnf} \in$ **GetPolicyParents** $(p_{cnf})\, pc$:

NotifyCurrentAuthor (p_{parcnf})

NotifyCurrentAuthor (p_{cnf})

$return$ **false**

else

$\forall p_{\mathrm{oldcld}} \in$ **GetPolicyChildren** (p_{old}, pc) :

DeletePolicy (p_{oldcld}, pc)

$\forall p_{ref} =$ **RefinePolicy** (p_{new}, pc) :

AddPolicy $\left(\pi^1 \circ pc\,(p_{new})\,, p_{ref}\right) pc$

VerifyPolicyContinuum (p_{new}, p_{ref}, pc)

CommitChange (p_{old}, p_{new}, pc)

The third extension to the policy continuum model is outlined in Algorithm 3 and outlines the steps to be taken when an external federated organisation seeks to modify its policies, affecting an organisation's policies. Again some of these steps can be automated via policy analysis processes that use system models to assess relationships between polices, whilst others require decisions from (human) policy authors responsible for policies relating to different policy continuum levels and federations. In this extension, the process identifies any previously deployed policies (p_{old}) that a modified (p_{new}) affects and investigates if any of these federation agreements (ag_f) need to be re-negotiated. If a modified policy fails or is rejected, the current federation policy author is notified accordingly of the affected policies and the affected federation agreement (p_{old}, agf) and re-negotiation of the modified policy and/or the federation agreement can commence. Otherwise the modified policy (p_{new}) can be refined to the appropriate level of the policy continuum and analysed for potential conflicts at that level of the policy continuum. If a policy conflict is detected, the policy author at that policy continuum level is notified of (p_{new}, p_{pc1}). If federation policies deployed at that policy continuum level are invalidated, then re-negotiation of federation policies for that level of the policy continuum begins. This is a recursive process where related federation policies at each level of the policy continuum are retrieved and analysed against the modified policy (p_{new}). At each level, federation agreements are validated and if required negotiation of the federation agreement is performed. The remaining steps in the process continue with the original authoring process, see Algorithm 2, where the process traces up the policy continuum and retrieves the parent policies (P_{parent}) related to the local policy (P_{old}). The old policy may have been created as a refinement to a higher-level policy; therefore, if it is modified the process needs to ensure that the related high-level policies can still meet their specified objectives. 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. If a conflict is detected, a ($PotentialConflictList$) is created which contains the set of associated parent policies that are indirectly involved in the conflict that can be relayed back to the current candidate policy author. If no conflict is detected at the current policy continuum level, the process returns the child policies (p_{oldcld}) all the way down the policy continuum. These child policies are then removed from the policy continuum and (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, the appropriate 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.

It is assumed that members of a federation each realise a policy continuum to manage their local resources and that sets of "federation-level" policies relate to their participation in a given federation. For the federation as a whole, governance will be provided by the collection of federation-level policies from each federation member, so these policies must be negotiated by the federation members in line with federation goals. Furthermore, each federation member will need to ensure that its local policies are consistent with the federation-level policies of each of the federations of which it is a member. Maintaining this consistency is a complex task since changes to federation-level policies may have implications for local policies and vice versa. When a network is part of a federation there are "local" policies and also "federation-level" policies, so that, as depicted in Figure 3.4, sub-groups of the former embody the management logic required to realise the latter. Federation-level policies associated with one network operation are linked with other policies in other federation members, thereby enforcing the federation agreement in place between the participants. For example, a federation agreement may allow a service provider configure a set-top box in a customer's home area network; thus the home area network management system will have a federation-level policy allowing the service provider configure aspects of the set-top box, whilst the service provider may have policies controlling when such configurations are applied.

Whilst Figure 3.4 shows a federation involving two members it should be noted that many federations will have many members (possibly with members continuously joining and leaving) and that individual members may be involved in many federations. Given this, a service provider will be faced with the challenge of managing its resources in a manner that meets its own local business goals, whilst also meeting its commitments to participation in one or more federations. In a dynamic environment it is expected that changes in local or federation-level policies will be relatively frequent; each such change may require the service provider to re-assess whether its deployed policies are *consistent* and whether they achieve the best possible result given available resources and demand levels. The overall goal is to develop a policy authoring process and associated analysis tools that will help automate this assessment process. The following cases have been

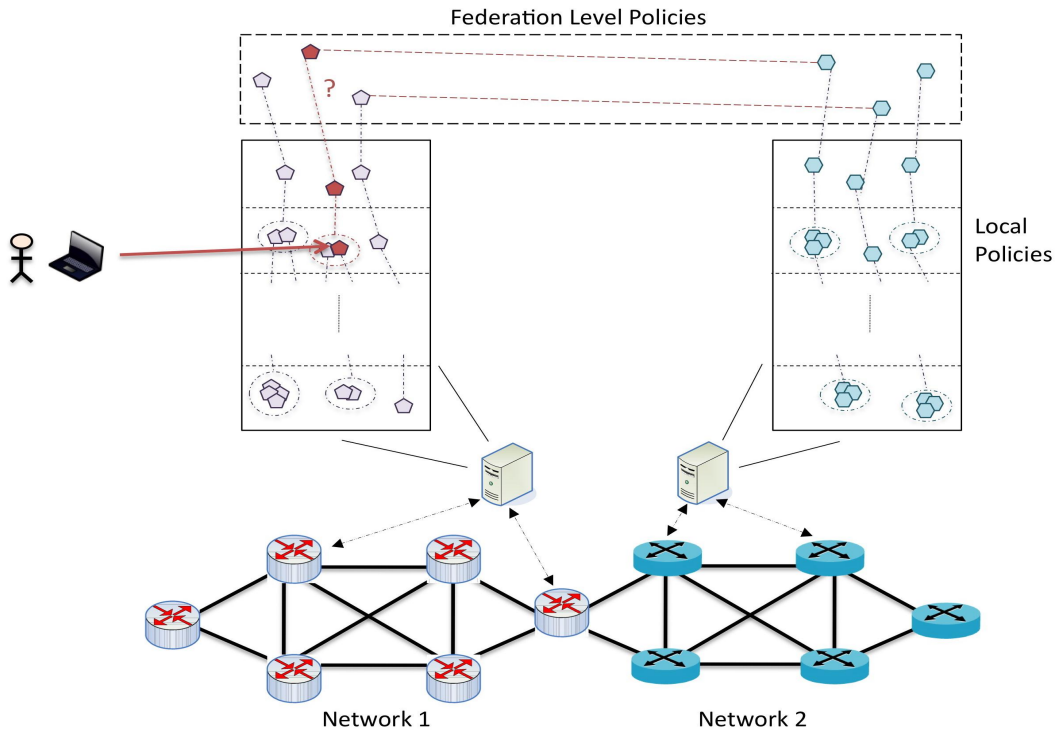

Figure 3.4: This figure illustrates the relationship between "local" and "federation-level" policies where sub-groups of the former embody the management logic required to realise the latter. Federation-level policies associated with one network operation are linked with other policies in other federation members, thereby enforcing the federation agreement in place between the participants.

identified in which policy changes will require assessment of the consistency between local and federation-level policies:

1. *Local Policy or Policies Created, Modified or Deleted*

   At some level of the local policy continuum a policy author attempts to change one or more of the deployed policies. According to the single domain authoring process outlined by Davy *et al.* (2008b), the changes are analysed to ascertain if the modified policies are still valid refinements of the linked policies in the next level up the continuum. If this is not the case the policy author at that upper level is notified. If that author decrees that a change can be made at its level then the process is iterated until the top level of the continuum is reached. In

the enhanced process, the proposed change in local policies will (as illustrated in Figure 3.4) be assessed in terms of its impact on the relevant federation-level policies in each of the federations these local policies are associated with. If the proposed changes mean that the federation level policies are invalidated, there are two options: either the proposed change is rejected, or the relevant federation agreements (and federation-level policies) are re-negotiated;

2. *Re-negotiation of Federation-level Policies*
   As well as local policy changes triggering re-negotiation of federation agreements and federation-level policies, the service provider may itself trigger the re-negotiation. Assuming that federation negotiation is carried out by some multi-step process of bid and counter-bid, then at each step a proposed set of federation-level policies will need to be assessed. Part of this assessment is likely to involve testing whether a combination of already deployed local policies and new policies can effect the behaviour embodied in the federation-level policies. If this is possible, then the impact of introducing new local policies—in terms of the possibility of them introducing conflicts as they are refined down through the policy continuum and in terms of any impact they may have on the ability to meet commitments to other federations—will need to be analysed. The federation-level policies resulting from a negotiation process would be expected to be consistent with pre-existing local policies and the operator's participation in other federations;

3. *Another Member Triggers Re-negotiation of Federation Policies*
   As well as a federation member wishing to change its own local or federation-level policies other federation members can trigger re-negotiations, in which case the steps outlined in point 2 will again be required.

Some assumptions made regarding this approach to federation policy authoring are:

- Policies are stored in a knowledge base with unique identifiers
- Policy element structure adheres to the DEN-ng policy semantics and are uniquely addressable
- Ontology model inspired by DEN-ng information model

- Policies are specified separately in a continuum targeted towards each level of policy author in an organisation
- The structure of the domain is modelled separately to the policy for that domain

3.3 Ontology Models

The DEN-ng information model is fully compliant with the UML and its associated software engineering techniques such as model-based translation that provides the ability to specify policies that adhere to a pre-defined system model. Unfortunately, there remains some open issues with using UML to model management information for a system as highlighted by Wong *et al.* (2005). An information model is used to completely model a managed system and can then be leveraged to build application specific and vendor specific data models from the common representation format. In this way all the information can then be mapped back to a common representation; however, a crucial problem occurs when attempting to compare different application or vendor specific representations togother. One possible solution to this problem is to augment the information and data models with ontological models as described by Wong *et al.* (2005). Ontological models 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 that can be harnessed to enhance policy consistency analysis processes (Davy *et al.* (2008c)). The rest of this section is devoted to discussing how ontologies can benefit information modelling and policy based management. In particular this section describes ontological models that offer powerful formal semantic modelling constructs to model the rich semantic relationships that hold between managed entities defined in the information and data models. More importantly, semantic reasoning can be performed over instances of the ontological models to detect implicit relationships that exist between managed entities and could indicate possible occurrences of policy inconsistencies. This section will highlight that when ontological models are leveraged in conjunction with semantic web rules more powerful modelling and reasoning can be provided over managed domain information and data models. Specifically, ontological models allow context data relevant to managed entities to be represented in a logical form that can be automatically reasoned over using off-the-shelf semantic reasoners.

A subset of the DEN-ng information model, the DEN-ng federated domain model has been transformed into an OWL-DL representation that allows the structure of managed domains to be modelled. The OWL-DL model can then be enhanced by defining dynamic relationships between managed entities within the managed domains. The steps taken in transformation of a subset of the DEN-ng information model to an ontology representation are outlined below:

1. Tag relevant subset of DEN-ng information model
2. Create the ontology based on the tagged subset of the DEN-ng model
3. Create Ontology concepts based on classes of subset of DEN-ng model
4. Properties of classes are mapped to properties of ontology concepts
5. Associations between classes are mapped to properties between ontology concepts
6. Additional domain-specific semantics can be added to the model

It should be noted that the domain and policy models are required for policy consistency analysis and that the domain model is applicable to a particular application-specific area. Each application specific area should provide its own domain model. Sample domain and policy models are provided in this thesis to demonstrate their use in the policy consistency analysis process. The element match algorithm is application-independent and not dependent on any particular domain or policy model. The approach is expected to be adapted by each service provider by using domain and policy models specific to their operating scenarios that can be changed as often as is required.

3.3.1 Domain Ontology Model

The sample domain model is based on the DEN-ng domain model (Strassner (2003)) that represents the arbitrary levels of managed entities in a typical managed domain, starting at the business-level through the system-level and down to the device-level. Each level has a domain model that describes the characteristics of managed entities and a policy model used to control the behaviour of the managed entities. Each layer can import the domain and policy models from the above layer allowing additional application-specific semantics to be added to the model relevant to that constituency

of policy author. Each successive layer will add additional expressive semantics to the previous layer based on concepts familiar to that constituency of policy author.

At the business level, the business domain model represents terms that are familiar to business people (e.g. groups within the organisation, members of those groups and the types of resources/services that those groups and members use). In essence, this means modelling the managed domain entities which are of interest to the business using concepts that are familiar to business people (i.e. business view for business people). The system layer domain model can import the business domain model and extend the concepts as required specific to the system level. The system model represents the managed entities that exist at the system level and are of interest to people working at the system level such as system administrators. The device layer domain model can import the system model and extend concepts from the system model as required for correctly modelling devices and/or device instances. The device domain model represents the devices or instances of devices used for hosting services/resources within a typical managed domain.

This section provides some typical TBox axioms applicable to managed domains and policies represented in description logic formalism that are derived from the extended DEN-ng model following the methodology outlined in Section 3.3. Note, that a more complete version of the DEN-ng model axioms can be found in Appendix C. For the description logics notation see Table 3.2, for a brief description of the axioms.

Notation	Description
C ≡ D	Equivalence class relationship, C is equivalent to D
B ≡ C ⊓ D	B is equivalent to C and D (C intersection D)
B ≡ C ⊔ D	B is equivalent to C or D (C union D)
B ≡ ∃x.D	B is equivalent to anything that has a property x with some value D (necessary)
B ≡ ∀x.D	B is equivalent to anything that has a property x with value only in D (necessary and sufficient)
A ⊑ B	A is a sub type of B

Table 3.2: Description Logic notation

3.3.1.1 Federated Domain

$$\begin{aligned} FederatedDomain \sqsubseteq & ManagementDomain \sqcap \\ & \exists_{\geq 1} federatesDomains.ManagementDomain \sqcap \\ & \exists_{\geq 1} hasFederatedContext.FederatedContext \sqcap \\ & \exists_{\geq 1} hasFederatedContextData.FederatedContextData \end{aligned} \tag{3.1}$$

A *FederatedDomain* concept, defined by the axiom in Equation 3.1, is a collection of managed domains in which each domain in the federation agrees to use zero or more global and zero or more local policy rules to govern the operation of the ManagedEntities that are contained within the managed domain. The federated domain has an overall operating federated context and individual federated context data aspects associated with it.

3.3.1.2 Management Domain

$$\begin{aligned} ManagementDomain \sqsubseteq & Domain \sqcap \\ & \exists_{\geq 1} hasManagedEntity.ManagedEntity \sqcap \\ & \exists_{\geq 1} hasGoverningAuthority.GoverningAuthority \sqcap \\ & \exists_{\geq 1} usesGovernanceApplications.GovernanceApplications \end{aligned} \tag{3.2}$$

A *ManagementDomain* concept, defined by the axiom in Equation 3.2, defines a common set of administrators that govern the managed entities that it contains. In other words, all constituent managed entities in the ManagementDomain are administered by the same user, group of users, and/or organisation(s). Second, it defines a set of applications that are responsible for different governance operations, such as monitoring, configuration, and so forth. Third, it defines a common set of management mechanisms, such as policy rules, that are used by the management mechanisms.

3.3.1.3 Governing Authority

$$\begin{aligned} GoverningAuthority \equiv & ManagedEntity \sqcap \\ & \exists_{\geq 1} governsDomain.ManagementDomain \sqcap \\ & \exists_{\geq 1} usesManagementPolicy.ManagementPolicy \end{aligned} \tag{3.3}$$

A *GoverningAuthority* concept, defined by the axiom in Equation 3.3, represents an individual or collection of managed entities that are responsible for performing governance operations. The governing authority uses appropriate management policies to govern both the managed entities in the domain as well as the management of the domain itself. It is important to note that the domain itself is not capable of management. Management actions are performed by the GoverningAuthority, and use ManagementPolicy instances for management actions (and other types of policy rules descended from PolicyRuleStructure for other types of policy actions).

3.3.1.4 Federated Context

$$\begin{aligned} FederatedContext \equiv & Context \sqcap \\ & \exists_{\geq 1} federatesContext.Context \sqcap \\ & \exists_{\geq 1} usesContextAwarePolicyRules.ContextAwarePolicyRules \end{aligned} \tag{3.4}$$

A *FederatedContext* concept, defined by the axiom in Equation 3.4, represents the overall aggregate contextual information for a federation. It collects local context data from each local domain in the federation and then filters the contextual information according to a set of context-aware policy rules. Federated Context is used to represent a completely aggregated representation of the context of a federation.

3.3.1.5 Federated Context Data

$$\begin{aligned} FederatedContextData \equiv & ContextData \sqcap \\ & \exists_{\geq 1} federatesContextData.ContextData \sqcap \\ & \exists_{\geq 1} usesContextAwarePolicyRules.ContextAwarePolicyRules \end{aligned} \tag{3.5}$$

The *FederatedContextData* concept, defined by the axiom in Equation 3.5, represents the individual aspects of contextual information for each participant managed domain in a federation. Since each context data object represents a different aspect of contextual information, each context data object may have different visibility, access, and other rules that govern its usage. Hence, a federated context data object collects local context data information from the local domain in the federation and then filters the contextual information according to a set of context-aware policy rules.

3.3.1.6 Context

$$Context \equiv \exists_{\geq 1} hasContextData.ContextData \sqcap \exists_{\geq 1} selectsPolicyRules.PolicyRules \tag{3.6}$$

The *Context* concept is defined by the axiom in Equation 3.6. The context of an entity is a collection of knowledge and data that result from the set of all interrelated conditions in which an entity exists. Events point out changing conditions that may affect that entity; an appropriate governance mechanism, such as policy rules, then defines a set of actions in response to the event(s) to change or maintain the state of the entity according to these conditions and actions. Context can have multiple distinct sets of related data and knowledge that are used to adjust its state in accordance with the changes in the environment that it exists in.

3.3.1.7 Context Data

$$\begin{aligned} ContextData \sqsubseteq\ & \exists_{\geq 1} hasContextData.ManagedEntity \sqcup \\ & \exists_{\geq 1} hasContextData.Time \sqcup \\ & \exists_{\geq 1} hasContextData.Location \sqcup \\ & \exists_{\geq 1} hasContextData.Activity \sqcup \\ & \exists_{\geq 1} hasContextData.PersonOrGroup \end{aligned} \tag{3.7}$$

The *ContextData* concept is defined by the axiom in Equation 3.7. Any managed entity can have multiple context data aspects associated with it. Context data focuses on one specific type of data and/or knowledge that is aggregated by the entity's context. The context data concept is used when context contains multiple distinct types of different data that need to be combined in order to determine the overall context of an entity. Some typical types of context data used to describe the state of an entity include time, location, activity, and person/group.

3.3.1.8 Managed Entity

$$\begin{aligned} Product, Resource, Service, \\ PersonOrGroup, Policy \sqsubseteq ManagedEntity \end{aligned} \tag{3.8}$$

A *ManagedEntity* concept, defined by the axiom in Equation 3.8, is an entity that is uniquely addressable and manageable through the use of management policies. A ManagedEntity is something of interest that can be managed. Typical examples include Products (e.g., an application suite), Resources (e.g., network devices and computers), and Services (e.g., VPNs and protocols). Any ManagedEntity can have Context or ContextData associated with it.

3.3.1.9 Product

$$Product \sqsubseteq \exists_{\geq 1} providesService.Service \sqcup \exists_{\geq 1} hasResource.Resource \tag{3.9}$$

A *Product* concept, defined by the axiom in Equation 3.9, provides some type of a service to customers and a product has resources of some form that can be used to provision the service. DEN-ng defines a service/resource as part of a product meaning that a product can contain services and/or resources used by a customer. DEN-ng models a product as an externally facing representation of a service and/or resource procured by customers. A product may be implemented through one or more services which utilise resources.

3.3.1.10 Resource

$$PhysicalResource, LogicalResource \sqsubseteq Resource \tag{3.10}$$

A *Resource* concept, defined by the axiom in Equation 3.10, is defined in DEN-ng as either a physical or a logical resource. This enables a complex entity like a router to be split into its physical and logical components. An example of a physical component would be a piece of hardware such as a networking card, whereas a logical component may be a piece of software that runs on the device (e.g. protocols, ports, etc). Resource is the infrastructure that supports the provision of services and therefore the delivery of products to customers.

3.3.1.11 Service

$$CustomerFacingService, ResourceFacingServiceComposite \sqsubseteq Service \tag{3.11}$$

A *Service* concept, defined by the axiom in Equation 3.11, is either a customer facing service or a resource facing service. A customer facing service defines the characteristics and behaviour of a particular service as seen by the customer. This means that a customer purchases and/or is directly aware of the type of service and is in direct contrast to a resource facing service which support customer facing services, but are not seen or purchased directly by the customer. An example of a customer facing service is voice, video and/or instant messaging (IM) communication as it is seen and consumed by the customer, whereas, an example of a resource facing service may be the XMPP protocol used to provision the voice, video and/or instant messaging (IM) communication as it is utilised by resources, but not seen by the customer. It should be noted that this is an example of one type of service and that many other types of services exist that can be extended from the service concept in the domain model.

The domain ontology model is imported by the policy ontology model as this allows specific rules to be specified for governing the behaviour of the managed domain model entities. Such rules may include access control rules that govern the set of subjects/targets that are permitted/restricted from access to some resource/service within or external to the managed domain.

3.3.2 Policy Ontology Model

The policy ontology model is a generic model inspired by DEN-ng (Strassner *et al.* (2008, 2009)) that is capable of modelling policy languages specified at arbitrary levels of abstraction. Security policies are specified to control access to a managed domain's resources/services. There can be arbitrary levels of policy. An example of the typical policy "levels" within a managed domain are defined as follows:

1. *Business:*

   At the business-level, policies specify in abstract terms the goals governing the overall behaviour of a managed domain. These policies specify the high level rules governing the federation, such as "Group A can talk to Group B, but cannot share business documents with Group B". The focus is on conforming with good business practices, controlling information flows based on "need to know" principles, and employing security and risk principles such as *separation of duty*.

2. *System:*

   At the system-level, system policies provide an implementation to realise the goals of the higher-level business policies. These policies refine the business policies by specifying missing details such as what the groups are, who their members are, and what media types are covered by policies. Typically this is where much of the domain knowledge, captured in the form of ontologies and relational models, is added to the existing policies.

3. *Device:*

   Policies specified at this level are directly implementable and may be deployed onto policy decision points (i.e. access control policies, firewall policies). This in effect creates a continuum of policies (Davy *et al.* (2008b)) that enables the managed domain and policy models to be extended as required by policy authors using concepts relevant to their area of expertise. These policies are ready to be loaded into policy decision points and are consistent with the requests that are generated whenever a managed entity wishes to perform an action on a digital resource (example resources include a communication session, a webcam and a spreadsheet).

The generic policy file structure ontology, depicted in Figure 3.5, shows how a policy model can be subclassed to create a subset policy file structure to model both candidate and federation policies. The subset policy structure contains some common policy concepts specified in the generic policy file structure that can be extended to include policy concepts relevant to modelling a particular policy application (e.g. security, network management, etc.). The subset policy structure does not contain all the concepts present in a standard policy model, but is a simplified policy structure that can be extended to include pertinent policy model concepts as required. The policy file structure can be extended relatively easily to model many common management policy languages used for network management and/or security or even extended to model multiple policy applications simultaneously (e.g. access control and firewall policies). The use of formal semantics to model policies allows for the execution of semantic rules over policies in the policy file structure from disparate policies languages that are specified against the same managed entities. It should be noted that the policy file structure ontology is used primarily for representation and analysis purposes and not for evaluation. A

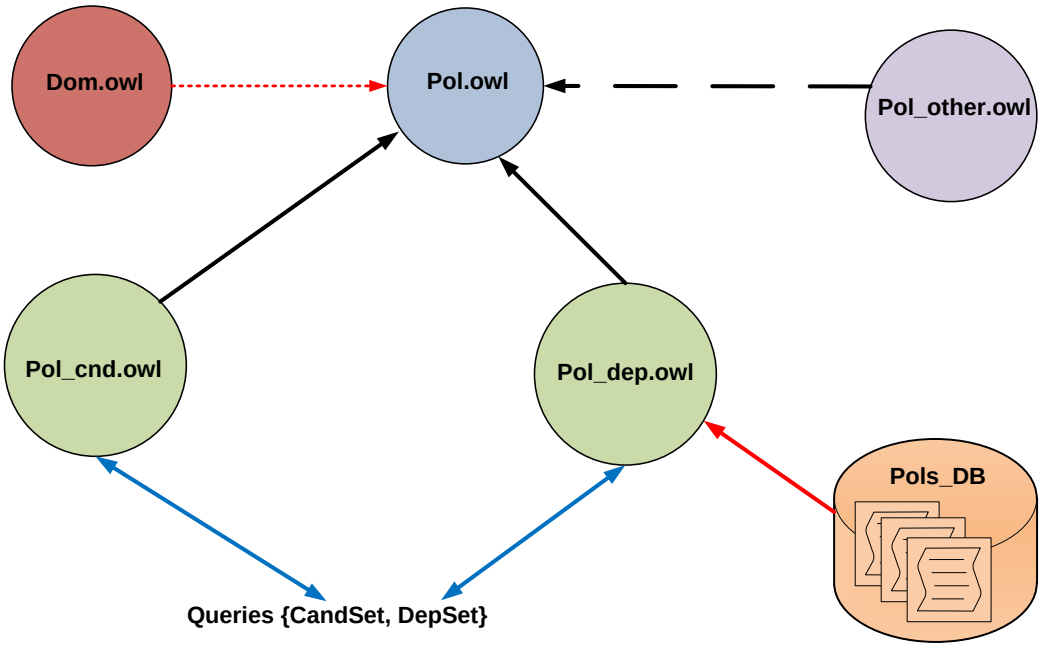

Figure 3.5: This figure illustrates a generic policy file structure suited to ontological modelling that imports a domain model and possibly multiple distinct policy models. The generic policy file structure can then be subclassed as required to model the structure of deployed and candidate policies. Semantic web queries are executed over the policy file structure to return pertinent policies for further analysis.

typical management policy is composed of sets of subjects, targets and rules that can take the form of condition, action {CA} or event, condition, action {ECA}.

3.3.2.1 Policy Concept

$$\begin{aligned} &PolicySubject, PolicyTarget, \\ &ManagementPolicy, DeonticPolicy, ManagementMetaPolicy, \\ &PolicyRuleStructure, ECAPolicyRule, CAPolicyRule, \\ &PolicyRuleComponentStructure, PolicyEvent \\ &PolicyCondition, PolicyAction \\ &\sqsubseteq PolicyConcept \end{aligned} \tag{3.12}$$

A *PolicyConcept* concept, defined by the axiom in Equation 3.12, is the superclass for the classes that together constitute the definition and representation of a Policy Rule and its components. A PolicyConcept is the root of the DEN-ng Policy model. As such, it defines common attributes, methods and relationships that all policy subclasses use and take part in. This class is named PolicyConcept because it does not define the characteristics and behaviour of a policy rule (or any of its components); it defines a part of the overall DEN-ng model that is concerned with modelling generic concepts related to policy.

3.3.2.2 Policy Subject

$$\begin{aligned} PolicySubject \sqsubseteq \\ &\exists_{\geq 1} isSubjectOf.ManagedEntity \sqcap \\ &\exists_{\geq 1} policySubjectGovernedBy.ManagementPolicy \end{aligned} \tag{3.13}$$

A typical policy refers to a *PolicySubject* concept, defined by the axiom in Equation 3.13, refers to a domain managed entity that a policy is specified over. A PolicySubject is a set of entities that represent the authority imposing policy. The PolicySubject can make policy decision and information requests, and it can direct policies to be enforced at a set of PolicyTargets.

3.3.2.3 Policy Target

$$\begin{aligned} PolicyTarget \sqsubseteq \\ &\exists_{\geq 1} isTargetOf.ManagedEntity \sqcap \\ &\exists_{\geq 1} policyTargetGovernedBy.ManagementPolicy \end{aligned} \tag{3.14}$$

Similarly, a policy refers to a *PolicyTarget* concept, defined by the axiom in Equation 3.14, refers to a domain managed entity that a policy is specified against. 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 PolicyTarget, or to transition the PolicyTarget to a new state.

3.3.2.4 Management Policy

$$\begin{aligned} ManagementPolicy \sqsubseteq \\ & \exists_{\geq 1} subjectInteractsWith.PolicySubject \\ & \exists_{\geq 1} targetInteractsWith.PolicyTarget \\ & \exists_{\geq 1} hasPolicyRules.PolicyRuleStructure \\ & \exists_{\geq 1} hasMetaData.ManagementPolicyMetaData \end{aligned} \tag{3.15}$$

A *ManagementPolicy* concept, defined by the axiom in Equation 3.15, is the superclass for PolicyRules that manage a system. As such, ManagementPolicy has explicit associations with PolicySubjects and PolicyTargets, defined through the SubjectInteractsWith and TargetInteractsWith relationships. ManagementPolicy aggregates one or more PolicyRuleStructure objects through the hasPolicyRules relationship to represent different types of policy rules (i.e ECA, CA, etc.) to realise many different forms of management directives. In order to represent popular policy types such as deontic policies, new ManagementPolicy subclasses are created.

$$DeonticPolicy, ManagementMetaPolicy, \sqsubseteq ManagementPolicy \tag{3.16}$$

The *ManagementPolicy* concept is subclassed to realise deontic actions (i.e., obligations and permissions) that will be used to make management decisions. DeonticPolicy is derived from ManagementPolicy, therefore inheriting the optional association to PolicySubject and PolicyTarget. DEN-ng adds the notion of delegations as well. A subclass of the management policy concept, defined in the axiom in Equation 3.16, is either of type deontic policy or management meta policy.

3.3.2.5 Deontic Policy

$$AuthorisationPolicy, ExemptionPolicy, ObligationPolicy, ProhibitionPolicy \sqsubseteq DeonticPolicy \tag{3.17}$$

A *DeonticPolicy* concept, defined by the axiom in Equation 3.17, is deontic policy as modelled in DEN-ng and defined as being either an authorisation, exemption, obligation or prohibition policy. DEN-ng defines four types of deontic policies - authorization (may), obligation (must), prohibition (may not), and exemption (need not), and two types of metapolicies (delegation and revocation). Each of these has two subclasses, one for a subject-based version and another for a target-based version. Authorization embodies the concept of "is permitted to" or "is allowed to". Deontic logicians assign the concept "may" to authorization. A subject-based authorization 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. Prohibition defines the set of actions that an entity is forbidden to execute on another entity. Deontic logicians assign the concept "may not" to authorization. Obligation defines the set of actions that one entity must perform on another entity. Deontic logicians assign the concept "must" to authorization. This facilitates the representation of various types of access control policies. An authorisation policy is specified to permit access to a resource or service and conversely a prohibition policy is specified to forbid access to a resource or service.

3.3.2.6 Management Meta Policy

$$DelegationPolicy, RevocationPolicy \sqsubseteq ManagementMetaPolicy \tag{3.18}$$

A *ManagementMetaPolicy* concept, defined by the axiom in Equation 3.18, is a meta policy that provides additional details regarding the semantics of a management policy. Exemption is, literally, immunity or release from an obligation. Deontic logicians assign the concept "need not" to authorization. A management meta policy is either a delegation policy or a revocation policy. A delegation policy grants some permissions from a delegator to a delegatee to access or interact with some resources. A revocation policy retracts the permissions granted from a delegator to a delegatee.

3.3.2.7 Policy Rule Structure

The *PolicyRuleStructure* concept is used to define the notion of representing the structure of a policy rule independent of any other semantics. A policy rule as defined in the policy model can take the semantics of either a condition-action rule or an event-condition-action rule so the reason for abstracting out the policy rule element is that it allows for representation of a wider range of policies.

$$\begin{aligned} &GoalPolicyRule, CAPolicyRule, ECAPolicyRule, \\ &UtilityFunctionPolicyRule \equiv PolicyRuleStructure \end{aligned} \tag{3.19}$$

A *PolicyRuleStructure* concept, defined by the axiom in Equation 3.19, is a superclass for a variety of policy rules defined in the policy model. This includes goal, condition-action (CA), event-condition-action (ECA) and utility function policy rules. The most common types of policy rules are defined as being either an event-condition-action (ECA) rule or a condition-action (CA) rule. The difference being that with an ECA rule a particular event (or set of events) must occur in order for the rule's conditional entity to be evaluated. With the CA rule the event is considered implicit as is the case for access control policies where the event is a request to access some resource or service.

3.3.2.8 ECA Policy Rule

$$\begin{aligned} ECAPolicyRule \equiv \\ &\exists_{\geq 1} usesPolicyEvent.PolicyEventStructure \sqcap \\ &\exists_{\geq 1} usesPolicyCondition.PolicyConditionStructure \sqcap \\ &\exists_{\geq 1} usesPolicyAction.PolicyActionStructure \sqcap \\ &\exists_{\geq 1} hasPolicyRuleMetaData \end{aligned} \tag{3.20}$$

An *ECAPolicyRule* concept, defined by the axiom in Equation 3.20, is a policy rule that is triggered by an event. When the event occurs, the condition part of the rule is evaluated. If the condition is evaluated to be true then the action part of the rule is executed otherwise if the condition is evaluated to false no action is taken.

3.3.2.9 CA Policy Rule

$$\begin{aligned} CARule \equiv \\ & \exists_{\geq 1} usesPolicyCondition.PolicyConditionStructure \sqcap \\ & \exists_{\geq 1} usesPolicyAction.PolicyActionStructure \sqcap \\ & \exists_{\geq 1} hasPolicyRuleMetaData \end{aligned} \tag{3.21}$$

A *CAPolicy* concept, defined by the axiom in Equation 3.21, is a policy rule that only comprises of a condition and an action. The event is an implicit event, the conditional part of the rule is evaluated and if evaluated to true then the action is executed otherwise if the condition evaluates to false no action is taken. Firewall policies and access control policies use the condition-action rule semantics in their representation.

3.3.2.10 Policy Rule Component Structure

$$\begin{aligned} PolicyEventStructure, PolicyConditionStructure, \\ PolicyActionStructure \sqsubseteq PolicyRuleComponentStructure \end{aligned} \tag{3.22}$$

The *PolicyRuleComponent* structure concept, defined by the axiom in Equation 3.22, is a superclass for the policy condition structure, the policy action structure and the policy event structure classes. Since different types of Policy Rules have different structural components, the PolicyRuleComponentStructure class is used to represent the different types of Policy Rule Components that can be used in a PolicyRule. Typical example subclasses of this class include PolicyEvent, PolicyCondition, and PolicyAction (which are used together to define an ECAPolicyRule).

3.3.2.11 Policy Event

$$\begin{aligned} PolicyEventAtomic, PolicyEventComposite, \\ PolicyEventNonStd \equiv PolicyEvent \end{aligned} \tag{3.23}$$

A *PolicyEvent* concept, defined by the axiom in Equation 3.23, is a superclass for atomic policy events, composite policy events, and non-standard policy events. PolicyEvent represents the occurrence of something significant and whose detection is used to trigger the evaluation of PolicyRule conditions. There are three types of PolicyEvent namely, PolicyEventAtomic, PolicyEventComposite, and PolicyEventNonStd. PolicyEventAtomic represents the happening of a single atomic occurrence. PolicyEventComposite represents the happening of multiple occurrences while PolicyEventNonStd

is a generic extension mechanism for representing event instances that have not been modelled with the attributes specified in the DEN-ng model. PolicyEvent is similar to the concept Event in the policy description language and also the concept Event in the Ponder policy language.

3.3.2.12 Policy Condition

$$PolicyConditionComposite, PolicyConditionAtomic, PolicyConditionNonStd \equiv PolicyCondition \tag{3.24}$$

A *PolicyCondition* concept, defined by the axiom in Equation 3.24, is a superclass for composite policy conditions, atomic policy conditions and non-standard policy conditions. PolicyCondition represents constraints that must hold true before the action of a PolicyRule can be executed. There are three types of PolicyCondition namely, PolicyConditionAtomic, PolicyConditionComposite and PolicyConditionNonStd. PolicyConditionAtomic represents a singular constraint that must hold true before the action of a PolicyRule can be executed. While a PolicyConditionComposite represents multiple constraints, some or all of which must hold true before the action of a PolicyRule can be executed. PolicyConditionNonStd is a generic extension mechanism for representing event instances that have not been modelled with the attributes specified in the DEN-ng model. A concept analogous to PolicyCondition can be represented by many policy languages.

3.3.2.13 Policy Action

$$PolicyActionAtomic, PolicyActionComposite, PolicyActionNonStd \equiv PolicyAction \tag{3.25}$$

A *PolicyAction* concept, defined by the axiom in Equation 3.25, is a superclass for atomic policy actions, composite policy actions, and non-standard policy actions. PolicyAction represents some undertaking on a resource. There are three types of PolicyAction namely, PolicyActionAtomic, PolicyActionComposite and PolicyActionNonStd. PolicyActionAtomic represents the execution of one singular action while PolicyActionComposite represents multiple actions. PolicyActionNonStd is a generic extension mechanism for representing event instances that have not been modelled with the at-

tributes specified in the DEN-ng model. A concept analogous to PolicyCondition can be represented by many policy languages.

$$AccessControlAction, FirewallAction, \\ IPSecVPNAction \sqsubseteq PolicyActionAtomic \tag{3.26}$$

An example atomic policy action concept, defined by the axiom in Equation 3.26, is a superclass for access control actions, firewall actions and IPSec VPN actions.

3.4 Federation Policy Authoring Process

Business policies may define the access control requirements for an organisation at an abstract level and typically in a language that is designed to be interpreted by a human administrator and not a computer. However, many human experts must interpret these policies and define access control policies using a computer understandable language so that access control decisions can be enforced upon requests for resource access. A concern with this existing approach is that there are many different target access control policy languages for the different types of electronic resources. For example, XACML policies may be used to control access to document repositories or hospital records, and IP table rules may be used to control access to network ports on a web server or file server. Often there must be strong collaboration between the authors of these disparate system's specific policy languages, otherwise security holes can occur.

Individual access control policies are authored by system experts with extensive local knowledge, but have limited knowledge of other domains. This harms the ability of security experts to collaborate and ensure their policies are all fulfilling their access control requirements in a consistent way. Additionally, as business policies change, these changes must be propagated to all affected access control systems which is a cumbersome and slow process. The challenges associated to ensuring consistent access control across a federation becomes increasingly difficult as more access control systems are deployed. A promising alternative is to define the access control policies in a structured language tailored for use by business level administrators. These high level policies can then be processed and translated into the various system specific access control policies to be validated and verified by local security experts. Another advantage is that the high level policies can be analysed for correctness and any potential conflicts that may occur

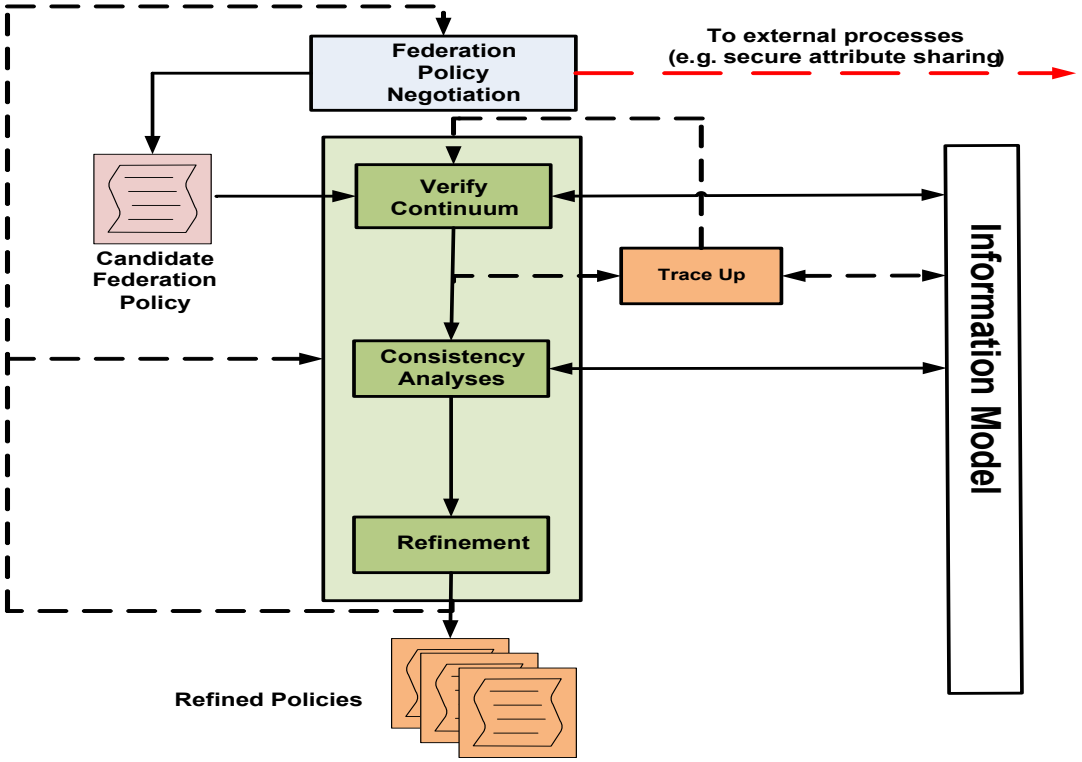

Figure 3.6: This figure illustrates the processes of the federation policy authoring framework. The process firstly requires negotiation of the candidate federation policy aspects. Then the process ascertains if the "candidate" policy still fulfils its role of being one of a group of policies at that policy continuum level. The process checks whether the policy conflicts with any other policies at the same continuum level and finally the policy is refined into one or more policies in the next level down the policy continuum.

across multiple security systems can be detected and highlighted to the appropriate policy author. This rest of this chapter presents a model driven framework to be used by business level policy authors, that can be extended to cater for many target access control systems. A prototype solution is described that has a business level access control policy language, that is refined into multiple system level policy specifications such as the XACML policy language (Rissanen (2013)) and also into IP tables based firewall policies.

The federation policy authoring framework is based on a single-domain policy authoring process outlined by Davy *et al.* (2007, 2008b), that has been extended to cater for authoring federation policies as depicted in Figure 3.6. The federation policy author-

ing framework extends the single domain policy authoring process by providing extra steps necessary for authoring federation policies. This process enables experts responsible for authoring policies at different levels of a policy continuum to identify if their new or modified policies potentially conflict with currently deployed policies, at that level or at other levels of the policy continuum. The process comprises the following steps.

1. If a policy is created, modified at a given policy continuum level the process initially ascertains if the "candidate" policy still fulfils its role of being one of a group of policies at that level which form a valid refinement of one or more policies at the next level up in the continuum.

2. If this is the case the process checks whether the policy conflicts with any other policies at the same continuum level.

3. Finally, the policy is refined into one or more policies in the next level down in the hierarchy and a consistency check is made at that level. This refinement/-consistency analysis is iterated until a set of deployable policies (typically device configuration commands) are generated.

If, at any stage a potential conflict is identified the policy author(s) at that level of the policy continuum are notified and are responsible if the warning should be ignored or if the conflict should be resolved. All of the analysis steps assume the presence of a rich system model used by the various policy analysis processes (policy refinement, policy refinement validation and policy conflict analysis).

3.4.1 Generic Policy Authoring Process

The generic policy authoring process is comprised of three phases. The first phase involves the transformation of relevant aspects of the DEN-ng information model into the required ontologies and is based on a modified version of the algorithm described by Barrett *et al.* (2007a). An information model based on a combination of UML augmented with description logic modelling techniques can be used to represent the static characteristics of federated managed domains and the semantic relationships that exist between managed entities. The relevant aspects of the information model required to model both the managed domain and the associated policies applicable to the domain

are transformed into ontological models, namely a domain ontology and a policy ontology. The policy ontology imports the domain ontology thereby linking the domain model to the policy model and ultimately facilitating semantic querying of deployed policies specified against domain managed entities. The approach to systems modelling can be repeated as required in each managed domain to allow for that domain's heterogeneity.

The second phase involves the execution of semantic web queries for the retrieval of currently deployed local policies that are related over a policy's elements to the candidate federation policy. The set of retrieved deployed policies are analysed by the policy element match algorithm to determine their semantic relationship and overall operating context relative to the candidate federation policy which is used to flag potential inconsistencies.

High-level candidate federation policies require consistent refinement into lower-level enforceable policy (i.e. XACML, IP tables, Ponder, WSPL, etc.) specifications before being loaded into policy decision points deployed in each managed domain. Hence, the third phase invokes a refinement process to transform the more abstract high-level business goals into more concrete lower-level enforceable policy specifications. The federation policy authoring framework facilitates the automatic generation of multiple executable policy specifications that conform to different policy implementation models from a single candidate federation policy. A constraint on any refinement process requires that refined policies are semantically correct and remain consistent with the pre-defined domain model and previously deployed local policies. This can be achieved by employing a policy authoring process that combines both model-based policy specification techniques with semantic policy consistency analysis processes.

3.4.2 Federation Policy Authoring Cases

To aid in the detection of policy inconsistencies when authoring federation policies, context information pertaining to the local domain operating environment, the federated domain environment and the dependencies between policies at arbitrary levels of abstraction within each domain will be required. However, system-level policy languages typically do not convey sufficient context data in their specification to aid consistency analysis, so context data would need to be retrieved from higher-level business policies that are related to the system level policies to provide context information regarding

the local domain's participation in the federation. Federation (business) policies contain context data regarding the operating semantics of the federation. Some typical examples of concepts that constitute context data include the managed entities involved in the federation, time, location, activity and whether the managed entity/entities was a person/group or resource. The federation context data can be combined with local system context data related to deployed local policies to give an overall view of context information pertaining to the local operating domain and the federated domain environments to aid policy conflict analysis processes. Reasoning can then be performed on this overall context information to determine potential situations under which inconsistencies between proposed federation policies and deployed local policies are likely to occur.

3.4.3 Federation Policy Specification

In order to realise federation agreements such as permitting communication services or access to federation resources, high-level federation policies need to be specified and refined into lower-level enforceable policy specifications while maintaining consistency at each step of the process. A negotiation process is required between collaborating (federating) domains in order to create and agree upon these federation (business) policies. This negotiation process is assumed to exist between domains (through some secure attribute sharing framework such as the FRM (Feeney *et al.* (2010)) or any other suitable secure attribute sharing frameworks) and is not part of the work undertaken in this thesis. The shared attributes of federating enterprises can then be used in the policy specification process. The federation policies are refined into the appropriate policy model specific enforceable policy specifications and loaded by policy decision points. Davy *et al.* (2008b) describe how the *policy continuum* model provides the basis of an approach for policy authoring in the domain of autonomic network management. In such scenarios, policies are specified in terms of an information model that is hierarchical in nature. The resulting policies can be arranged in a continuum, ranging from semi-abstract business policies to policies that can be implemented on devices with specific configuration parameters. Between the two extremes are policies with differing levels of detail and refinement, supporting different perspectives. In a federation access control scenario, the (competing) business goals are:

- *Allow* Principals to act upon resources within and between domains, so that *business operations* can proceed
- *Prevent* Principals from acting upon resources within and between domains, so that *security objectives* are met

This basic structure is more limited than that of network management policies described in Davy *et al.* (2008b). However, when authoring access control policies such as those required in the federation scenario above, it is very difficult to define policies that are correct both at the "macro" and at the "micro" level. Thus the policies should be:

1. based on sound security principles addressing modern complex business relationships; *and*
2. specify system-level rules in sufficient detail that no weaknesses exist in their implementation.

Thus access control policies for federation scenarios need to be consistent across multiple perspectives, in a similar way to network management policies more generally. Consequently work in this chapter considers how to take techniques such as *model driven development* that were applied to the network management policy continuum, and apply them to federation scenarios where multiple systems need to be managed. The policy refinement process itself is based on previous work described by Barrett *et al.* (2007a) that involves the creation of required DSLs for each policy level. Each policy level is associated with a DSL that balances the competing objectives of being both expressive and concise. In turn, the language hierarchy is realised in a layered architecture. The number of policy levels required within the layered architecture is arbitrary and largely application dependent. For example, in certain operating scenarios (such as network management) three policy "levels" are required, while other operating scenarios (such as security) only require two policy "levels".

There are two types of repositories, namely a knowledge base and a rule base, in order to store different kinds of information. Static attributes and their relationships are captured in a knowledge base. This knowledge base serves two purposes, namely maintaining relationships (write-oriented) and supporting reasoning over the rules (read-oriented). Rules defined in terms of the semi-static attributes are captured in a rule base. The rule structure is *For* (*Resources*) *IF* (*Conditions*) *THEN permit / deny.*

Rule-combining algorithms combine the effects of all the rules in a policy to arrive at a final authorisation decision, which include deny-overrides, ordered-deny-overrides, permit-overrides, ordered-permit-overrides, as well as first-applicable algorithms. There are some advantages of separating the knowledge base and rule base. First, one attribute knowledge base implementation can be replaced with another that is more suitable for that use case, which offers the same functionalities. Second, the knowledge base can facilitate sharing common knowledge. For example, the business knowledge required to define firewall rules (IP traffic level) is related to that needed to define group chat and media sharing rules (XMPP traffic level). The system consistency can be achieved by sharing the business knowledge across different policy components. Third, it is easier to identify overlap and discrepancies between separate rule sets in the rule base. One or more legacy systems, each operating in its own closed world with its own administrative procedures, can have hidden problems that only become apparent when one of the systems are changed. By consolidating them into a single rule base, conflict and consistency analysis can be performed to alleviate the rule overlap and discrepancies.

The *Xtext* Framework (Efftinge & Völter (2006)) is used to automate the policy specification process. State transitions are defined by rules that are specified using a grammar. Transforming from an abstract language to a less abstract language (*refinement*) can be performed by inspecting both grammars and by defining the transformation rules that map one grammar element into one or more grammar elements in the less abstract language. Closer integration with static attributes reduces the abstraction level and leads to more concrete rules. XText allows a system expert to specify a DSL that can be used by non-system experts to describe relatively complex statements in a constrained and informed manner. The system experts in the example case provided in this chapter are individuals that have in-depth experience in defining communication access control policies to be applied using XACML and firewall policies. They abstract the concepts from these low level policy languages into concepts amenable to non-expert users such as "Allow" and "Permit" keywords and identifying groups of users and resources based on common name references that can later be resolved to concrete references. The authoring application guides the non-system experts while they are defining the high level policies by providing prompt and informative feedback at runtime. The feedback, facilitated by XText warning and error notification system, can alert the authors to inconsistencies or anomalies in the policies they defined. The *Xtext*

warning and error notification system has been augmented with linkages to the policy analysis processes. Subsequently, queries can be made to knowledge bases, stored in OWL based ontologies, to ascertain if there are 1) existing deployed policy rules that may be negatively affected by the deployment of these policies, or 2) semantic based inconsistencies are asserted if the policies are deployed. The analysis processes are highly extensible and can be readily upgraded to detect for more elaborate forms of policy anomalies. This policy management tool chain enables authors to isolate where changes (inserts, updates, deletes) are made, and thereafter to generate artefacts that are consistent with the new policy state, removing the need for tedious and error-prone editing of complex and user-unfriendly editing of multiple device-level policy artefacts.

A typical information model contains additional data not required to produce a DSL, hence only an essential subset of the overall information model pertinent to federations is tagged and extracted from the DEN-ng federated domain model for generation of federation DSLs and accompanying editor tools. A federation DSL generated from the DEN-ng federated domain model, depicted in Figure 3.7, is used to construct an object model of the federation that represents the managed domain (e.g. a communications network) using concepts specified from the information model. This federation DSL includes an accompanying parser and editor that are aware of the types of managed entities that can be linked to one another and in exactly what manner, as this information has been previously specified in the information model. This has the added bonus of preventing the user from describing configurations of the federated domains that are inconsistent with the information model. The generated 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 within each managed domain that make up the federation. Multiple structural DSLs, along with associated parsers and editors can be generated from separate, or possibly overlapping, identified and tagged subsets of the federated domain model. This enables different object models to be defined that correspond to the different policy continuum levels of the managed domains within the federation. A federation policy DSL, see Listing 3.1, is used to model not only the managed entities that form part of the federation, but more critically the federated context under which those entities are permitted to communicate along with federated context data associated with the managed entities.

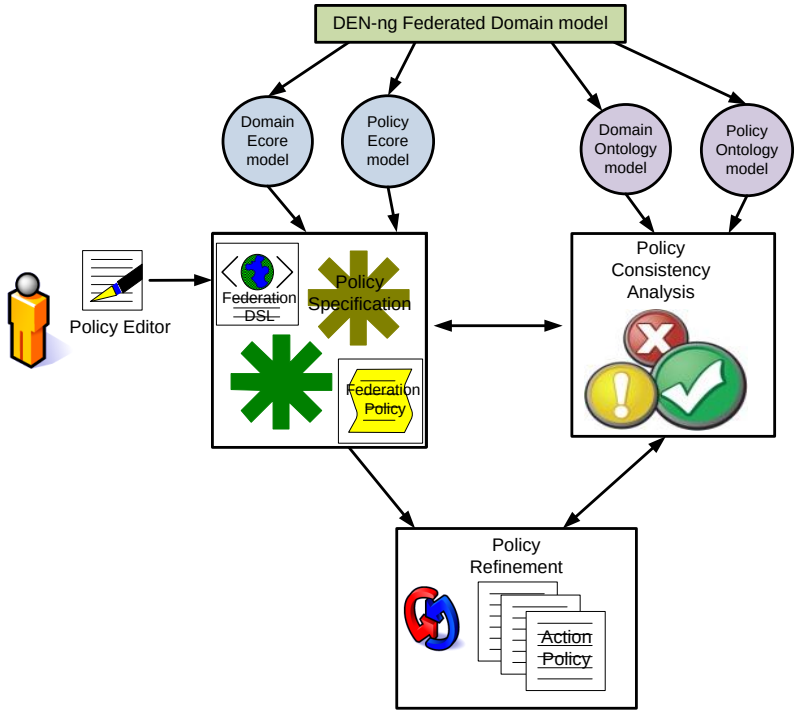

Figure 3.7: This figure illustrates the federation policy specification process. The process requires construction of both an object model and ontological model of the federation that represents the managed domain using concepts specified from the information model. The object model is utilised by the policy specification process to ensure specified policies adhere to the model; while the ontological model is reasoned over by the policy consistency analysis process at each step of the refinement process to ensure policies remain consistent.

```
// Federation Policy DSL generated with Xtext

grammar org.xtext.example.mydsl3.MyDsl with org.eclipse.xtext.common.Terminals

import "platform:/resource/org.tssg.fedlan.datamodel/metamodel/fedlan.ecore"

import "http://www.eclipse.org/emf/2002/Ecore" as ecore

FederatedDomain returns FederatedDomain:
        {FederatedDomain}
        'FederatedDomain'
        '{'
        'GovernedCentrally' isGovernedCentrally=EBoolean
```

```
        ( (hasDomain+=Domain) )*
        '}'
;

Domain returns Domain:
        {Domain}
        'Domain'
        (hasContextData+=ContextData)*
;

ManagementDomain returns ManagementDomain:
        {ManagementDomain}
        'ManagementDomain'
        (hasManagedEntity+=ManagedEntity)*
        (hasGoverningAuthority+=GoverningAuthority)+
        ;

ManagedEntity returns ManagedEntity:
        {ManagedEntity}
        'ManagedEntity'
        '('
                'objectID' objectID=EString
                'description' description=EString
                'commonName' commonName=EString
                ;

GoverningAuthority returns GoverningAuthority:
        {GoverningAuthority}
        'GoverningAuthority'
        (hasManagementDomain+=ManagementDomain)+
        (hasManagementPolicy+=ManagementPolicy)+
        ;

ManagementPolicy returns ManagementPolicy:
        {ManagementPolicy}
        'ManagementPolicy'
        (hasFederatedDomain+=FederatedDomain)+
        (hasPolicyRuleStructure+=PolicyRuleStructure)+
        ;

PolicyRuleStructure returns PolicyRuleStructure:
        {PolicyRuleStructure}
        'PolicyRuleStructure'
        //(theECAPolicyRule+=ECAPolicyRule)*
        ;

ECAPolicyRule returns ECAPolicyRule:
        {ECAPolicyRule}
        'ECAPolicyRule'
        ;

Context returns Context:
```

```
        {Context}
        'Context'
        '{'
                ('hasContextData' '(' hasContextData+=[ContextData|EString] (
                 ↪ "," hasContextData+=[ContextData|EString])* ')' )?
                ('hasFederatedContext' '('
                 ↪ hasFederatedContext+=[FederatedContext|EString] ( ","
                 ↪ hasFederatedContext+=[FederatedContext|EString])* ')' )?
    '}';


ContextData returns ContextData:
        'ContextData'
        '{'
                ('definition' '{' definition+=EString ( ","
                 ↪ definition+=EString)* '}' )?
                'hasFederatedContextData' '('
                 ↪ hasFederatedContextData+=[FederatedContextData|EString] (
                 ↪ ","
                 ↪ hasFederatedContextData+=[FederatedContextData|EString])* ')'
    '}';

FederatedContext returns FederatedContext:
        {FederatedContext}
        'FederatedContext'
        '{'
                ('hasFederatedContextData' '('
                 ↪ hasFederatedContextData+=[FederatedContextData|EString] (
                 ↪ ","
                 ↪ hasFederatedContextData+=[FederatedContextData|EString])*
                 ↪ ')' )?
    '}';

FederatedContextData returns FederatedContextData:
        {FederatedContextData}
        'FederatedContextData'
        '{'
                ('federateddefinition' '{' federateddefinition+=EString ( ","
                 ↪ federateddefinition+=EString)* '}' )?
    '}';

EString returns ecore::EString:
        STRING | ID;

EBoolean returns ecore::EBoolean:
'TRUE' | 'FALSE' ;

EPolicyRuleStructure:
```

```
'CAPolicyRule' | 'ECAPolicyRule' | 'GoalPolicyRule' |
↪ 'UtilityFunctionPolicyRule' ;
```

Listing 3.1: Federation Policy DSL

Following on from the federation DSL creation process, required federation policies can be specified. Each federation participant can make use of the federation policy authoring process to aid in the specification of federation policies as this process can check if a newly created or modified federation policy will conflict with previously deployed local policies. Once a federation policy has been checked for any possible inconsistencies, each federation participant can then transform (refine) the federation policy into lower-level policies suited to the policy systems running within their own domains.

Rule 1:	allow **CISCO** UserGroup **Developers** to use CommunicationMode **chat** with **CISCO** UserGroup **Managers**
Rule 2:	allow **CISCO** UserGroup **Managers** to use CommunicationMode **chat** with **CISCO** UserGroup **Developers**
Rule 3:	allow **CISCO** UserGroup **Developers** to use CommunicationMode **chat** with **TSSG** UserGroup **TSSGStudents**
Rule 4:	allow **CISCO** UserGroup **Developers** to use CommunicationMode **chat** with **TSSG** UserGroup **TSSGPostDocs**
Rule 5:	allow **TSSG** UserGroup **TSSGStudents** to use CommunicationMode **chat** with **CISCO** UserGroup **Developers**

Table 3.3: Federation Goals

Table 3.3 provides some sample *business-level* goals for a federation. Identifiers specific to this federation include `CISCO`, `Developers`, `chat`, etc. Such terms have entries in a *business* glossary and their relationships are specified in a *system* knowledge base. Listing 3.2 illustrates a sample federation (business) policy created using the XText framework that adheres to the federation domain model specified using the federation DSL. Notice that the language used is very close to natural structured sentences, ensuring maximum accessibility to non-experts. This policy specifies the characteristics and intended behaviour of the federated domains and the constraints under which this federation must adhere to. As an example, in order for XMPP communication to take place between these federated domains, a single federation policy that maps to multiple system-level (XACML) policies will need to be specified and deployed (multiple policies per federation participant) within each federated domain. In this particular case

XACML policies are required to allow outgoing XMPP traffic and to permit incoming XMPP traffic in each managed domain.

```
FederatedDomain GovernedCentrally false {

ManagementDomain( objectID("MD1") description("") commonName("TSSG")){

GoverningAuthority( objectID("GA1") description("") commonName("GA1"))

ManagedEntity( objectID("ME1") description("") commonName("TSSGStudents"))

ManagedEntity( objectID("ME2") description("") commonName("TSSGPostDocs"))

Context{
          FederatedContext definition("TSSG contracted to provide research
           ↪ consultancy services to CISCO") }//end Context

ContextData{
        FederatedContextData{ definition("TSSGStudent locatedAt CISCO site")
                                }//end FederatedContextData
                        }//end ContextData
ContextData{
        FederatedContextData{ definition("TSSGStudent requires XMPP comms")
                                }//end FederatedContextData
                        }//end ContextData

        }//end ManagementDomain

ManagementDomain( objectID("MD2") description("") commonName("CISCO")){

GoverningAuthority( objectID("GA2") description("") commonName("GA2"))

ManagedEntity( objectID("ME4") description("") commonName("Managers"))

ManagedEntity( objectID("ME5") description("") commonName("Developers"))

Context{
          FederatedContext definition("CISCO partnered with TSSG to develop new
           ↪ enterprise social networking technologies") }//end Context

ContextData{
        FederatedContextData{ definition("Developers workWith TSSGStudents")
                                }//end FederatedContextData
                        }//end ContextData
ContextData{
        FederatedContextData{ definition("Developers require XMPP comms ")
                                }//end FederatedContextData
                        }//end ContextData

        }//end ManagementDomain
```

```
}//end FederatedDomain
```

Listing 3.2: Federation Policy

The generic business-level (federation) policy is translated into more concrete and meaningful system-level policies by making use of the knowledge base. For example, at the business policy level, resource access polices are generic. More detailed information will be taken from the knowledge base and considered in the lower level polices to complement the business-level generic "access" rules. Two examples are used: one is used to "access" the communication method and the other one to "access" a code repository. When allowing members of a specific group to access a "communication method", e.g, video conference, the information from the knowledge base about membership information, the associated firewall rules and QoS policies should be taken into account. When accessing the code repository, for example a Git-based code repository, the system-level policies must be compatible with the access policies associated with the code repository and policies related to system securities. Listing 3.2 illustrates the intermediate language that was developed to make the high level business (federation) goals more concrete. The language should typically be used by a policy systems expert having knowledge of policy priorities and conflict handling.

```
<Policy PolicyId="abc123"
 RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:
␣rule-combining-algorithm:ordered-permit-overrides">
<Description>This is a federation IM policy.</Description>

<Target>
<Subjects>
<AnySubject/>
</Subjects>
<Resources>
<AnyResource/>
</Resources>
<Actions>
<AnyAction/>
</Actions>
</Target>

<Rule RuleId="rule1" Effect="Permit">

<Target>
<Subjects>
<AnySubject/>
</Subjects>
<Resources>
```

```
<AnyResource/>
</Resources>
<Actions>
<Action>
   <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
     <AttributeValue
       ↪ DataType="http://www.w3.org/2001/XMLSchema#string">chat</AttributeValue>
    <ActionAttributeDesignator
      ↪ AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
      ↪ DataType="http://www.w3.org/2001/XMLSchema#string"/>
  </ActionMatch>
</Action>
</Actions>
</Target>

<Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:or">
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
<SubjectAttributeDesignator AttributeId="group"
 ↪ DataType="http://www.w3.org/2001/XMLSchema#string"/>
</Apply>
<AttributeValue
 ↪ DataType="http://www.w3.org/2001/XMLSchema#string">Developers</AttributeValue>
</Apply>
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
<SubjectAttributeDesignator AttributeId="group"
 ↪ DataType="http://www.w3.org/2001/XMLSchema#string"/>
</Apply>
<AttributeValue
 ↪ DataType="http://www.w3.org/2001/XMLSchema#string">Managers</AttributeValue>
</Apply>
</Condition>

</Rule>

</Policy>
```

Listing 3.3: XACML Policy

The end result of the federation policy specification process, see Listing 3.3, is a low-level XACML policy (one of many) derived from those in Listing 3.2 that is ready to be deployed onto a PDP in order to control communication flows between the managed domains within this federation. Note the same set of identifiers is used in each policy, but the XACML policy is in an enforceable specification ready for loading into a policy decision point. This policy describes that within two organisations, there are particular teams that must not communicate over XMPP. This can be facilitated using XACML and IP firewall rules. However, the firewall rules cannot be too stringent as they need

to be flexible enough to allow other groups to communicate within the federation. This policy may also be incompatible with the existing policy deployment, or may even be redundant if it is fulfilled by an existing set of policies. The policy analysis techniques outlined in Chapter 4 are designed specifically to cater for detection of these types of conflict scenarios.

Each policy level within a federated domain is associated with a policy DSL specific to that domain level, this balances the competing objectives of being both expressive and concise. The language hierarchy is realised in a layered architecture where the generic federation-level polices are translated into more concrete and meaningful system-level policies by making use of detailed system knowledge contained within the federated domain model. The federation policy specification process can achieve automaton through the use of state transitions pre-defined by a system expert using static rules that are specified using a grammar (i.e. DSL). The transformation from an abstract language (i.e. goal policy) to a less abstract language (*refinement*) (i.e. action policy) should be performed by a system expert by inspecting both grammars and by statically defining the transformation rules required to map one grammar element into one or more grammar element(s) in the less abstract language. For example, at the federation policy level, resource access control polices are generic in nature. More detailed information will be taken from the federated domain model and utilised in the lower-level system polices to complement the federation-level generic "access control" rules. The federation policy specification process enables multiple constituencies of policy authors to identify where changes (inserts, updates, deletes) are made, and accordingly to generate policy updates that are consistent with the new policy state, removing the need for tedious and error-prone editing of complex and user-unfriendly editing of multiple system-level policy artefacts.

This section described the development of the *Federation Policy Authoring Process* which is used by non-system experts to create and modify access control policies for a number of systems under influence by the terms of the federation contract. Specifically, policies were defined that constrain instant messaging between two organisations through the use of a policy-controlled message Intercepter. The Intercepter receives IM communications leaving and entering each organisation and can query and enforce XACML based policies on these messages. Access is also constrained to network attached storage devices by blocking access to network address ranges and ports at the

IP level using a linux based firewall. Thus policies can be enforced that can constrain communication via instant messaging and access to network based resources across the federation. The approach was evaluated in the context of a federated access control scenario, where two organisations agree to federate in accordance with a pre-determined contract. Within this contract are the terms on which resources can be shared. These resources may be source code repositories, document servers, or communication services.

3.5 Federation Test-bed Implementation

In order to evaluate the newly developed processes and algorithms outlined in this thesis an exemplar federated policy based management test-bed was developed and implemented. The test-bed includes separate, but consolidated processes for authoring federation policies that include the sub-processes of policy specification (including refinement into multiple distinct lower-level system languages), policy consistency analysis and policy evaluation. The main advantage of developing and implementing a federation test-bed is that it enables testing and verification of any newly developed federated policy specification and policy consistency analysis processes in a real-life scenario. Unfortunately, it was only possible to have a maximum of two simulated federated domains in the test-bed. For proper scalability testing of the newly developed federation processes, many more federation members are required.

The diagram in Figure 3.8 depicts the policy controlled federated test-bed architecture. The architecture illustrates two service providers each implementing a continuum of policies to control their federated resources and services. The illustration also depicts the arbitrary levels of network management that can exist within each domain. This allows policy authors to specify policies relative to their area of network management. Policies can exist solely or within a policy set at lower levels of network management and may also have dependencies to policies at higher levels of abstraction within the policy continuum. The top level box shows federation-level policies (specifying the business goals to oversee a typical end-to-end communication flow such as XMPP) that need to be refined to lower-level policies before being enforced to control network resources

and services within the domains of service providers participating in a federation. Some reasonable assumptions made regarding the approach taken are:

- Service providers participating in a federation are assumed to have a hierarchical XMPP grouping structure based on their internal organisational hierarchy
- Service providers implement a policy based management system and specifically policy rules to control their network resources and services
- Policies are implemented in a continuum from business-level to system-level to manage device configurations
- Service providers will need to enforce federated policy rules to control the behaviour of their federated XMPP communication services
- A secure attribute sharing negotiation process is required and assumed to be in place between service providers in order to create and agree upon federation-level policies.
- There should be no change to client software code or server software code (this makes the approach extensible to other XMPP servers and clients)
- There should be no change to the architecture of the policy system (this enables policy systems to be extended and if required alternative policy models could be swapped in or out of the architecture on an 'as-needed' basis)

The federated XMPP test-bed, depicted in Figure 3.9, is deployed within each of the two managed domains participating in the federation and consists of the following components, an open-source XMPP server, an Intercepter and a XACML policy server.

3.5.1 XMPP Server

XMPP (Saint-Andre (2011)) is an open and extensible protocol for providing near real-time communication services such as messaging, presence, and request-response services. These near real-time communication services when implemented between enterprises are a perfect example of loosely coupled value networks (a.k.a federation). The basic syntax

and semantics were developed originally within the Jabber open-source community, in 1999. The communication services are defined in two primary specifications published by the IETF (RFC3920), (RFC3921)(the "RFC" series), and in dozens of extension specifications published by the XMPP Standards Foundation (XSF) (the "XEP" series). XMPP communication services (Instant Messaging (IM), Groupchat, filesharing, etc.) are an efficient and powerful means of communication within and between enterprises. This relatively open communication method has been growing in popularity with the inception of social networking, however, it's usage has remained largely unchecked. Openfire (Realtime (2011)) is an open-source XMPP server, written in Java. It supports various XMPP communications services such as IM, group chat, file transfer, etc., and uses the XMPP protocol for near real time communication. XMPP is designed using a federated, client-server architecture. Server federation is a common means of spreading resource usage and control between Internet services. In a federated architecture, each server is responsible for controlling all activities within its own domain and works cooperatively with servers in other domains as equal peers. In XMPP, each client connects to the server that controls its XMPP domain. This server is responsible for authentication, message delivery and maintaining presence information for all users within the domain.

3.5.2 Intercepter

An Intercepter (i.e software based agent) was implemented in Java to intercept XMPP packets travelling between the XMPP client and server with the aim of applying policy rules to the type of communication being sought. The Intercepter is basically a modified XMPP server that can intercept all XMPP packets travelling through it or individual types of XMPP packets such as Info/Query (IQ) packets, message packets or presence packets. The main task of the Intercepter is to act as a PEP by intercepting XMPP packets, formulating and forwarding access requests based on the XMPP packets to the XACML policy server, enforcing the responses to the policy decisions received from the XACML policy server by either routing the XMPP packet to the XMPP server for delivery if the policy rule permits or returning an information message back to the XMPP client advising that the type of XMPP communication service being sought was denied.

3.5.3 XACML Policy Server

The test-bed has been designed to apply XACML (Rissanen (2013)) policy rules to XMPP communication flows. Unchecked XMPP communication flows, such as IM for personal communication among employees can have a positive or negative impact on an enterprise's operations and even its bottom line. On the positive side, XMPP communication services facilitate collaboration via real-time communication tools; while a negative impact of having XMPP communication services freely available is that employees may abuse them for their own personal benefit or may inadvertently cause a conflict of interest if the wrong parties are permitted to communicate (i.e hedge fund investors, bank officials, etc.). This implies the need to enforce intelligent constraints in the form of XACML policy rules against all types of XMPP communication in a bid to restrict communication flows between employees within federating enterprises.

Sun's XACML2 (Proctor (2006)), is an open-source implementation of the OASIS XACML standard, written in the Java programming language. It consists of a set of Java classes that understand the XACML language, as well as the rules about how to process requests and how to manage attributes and other related data. The XACML policy server consists of two main components, a PEP for creating policy requests and a PDP for loading policies and making policy decisions. The PDP loads policies from a policy repository and makes decisions based on loaded policies and requests forwarded to it from the PEP. The policy repository can be a database or a LDAP system.

The typical sequence of messages that occur for a basic XMPP communication request are described as follows. Firstly, in a federated environment, XACML policy servers will need to communicate with each other to negotiate context, federated policy rules, etc., as these rules will need to be pre-loaded into the PDP. An XMPP communication request arrives to the Intercepter from the user. The Intercepter forwards the request to the XACML policy server which consists of two main components, a PEP for creating policy requests and a PDP for loading policies and making policy decisions. The PEP creates a policy request based on the contents of the XMPP packet that was forwarded to it from the Intecepter. This policy request encoded in XML is then forwarded to the PDP for a policy decision based on the type of communication being requested. Upon receiving a policy request from the PEP, the PDP decodes the XML request, loads a set of relevant policies based on a target object of the request and

makes a policy decision after comparing the policy request against the policies it has loaded. Once a policy decision has been determined, the PDP encodes the response in XML and returns this response to the PEP. The PEP then instructs the Intercepter to either forward the XMPP packet to the XMPP server or return an error message back to the client. In the case that the XMPP communication request is permitted, then no response is returned to the initiator of the request as this is deemed unnecessary and would only generate extraneous traffic.

The XMPP test-bed highlights the suitability of XACML as a security policy language capable of controlling XMPP communication services in federated environments. XACML is leveraged as a first step in applying access control to XMPP communication services in an intra-domain and inter-domain federation environment. The inter-domain case adds increased complexity and more stringent requirements to the solution due to conflicting objectives and organisational principles.

The sequence of steps in the federation policy authoring process that make use of the newly developed processes (i.e algorithms, DSLs, etc.), from initial negotiation of the candidate federation policy aspects through federation policy specification, consistency analysis against local deployed policies and finally refinement into (possibly multiple) lower-level enforceable policies include:

- Specification of federation policies using the federation policy specification processes that follow model driven development principles and adhere to the federated domain & policy continuum models.
- Consistency analysis of federation policies using consistency analysis processes that include semantic web queries and element match algorithm.
- Optimum policy evaluation using the probabilistic access control router in federating enterprise scenarios (a.k.a. enterprise social networks)

3.6 Summary and Discussion

This chapter described extensions to the DEN-ng information model and policy continuum model to cater for the consistent specification of federation policies. An associated federation policy authoring process is introduced that makes extensive use of model

driven development principles applied to derive multiple heterogeneous executable policies from a single high-level federation policy that demonstrates use of the new DEN-ng and policy continuum extensions. Firstly, the extensions to the DEN-ng information model address federated management domains, wherein policies are used as a means of governing the interactions between federated domains. Secondly, extensions to the policy continuum are introduced to allow policy authors to specify federation policies that are consistent with local policies at various abstraction levels within each federated domain.

Other notable information models, specifically the CIM (Lamers *et al.* (2010)), or the SID (Faurer *et al.* (2004)) do not provide support for representing certain concepts required to support the modelling of federations. In particular neither of these information models provides the concept of a FederatedDomain class required for modelling a collection of managed domains participating in a federation. These information models do not model FederatedContext or FederatedContextData classes required to represent the overall aggregate contextual information applicable to FederatedDomains and used to select a set of deployable polices for a particular context. Neither information model supports a GoverningAuthority class which models the governance structure of the federation and uses appropriate management policies to control the federation. In consideration of these important consolidated modelling capabilities, the DEN-ng information model has been selected as a suitable candidate to extend for complete representation of both the managed domain and policy models applicable to federation management.

The federation policy authoring process includes the sub-processes of policy specification and consistency analysis. Federation policy specification processes require detailed system models to aid in the consistent specification of federation policies, so extensions to the DEN-ng information model were introduced to facilitate modelling of federated domains and the policies that govern them. This is a useful first step towards using DEN-ng for specification and consistency analysis of federation policies. Federation and local system policies are defined at different abstraction levels which makes the processes of policy specification and conflict analysis more difficult to perform. To cater for this, extensions to the policy continuum model were provided that enables experts responsible for authoring federation policies, a way to identify if their new or modified policies potentially conflict with current deployed policies, or if existing

federation agreements need to be re-negotiated. Federation policies convey context information pertaining to their deployment. In order to aid consistent policy specification and perform consistency analysis between deployed local system policies and federation policies, there needs to be a means of representing the context information available in federation policies representing the linkage between federation and local system policies in a generic fashion. Semantic web technologies such as ontological models and semantic web rules are a worthy candidate for this purpose as they allow for the representation of policies in a formal representation that can be reasoned over to make implicit knowledge contained within the models explicit.

Defining policies for a large organisation that need to be deployed onto multiple target policy systems is a cumbersome process that involves the participation of many domain experts. The policy specification process described in this chapter aims to alleviate the need for the involvement of many domain experts as the various policy languages required are separated out into distinct levels of the policy continuum. A proof of concept system was implemented that can generate validated XACML and firewall policies in a controlled scenario involving communication across administrative boundaries. A model driven approach was adopted and facilitated by the XText framework. This can be viewed as a very promising approach towards realising the policy continuum concept for targeted scenarios such as the one presented in this chapter.

Research question 1, outlined in Chapter 1 asks *how can a policy authoring process be defined to support the specification of federation-level policies by multiple constituencies of policy authors aimed at controlling federations of services and resources.* The federation policy authoring process defined in this chapter addresses this research question as it is tailored to operate across federated domains to aid in federation policy specification. The extended DEN-ng federated domain model can represent context data pertaining to federated and local domain environments. This context data is crucial as input to any policy specification or policy consistency analysis processes. Context data can be obtained from each federated domain participant to give an overall view of context for that federation. The context data related to federation policies can be used to assist policy consistency analysis processes to determine situations in which deployed policies may conflict with a candidate federation policy. As an initial step toward developing such policy specification and analysis tools this chapter introduced a policy authoring process whose steps ensure that local policies are kept consistent with federation policies

as individual policies are created, modified or withdrawn. Specifically, this chapter described an extension to DEN-ng to facilitate modelling all aspects relevant to federated domains.

This chapter described extensions to the DEN-ng information model and policy continuum model to cater for federation policy specification. However, newly specified federation policies may conflict during the refinement process, so what is required are policy consistency checks between candidate federation policies and previously deployed local policies. The next chapter will concentrate on the investigation of techniques to automatically assess, based on knowledge embodied in system models, the likely impact of changes to local policies on federation policies and vice versa. The next chapter outlines an approach to detect inconsistencies among groups of previously deployed policies (as opposed to only performing pair-wise analysis of policies). Analysing groups of deployed policies may detect inconsistencies not possible using pair-wise analysis techniques alone.

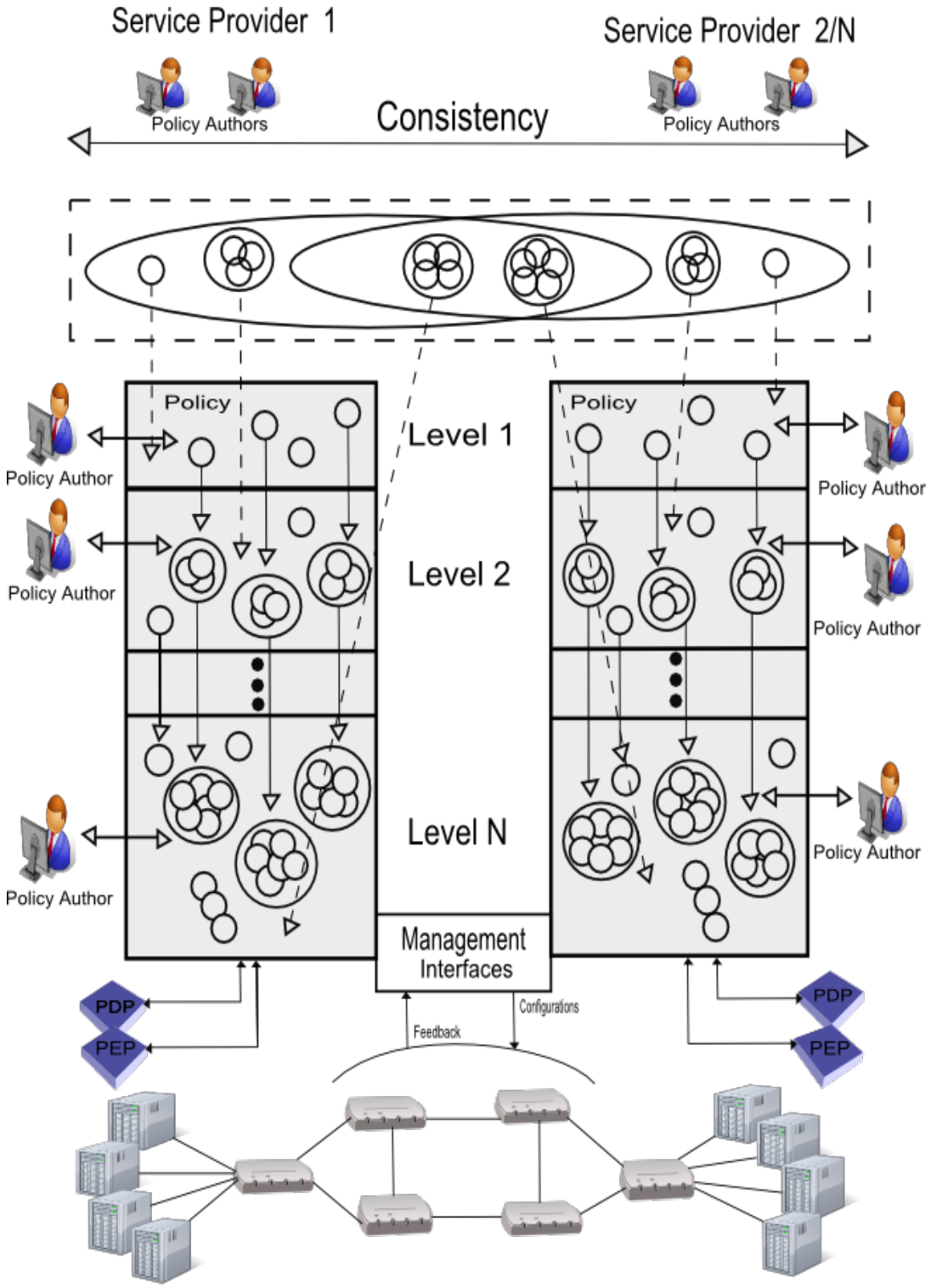

Figure 3.8: This figure illustrates an exemplar policy controlled federated test-bed architecture. The architecture illustrates two service providers each implementing a continuum of policies used to control their federated resources and services. The illustration also depicts the arbitrary levels of network management that can exist within each domain.

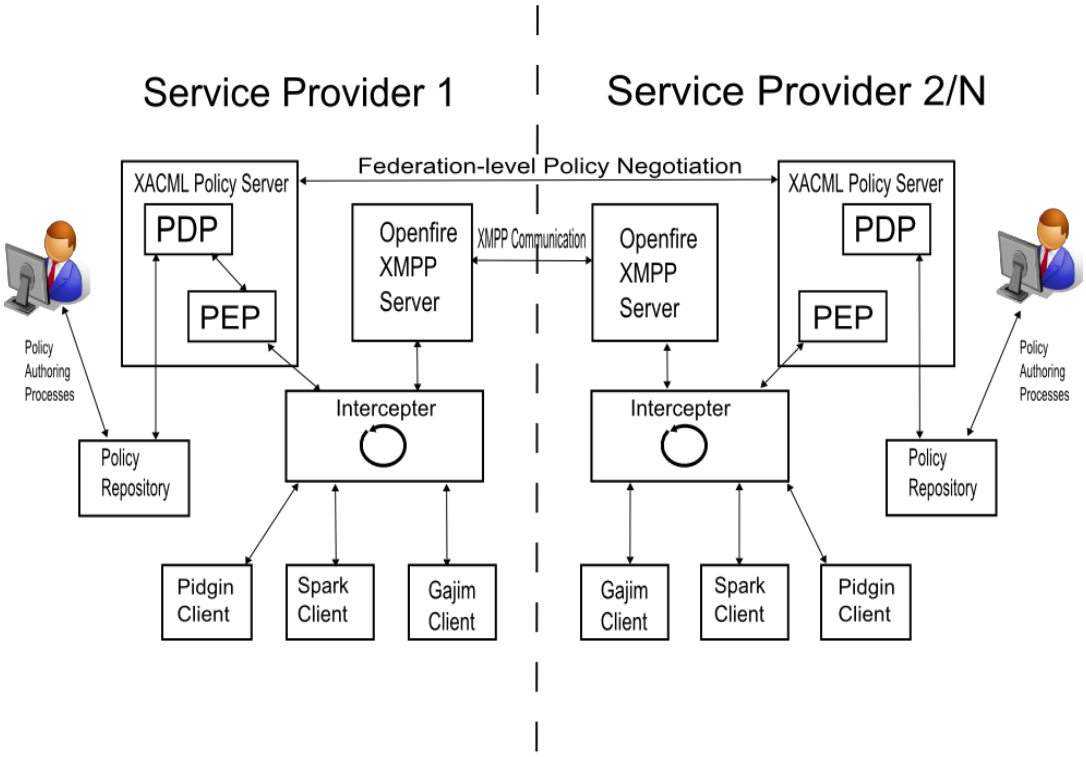

Figure 3.9: This figure illustrates the components of the exemplar federated XMPP test-bed that is in place between two federated service provider domains. The components of the test-bed include an open-source XMPP server, an Intercepter, a XACML policy server, and multiple types of XMPP clients.

Chapter 4

Federation Policy Analysis

Federated domains comprise of individual management domains that leverage policy based management systems using heterogeneous policy languages that can be deployed at different abstraction layers within a managed domain. The policy analysis approach presented in this chapter takes such operating constraints into account and is a first step in supporting policy consistency analysis across federated domain environments. However, the approach taken makes some limited assumptions in that the information model used by each managed domain must be capable of representing both context and context data pertaining to the operating semantics of the federation to aid policy consistency analysis processes. This chapter builds on the previous chapter, (Chapter 3), where the DEN-ng federated domain model extensions were proposed to assist in the specification of federation policies. Assuming that federation policies are specified using the policy authoring process outlined in Chapter 3, they will need to be checked for potential inconsistencies with deployed local policies during the transformation (refinement) phase. This chapter introduces a framework that facilitates the detection of policy inconsistencies that can occur during transformation of high-level policies into multiple low-level policy specifications in complex federated domain environments.

As part of the framework and to aid interoperability, extensibility, and reasoning; ontological models are used to describe both the management domain and management policies specified over individual/group managed entities of the management domain. Specifically, this chapter outlines a policy selection process that harnesses ontological models through semantic web rules specified against common patterns of policy inconsistencies to reduce the policy search space and return only pertinent groups of deployed

policies for further consistency analysis. The policy element match algorithm is specifically tailored towards matching groups of policy elements from an arbitrary number of policies. The approach can discover inconsistencies where an entire set of policies is returned that matches a single candidate policy. Previous approaches to policy consistency analysis mentioned in the literature could be used to inefficiently detect (albeit on a pair-wise basis) the same inconsistent policies but would require many more iterations (policy comparisons) to ensure that the deployed policies completely matched the semantics of the candidate policy. The reason for this is that previous approaches failed to utilise both the powerful modelling constructs provided by ontologies and query capabilities provided by their associated semantic web rules that can be leveraged by policy consistency analysis processes to detect occurrences of complex policy inconsistencies such as those that occur in federated domain environments. The semantic web queries coupled with the policy element match algorithm defined in this chapter should be deployed as part of the policy authoring process described in Chapter 3 to aid policy consistency analysis.

The rest of this chapter is structured as follows, Section 4.1 frames the policy consistency analysis problem as a combinatorial problem that aims to identify the minimal combination of deployed policies that match a candidate policy. Section 4.2 outlines the policy analysis processes that are core to the theme of this chapter. The policy analysis process takes a two phase approach, a policy selection phase and a policy element match phase. The policy selection phase executes semantic web queries over instantiations of the ontological models to return a much reduced set of deployed policies as input to the policy element match algorithm. The policy element match algorithm can be viewed as central to the framework by identifying combinations of matches over arbitrary numbers of policy elements which can highlight potential inconsistencies more efficiently than using pair-wise policy analysis techniques. Section 4.3 describes typical use cases to demonstrate the effectiveness of the policy consistency analysis approach in a federated enterprise communication scenario. Section 4.4 describes the policy consistency analysis prototype implementation, while 4.5 discusses experimental results with regards to evaluation of the approach. Finally, in 4.6 the contributions of this chapter are summarised and analysed.

4.1 Combinatorial Problem

$$\underset{X}{\text{minimize}} \quad \sum_{i=1}^{C}\sum_{j=1}^{D} c_{ij}x_{ij} \tag{4.1a}$$

$$\text{subject to} \quad \sum_{p\epsilon P} x_{ip} \geq 1, \quad \forall p\epsilon P, \forall i\epsilon C \tag{4.1b}$$

$$\sum_{i=1}^{C} x_{ij} = \big(0\ or\ C\big)\ \forall j\epsilon D \tag{4.1c}$$

$$x_{ij}\epsilon\{1,0\} \tag{4.1d}$$

The policy consistency analysis combinatorial problem aims to discover the minimal combination of deployed policies that, when considered together, relate to the entities of a given policy named the candidate policy. The combinatorial problem is described in equation 4.1 and is similar in form to the set cover combinatorial problem. The primary differences are that there may be multiple element sets related to the candidate policy, similarly there are multiple element sets related to each deployed policy.

C is the number of elements defined for a particular policy model. D is the number of deployed policies in a particular policy based management system. The decision variable x_{ij} has an integer value of 0 or 1 and indicates whether a particular deployed policy element is selected as part of the consistency analysis solution. The objective function aims to minimise the cost c_{ij} of including each deployed policy x_{ij} in the solution set. The constraints over the decision variable are that for each entity ($p\epsilon P$) of each element of the candidate policy the sum of deployed policies that include the candidate policy element should equal to 1 or more. This ensures that each candidate policy entity is covered. Also to ensure that each element of the candidate policy is covered entirely by the deployed policies, the number of covers should sum to the number of policy elements if selected at all.

Calculating the minimum number of deployed policies that overlap to cover a candidate policy is an NP hard problem. This is due to the fact that all combinations of deployed policies need to be considered together to ensure an optimal solution. The solution space for the problem is therefore 2^n where n is the number of deployed policies. Effectively the solution space doubles on the addition of each new policy. The approach taken in this thesis seeks to discover multiple possible combinations of policies that can be considered together to cover the candidate policy. In this thesis this type of analysis is termed policy consistency analysis.

4.2 Policy Analysis Processes

"Federation-level" policies required to control a federation cannot be specified and deployed within federated domains without checking if their deployment will lead to inconsistencies with the "local" systems comprising the federation. Newly specified federation policies have the potential to conflict with previously deployed local/federation policies at arbitrary levels of abstraction within each domain. This requires implementation of policy consistency analysis processes as part of an overall federated policy authoring process. These processes need to be generic in nature so that they can be used by each federation member to maintain the consistency of the federation with the local operating environment. They would also need to take into account the heterogeneity of each domain with regard to information models, policy-based management systems and more specifically policy languages employed.

The policy consistency analysis process, depicted in Figure 4.1, takes a two phase approach, a policy element selection phase and a policy element match phase. In the first phase semantic queries are executed over instantiations of the ontological models to return deployed policy element identifiers. The second phase in the process attempts to identify matches between a candidate policy element and deployed policy elements. The approach can easily accommodate many types of inconsistencies between policies such as modality conflicts, application-specific conflicts without modification to the domain and policy ontology models or the element match algorithm. The policy consistency analysis process is one step in an overall policy authoring framework for policies. The primary requirement on the policy consistency analysis process is that it remain application-independent so as to be applicable to multiple policy-based management scenarios. Work carried out by Barron *et al.* (2011) outlined policy selection and consistency analysis processes required for authoring policies in a federated environment.

By employing ontological models and semantic web rules for modelling managed domains where rule conflicts can occur provides different ways to detect conflicts in the knowledge base. The models are specified in formal representation format where an off-the-shelf reasoner can be applied to detect occurrences of inconsistencies in the knowledge base or to make implicit knowledge explicit. When contradictory facts are

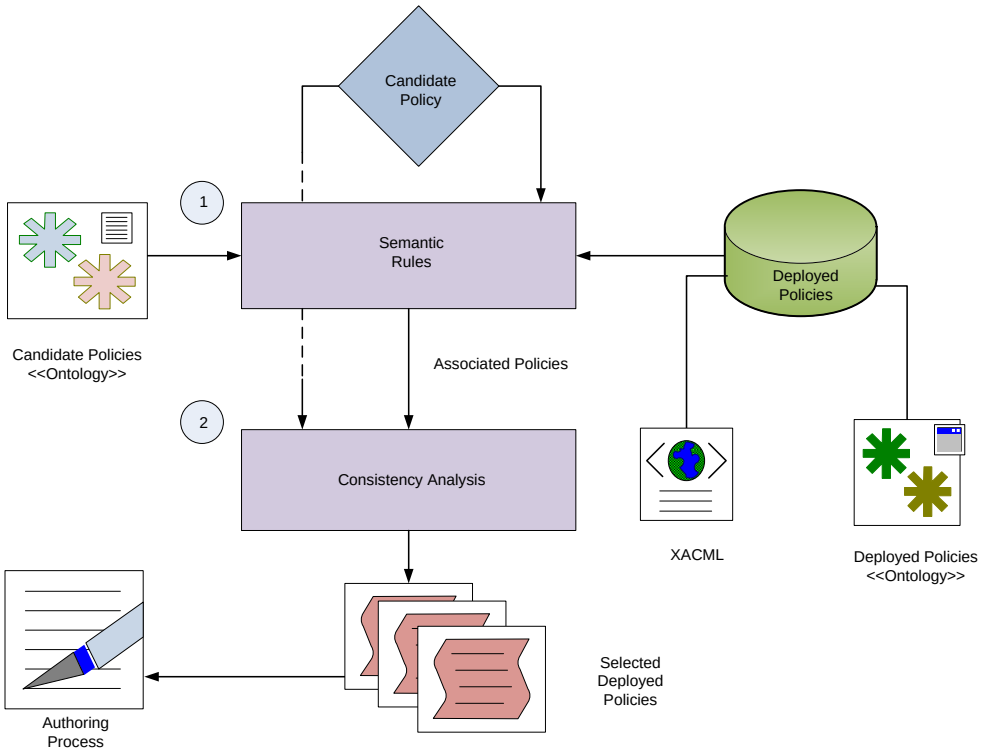

Figure 4.1: This figure illustrates the policy analysis process that takes a two phase approach, a policy element selection phase that executes semantic web queries over instantiations of the ontological models to return deployed policy element identifiers and a policy element match phase that attempts to identify matches between a candidate policy element and deployed policy elements.

defined in the knowledge base, the knowledge base is deemed to be inconsistent. Reasoners are capable of detecting these inconsistencies by default. Hence, the consistency checking processes used by knowledge base reasoners can be used to detect occurrences of semantic conflicts between policies. The default types of semantic conflicts that can be detected are:

- *Detection based on disjoint classes*

  Two classes can be defined as being disjoint in the ontology. This means that an individual cannot be a member of both classes simultaneously. This can be used to ensure that entities cannot be members of two different classes at the same time. An example of this may be to check that a policy does not have conflicting actions

such as Permit/Deny. The reasoner has the ability to check if an individual is in an inconsistent state.

- *Detection based on complement classes*
  Consider a class of a certain type, a new class can be defined as the complement of this class, so an individual that is not an instance of the first class will be deemed to be an instance of the second class. This functionality can be achieved using the disjointWith construct. However, if an individual is a member of the first class and also the second class, then this will be defined as an inconsistency in the knowledge base and can be detected by the reasoner.

- *Detection based on the empty set membership*
  An empty set is a set that is defined in the knowledge base as containing no individuals. The nothing class is used to specify an empty set. If an individual exists in the knowledge base as a member of the empty set then this is deemed to be an inconsistency and can be considered a semantic conflict. A reasoner can detect this type of semantic conflict.

4.2.1 Policy Selection

Policy selection is the process of retrieving deployed policies related to a candidate policy for inclusion in subsequent consistency analysis processes. Medium to large scale networks will contain a large number of deployed local policies. Returning all deployed policies for consistency analysis would lead to inefficiency of the consistency analysis algorithm as not all deployed policies will be related to the candidate federation policy and hence will not need to be checked for inconsistencies. A selection process can be used to return only those deployed policies that are in some way (over policy elements) related to the candidate federation policy for further analysis.

Semantic web queries can be created and used for the selection of related previously deployed local policies based on policy rule element attributes such as subject, target, and action from the federation policy. The subject, target or action can be a superset, subset or equivalent to the policy rule elements from the local deployed policies. The deployed policies need to be related by at least one of these rule elements in order for them to be selected. The execution of a semantic web query is far less costly in terms of algorithmic efficiency than performing consistency analysis on all deployed policies.

The steps in the second phase of the process which fires the policy semantic web query are outlined as:

1. Instance of candidate federation policy added to policy ontology
2. Semantic web queries retrieved from rule KB/created and stored in rule KB, related to candidate federation policy elements
3. Related deployed policy instances searched and returned using semantic web queries
4. Returned deployed policies instances used as input to policy element match algorithm

Semantic web queries are executed over the properties of concepts specified in a knowledge model (i.e domain model, policy model) and are defined over sets of triples to form a basic graph pattern. Semantic web queries take the form of *{(S),(P),(O)}* where the subject *{S}* is related to the object *{O}* via the predicate *{P}*. The predicate can be specified over either data type or object properties. The semantic web queries act as a filter returning only a subset of the complete set of deployed policies as input to the element match algorithm. This makes the policy consistency analysis process more efficient as not all deployed policies need to be analysed against the candidate federation policy. The semantic web queries are generic and can be executed over instantiations of the ontology policy model because they refer to generic policy element concepts. The semantic web queries can be extended to include triples based on concepts from the extended policy model to search for application-specific conflicts. Some concepts defined in the domain ontology model can be generalised or specialised so it is important to return these subclass/superclass concepts for further analysis. For example, policy actions can be specified over access to communication services. These communication services may form part of an implicit hierarchy of communication services. For this reason semantic queries over policy elements will be required to firstly, return the type of managed entity that the policy element is specified over and secondly, return any generalisations and/or specialisations of that managed entity from the domain model. The semantic web queries outlined in this Section 4.2.1 are focused on returning those deployed policies that are pertinent to the detection of policy inconsistencies. However,

the type of semantic web queries executed can easily be extended or modified to return deployed policies that are relevant to other types of application-independent conflicts (i.e. modality conflicts) or application-specific conflicts.

4.2.1.1 Policy Type Query

The first semantic web query outlined in Query 4.2 is a query over the policy effect element. The policy effect element dictates whether the policy is a positive or negative authorisation policy. A semantic query over the policy effect element can return deployed policy effect elements with the same or opposing policy effect element.

$$\{P_x, hasEffect, E_x\} \rightarrow \{P1001, hasEffect, Permit\} \tag{4.2}$$

4.2.1.2 Policy Action Query

A semantic web query defined over policy actions is used to retrieve all managed entities associated with a particular policy action. However, it is not sufficient to return only the domain managed entities from exactly matching policy action elements, what is also required is to return the complete hierarchy of domain managed entities from a policy action element. Returning the entity hierarchy is required in order to detect superclass and subclass policies and can be incorporated into the semantic web rule. The semantic web query specified in Equation (4.3) is an example of a typical query defined over policy actions used to retrieve all individual and groups of domain managed entities associated with a particular policy even when those domain managed entities form a hierarchy of domain managed entities.

$$Policy(p) : hasAction : Action(a) \tag{4.3a}$$
$$Action(a) : actionOver : ComServInst(csi) \tag{4.3b}$$
$$ComServInst(csi) : typeOf : ComServClass(cs) \tag{4.3c}$$
$$ComServInsts(csis) : instOf : ComServClass(cs) \tag{4.3d}$$
$$ComServClass(cs) : subClassOf : ServClass(sc) \tag{4.3e}$$
$$ServInsts(si) : instOf : ServClass(sc) \tag{4.3f}$$

The first part of the semantic web query over a policy action rule, outlined in Equation (4.3a), is issued to select all actions associated with a particular policy. This part of the query is used to associate policies with their respective policy actions. Policy actions are defined over the services/resources provided by a managed domain in order to control access to that domain's services/resources. The query snippet in Equation (4.3b), is used to make an association between a policy action and the communication service that the action is specified over, thereby essentially linking the domain model to the policy model. In this particular example, the communication service in question is XMPP communication (i.e the general concept) that includes the sub-communication services of Instant Messaging, Groupchat, and FileTransfer (i.e. specialised concepts). Actions are specified over a particular instance of a communication service. The query snippet outlined in Equation (4.3c) is part of the action rule that can be used to determine the type of DL concept class of a particular service, along with all the related services of that class. A communication service instance is a member of a communication service class defined in the domain model ontology. The query snippet outlined in Equation (4.3d) is part of the semantic web query that attempts to identify all the instances of a particular communication service specified in the domain model. The super class / subclass hierarchy of communication services is implicitly specified in the domain model. In order to return the complete service hierarchy related to this service instance the query snippet outlined in Equation (4.3e) is used. It is important to determine, firstly, the class that this individual is an instance of in the domain model and then to return all instances of that class. This will in effect bring back the complete hierarchy of a particular communication service. A communication service is a subclass of a service. Instances of the service class are also returned for inclusion in policy consistency analysis processes. This allows for the inclusion of domain managed subclass/superclass hierarchies to be included in the analysis process. The last part of the semantic web query over policy actions outlined in Equation (4.3f) returns instances of the same communication service class. This approach allows for the retrieval of individual or hierarchies of domain managed entities mentioned in policy action terms.

4.2.1.3 Policy Subject Query

A semantic web query is executed to return the set of managed domain individuals/-groups that are the subjects of deployed polices related to the candidate policy. An

example semantic web rule specified over policy subject, outlined in Equation (4.4), attempts to identify previously deployed policies that have been specified over the same domain managed entities as the candidate policy subjects. If previously deployed policies are identified then those policies are returned for inclusion in subsequent policy consistency analysis processes.

$$Policy(p) : hasSubject : Subject(s) \tag{4.4a}$$

$$Subject(s) : refersTo : Group(g) \tag{4.4b}$$

$$Group(g) : inGroup : Person(p) \tag{4.4c}$$

A management policy contains a subject element that performs an action or requests access to a service/resource. The first part of the semantic web query over policy subject, specified in query snippet in Equation (4.4a) returns deployed policy subject identifiers from the policy model related to the candidate policy over the subject element. The policy subject element refers to a group of domain entities and in this specific case the policies applies to groups belonging to the managed domain participating in a federation. The semantic web query snippet specified in Equation (4.4b) is used to retrieve the domain group identifier that is the subject of the candidate policy. Once the domain group identifier from the candidate policy subject has been identified. The last part of the semantic web query specified in Equation (4.4c) returns the individual members of the managed domain group which are specified in the policy model as the policy subjects. If a domain entity is a member of multiple domain groups and those domain groups have policies specified over them, then all those related policies are returned also for policy consistency analysis. A group is comprised of members, so the query returns the identifiers of the members of the group.

It should be noted that the *inGroup* part of the semantic web query used in the policy subject and policy target queries is not required to return individual subject identifiers, but is used to retrieve the group identifier which is useful in retrieving those policies that have been specified against groups of users (as opposed to individual users). This means policies specified against individuals and groups can be analysed against each other.

4.2.1.4 Policy Target Query

A semantic web query similar to that defined for policy subject can be specified for policy target to return the domain managed entities that are the target of a policy. The semantic web query over the policy target element is used to identify the set of individuals/groups that are the target of deployed policies related over the target element to the candidate policy. An example semantic web query over deployed policy targets, outlined in Equation (4.5), attempts to identify previously deployed policies that have been specified over the same domain managed entities as the candidate policy target. If previously deployed policies are identified then those policies are returned for inclusion in subsequent policy consistency analysis processes.

$$Policy(p) : hasTarget : Target(t) \tag{4.5a}$$

$$Target(t) : refersTo : Group(g) \tag{4.5b}$$

$$Group(g) : inGroup : Person(p) \tag{4.5c}$$

A management policy contains a target element that specifies the service/resource that is to be acted upon or accessed by a policy subject. The first part of the semantic web query specified in Equation (4.5a) returns deployed policy target identifiers from the policy model related to the candidate policy over the target element. The policy target element refers to a groups of managed domain entities and in this specific case the policies apply to groups belonging to the managed domain participating in a federation. The semantic web query snippet specified in Equation (4.5b) is used to retrieve the managed domain group identifiers that are specified in the target element of deployed policies. Once the managed domain group identifiers from the deployed policies' targets have been retrieved the last part of the semantic web query, specified in the snippet in Equation (4.5c), returns the individual domain members' identifiers of the managed domain group. If a managed domain entity is a member of multiple domain groups and those domain groups have policies specified over them, then all those related policies are returned also for consistency analysis. A managed domain group is comprised of members, so the semantic web query returns the identifiers of the all members of all managed domain groups identified.

4.2.2 Policy Consistency Analysis

The policy consistency analysis approach outlined in this section is specifically suited to analysing policies specified over either individuals or groups (even large groups of social network users and resources). Semantic web rules are executed over ontology instances to further reduce the search space and return pertinent policies for consistency analysis. The policy consistency analysis approach employs a (greedy) policy element match algorithm to identify matches between groups of policies (as opposed to pair-wise analysis) that are then notified to a policy author as potentially conflicting policies. The element match algorithm can take as input a single candidate policy or a group of candidate policies to match against a single or group of deployed policies. If a group of candidate policies are used as input the entities of the individual policies are considered in union to match against deployed policies.

The selected policies are checked for overlap based on matches (partial or fully) over policy rule elements. The types of policy rule element relationships that can be checked for are subset, superset, overlap and equality. Some of these typical policy element relationships are highlighted in Figure 4.2. The constraints on the algorithm are that it needs to be generic enough to be deployed across multiple managed domains participating in a federation and applicable to multiple policy applications. The steps in the third phase of the process which employs the policy conflict analysis algorithm process are outlined as:

1. Policy instances are reasoned over based on policy rule element relationships
2. Additional reasoning can be applied to the instances
3. Policy author is notified of reasoning results

4.2.2.1 Policy Element Match Algorithm

Typical management and security policies are composed of an arbitrary number of elements (i.e. subject, target, action, conditions, etc). Semantic web queries can be executed over the instantiations of elements from a candidate and deployed policies to determine if the policies are specified against the same domain entities. The policy element match algorithm outlined in Algorithm 4 is used to create a list of related policies. The algorithm first attempts to find an identical match between the policy

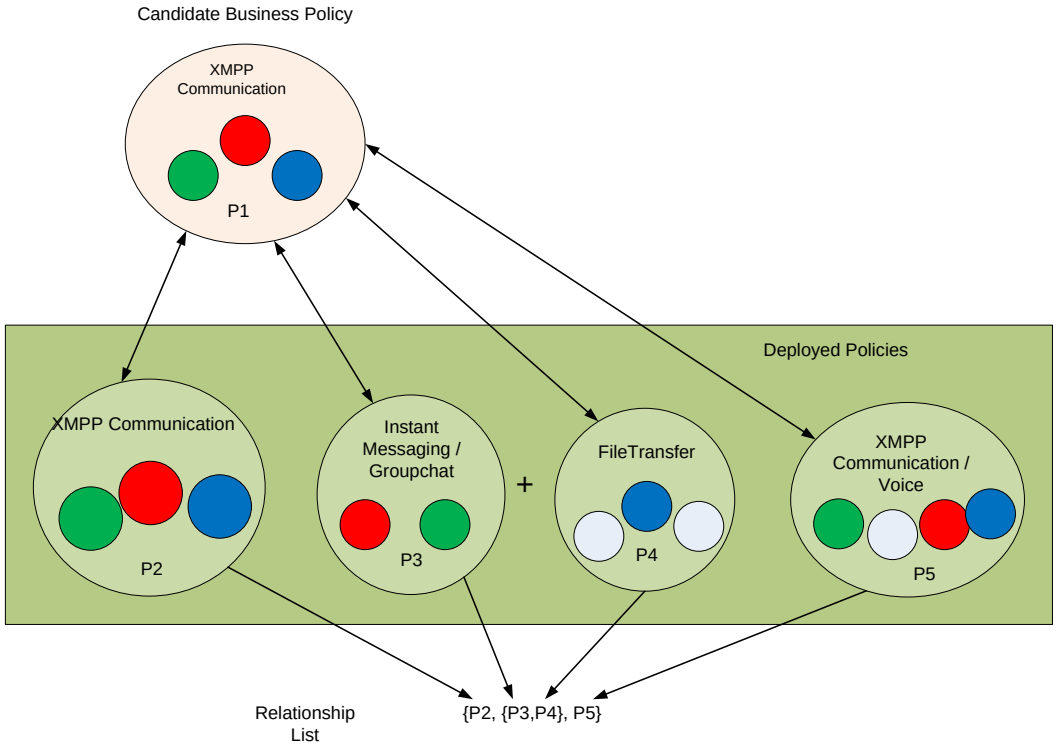

Figure 4.2: This figure illustrates some typical policy element relationships. For example, a candidate policy may be equivalent to, a superset of, or a subset of deployed policies. When certain deployed policies are considered in union, they may realise the same of opposing behaviour as a candidate policy.

element identifiers from each set (both candidate and deployed) and if a match is found between the policy elements they are added to the relationship list. Once all identical matching policy elements have been identified, the algorithm attempts to union deployed policy elements to discover if the union of partially matched policy elements matches the candidate policy element. If a match is discovered between the union of deployed policy elements and candidate policy element, the deployed policy elements are added to the relationship list and associated together. The reason for associating the union of deployed policy elements is that any future analysis on these policy elements would have to consider these policy elements together. The algorithm continues to union partially matched deployed policy elements until no more policy element matches can be identified. The final step of the process, is to intersect the policy element set identifiers

and if a deployed policy has a policy element in each set then this deployed policy (or union of deployed policies) matches the candidate policy. The policy author is notified regarding the list of matched deployed policies and can make a decision regarding the deployment of the candidate policy.

Algorithm 4 Policy Element Match Algorithm

element-Match:(CandSet, ℙdepSet) → ℙdepSet
element-Match(c, d) ≙
 $d_{list} \equiv 0$
 $U_d \equiv d$
 $d_p \equiv 0$
 do
  $U_c \equiv c$
  $d_c \equiv 0$
  do
   select an $S \in U_d$ max $| S \cap U_c |$
   $U_c \equiv U_c$ - S
   $d_c \equiv d_c \cup \{S\}$
   if ($S \equiv 0$ **and** $U_c \neq 0$)
    $d_p \equiv d_c$
    $d_c \equiv 0$
  while ($S \neq 0$ **or** $U_c \neq 0$)
  $U_d \equiv U_d$ - d_c
  $d_{list} \equiv d_{list} \cup \{d_c\}$
 while ($d_c \neq 0$)
return d_{list}

The list d_{list} contains identified matched deployed policies and is initially set to zero. The set U_d contains the set of remaining unmatched deployed policy elements. The set U_c contains the set of candidate policy elements for the algorithm to match against. The set d_c contains at each step the identified matched policy elements and may hold partially identified matches. When the inner loop is entered the maximum subset S is chosen from the set U_d. This maximum matched subset S is then removed from set U_c

and placed in set d_c. If the subset S only partially matches the identifiers of the set U_c, that partially matching identifier is removed from the set U_c while the algorithm attempts to discover if other subsets of S can match the remaining identifiers in the set U_c.

If there are deployed policy elements remaining in the set U_d, the algorithm attempts to union the remaining policy elements in U_d to discover if the union of policy elements can match the candidate policy set U_c. If the union does match, the policies in union are removed from the set U_d and are added to the list d_{list} and associated together. If only a partial match of the candidate policy is discovered and there are no more policies contained in S to complete the match, then the partially matched policy element contained in the set d_c is placed in the set d_p (where it will be removed later) and the policy element is removed from the set d_c which will allow the loop to exit. When the algorithm terminates, the list d_{list} contains a list of matched policy elements that can be used for further iterations of the algorithm or notified to the policy author.

4.2.2.2 Federated Policy Consistency Analysis

In general federation policies are realised as local policies within each managed domain participating in a federation. Federation policies can be divided into two categories, namely intra-federation policies and inter-federation policies. Intra-federation policies can be viewed as private policies in the sense that they are applied to local domain members participating in a federation and are not shared with external federation partners. Inter-federation policies are viewed as publicly distributable policies, even though they are realised as local policies within the local domain they are applied to external federation members providing some service within the local domain. These federation policies and/or their federated context are shared and possibly negotiated with other external federation members. Federation policy negotiation is required under certain circumstances such as when federation policies are created, updated or removed to ensure the consistent operation of the federation. The policy consistency analysis process will need to be employed under certain circumstances and the following cases have been identified as possible situations where inconsistency could be introduced into the system and so warrants policy consistency analysis to be carried out.

- *Local policy or policies created, modified or deleted*
  A policy author at some level of the policy continuum may change a deployed local policy. This change could have a potential impact on the federation and as such will need to employ the policy consistency analysis processes to ensure the consistency of the federation is maintained.

- *Re-negotiation of federation-level policies*
  A service provider as a member of the federation may wish to re-negotiate certain aspects of the federation. This may lead to new federation policies being specified or the updating of previously deployed federation policies. Any changes made to policies will need to be analysed for potential conflicting situations.

- *Another federation member triggers re-negotiation of federation policies*
  As mentioned in the previous case a federation member may wish to change its own local or federation policies. However, there also exists the case where another federation member wishes to implement a change to its local or federation policies. This will also require analysis of any changed local or federation policies.

Federation to local policy (deployment) negotiation proceeds in three directions:

1. The federation policy has been realised in local policies, checked for conflicts, and are ready to be deployed. A final deploy message is required to be sent between all participants before final deployment of the policies. This is to ensure that all participants are in a position to deploy related federation policies simultaneously to ensure consistency of policy systems and that no redundant policies or security risks are introduced.

2. The federation policy has been realised in local policies, checked for conflicts, a conflict has been detected so certain aspects of the policy need to be negotiated.

3. The federation policy has been realised in local policies, checked for conflicts, it is found to be not viable/possible to deploy the policy, so deployment is aborted (and possibly re-negotiation begins)

The federated policy consistency analysis process, depicted in Figure 4.3, is spread across multiple domains participating in a federation. A service provider who wishes

to implement a federation-level policy employs the policy selection and conflict analysis processes, once the federation-level policy has been deemed to be non-conflicting then the initiating service provider requests (through a secure attribute sharing process) the other service providers participating in the federation to implement the federation-level policy. Those service providers then employ the policy selection and conflict analysis processes and return a status report to the initiating service provider on whether the federation-level policy is deployable or whether further (re)negotiation of the federation policy needs to take place. The steps in the federated policy conflict analysis process are:

1. Service provider A (SPA) specifies candidate federation policy
2. Candidate federation policy verified against policy continuum levels
3. Semantic web queries executed to retrieve previously deployed policies
4. Policy element match algorithm analyses retrieved policies for inconsistencies
5. SPA creates policy deployment request
6. Policy deployment request sent through secure attribute sharing process to other federation members
7. SPB/SPN repeats Step 1, Step 2, Step 3 and Step 4
8. Status report sent back to SPA (if policy not capable of being deployed, re-negotiation of policy occurs (i.e. step5))
9. Each service provider either commits the federation policy or preforms a roll-back
10. If a commit is achieved, each service provider refines the federation policy into local policies that adhere to the local policy model.

The two processes would need to be employed independently by each participant of a federation as opposed to one service provider hosting the two processes and then deploying them remotely as service providers would more than likely not share local policies with other service providers.

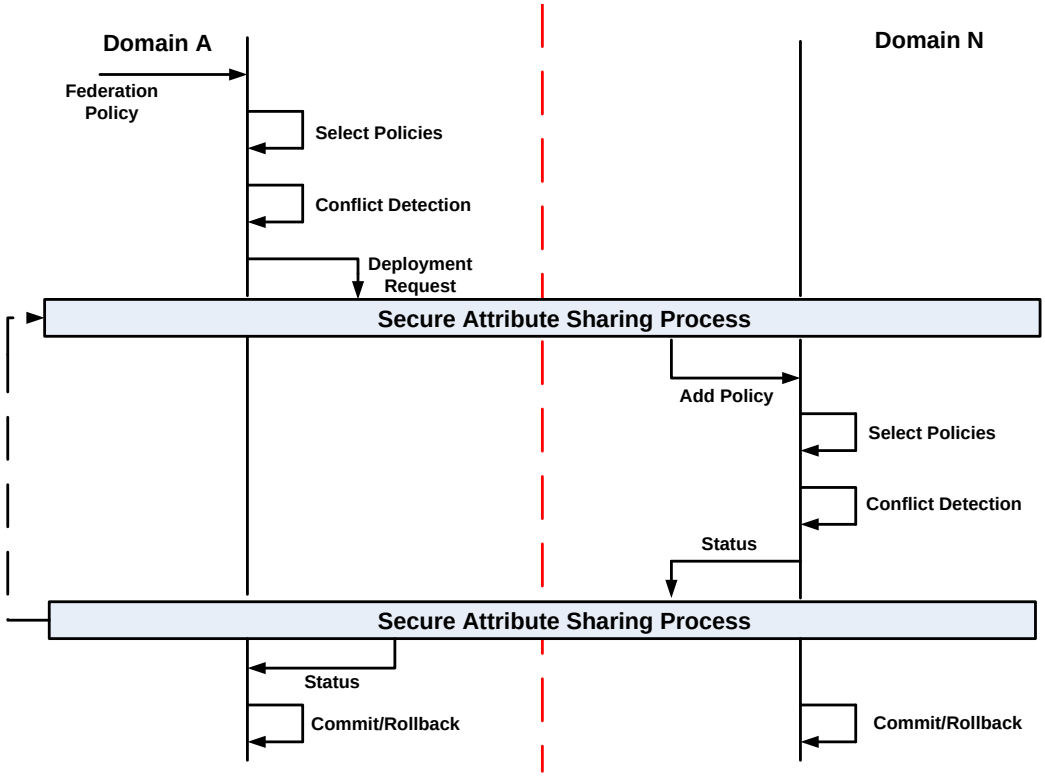

Figure 4.3: This figure illustrates the federated policy authoring process involving two federated domains (although there could be an arbitrary number of domains involved in the federation). It depicts the sequence of steps required within and between domains in a federation in order to deploy a federation-level policy and check for conflicts with previously deployed local policies.

Figure 4.3 shows the federated policy authoring process involving two federated domains (although there could be an arbitrary number of domains involved in the federation). It depicts the sequence of steps required within and between domains in a federation in order to deploy a federation-level policy and check for conflicts with previously deployed local policies. The illustration depicts a situation where *Domain A* wishes to deploy a federation policy which will require realisation in the form of lower-level local policies. A newly specified or modified federation-level policy is added. The semantic web queries select related previously deployed local policies. Once these policies have been selected, they are checked for conflicts. An example of which would be *Domain A* selects policies, checks for conflicts, if no conflicts are detected or are detected,

but resolved *Domain A* notifies *Domain N* of policies that need to be deployed (through secure attribute sharing process), *Domain N* uses semantic web queries to select local policies and then employs the element match algorithm to check for conflicts, if no conflicts have been detected or have been detected, but resolved *Domain N* notifies *Domain A* of the deployment status, so the federation policy is either committed or rolled back by each domain.

At the final stage in the process, the originating domain (i.e Domain A in this case) acts as coordinator in the policy deployment process to ensure commit/rollback decisions are synchronised and uses a two phase process similar to that used in database transactional systems. In the first phase, the coordinator sends a *prepare to commit* message to all federation members. Each federation member responds with either a *ready to commit* or *cannot commit* message back to the coordinator, depending on whether they can deploy the policy or not. In phase two, only if the coordinator has received a *ready to commit* reply from all federation members in phase one, can it send a commit message to all federation members. Otherwise, it sends a *rollback* message to all federation members. Assuming that the coordinator has sent a *commit* message, all federation members deploy the policy and then send a *deployed* message back to the coordinator. If any of the federation members fail in the commit process, for any reason, that federation member sends a *rollback* message back to the coordinator. If the coordinator receives even one *rollback* message, it asks all federation members to rollback. Otherwise the whole policy deployment process is considered successful.

4.3 Policy Consistency Analysis Use Case

This section outlines a use case to evaluate the effectiveness of the policy selection and policy element match algorithm approach in the detection of redundancy inconsistencies for a federation access control scenario.

Policy based management offers effective techniques to control the use of communication services both within and between federated domains (Feeney *et al.* (2010)). Instant Messaging communication uses XMPP (Saint-Andre (2011)) and is an example of one such communication service that can be federated and controlled through the use of an access control policy language. Federation policies are high level (business) policies that specify the overall goals of a federation. These goals are refined and realised through

the deployment of lower-level policies such as XACML (Rissanen (2013)) policies. A newly created or modified policy (known as a candidate policy) may realise the same behaviour as previously deployed policies. However, this may not be discovered until after refinement and deployment of the candidate policy, thereby introducing a redundant policy into the system. Redundant policies have an adverse effect on the performance of analysis and evaluation processes carried out for policies as they needlessly consume system resources and require additional time for processing. Additionally, a significant number of redundant policies deployed in a system impacts severely on the processing time for policy requests on policy decision points. By analysing policies before refinement, the introduction of potentially redundant policies can be drastically reduced in most cases, although there remains the possibility that some low-level policies derived from non-overlapping high-level policies my be inconsistent by overlapping themselves.

The federation access control use case requires a decision on whether the intended behaviour of a candidate federation policy has been realised (either completely or partially) by the previous deployment of access control policies (Barron & Davy (2013)). Through analysis of a policy's elements, the process attempts to identify a match (or matches) between a candidate and deployed policies over their elements that would indicate policy dominance. By maintaining application specific information in ontologies, modifications are not required to the element match algorithm in order to detect other forms of policy inconsistency or apply the dominance detection process to other policy based management scenarios. The approach exploits the powerful modelling concepts provided by ontologies to model not only the static aspects of a domain such as the characteristics of managed entities and relationships between managed entities, but also the dynamic aspects such as policies deployed to control the behaviour of the managed entities. The approach is also extensible, application independent and caters for the arbitrary levels of policy authoring required within a system by layering the ontology domain models so lower-level domain models can import and extend higher-level domain model semantics as required. The use of separate, but consolidated policy ontology models is key to modelling the various policy languages required to control a managed domain. The semantic web queries are inherently extensible and provide a minimal form of analysis across all deployed policies in order to reduce the search space for policy comparison by the policy element match algorithm thereby increasing the overall performance of the dominance detection process. Central to the policy selection process

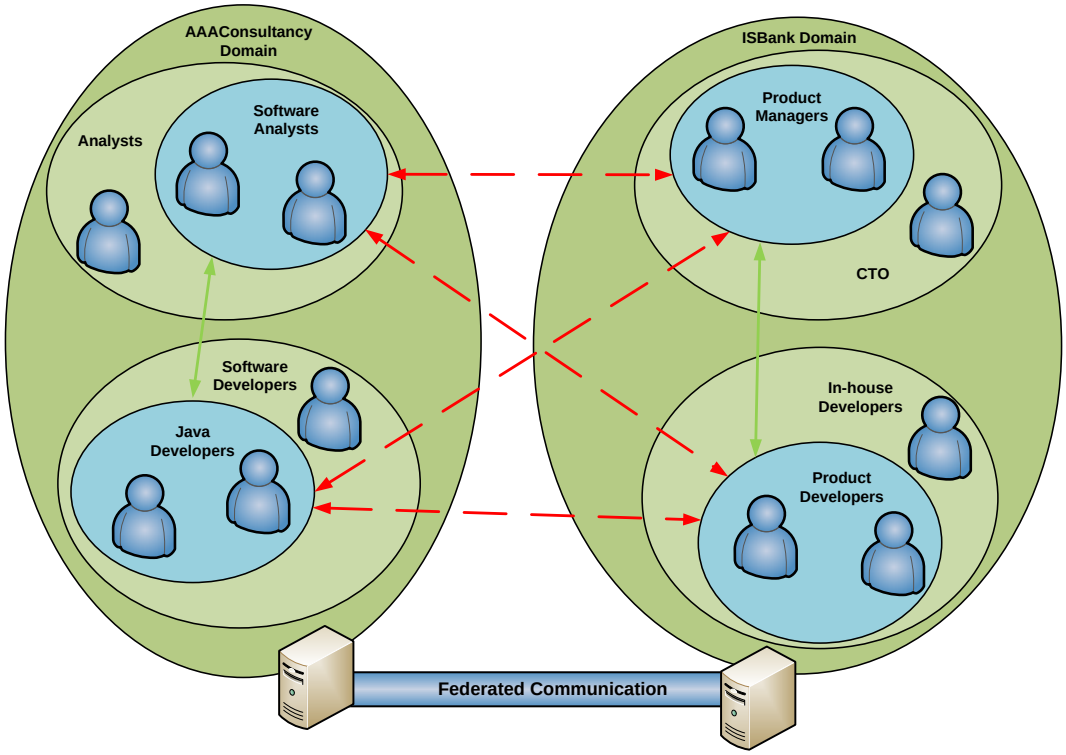

Figure 4.4: This figure illustrates a typical federation use case scenario that is based on a software consultancy company contracted to provide consultancy services to a financial institution. There are multiple groups involved in the scenario where the members of these groups require real-time communication services that cross organisational boundaries.

is the use of semantic web queries to return a much reduced set of deployed policies (pertinent to dominance detection) as input to the element match algorithm. However, other forms of policy inconsistency can easily be accommodated by the policy selection process as only minimal modification is required to the semantic web queries in order to return the relevant set of deployed policies for a particular type of inconsistency check.

The scenario depicted in Figure 4.4 is based on a software consultancy company contracted to provide consultancy services to a financial institution. Each domain uses XMPP to communicate in real-time and has a hierarchical XMPP grouping structure

based on its internal organisational hierarchy, outlined in the Table 4.1. Two financial software consultants have been selected to provide consultancy services and will need to be in direct communication with the CTO of the financial institution responsible for the project. The financial software consultants that are part of the project can be considered a sub group (FinancialSoftwareConsultants) of the overall software consultants group and likewise the financial product managers will be part of a project sub group (FinancialProductManagers). A sub group (JavaSoftwareDevelopers) containing two software developers from the SoftwareDevelopers group will work in collaboration with a sub group of developers from the financial institution's in-house developers group (FinancialProductDevelopers) to provide software engineering for the project. This will require both enterprises to federate their policy rules for their XMPP communication services. A negotiation process will be required between the two policy servers (PS) in order to create and agree upon federation-level policies, so as to manage XMPP communications traffic flowing between them. A number of requirements of the policy system are specified:

- There should be no change to client software code or server software code (this makes the system extensible to other XMPP servers and clients).
- There should be no change to the policy system itself (the policy system consists of a PEP and a PDP; the PDP loads policies from a policy repository; this repository can be a database or a LDAP system).
- The policy system should be capable of utilising Intra and Inter-domain policies.
- There should be flexible use of policy systems (this enables policy systems to be extended and if required alternative policy models could be swapped in or out of the architecture on an 'as-needed' basis).

An automated process is assumed to be in place to abstract previously deployed XACML policies and for the creation of ontology individuals based on these abstracted XACML policies. These individuals are then added to the ontology along with the newly specified federation-level policy. Reasoning is performed to determine the relationships between the elements of these policy individuals. The types of element relationships between policy rules that need to be checked for using DL concepts include superset,

subset, equality and correlation. The reasoner can be used to detect these types of relationships. Deployed XACML policies are assumed to map to higher-level business policies. These business-level policies will contain more semantic information regarding the context of the deployed XACML policy. Due to the lack of semantic information conveyed by XACML policies, these higher level policies will need to be retrieved to aid any conflict analysis processes should two policies be determined to possibly conflict by the reasoner. If the reasoner returns a policy related by an element, a second SPARQL query may be fired to retrieve the business-level policy associated with the related XACML policy.

Example candidate federation policies required to realise the scenario are specified in the Table 4.2. It should be noted that the exemplar candidate policies are high-level policies written in semi-structured natural English. However, the deployed policies have been re-written to make them more concise and would normally be specified in a much more verbose language such as XACML.

XMPP Group	**XMPP Group Members**
JavaDevelopers	JD1, JD2, JD3, JD4, JD5
SoftwareAnalysts	SA1, SA2, SA3, SA4
ProductManagers	PM1, PM2
ProductDevelopers	PD1, PD2, PD3, PD4

Table 4.1: XMPP Groups & Members

PolicyID	**Policy Goal**
CP1	SoftwareAnalysts **can** *IM* ProductManagers
CP2	SoftwareAnalysts **can** use *all XMPP comms* with ProductDevelopers
CP3	JavaDevelopers **cannot** *IM* SoftwareAnalysts

Table 4.2: Candidate Policies

Please also note in the following cases that the term *domDetect* is just a call to use the policy element match algorithm and does not hold any meaning in itself.

4.3.1 Case 1

The first case analyses candidate policy *CP1* from Table 4.2 and the deployed policy *DP1* from Table 4.3. The deployed policy *DP1* has been specified over the same subjects, targets and pertaining to the same action as the candidate policy *CP1*. If the candidate policy were to be refined and deployed, it will lead to a redundant policy in the system.

$$\begin{aligned} E \equiv \{Permit\} &= domDetect(DP1, ..., DP4) \\ S \equiv \{SA1, ..., SA4\} &= domDetect(DP1, ..., DP4) \\ T \equiv \{PM1, PM2\} &= domDetect(DP1) \\ A \equiv \{IM\} &= domDetect(DP1, DP2, \\ &= DP5, ..., DP9) \\ R \equiv E \cap S \cap T \cap A &= \{DP1\} \end{aligned}$$

4.3.2 Case 2

The second case analyses candidate policy *CP2* from Table 4.2 and deployed policies *DP2*, *DP3*,*DP4* from Table 4.3. The candidate policy *CP2* is a more generic policy than any of the deployed policies. The deployed policies refer to the same subjects and targets, but with different actions. If the deployed policies were considered in union, the intended behaviour of the candidate policy will be realised through the combination of deployed policies.

$$\begin{aligned} E \equiv \{Permit\} &= domDetect(DP1, ..., DP4) \\ S \equiv \{SA1, ..., SA4\} &= domDetect(DP1, ..., DP4) \\ T \equiv \{PD1, ..., PD4\} &= domDetect(DP2, ..., DP4) \\ A \equiv \{XMPP\} &= domDetect(DP2, ..., DP4) \\ R \equiv E \cap S \cap T \cap A &= \{(DP2, ..., DP4)\} \end{aligned}$$

4.3.3 Case 3

The third case analyses candidate policy *CP3* from Table 4.2 and the deployed policies *DP5*, *DP6*,*DP7*, *DP8*,*DP9* from Table 4.3. The subjects and targets from each policy are different, but the actions are the same. If the deployed policies were considered in union, the set of subjects and targets defined in the deployed policies refer to the same subjects, targets respectively identified in the candidate policy. The candidate policy is specified over a group of individuals whereas the deployed policies are specified over the individual members of the same groups.

$$
\begin{aligned}
E \equiv \{Deny\} &= domDetect(DP5, ..., DP9) \\
S \equiv \{JD1, ..., JD5\} &= domDetect(DP5, ..., DP9) \\
T \equiv \{SA1, ..., SA4\} &= domDetect(DP5, ..., DP9) \\
A \equiv \{IM\} &= domDetect(DP1, DP2, \\
&= DP5, ..., DP9) \\
R \equiv E \cap S \cap T \cap A &= \{(DP5, ..., DP9)\}
\end{aligned}
$$

PolicyID	Subject	Target	Action	Effect
DP1	SoftwareAnalysts	ProductManagers	IM	Permit
DP2	SoftwareAnalysts	ProductDevelopers	IM	Permit
DP3	SoftwareAnalysts	ProductDevelopers	FileTransfer	Permit
DP4	SoftwareAnalysts	ProductDevelopers	Groupchat	Permit
DP5	JD1	SoftwareAnalysts	IM	Deny
DP6	JD2	SoftwareAnalysts	IM	Deny
DP7	JD3	SoftwareAnalysts	IM	Deny
DP8	JD4	SoftwareAnalysts	IM	Deny
DP9	JD5	SoftwareAnalysts	IM	Deny

Table 4.3: Deployed Policies

4.4 Prototype Implementation

The prototype implementation includes the creation and instantiation of ontology models and is based on a modified version of the algorithm used by Barrett *et al.* (2007a). The domain and policy models were generated based on a subset of the complete ontological models outlined in in Chapter 3 and created in OWL (Motik *et al.* (2009)) using Protege (Gennari *et al.* (2003)) (a tool for creating and editing ontologies) where a domain model and a policy model have been defined for each level of each organisation. Domain model instances representing the managed entities were added to the domain ontology thereby essentially creating an instantiation of the domain model. Similarly, policy instances were created to represent the actual deployed policies in place to control the behaviour of the managed entities. Jena (McBride (2002)) is an api implemented in the Java programming language used for manipulating ontology models. In particular Jena can be used to load the required domain and policy ontologies and execute semantic queries over the loaded ontologies. The policy element match algorithm was

implemented in the Java programming language. SPARQL (Prud *et al.* (2008)) is a semantic web query language used to query RDF graphs and return the queried data. In this particular implementation SPARQL was used to query the instantiations from the policy ontology and return the policy element IDs along with the policy IDs that those elements belong to in order to build sets of deployed policy elements. The semantic web queries act as filter returning only a subset of the complete set of deployed policies as input to the element match algorithm. This makes the dominance detection process more efficient as not all deployed policies need to be analysed against the candidate policy. It should be noted that the semantic web queries defined in this section to test the prototype implementation are focused on returning those deployed policies that are pertinent to the detection of redundancy conflicts. However, the semantic web queries could easily be changed to return deployed policies that are relevant to other types of inconsistencies such as modality conflicts, application-specific conflicts, etc..

4.5 Prototype Evaluation

This section will presents results on experiments carried out to test the performance and scalability of the policy consistency analysis approach with particular emphasis on the element match algorithm. All experiments were carried out on a virtual machine running the Linux Ubuntu 12.04 operating system with a 2.7GHz processor and 2GB of RAM allocated. For each experiment, a single candidate policy element set and multiple deployed policy element sets were input into the policy element match algorithm and analysed off-line (i.e. statically) 100 times to achieve a statistical average with the resultant average then graphed. With regards to the performance of the algorithm over arbitrary numbers of policy elements, it is expected that analysis time would increase linearly as the number of policy elements is increased (i.e. performance of CA rules verses ECA rules).

4.5.1 Experiment 1

In order to determine if the number of deployed policies impacts on the performance of the policy element match algorithm. An experiment was designed that contained a single matching candidate policy and sets of deployed policies with different numbers of deployed policies in each set. The number of deployed policies in the initial set was

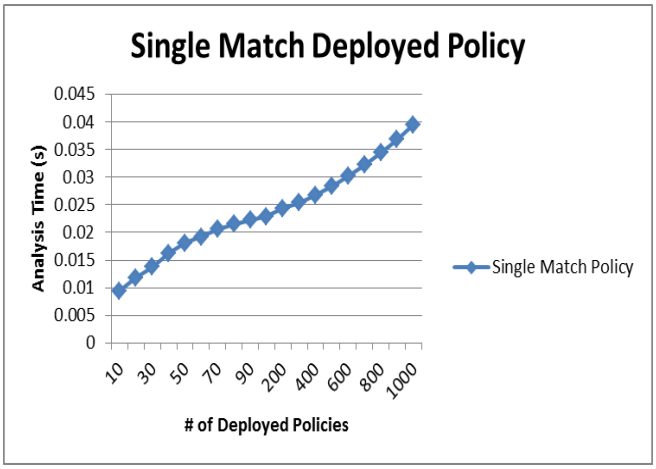

Figure 4.5: This figure illustrates the single match experiment results that indicates the time required to detect redundant policies increases marginally as the number of deployed policies is increased.

ten and this was gradually increased over a number of iterations of the experiment to a total of 1,000 deployed policies in the set. The results of this experiment are depicted in Figure 4.5 and indicate that the time required to detect redundant policies increased marginally as the number of deployed policies is increased.

4.5.2 Experiment 2

To ascertain if the number of matched deployed policies degrades the overall performance of the policy element match algorithm a second experiment was devised. Again for this experiment a single candidate federation policy was used with the number of matching deployed policies in each set increased gradually over the experiment from an initial one matching deployed policy up to 120 matching deployed policies in the set. The results of this experiment are depicted in Figure 4.6 and illustrate that the analysis time required to detect redundancy increases steadily as the number of multiple matching deployed policies is increased.

4.5.3 Experiment 3

A third experiment was devised to determine if the actual number of matched union deployed policies has an impact on the performance of the policy element match al-

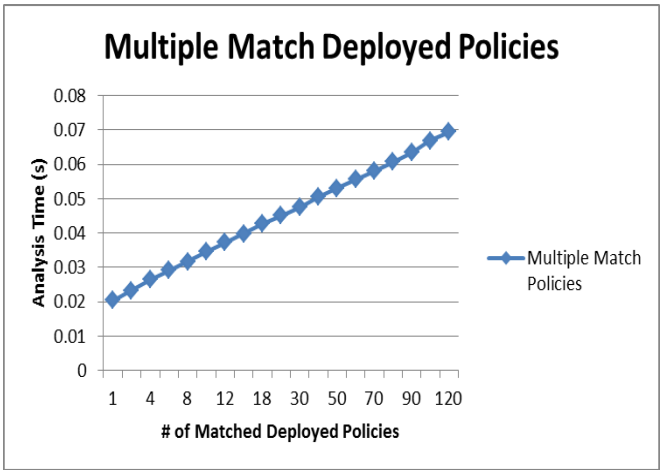

Figure 4.6: This figure illustrates the multiple matches experiment results that shows the analysis time required to detect redundancy increases steadily as the number of multiple matching deployed policies is increased.

gorithm. In this experiment the number matched deployed policies would need to be considered in union to dominate (i.e make redundant) the candidate policy. A single candidate federation policy was also used for this experiment with the number of matched union deployed policies initialised at two in the set and gradually increased over the duration of the experiment to a total of 95 matched union deployed policies in the set. The results are depicted in Figure 4.7 and indicate that the time required to detect redundancy increases significantly as the number of matched union deployed policies increases. This is due to the complexity of maintaining the identified matches between the policy element sets from multiple distinct deployed policy element sets. It is clear from the results of the experiment that an increase in the number of union matching deployed policies has an impact on the analysis time of the policy element match algorithm.

4.6 Summary and Discussion

Policy refinement is an essential part of the policy authoring process as it enables high-level business goals to be refined into low-level implementable policies. However, inconsistent policies may be introduced if policy analysis is not performed to determine

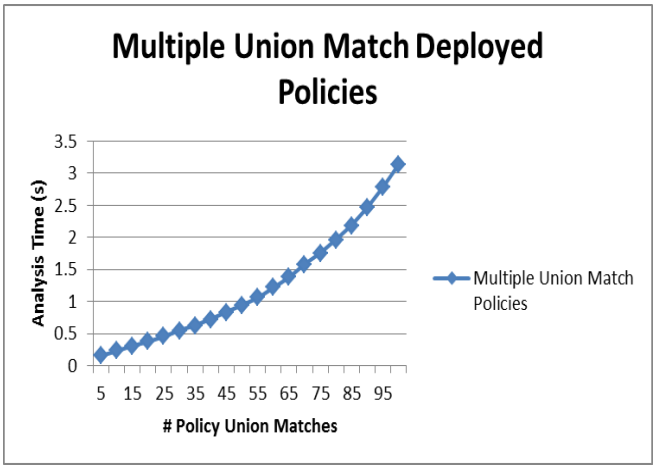

Figure 4.7: This figure illustrates the multiple union matches experiment results that indicates the time required to detect redundancy increases significantly as the number of matched union deployed policies increases. This is due to the complexity of maintaining the identified matches between the policy element sets from multiple distinct deployed policy element sets.

if the high-level goals are consistent with the previous deployment of policies. This chapter described how such policy inconsistencies between candidate federation policies and previously deployed local policies can be detected when new federation policies are specified or modified. The consistency analysis process is a generic process where deployed policies are represented semantically as instances in a domain specific policy ontology expressed in a formal representation using an ontological language such as OWL-DL as described in Chapter 3.

Research question 2, outlined in Chapter 1 asks *what conflict analysis algorithms need to be developed to assess the consistency of candidate federation-level and local policies when policies are created, modified or withdrawn.* In a federated domain environment where multiple heterogeneous policy models may be used, analysis of policies becomes a much more complex task. An ontological approach to modelling both the managed domains and their associated policies can cater for this complexity by harnessing semantic models and semantic web rules to assist in the detection of federation/local policy inconsistencies. Previous approaches to policy consistency analysis are targeted towards the detection of inconsistencies over application specific low-level policy models

(i.e. firewall, routing, etc.) that cannot be easily extended to cater for inconsistency checks over policies specified in other policy models. However, as demonstrated in this chapter taking an ontological approach to modelling the domain and using ontological models augmented with semantic web queries provides the capability to detect inconsistencies over a number of policy models that may be specified at different abstraction levels simultaneously.

The consistency analysis processes outlined in this chapter were used to analyse policies by firstly, retrieving a minimal set of deployed policies that may potentially match a candidate policy and secondly, identifying matches over specific policy elements. In the first phase semantic web queries are executed over instantiations of the ontological models to return deployed policy element identifiers. The second phase in the process attempts to identify matches between a candidate policy element and deployed policy elements. Identified matches over policy elements may be considered either complete or partial matches through the combination of policy elements. Partial matches indicate deployed policies that may implicitly realise the same behaviour as a candidate policy. However, considered solely they may not be inconsistent with a candidate policy, hence deployment of the candidate policy should proceed as normal.

The use case presented in this chapter demonstrates the extensibility and scalability of the approach to using semantic web queries combined with a policy element match algorithm as part of a policy consistency analysis process tailored specifically towards groups of deployed policies. A number of experiments were conducted to determine what impact an increase in the number of deployed policies and deployed matching union policies has on the performance of the policy element match algorithm. In the example scenario, this relates to pre-existing policies deployed for a large organisation where individuals already have policies defined for them, and a new group policy is being deployed. The results obtained from these experiments demonstrate the scalability of the consistency analysis approach even in situations where there are large numbers of union matched deployed policies. The use case makes extensive use of semantic models and semantic web queries validated through a federated XMPP enterprise communication scenario that is only concerned with the detection of dominant policies (as an example of policy inconsistency) that can cause redundancy in the local system. However, the policy consistency analysis approach can easily accommodate other types of application-independent policy inconsistencies such as modality conflicts, goal conflicts,

etc. and/or application-specific policy inconsistencies without modification to the ontological models (domain/policy ontology models) or the policy element match algorithm. A systems expert having intimate knowledge of an application domain would need to specify new semantic web queries specifically to detect application-specific inconsistencies and all that is required is to execute the new semantic web queries to return the applicable sets of deployed policies for subsequent analysis.

Most policy analysis approaches are based on pair-wise analysis of policies to detect potential inconsistencies and in some cases certain inconsistencies cannot not be detected by pair-wise analysis alone (such as when a group of policies dominate a single policy). However, the policy consistency analysis approach outlined in this chapter compares arbitrary combinations of deployed policies (i.e. can identify if a single/group of deployed policies is inconsistent with a candidate federation policy) which means the technique is capable of detecting implicit relationships between policies that cannot otherwise be detected using pair-wise analysis techniques alone. As an example, consider XMPP communication, if a policy action was specified over XMPP communication, this policy action would then apply to the specialised classes of Instant Messaging and File Transfer. Similarly, if a policy action was specified over Instant Messaging and another policy action was specified over FileTransfer, then these two policy action elements considered in union would realise the same behaviour as the more general class of XMPP Communication.

An important and often overlooked aspect of the overall policy authoring process is that of efficient and consistent policy evaluation for access requests. This is becoming an ever more important issue for enterprises in a bid to efficiently control the level of access requests generated through typical everyday use. In enterprise social networks, there is an emphasis on preserving the openness of the network while maintaining safety (i.e. privacy, access control) under certain operating environments without hindering communication flows through excessive delays caused by security checks. The next chapter (Chapter 5) presents a novel access request router framework that utilises social network analyses in order to categorise a social network's users and policies in order to evaluate specific categories of policies against specific access requests from the social network's users/groups in a bid to increase overall access request evaluation performance while maintaining an acceptable level of safety mandated by the business processes of that enterprise social network.

Chapter 5

Probabilistic Access Control for Enterprise Communication Systems

In enterprise social networks, there is an emphasis on preserving the openness of the network while maintaining safety (i.e. security, privacy, etc.) under certain operating environments without hindering communication flows through excessive delays caused by security checks. A trade-off can be observed between the responsiveness and openness of social networks in terms of communication, access to resources, etc. Based on these observations the approach described in this chapter attempts to discover an equilibrium between efficiencies gained by maintaining the openness of social networks and secure access management to appease the safety concerns of enterprises. The challenge is to strike a reasonable balance between optimisation of access request evaluation performance while maintaining sufficient safety levels for a particular enterprise social network's operating environment. This chapter will demonstrate that a probabilistic approach to policy evaluation can optimise access request evaluation performance while maintaining acceptable safety levels. Central to the approach is an Access Request Router (ARR) that was developed to route access requests to specific PDPs based on some pre-defined domain categorisation criteria. The categorisation technique is based on the principle that not all access requests require needlessly repeated policy evaluation. The approach is tailored for use within an enterprise social network by basing the categorisation criteria on results of analyses of the social network to identify the strength of relations between social network users/groups, but can be adapted for use in other scenarios.

The strength of relations (also known as tie strength) between social network users was first introduced by Granovetter (1973) during his interviews with users regarding how they came to hear about openings for their current job positions. One can posit that social network users that have a greater number of relations (i.e. higher tie strength) communicate more frequently and hence are more likely to have needlessly repeated policy evaluations performed against their access requests. The results of social network analyses is leveraged to decide how an access request should be evaluated (i.e. against a particular set of policy rules). By taking this approach, the ARR can evaluate specific sets of policies against specific social network users/groups (i.e. those that do not communicate often and may pose a higher security risk) and limit the level of policy evaluation performed against other specific users/groups (e.g. those that communicate often) thereby increasing overall policy evaluation performance while maintaining an acceptable level of safety. The access request router acts like a load balancer for access requests and can be harnessed in two very different situations 1) it is always on and continually balancing access requests or 2) it is only used when the access request load on PDPs passes a certain threshold. Due to the fact that the access request router balances safety against performance, enterprises may not see the benefit of trading performance for safety under all operating scenarios. However, under certain operating scenarios when given a choice between the whole network becoming overly congested from too many access requests made at a single point in time and the complete network grinding to a halt or inconsistently evaluating (permitting/denying) a certain number of less important access requests, but being able to keep the network up and running, enterprises can choose to balance performance against safety.

The rest of this chapter is structured as follows, Section 5.1 describes the access request router's framework. This includes a description of the functionality of the access request router and the types of categorisation leveraged by the access request router to intelligently route access requests, namely social network categorisation and policy categorisation. Section 5.2 describes a prototype implementation of the access request router framework designed specifically to evaluate the efficiency and safety of taking a probabilistic approach to access request evaluation. Section 5.3 discusses experimental results obtained from evaluation of the access request router framework with regards to performance and safety metrics. Finally, in Section 5.4 the contributions of this chapter are summarised and analysed.

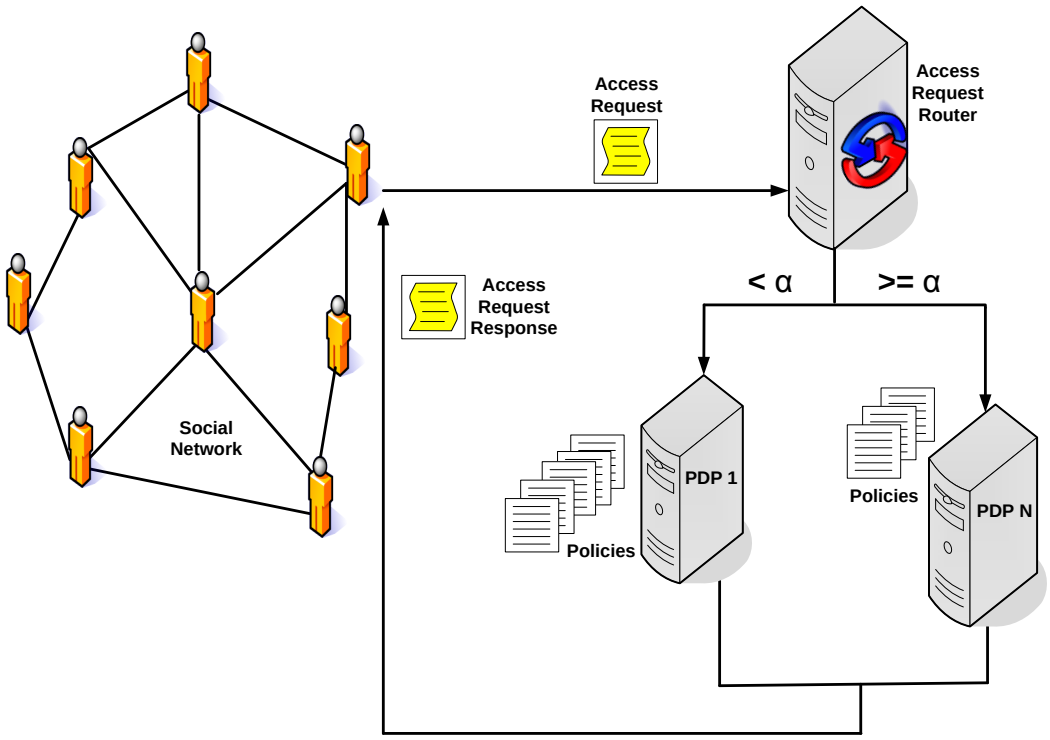

Figure 5.1: This figure illustrates the access request router framework applied to a social network scenario. Some typical components of the framework include an access request router coupled with an arbitrary number of unmodified policy decision points with deployed access control policies pre-loaded.

5.1 Access Request Router Framework

This section describes the access request router framework; depicted in Figure 5.1, that includes an access request router coupled with unmodified policy decision points with deployed access control policies pre-loaded. The access request router framework can be deployed in most typical enterprise social network scenarios with no modifications to code required. The access request router is highly extensible in that it only relies on categorisation of the enterprise social network (using standard social network analysis techniques) and deployed access control policies (these need to be categorised by a policy expert) from the operating environment in which it is to be deployed.

5.1.1 Access Request Router

The access request router's function is to determine which PDP a request should be forwarded to and forward the request accordingly. Upon receipt of an access request, the access request router extracts specific attributes from the request (e.g. subject, target, etc.) and uses this information to determine the relation strength metric value (i.e. tie strength) between the subject and target (i.e. users, groups, resources, etc.) of the access request. Using the value obtained for the relation strength metric, a (boolean) comparison can then be made against a pre-specified (user/group) trust value parameter in order to determine the PDP to which an access request should be routed and ultimately the policies that will be evaluated against that access request.

Definition 1: *(Relation strength metric) A relation strength variable γ is defined to represent the aggregate number of relations between two users of the social network where $\gamma(s,t) \rightarrow R$, $s \neq t$, $R \in [0,1]$.*

Social network categorisation was chosen based on the aggregate strength of relations between users of the social network. Some typical examples of relations are *supervisorOf, colleagueOf, worksOnProjectWith, co-authoredPaperWith, etc,*. The aggregate relation strength metric is determined by the total number of relations that exist between any two users in the social network. The relation strength metric was normalised for social network users between 0 and 1 inclusive. After normalisation of the social network relation metrics has been performed γ will contain a value between [0,1] inclusive.

Definition 2: *(Cautious value metric) A cautious variable β is defined as an artificially weighted value in the range [0,1] inclusive. The cautious value metric can skew between conservative security where β equates to 0 and liberal security measures where β equates to 1.*

β is introduced to add an extra level of security into the process. As an example of its practical use, in a health care environment there is a legal requirement to exercise extreme caution over access to patient's records as unauthorised access (i.e information leakage) can have serious legal implications, so in an operating environment like this β should be set to 0 for maximum security, whereas in a research institution project collaboration scenario security over communication is not as critical so β should be set to 1 for minimum security.

Definition 3: *(Trust value metric) A trust value α is defined as a constant value used to indicate a benchmark level of trust between social network users/groups. α is compared against the relation strength metric value γ in order to route an access request to a particular PDP. The trust value α should be in the range of [0,1] inclusive.*

The trust value α is a constant value that should be defined by a policy administrator and used to manipulate the category of social network user's access requests that get forwarded to a particular PDP. By toggling α between a low and high value, a policy administrator can vary the volume of access requests that are forwarded to particular PDPs.

Definition 4: *(Probability value metric) A probability value ρ is defined as an artificially weighted value in the range of [0,1] inclusive used to introduce some randomness into the relation strength metric calculation process.*

In typical social networks, some users have very high numbers of connections to other users. The justification for the introduction of the random ρ variable is to counter act the fact that these social network users with high relation strength metric values may not have their access requests evaluated under any circumstances, so the variable ρ introduces some randomness into the process in a bid to mimic real social network behaviour.

$$\frac{(\rho)(\beta)+\gamma}{2} \leqslant \alpha \tag{5.1}$$

The routing decision for an access request is calculated according to Equation (5.1). Based on the outcome of this equation an access request is routed to a particular PDP. For example, if an access request is calculated to be lower than (or equal to) α then the access request can be routed to the PDP that contains all deployed policies; whereas, if an access request is calculated to be greater than α then the access request can be forwarded to a PDP with a default policy action (either permit or deny).

5.1.2 Social Network Categorisation

Analysis of enterprise social networks can highlight the users, groups and resources most prominent (i.e. the active network) in the social network. For example, the authors in Smith *et al.* (2009) analysed an organisation's social network of employees and more specifically the frequency of modifications made by employees to the organisation's wiki pages. This form of social network analysis can be harnessed and utilised by policy based management processes in order to intelligently apply policies to specific groups of social network users. The enterprise social network models communication and access to resources through the use of any type of social media platform or social media tools for the purposes of collaboration and information dissemination. A social network can be categorised into an arbitrary number of categories based on application-specific purposes or purely to increase performance. Extensibility is introduced at this point as system administrators have full control over how the social network is categorised. Categorisation involves ordering the users/groups/resources and can be based on many factors such as *identifying individuals with highest number of relations, most frequent access/modifications (read or updates) performed on social network resources or most frequent communication between users, etc.* Social network categorisation can be used to divide users into particular groups that can help decide if particular communication patterns are more sensitive than others.

5.1.3 Policy Categorisation

The categorisation of polices has been carried out in a number of different areas, for access control policies (Marouf *et al.* (2011)), for grid-based policies (Rajendran & Huber (2011)) and for packet classification for firewall policies (Dong *et al.* (2006)). The categorisation process involves splitting a group of policies and/or policy sets into various categories based on some pre-defined criterion such as *prioritisation of the policies, the policy domain (e.g. security, network management, etc), be application-specific (e.g. tied to the social network) or based on request evaluation time* thereby directly linking the categorisation of policies to policy evaluation performance. Again extensibility is introduced at this point as a policy expert is responsible for choosing the number of policy categories and the criterion under which policies should be categorised. The policy categories are aligned to the social network user categories thereby allowing a policy

expert to dictate which category of policies should be evaluated against particular social network users. The overall aim of policy categorisation and alignment to the social network categories should be to increase access request evaluation performance. However, there are some constraints for a policy author with respect to policy categorisation. Policy categorisation is a complex process with difficult challenges such as considering policy set execution strategy semantics (i.e. first-applicable, most-specific, etc.). An access request may require a history of previous policy evaluation decisions (i.e. state information). This information would have to be distributed between PDPs to ensure consistent access request evaluation. To overcome some of these policy categorisation limitations policies were categorised into two categories only, one category contains all deployed polices while the other category contains only a single default policy (either permit or deny).

5.1.4 PDP

The access request router is application-independent in that it does not depend on any specific access control policy language or any particular policy decision point implementation (i.e whether the policies comprise of string literals or are numerical, etc.), as a result the framework is adaptable for use by the most common types of PDPs (i.e Sun XACML, XEngine, etc.), under the assumption that the PDP merely loads a pre-defined category of policies (or policy sets) or multiple categories of policies. The approach taken proposes the use of separate PDPs, where one PDP is assigned specifically for each category of policy and/or policyset. This should yield a performance boost when compared with having a single PDP and continuously having to load and unload policies/policysets as the category of policy changes on a per-request basis. The choice of which policy category is loaded into a PDP for request evaluation is pre-defined, arbitrary and should be configured by a policy expert.

5.2 Prototype Implementation

The prototype implementation to demonstrate the effectiveness of this approach included development of the following access request router framework components. The access request router was developed using the Java programming language. It consists of a number of classes for processing access requests to parse subject details, query

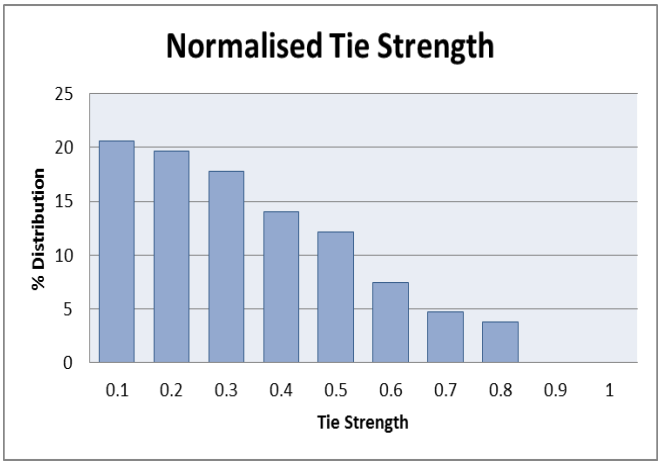

Figure 5.2: Tie Strength Distribution

a database for social network user tie strength, determining the overall tie strength and routing the request to a particular PDP. Sun's XACML 2 (Proctor (2006)), is an open-source implementation of the OASIS XACML standard Rissanen (2013), written in the Java programming language. The XACML policy server consists of two main components, a PEP for creating policy requests and a PDP for loading policies and making policy decisions. The PDP loads policies from a policy repository and makes decisions based on loaded policies and requests forwarded to it from the PEP. The policy repository can be a database or a LDAP system. Two Sun XACML PDPs were deployed in the network and interfaced with the access request router. One PDP had all deployed policies pre-loaded while the other had a single default policy pre-loaded that either permitted or denied all access requests that it received. A third Sun XACML PDP was deployed in the network (this PDP was not part of the ARR framework) to use as a benchmark for evaluation experiments. For a realistic policy set benchmark the 'continue-a' policy set referenced in Fisler *et al.* (2005) was used which is based on a conceptional conference submission system and includes four hundred representative access requests comprised of single access requests and multi-access requests.

An anonymised social network data set taken from Viswanath *et al.* (2009) was used as a realistic simulation of a social network. This data set contained a list of all the user-to-user links from the Facebook New Orleans network and more importantly the number of posts made by users to other facebook users' walls. This provides interesting and

more useful statistics on the activity of users within the facebook network as opposed to just analysing static user-to-user links. Specifically, the data set contained an identifier (node ID) and a wall post counter that represented the number of wall posts facebook members had made to other facebook members' walls (this metric was used to define the relation strength) which is a clearer indication of closer interaction between facebook users. Figure 5.2 shows the distribution of users with varying relation strength on a scale from 0.1 to 1.0 inclusive after normalisation has been performed. It is clear from Figure 5.2 that the majority of relation strengths are clustered around 0.1 to 0.5 inclusive. The facebook users' node identifiers (node ID) were mapped from the social network data set to access request policy subjects from the 'continue-a' access request set which was used for all experiments.

5.3 Prototype Evaluation

This section presents initial results of a number of extensive experiments carried out to evaluate the performance and safety of the probabilistic access request router approach. All experiments were carried out on a virtual machine with 2.7GHz processor, 2GB of RAM running a Linux Ubuntu operating system. According to the XACML standard (Rissanen (2013)), an access request can be evaluated to either explicitly *permit* or *deny* an access request or considered *not applicable* or *indeterminate* to an access request. In the straight forward case of a *permit* or *deny* decision, the access request is accordingly permitted or denied. In cases where the decision is evaluated to either *not applicable* or *indeterminate* an assumption is made that the default decision is upheld which in most cases is a default deny decision. The Sun XACML PDP implementation was used as a benchmark to analyse the performance and safety achieved through each of the trust values. Experiments were devised to determine the length of time required to evaluate random access requests over each trust value with the precise number of permitted, denied or not applicable access requests recorded using both the Sun XACML PDP and the access request router PDP. Each access request set was sent through the Sun XACML PDP and ARR PDP with results obtained from the evaluation of each access request. These results could then be used to gauge the overall evaluation time and safety level achieved over each of the trust values using the ARR approach. For each experiment, all simulated access requests contained only single-valued attributes

for subject, resource and action request elements. Each experiment was repeated one hundred times with the average of each experiment graphed.

5.3.1 Performance

In an access control scenario (using a typical Sun XACML PDP implementation) access requests are received sequentially on a first-in first-out (FIFO) basis where each request is evaluated (through brute force method) sequentially against all deployed policies. The Sun XACML PDP in not adept to deal with bursty access request traffic that is typical in social network (near real-time communication) scenarios. The technique described in this chapter opts to take a different approach to access request evaluation by making informed decisions regarding which access requests should be evaluated based on the results obtained from analysis of the social network. The performance of the access request router is based on a number of underlying factors such as the number of social network and policy categories defined. The request evaluation rate was measured which indicates the number of request evaluations performed contrasting both the Sun XACML PDP and the ARR PDP. For the ARR PDP the request evaluation rate was measured specifically over each of the trust values and a comparison of the results using both approaches was made. This experiment was designed to measure the request evaluation rate over each of the trust values. As most users have at least some degree of relations with other users (i.e. most users would at least fall into the *0.1* through to & *0.3* trust level range. The default action can be either severely restrictive by having a default deny policy or openly permissive by having a default permit policy in place. When the trust value is low (i.e. most users fall into these categories) the level of request evaluation performed is at a minimum. Most access requests are evaluated against the default policy which can be either a permit or a deny action.

The request evaluation rate was measured using a default permit policy as in extreme cases where a low trust value is set, all requests are permitted (i.e. no request evaluation is performed) and when the trust value is high, the request evaluation rate is on a par with the Sun XACML PDP (i.e. the same number of requests are evaluated). The results of this experiment, depicted in Figure 5.3, demonstrate that when the trust value is set to a low value *0.1* through to & *0.3* minimal request evaluation is performed, the request evaluation rate steadily increases as the trust value is increased for trust values *0.4* through to & *0.7*,while at *0.8* through to & *1.0* request evaluation is on a par

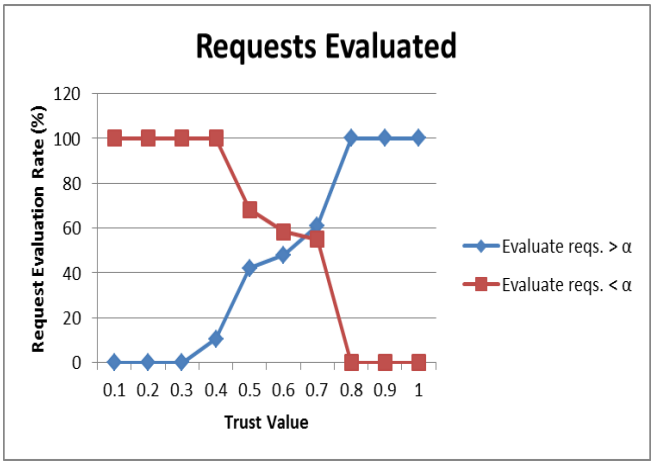

Figure 5.3: Trust Value Request Evaluation Rate

with the Sun XACML PDP with the maximum number of requests being evaluated. Experiments were devised to analyse the impact of the trust value weighting on the performance of access request evaluations. The experimental results obtained, depicted in Figure 5.4 indicate that certain trust values achieve a significant performance gain such as the trust values *0.1* through to & *0.5* with a performance improvement of 60% or higher; while other trust values such as *0.6* through to *1.0* perform quite poorly with a performance improvement of 40% or less. One can postulate that the justification for this is that most tie strength values fall in the range of [0.1, 0.5], see Figure 5.2.

A default deny policy was observed to produce the exact same performance improvement as a default permit policy with regards to the policy evaluation rate. It was noted that the optimum trust value for this experiment was determined to be *0.5* as it is produces a 60% improvement in request evaluation performance when compared with the Sun XACML PDP, although it was also noted that there is a minimal negative impact with regards to the safety (i.e. false negative evaluations) of evaluation decisions for achieving this performance boost.

5.3.2 Safety

The proposed technique either evaluates an access request against all deployed policies or a single default (either permit or deny) policy evaluation is performed against the access request, so it is imperative to investigate the level of safety achieved using the

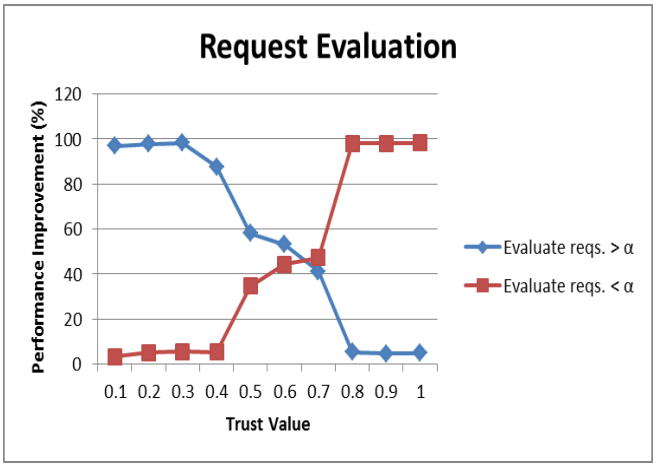

Figure 5.4: Trust Value Performance Improvement

approach so as not to compromise security or privacy. Comprehensive experiments were devised to analyse the safety of the probabilistic access request approach using random access requests that were run through both the Sun XACML PDP and access request router PDP implementations with the number of access requests evaluated to permit, deny, not applicable or indeterminate recorded. Each PDP maintained a counter to record the exact number of permitted, denied, not applicable or indeterminate access requests. This provided fine grained details regarding outcomes of policy evaluations on access requests from each of the PDPs. Experiments were conducted to analyse the sensitivity of each trust value weighting against the safety achieved for each access request. The ARR framework allows the forwarding of access requests to particular PDPs. Two test cases were evaluated. In the first case, all access requests with a relation strength metric value less than the specified trust value would be evaluated against all policies while access requests with a relation strength metric value greater than the specified trust value were evaluated against the single default policy (permit policy for this case). For the second case, all access requests with a relation strength value less than the specified trust value were evaluated against a single default policy (deny policy for this case), while access requests with a relation strength metric value greater than the specified trust value were evaluated against all deployed policies.

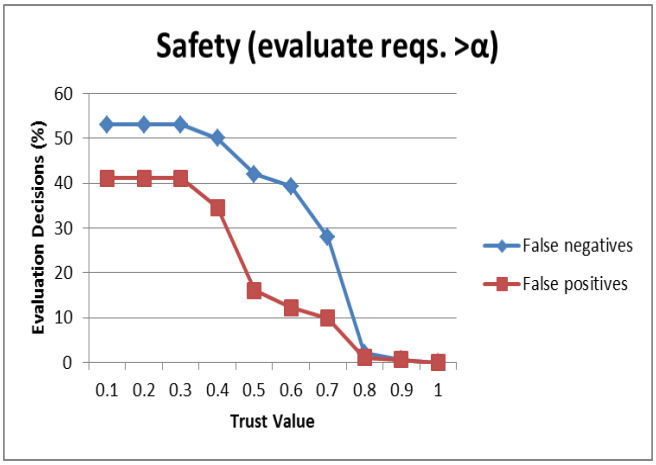

Figure 5.5: Trust Value Safety

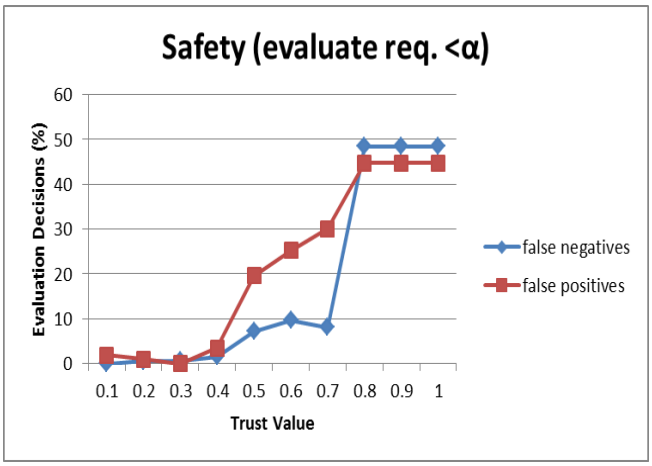

Figure 5.6: Trust Value Safety

Experimental results obtained from the two cases, depicted in Figures 5.5 & 5.6, demonstrate that there is a marginal difference in the number of access requests evaluated to either false positives or false negatives when the default policy is used for evaluation of access requests above and below the trust value. For example, while the trust value *0.1* provides the best performance, see Figure 5.3, it produces more incorrect request evaluations with regards to producing more false negatives with over 50% false negatives for a default deny policy and over 40% false positives for a default permit policy when comparison is made between the ARR PDP and Sun XACML PDP. While conversely, the trust value *1.0* provides the worst performance for access request evalu-

ation with request evaluation time to the maximum, see Figure 5.3, however, it provides a level of safety equivalent to that achieved by the Sun PDP by evaluating fewer access requests to false negative or false positive decisions. It was also noted that the trust value *0.5* provides an optimum level of performance improvement while maintaining a reasonable level of safety using a default permit policy that produces roughly only 16% false positives for achieving a 60% increase in request evaluation performance.

5.4 Summary and Discussion

Enterprise social networks are becoming an increasingly important tool for enterprises to harness in a bid to increase information dissemination and collaboration among employees and partners. These networks are trying to mimic the open behaviour (i.e. social utility) of public social networks, but operate within business processes that have strict security and privacy concerns. Communication between users/groups or access to enterprise resources needs to be securely managed using access control policies. An enterprise's policy systems have limited resources (i.e, RAM, CPU, etc.) to cope efficiently with large numbers of access requests generated through social network use. This chapter described an approach to probabilistically optimise access request evaluation by tailoring request evaluation towards specific users/groups of the social network deemed to pose a greater security threat. The approach harnessed analyses of a social network to categorise the users/groups and access control policies applicable to those users/groups to increase access request evaluation performance while maintaining adequate security and privacy levels. Research question 3, outlined in Chapter 1 asks *"What strategies can federating enterprises adopt to increase access request evaluation performance while maintaining an acceptable level of risk?"*. A trade-off was observed between the responsiveness and openness of social networks in terms of communication, access to resources, etc. Based on these observations the approach attempts to discover an equilibrium between efficiencies gained by maintaining the openness of social networks and secure access management to appease the safety concerns of enterprises. The challenge is to strike a reasonable balance between optimisation of access request evaluation performance while maintaining sufficient safety levels for a particular enterprise social network's operating environment. As a solution the approach outlined in this chapter proposed a framework to ensure that access requests received from specific

social network users (e.g. most active users generate more access requests) are processed more efficiently by performing less (repeated) policy evaluations on these requests while access requests from other social network users (e.g. whom do not communicate frequently) are evaluated against more policies as these social network users may pose a greater safety risk overall. Extensive experiments were conducted to analyse the efficiency of the approach and to determine if policy evaluation outcomes were affected by using the approach as in some instances not all policies are available for evaluation against each access request. A balance is required between increasing access request evaluation performance and maintaining particular safety levels by taking a probabilistic approach to access request evaluation as depending on the strength of the relations between enterprise social network users/groups will ultimately decide the category of policies that will be evaluated against any given request. The ARR framework was implemented and evaluated against real social network data and experimental results indicated that the probabilistic access request approach can easily deliver modest policy evaluation performance gains for access requests while maintaining reasonable access control safety levels.

Chapter 6

Conclusion and Future Work

This thesis presented contributions to the research area of policy based management, specifically to progress upon the methods currently used for policy authoring and in particular policy specification and conflict analysis of federation policies.

Chapter 2 presented a state of the art review of research in the area of policy based management, with particular emphasis on policy authoring specifically, policy specification, consistency analysis, and refinement aspects to support federation policy authoring processes. The chapter then discussed technologies such as information models, ontological models, and semantic web technologies to provide support for policy specification and consistency analysis processes. This motivated the requirement for policy specification and policy consistency analysis techniques that are specifically tailored for use within federated domain environments. Following on from analysis of the current state of the art in policy based management and the lack of support for federations, the requirements of the research conducted in this thesis were then presented.

Chapter 3 presented extensions to the DEN-ng information model and policy continuum model to facilitate federation policy specification and consistency analysis processes. A subset of the DEN-ng information model, the DEN-ng federated domain model has been transformed into an OWL-DL representation that facilitates additional structural aspects of managed domains to be modelled. The OWL-DL model can then be enhanced by defining dynamic relationships that hold between managed entities within the managed domains. This OWL-DL model can then be queried and reasoned over using semantic web technologies to detect implicit relationships that hold between domain managed entities and highlight potential inconsistencies that may occur among

groups of deployed policies. As an initial step toward developing federation policy specification and consistency analysis tools, this chapter introduced a federation policy authoring process whose steps ensure that local policies are kept consistent with federation policies as individual policies are created, modified or withdrawn. In particular, a federation policy specification process was described that uses model driven development techniques to specify multiple low-level enforceable policies from a single high-level candidate federation policy that adheres to the semantics of a sample federation model. Finally, an exemplar federated policy based management test-bed was described that was developed and implemented in order to evaluate newly developed processes and algorithms outlined in this thesis.

Chapter 4 framed policy consistency analysis as an optimisation problem and then described how inconsistencies between candidate federation policies and previously deployed local policies can be detected when new federation policies are specified, modified or removed. The federation policy consistency analysis process takes a two phase approach, a policy element selection phase and a policy element match phase. The consistency analysis processes firstly uses semantic web queries to retrieve a minimal set of deployed policies that may potentially match a candidate federation policy and secondly identifies matches over policy elements that may potentially indicate cases of policy inconsistency. Finally, a number of federation use cases were provided to evaluate the effectiveness of the policy selection and policy element match approach in federation scenarios.

Chapter 5 presented a novel access request router framework that leveraged social network analyses to categorise an enterprise social network's users and policies in order to evaluate specific categories of policies against specific access requests from the social network's users/groups in a bid to increase overall access request evaluation performance while maintaining an acceptable level of risk mandated by the business processes of that enterprise social network. The probabilistic approach taken ensures that access requests received from specific enterprise social network users (e.g. most active users generate more access requests) are processed more efficiently by performing less (repeated) policy evaluations on these types of requests while access requests from other enterprise social network users (e.g. whom do not communicate frequently) are evaluated against more policies as these social network users may pose a greater safety risk overall. This chapter

concluded with analysis of extensive experiments designed to evaluate the performance and safety aspects of taking a probabilistic approach to access request evaluation.

The hypothesis of this thesis is that policy based management techniques can be leveraged to support the federation of service providers thereby minimising the level of human participation in the federation set-up and maintenance phases through the use of management (federation) policies.

6.1 Appraisal of the Thesis

This thesis is reviewed from the perspective of applying standard policy based management techniques to facilitate the authoring and evaluation of federation policies. Chapter 2 discussed current policy based management processes which at present lack the ability to support the federation of service providers as these systems were designed to operate within single domain environments and not within federations. In particular policy based management systems are heterogeneous and as a result use management policies that may be specified in different policy languages which is problematic when participating in a federation as this heterogeneity makes it even more difficult to consistently specify federation policies as they may be related to different processes/managed entities. Policies may be enforced at various abstraction levels within a managed domain, (i.e. system, network, etc.) and as a consequence the specification of policies at arbitrary levels has an adverse affect on the enforcement of policies as it becomes much more difficult to maintain the consistency of policies at any one level. The federation policy authoring process encompasses a policy specification process for the derivation for low-level implementable policies from a high-level policy based on model driven development techniques and consistency analysis processes that leverage ontological models and semantic web technologies both specifically targeted towards the consistent specification and analysis of federation policies. A probabilistic approach to evaluate access requests from specific users/groups of the enterprise social network is provided to ease the burden of access request evaluation performance on policy evaluation systems. This approach can increase access request evaluation performance while maintaining an acceptable level of access control and security mandated by the internal business processes of the enterprise social network. The main contribution of the thesis is a federation policy authoring process that outlines the steps to be taken when local or

federation-level policies are created, modified or withdrawn that specifically includes policy specification, consistency analysis, and policy evaluation as processes that are central components of an overall policy authoring framework for federation policies. The main advantages of the federation policy authoring and analysis approach taken are:

- **Extensible**

  The approach presented in this thesis is extensible, as ontological models are used to model both the managed entities within a managed domain and the management polices specified to control the behaviour of those domain managed entities. The ontological models and semantic web rules can easily be modified and extended by a systems expert to identify various types of domain-independent and application-specific policy inconsistencies (deontic conflicts, redundancy, conflicts of interest, etc.) defined in the literature. The semantic web queries are used to retrieve those deployed policies that mention terms related to the terms in the candidate federation policy therefore reducing the search space to those deployed policies that can feasibly match the candidate federation policy. The approach also introduces flexibility at this step, as the semantic web queries can readily be adapted with minimal modification to any policy application domain or policy controlled environment. The policy element match algorithm has the ability to discover the combined behaviour of sets of deployed policies and remains extensible to many application domains due to the fact that essential knowledge specific to the managed domain is stored independently in a knowledge base that is not dependent on implementation specific details of the policy element match algorithm. In this thesis ontological models were harnessed to represent both the managed domain and its associated management policies due to the fact that ontological models can be extended with additional managed domain specific semantics if (and when) required and similarly the policy model can be swapped out in favour of other policy models or extended if required.

- **Efficient**

  The approach presented in this thesis is efficient, as groups of deployed polices are compared simultaneously to detect occurrences of potential policy inconsistencies

as opposed to pair wise comparison used by current policy analysis processes defined in the literature. This makes policy analysis a much more efficient process as less policy comparisons are required to detect typical policy inconsistency cases. Another point to note is that by analysing policies at a higher abstraction level policy inconsistencies can be detected before they are introduced into a managed system where they may prove much more difficult to detect and would require more expensive analysis processes and longer detection time. The semantic web queries are inherently extensible and provide a minimal form of analysis across all deployed policies in order to reduce the search space for policy comparison by the policy element match algorithm thereby increasing the overall performance of the policy inconsistency analysis process. The access request router approach delivered significant policy evaluation time gains for access requests by harnessing enterprise social network analyses to categorise an enterprise's social network users/groups, and policies in order to evaluate specific categories of access control policies against specific access requests from the enterprise social network's users/groups in a bid to increase overall access request evaluation performance while maintaining an acceptable level of risk mandated by the business processes of the enterprise social network. By taking this approach, access requests from those enterprise social network users/groups that are deemed to pose a greater security risk require policy evaluation against more policies, while access requests from enterprise social network users/groups that are not deemed to pose a security risk have a default policy (default permit/deny) evaluated against their requests for access in the enterprise social network.

The main advantages of the federation policy authoring and policy evaluation approaches as highlighted above have been discussed in detail from comparisons of current approaches as described in Chapter 2. The work presented in this thesis has achieved the requirements set forth to 1) *implement a federation policy authoring process to ensure consistent specification and consistency analysis of federation policies within federated domain environments*, and 2) *implement a policy evaluation process that can easily identify an optimum balance between increasing access request evaluation performance while maintaining secure access control and privacy levels.* The research questions of this thesis as outlined in Chapter 1 are now reviewed.

1. *How can a policy authoring process be defined to support the consistent specification of federation-level policies by multiple constituencies of policy authors at arbitrary abstraction levels aimed at controlling federations of services and resources?*

   This question is addressed in Chapter 3 where extensions to the DEN-ng information model and policy continuum model are provided to cater specifically for the consistent specification and consistency analysis of federation policies. The DEN-ng federated domain model has been specified in an OWL-DL representation that allows for additional domain specific semantics (entity relationships, etc.) to be added to the model as required. Chapter 3 introduced a federation policy authoring process whose steps ensure that local policies are kept consistent with federation policies as individual policies are created, modified or withdrawn. Specifically a federation policy specification process is introduced that makes extensive use of model driven development principles to derive multiple low-level (possibly heterogeneous) enforceable policies from a single high-level federation policy that adheres to a federation domain model and demonstrates the usefulness of the new DEN-ng and policy continuum extensions.

2. *What conflict analysis algorithms need to be developed to assess the consistency of candidate federation-level and local policies when policies are created, modified or withdrawn?*

   This question is addressed in Chapter 4 which describes how inconsistencies between candidate federation policies and previously deployed local policies can be detected when new federation policies are specified, modified or removed. The consistency analysis process is a generic process where deployed policies are semantically represented as instances in a domain specific policy ontology model. The policy ontology model can then be queried and reasoned over to detect implicit relationships and potential conflicts among groups of deployed policies. Semantic web rules are used for the selection of previously deployed policies related to the candidate federation policy over its policy elements to be used as input to the policy element match algorithm. The policy element match algorithm can be viewed as a central component of the framework by identifying matches over arbitrary

numbers of policy elements which leads to detection of potential inconsistencies more efficiently than using pair-wise policy analysis techniques alone. A number of federation use cases were provided to evaluate the effectiveness of the policy selection and policy element match approach in federation scenarios.

3. *What strategies can federating enterprises adopt to increase access request evaluation performance while maintaining an acceptable level of risk?*

   This question is addressed in Chapter 5 where a novel probabilistic approach that can balance policy evaluation performance against access request safety was described that utilised social network analyses to categorise a social network's users and policies in order to evaluate specific categories of policies against specific access requests from the social network's users/groups in a bid to increase overall access request evaluation performance while maintaining an acceptable level of safety mandated by the business processes of that enterprise social network. Central to the approach is harnessing analyses of the enterprise social network to determine the most prominent users/groups of the social network. The results of this social network analysis can then be used to segment the users/groups of the enterprise social network and the access control policies specified over those users/groups with the aim of increasing access request evaluation performance. Extensive experiments were conducted to analyse the efficiency of the approach and to determine if policy evaluation outcomes were significantly affected by using the approach as in some instances not all policies are available for evaluation against each access request received. A trade-off was observed between increasing access request evaluation performance and maintaining particular safety levels by taking a probabilistic approach to access request evaluation as depending on the strength of the relations between enterprise social network users/groups will ultimately decide the category of policies that any given access request will be evaluated against.

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

- It remains to be determined the effects in terms of reasoning efficiency when a large number of deployed policies are returned for analysis to the policy element match algorithm. Future work will investigate this along with methods for analysing context data specifically with the aim of suggesting possible solutions to detected policy inconsistencies.

- This thesis only considered federation policies as external policies that are shared among federation members. However, federation policies can be categorised as intra-federation policies and inter-federation policies. Even though federation policies are realised as local policies within each domain. Intra-federation policies can be viewed as private policies in the sense that they are applied to local domain members participating in a federation and are not shared with federation partners. Inter-federation policies can be viewed as public policies, even though they are deployed within the local domain they are applied to external federation members providing some service within the local domain. These policies can be shared without any privacy concerns and possibly negotiated with other federation members. Future work will investigate applying the policy specification and consistency analysis techniques outline in this thesis to intra-federation policies.

- It is imperative that applying probabilistic techniques to access requests yields the same outcomes to policy evaluation as a standard PDP and does not introduce any security vulnerabilities by inconsistently evaluating an access request. This is a problematic factor with using probabilistic techniques as the policies and policy sets are distributed among a number of PDPs and so not all policies and/or policy sets are available for evaluation against each access request received. Future work will investigate the distribution of policies/policy sets across a number of PDPs and the impact on access request evaluation consistency.

6.2 Future Work

The future work section has been divided into three areas. Section 6.2.1 discusses possible future extensions to the federation policy authoring process. Section 6.2.2 suggests further directions that can be taken with regards to the policy consistency analysis pro-

cess, while Section 6.2.3 outlines future extensions possible to increase performance of policy evaluation in enterprise social networks.

6.2.1 Federation Policy Authoring Extensions

Federation Policy Negotiation Protocol. As highlighted in Chapter 4 certain aspects of federation policies need to be negotiated between (possibly multiple) federation participants during the actual policy specification stage of the authoring process for federation policies. In order for multiple enterprises to consistently deploy federation policies within their domains, a negotiation process needs to be put in place to allow the negotiation of particular policy elements between federated domains. At present, the FRM (Brennan *et al.* (2009)) is used for secure attribute sharing of federation policy aspects only and does not provide a protocol to for the actual negotiation process of the federation policy aspects. However, future research will investigate processes and algorithms required to implement a negotiation protocol that can complement the FRM (or any other secure attribute sharing framework) for secure attribute sharing in federated domain environments. In essence, negotiation of federation policy aspects is required whenever federation policies are created, updated or removed. Certain federation aspects required for inclusion in the federation policy specification process and hence are required to be distributed among federation participants include governance structure, context, and context data. Typical examples of the types of context data that needs to be shared among federation participants include XMPP capabilities and XMPP grouping structures. XMPP capabilities include instant messaging, groupchat and file transfer and are specified in the action element of a policy rule, while, XMPP grouping structures indicate the XMPP groups involved as part of the communication, along with the individual members of those groups which are specified in the subject and target elements of a policy rule. These federation aspects are mentioned as terms in the federation policy and are used to specify the subject, target, action, and conditions elements of the federation policy. A federation policy negotiation protocol is required for implementation to facilitate the negotiation of particular policy elements between domains. It is essential to be able to negotiate these federation aspects as they implicitly indicate the managed entities and the type of communication permitted between federating enterprises. This type of information is critical to federation policy authoring

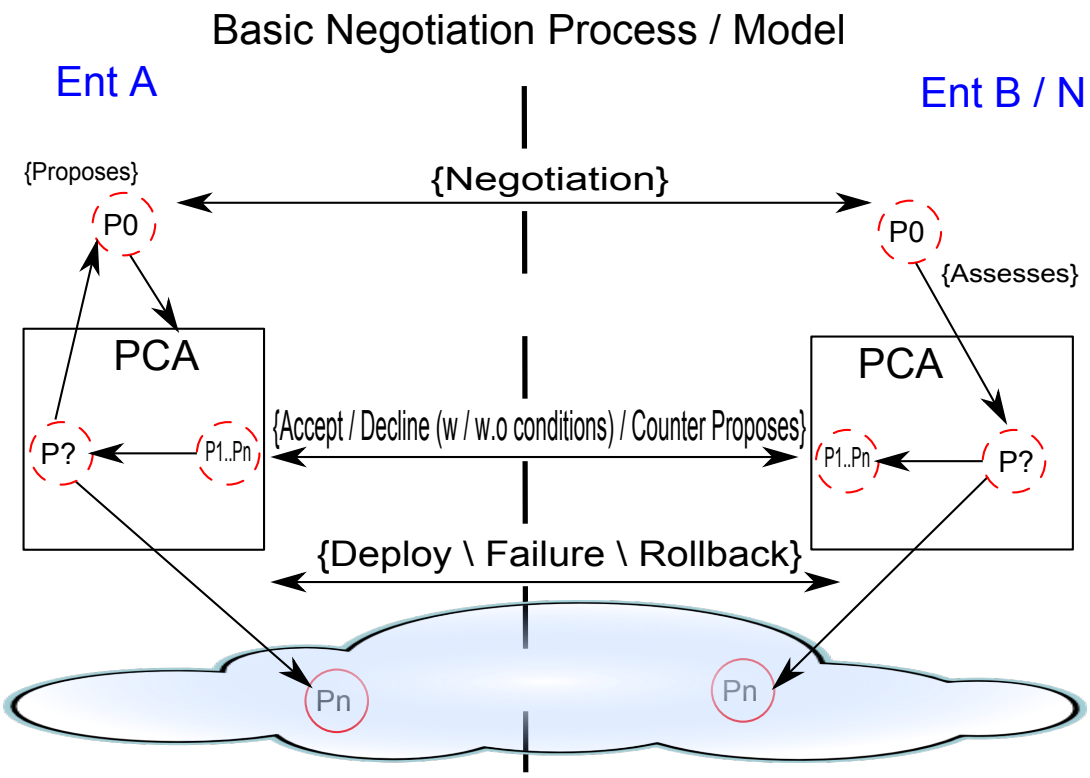

Figure 6.1: Negotiation Model

processes to facilitate consistent specification of federation policies in order to control the flow of communication between federating enterprises.

A typical example of a negotiation protocol used for specification of federation policies is depicted in Figure 6.1 and proceeds as follows. Enterprise A proposes a federation policy. Federation policy aspects (such as context/context data) conveyed through a secure attribute sharing framework (such as the FRM) to Enterprise N. Enterprise N uses the conveyed federation aspects in the specification of a federation policy, then the federation policy is analysed for consistency with local deployed policies by Enterprise N. If no inconsistencies are detected with local deployed policies by Enterprise N, a deployment message is sent from Enterprise N to Enterprise A to proceed with refinement and deployment of the federation policy. Both enterprises then proceed to refine and

deploy the federation policy within their respective domains and can (re-)use the negotiation protocol at each step as required. Once the local policies related to the federation policy are successfully refined and deployed (into possibly multiple policy languages), a final *commit* message declaring that deployment was successful is sent between the two enterprises. Failing a successful deployment, a roll-back message is sent to the other enterprise to indicate that a roll-back should be performed. At this point, negotiation fails and terminates or alternatively, re-negotiation starts.

6.2.2 Policy Consistency Analysis Directions

Social Network Policy Conflict Analysis. While investigating policy consistency analysis in federated domain environments in Chapter 4, the application of the policy consistency analysis techniques to enterprise social networks offered interesting and relevant uses cases. In particular, analyses of an enterprise social network's policies requires a large number of deployed policies to be retrieved and analysed efficiently. The policy selection process combined with the policy element match algorithm outlined in Chapter 4 are specifically tailored for efficient retrieval and analysis over large groups of deployed policies such as those typical of enterprise social network scenarios. Within federating enterprises (e.g. enterprise social networks) there are multiple policy systems being utilised and quite often these policy systems adhere to different policy models and languages (i.e access control, firewall, etc.). The policy consistency analysis techniques described in Chapter 4 are adept at performing analysis over multiple system policy models. An enterprise social network domain ontology models the structure of a social network such as the users, groups, and resources of the social network. Similarly, an enterprise social network policy ontology models the management policies specified against the managed entities from the domain ontology model. These enterprise social network ontology models can be consolidated and leveraged by policy consistency analysis processes such as the ones described in Chapter 4 to aid detection of policy inconsistencies by providing additional domain-specific semantic knowledge regarding users, groups, and resources as required. Semantic web technologies provide powerful modelling and reasoning tools such as ontology languages (OWL-DL, OWL-Lite, etc.) and semantic web rule languages (SWRL, SPARQL, etc.) that can be harnessed to represent and reason over an enterprise's complete social network to augment policy consistency analysis processes. According to Carminati *et al.* (2009) there are five categories of social

network data that can be modelled using semantic web technologies. These are: (1) personal information; (2) personal relationships; (3) social network resources; (4) relationships between users and resources; (5) actions that can be performed in a social network. For example, the FOAF ontology (Brickley & Miller (2010)) includes this type of knowledge and can easily be extended to model additional social network semantics pertaining to users, groups, resources and their complex relationships representative in enterprise social networks. The policy authoring processes outlined in Chapter 3 can harness existing enterprise social network ontology models of social network structures and the policies specified over the social network's users (i.e people, groups, etc.) and resources (i.e. code repositories, wikis, document repositories, etc.). The harnessed ontology models augment the policy specification and consistency analysis processes from Chapter 4 with social network specific knowledge (i.e. foaf/colleagues relationships, distributive actions (i.e. file access permissions)) to aid in the consistent specification of enterprise social network policies and in the detection of inconsistencies that can occur specific to enterprise social network policies.

6.2.3 Policy Evaluation Extensions

Optimum Policy Distribution. As highlighted in Chapter 5, the categorisation and deployment of polices/policy sets is a complex task. For example, deployed policies may be part of a policy set with a particular execution strategy (i.e. First-Applicable, Permit-Overrides, etc.). In a centralised policy repository, the PDP selects policies/policy sets and applies the execution strategy of the selected policy set. However, if the policy repository is distributed to an arbitrary number of PDPs not all policies in the policy set may be available at policy evaluation time. If all policies in the policy set are not available when the policy set is being evaluated it can lead to an incorrect execution strategy being applied based on the policies that are available as part of that policy set during evaluation time.

Analysis of social networks to-date has focused on the ego-centred links associating individuals/groups with other individuals/groups with the aim of identifying positions of power and persuasion. Easley & Kleinberg (2010) applied fundamental principles of graph theory to understand a number of different aspects to the relationship links among the members of a social network. These aspects include the degree of centrality (the number of links going into/out of a node), closeness centrality (the distance

of a particular node to all other nodes), betweenness centrality (the degree to which individuals/groups need to go through a particular user to collaborate with other individuals/groups). These forms of social network analysis are a valuable asset and can be harnessed to categorise the management policies applicable to an enterprise social network in many different ways. Enterprise social network graphs represent semantic relationships that hold between social network users/groups and/or resources (code repositories, wikis, documents, online document repositories, etc.). These semantic relationship graphs can be queried and analysed using modified graph searching techniques to aid in the policy deployment process by building a dynamic social network policy graph that can indicate a number of useful policy categories. Some example policy categories include the largest number of policies applicable to a particular policy target (user, group, resource, etc.), policies applicable to a particular subject/resource that are accessed most frequently, etc. Based on this dynamic social policy graph, algorithms can then be used to traverse the graph and return the selected category of policies first for consistency analysis. The graph-based selection algorithm can restrict its search space to retrieve deployed policies that are either direct or indirect policies (a specific number of degrees (i.e edges) away) attached to a user, group, or resource. The selection algorithm can query an edge of a graph that represents a particular relationship and return both vertices attached to the edge which represents a set of users and/or resources. The graph searching algorithms aim to reduce the overall policy search space where semantic web queries are executed over the reduced ontology instances to further reduce the search space and return pertinent policies for analysis.

Processes and algorithms are required to dynamically update the social network policy graph whenever a new policy is deployed, updated or removed from the managed system (and hence the graph) by updating the weightings of the vertices in the graph to reflect whatever changes have taken place. However, any processes or algorithms designed for optimum policy distribution should consider the various policy set execution strategies and attempt to ensure the correct execution strategy of that policy set is applied. Although a distributed policy deployment approach may be less accurate than that of a centralised policy deployment approach, the distributed policy deployment approach is expected to deliver significant policy evaluation time gains for policy requests as it avoids the potential policy request processing bottleneck that can occur when all policy requests are directed towards a centralised policy decision point as is the case in

social networks. This is due to the fact that a PDP with all deployed policies loaded has to traverse all the deployed policies stored in a centralised policy repository in an attempt to match against the policy request received as opposed to search a portion of all deployed policies stored in a policy repository that only stores a subset of all deployed policies.

References

Abedin, M., Nessa, S., Khan, L. & 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*, 15–29.

Al-Shaer, E.S. & Hamed, H.H. (2003). Firewall policy advisor for anomaly discovery and rule editing. In *Integrated Network Management, 2003. IFIP/IEEE Eighth International Symposium on*, 17–30, IEEE. 38

Al-Shaer, E.S. & Hamed, H.H. (2004a). Discovery of policy anomalies in distributed firewalls. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, vol. 4, 2605–2616, IEEE. 39

Al-Shaer, E.S. & Hamed, H.H. (2004b). Modeling and management of firewall policies. *Network and Service Management, IEEE Transactions on*, **1**, 2–10. 39

Alcaraz Calero, J., Marín Pérez, J., Bernal Bernabé, J., Garcia Clemente, F., Martínez Pérez, G. & Gómez Skarmeta, A. (2010). Detection of Semantic Conflicts in Ontology and Rule-Based Information Systems. *Data & Knowledge Engineering*. 39

Aleman-Meza, B., Nagarajan, M., Ramakrishnan, C., Ding, L., Kolari, P., Sheth, A.P., Arpinar, I.B., Joshi, A. & Finin, T. (2006). Semantic analytics on social networks: experiences in addressing the problem of conflict of interest detection. In *Proceedings of the 15th international conference on World Wide Web*, 407–416, ACM. 8, 56

Baader, F., Calvanese, D., McGuinness, D., Patel-Schneider, P. & Nardi, D. (2003). *The description logic handbook: theory, implementation, and applications*. Cambridge Univ Pr. 62, 64

Bajaj, S., Box, D., Chappell, D., Curbera, F., Daniels, G., Hallam-Baker, P., Hondo, M., Kaler, C., Langworthy, D., Malhotra, A. *et al.* (2006). Web services policy framework (ws-policy). *Version*, **1**, 2003–2006. 34, 53

Baliosian, J. & Serrat, J. (2004). Finite state transducers for policy evaluation and conflict resolution. In *Policies for Distributed Systems and Networks, 2004. POLICY 2004. Proceedings. Fifth IEEE International Workshop on*, 250–259, IEEE. 42

Bandara, A.K., Lupu, E.C. & Russo, A. (2003). Using event calculus to formalise policy specification and analysis. In *Policies for Distributed Systems and Networks, 2003. Proceedings. POLICY 2003. IEEE 4th International Workshop on*, 26–39, IEEE. 39, 40

Bandara, A.K., Lupu, E.C., Moffett, J. & Russo, A. (2004). A goal-based approach to policy refinement. In *Policies for Distributed Systems and Networks, 2004. POLICY 2004. Proceedings. Fifth IEEE International Workshop on*, 229–239, IEEE. 50

Bandara, A.K., Lupu, E.C., Russo, A., Dulay, N., Sloman, M., Flegkas, P., Charalambides, M. & Pavlou, G. (2006). Policy refinement for ip differentiated services quality of service management. *Network and Service Management, IEEE Transactions on*, **3**, 2–13. 50

Barrett, K., Davy, S., Strassner, J., Jennings, B., van der Meer, S. & Donnelly, W. (2007a). A model based approach for policy tool generation and policy analysis. In *Proceedings of the IEEE Global Information Infrastructure Symposium*, 99–105. 35, 111, 114, 159

Barrett, K., Strassner, J., van der Meer, S., Donnelly, W., Jennings, B. & Davy, S. (2007b). A policy representation format domain ontology for policy transformation. *2nd IEEE International Workshop on Modelling Autonomic Communications Environments (MACE2007), San Jose, CA, USA, Oct. 2007*. 29, 34

Barron, J. & Davy, S. (2013). Semantic web technologies to aid dominance detection for access control policies. In *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, 780–783, IEEE. 13, 154

Barron, J., Davy, S., Jennings, B. & Strassner, J. (2010). A Policy Authoring Process and DEN-ng Model Extension for Federation Governance. *Modelling Autonomic Communication Environments*, 73–86. 13

Barron, J., Davy, S. & Jennings, B. (2011). Conflict analysis during authoring of management policies for federations. In *Proc. 1st IFIP/IEEE International Workshop on Managing Federations and Cooperative Management (ManFed.com 2011). IEEE, 2011.*, 1176–1183, IEEE. 13, 29, 138

Barron, J., Davy, S. & Jennings, B. (2013). Probabilistic access control for enterprise communication systems. *Computer Communications (under review)*. 14

Beckett, D. & McBride, B. (2004). Rdf/xml syntax specification (revised). http://www.w3.org/TR/rdf-syntax-grammar/ Last Accessed: 01/05/2013, w3C recommendation. 64

Bell, D.E. & La Padula, L.J. (1976). Secure computer system: Unified exposition and multics interpretation. Tech. rep., DTIC Document. 20

Bhatti, R., Bertino, E. & Ghafoor, A. (2006). X-FEDERATE: A Policy Engineering Framework for Federated Access Management. *IEEE Transactions on Software Engineering*, **32**, 330–346. 54

Bjorner, D. & Jones, C. (1978). *The Vienna Development Method: The Meta-Language*. Springer-Verlag London, UK. 82

Boutaba, R. & Aib, I. (2007). Policy-based management: A historical perspective. *Journal of Network and Systems Management*, **15**, 447–480. 20

Brennan, R., Lewis, D., Keeney, J., Etzioni, Z., Feeney, K., O' Sullivan, D., Lozano, J.A. & Jennings, B. (2009). Policy-based Integration of Multi-provider Digital Home Services. *IEEE Network*, **23**, 50–56. 6, 189

Brennan, R., Feeney, K., Walsh, B., Thomas, H. & O'Sullivan, D. (2011). Explicit federal relationship management to support semantic integration. In *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*, 1148–1155, IEEE. 6

Brickley, D. & Miller, L. (2010). Foaf vocabulary specification 0.98. *Namespace Document*, **9**. 56, 192

Butler, B., Jennings, B. & Botvich, D. (2011). An experimental testbed to predict the performance of xacml policy decision points. In *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*, 353–360, IEEE. 52

Campbell, G. & Turner, K. (2008a). Goals and policies for sensor network management. In *Sensor Technologies and Applications, 2008. SENSORCOMM'08. Second International Conference on*, 354–359, IEEE. 46

Campbell, G. & Turner, K. (2008b). Policy conflict filtering for call control. In *Proc. 9th Int. Conf. on Feature Interactions in Software and Communications Systems*, 83–98, Amsterdam, Netherlands: IOS Press. 46

Campbell, G.A. & Turner, K.J. (2007). Ontologies to support call control policies. In *Telecommunications, 2007. AICT 2007. The Third Advanced International Conference on*, 18–18, IEEE. 45

Carminati, B., Ferrari, E. & Perego, A. (2006). Rule-based access control for social networks. In *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops*, 1734–1744, Springer. 8, 56

Carminati, B., Ferrari, E., Heatherly, R., Kantarcioglu, M. & Thuraisingham, B. (2009). A semantic web based framework for social network access control. In *Proceedings of the 14th ACM symposium on Access control models and technologies*, 177–186, ACM. 56, 191

Charalambides, M., Flegkas, P., Pavlou, G., Bandara, A., Lupu, E., Russo, A., Dulay, N., Sloman, M. & Rubio-Loyola, J. (2005). Policy Conflict Analysis for Quality of Service Management. In *Sixth IEEE Int. Workshop on Policies for Distributed Systems and Networks.(POLICY 2005), Stockholm, Sweden June*, 6–8. 40

Charalambides, M., Flegkas, P., Pavlou, G., Rubio-Loyola, J., Bandara, A.K., Lupu, E.C., Russo, A., Sloman, M. & Dulay, N. (2006). Dynamic policy analysis and conflict resolution for diffserv quality of service management. In *Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP*, 294–304, IEEE. 40

Chomicki, J., Lobo, J. & Naqvi, S. (2000). A Logic Programming Approach to Conflict Resolution in Policy Management. In *PRINCIPLES OF KNOWLEDGE REPRESENTATION AND REASONING-INTERNATIONAL CONFERENCE-*, 121–134, Morgan Kaufmann Publishers; 1998. 38, 46

Chomicki, J., Lobo, J. & Naqvi, S. (2003). Conflict Resolution Using Logic Programming. *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, 244–249. 38, 40, 46

Cisco (2013). Webex social. http://www.cisco.com/web/products/quad/index.html Last accessed: 03/04/2013. 7, 55

Connolly, D., van Harmelen, F., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F. & Stein, L.A. (2001). Daml+oil (march 2001) reference description. W3C Note. 41

Craven, R., Lobo, J., Ma, J., Russo, A., Lupu, E. & Bandara, A. (2009). Expressive policy analysis with enhanced system dynamicity. In *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, 239–250, ACM. 50

Craven, R., Lobo, J., Lupu, E., Russo, A. & Sloman, M. (2010). Decomposition techniques for policy refinement. In *Network and Service Management (CNSM), 2010 International Conference on*, 72–79, IEEE. 51

Craven, R., Lobo, J., Lupu, E., Russo, A. & Sloman, M. (2011). Policy refinement: decomposition and operationalization for dynamic domains. In *Network and Service Management (CNSM), 2011 7th International Conference on*, 1–9, IEEE. 51

Damianou, N., Dulay, N., Lupu, E. & Sloman, M. (2001). The Ponder Policy Specification Language. *LECTURE NOTES IN COMPUTER SCIENCE*, 18–38. 30, 31, 42

Davy, S. (2008). *Harnessing Information Models and Ontologies for Policy Conflict Analysis*. Ph.D. thesis, School of Science, Waterford Institute of Technology. 37

Davy, S. & Jennings, B. (2007). Harnessing Models for Policy Conflict Analysis. *LECTURE NOTES IN COMPUTER SCIENCE*, **4543**, 176. 60

Davy, S., Jennings, B. & Strassner, J. (2006). Policy Conflict Prevention via Model-driven Policy Refinement. *Proc 17th IFIP/IEEE Distributed Systems: Operations and Management (DSOM)*, 209–220. 44

Davy, S., Jennings, B. & Strassner, J. (2007). The Policy Continuum – A Formal Model. In *Proc. 2nd IEEE International Workshop on Modelling Autonomic Communications Environments (MACE 2007)*, 65–79. 28, 60, 82, 110

Davy, S., Jennings, B. & Strassner, J. (2008a). Efficient Policy Conflict Analysis for Autonomic Network Management. In *Engineering of Autonomic and Autonomous Systems, 2008. EASE 2008. Fifth IEEE Workshop on*, 16–24. 44

Davy, S., Jennings, B. & Strassner, J. (2008b). The Policy Continuum – Policy Authoring and Conflict Analysis. *Computer Communications*, **31**, 2981–2995. 3, 27, 28, 29, 91, 101, 110, 113, 114

Davy, S., Jennings, B. & Strassner, J. (2008c). Using an Information Model and Associated Ontology for Selection of Policies for Conflict Analysis. In *Policies for Distributed Systems and Networks, 2008. POLICY 2008. IEEE Workshop on*, 82–85. 44, 93

Davy, S., Barron, J., Shi, L., Butler, B., Jennings, B., Griffin, K. & Collins, K. (2013). A language driven approach to multi-system access control. In *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, 1004–1008, IEEE. 13

Dong, Q., Banerjee, S., Wang, J., Agrawal, D. & Shukla, A. (2006). Packet classifiers in ternary cams can be smaller. In *ACM SIGMETRICS Performance Evaluation Review*, vol. 34, 311–322, ACM. 171

Dunlop, N., Indulska, J. & Raymond, K. (2001). Dynamic policy model for large evolving enterprises. In *Enterprise Distributed Object Computing Conference, 2001. EDOC'01. Proceedings. Fifth IEEE International*, 193–197, IEEE. 38

Dunlop, N., Indulska, J. & Raymond, K. (2002). Dynamic conflict detection in policy-based management systems. In *Enterprise Distributed Object Computing Conference, 2002. EDOC'02. Proceedings. Sixth International*, 15–26, IEEE. 38

Dunlop, N., Indulska, J. & Raymond, K. (2003). Methods for conflict resolution in policy-based management systems. In *Enterprise Distributed Object Computing Conference, 2003. Proceedings. Seventh IEEE International*, 98–109, IEEE. 38

Durham, D., Boyle, J., Cohen, R., Herzog, S., Rajan, R. & Sastry, A. (2000). The cops (common open policy service) protocol. 23

Easley, D. & Kleinberg, J. (2010). *Networks, crowds, and markets*, vol. 8. Cambridge Univ Press. 192

Efftinge, S. & Völter, M. (2006). oaw xtext: A framework for textual dsls. In *Workshop on Modeling Symposium at Eclipse Summit*, vol. 32. 34, 35, 115

Erdos, M. & Cantor, S. (2002). Shibboleth architecture draft v05. *Internet2/MACE, May*, **2**. 33

Faurer, C., Hartley, C., Hepburn, H., Reilly, J., Sigley, W., Strassner, J., Elkady, A., Salomon, J., Grosso, E., Koppel, M., Cutts, R. & Izzo, M. (2004). Shared information/data (sid) model addendum 0 - sid primer. TeleManagement Forum, revision 4.0. 58, 130

Feeney, K., Lewis, D. & Wade, V. (2004). Policy based management for Internet communities. *Policies for Distributed Systems and Networks, 2004. POLICY 2004. Proceedings. Fifth IEEE International Workshop on*, 23–32. 5, 6, 42

Feeney, K., Brennan, R., Keeney, J., Thomas, H., Lewis, D., Boran, A. & O'Sullivan, D. (2010). Enabling Decentralised Management through Federation. *Computer Networks*. 6, 113, 153

Fisler, K., Krishnamurthi, S., Meyerovich, L. & Tschantz, M. (2005). Verification and change-impact analysis of access-control policies. In *Proceedings of the 27th international conference on Software engineering*, 196–205, ACM. 173

Fitzgerald, W. & Foley, S. (2009). Aligning Semantic Web applications with network access controls. *Computer Standards & Interfaces*. 48

Fitzgerald, W., Foley, S. & Foghlú, M. (2007). Confident firewall policy configuration management using description logic. In *Twelfth Nordic Workshop on Secure IT Systems, short presentations (unpublished), Reykjavik, Iceland, October 11*, vol. 12.

Fitzgerald, W., Foley, S. & Foghlú, M. (2008). Network access control interoperation using semantic web techniques. In *6th International Workshop on Security In Information Systems (WOSIS), Barcelona, Spain (June 2008)*, Citeseer.

Fitzgerald, W., Foley, S. & Foghlú, M. (2009). Network access control configuration management using semantic web techniques. *Journal of Research and Practice in Information Technology*, **41**, 99. 48

Frank, G., Jenkins, J. & Fikes, R. (2008). Jtp: an object-oriented modular reasoning system. 41

Fu, Z., Wu, S., Huang, H., Loh, K., Gong, F., Baldine, I. & 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.*, 39–56.

Gennari, J., Musen, M., Fergerson, R., Grosso, W., Crubézy, M., Eriksson, H., Noy, N. & Tu, S. (2003). The evolution of protégé: an environment for knowledge-based systems development. *International Journal of Human-Computer Studies*, **58**, 89–123. 159

Granovetter, M.S. (1973). The strength of weak ties. *American journal of sociology*, 1360–1380. 167

Griffin, L., Butler, B., de Leastar, E., Jennings, B. & Botvich, D. (2012). On the performance of access control policy evaluation. In *Policies for Distributed*

Systems and Networks (POLICY), 2012 IEEE International Symposium on, 25 – 32, IEEE. 52, 70

Gruber, T. *et al.* (1993). A translation approach to portable ontology specifications. *Knowledge acquisition*, **5**, 199–199. 62

Horrocks, I. (2005). Owl: A description logic based ontology language. In *Logic Programming*, 1–4, Springer. 65

Horrocks, I., Patel-Schneider, P. & Van Harmelen, F. (2003). From SHIQ and RDF to OWL: The making of a web ontology language. *Web semantics: science, services and agents on the World Wide Web*, **1**, 7–26. 66

Horrocks, I., Patel-Schneider, P., Boley, H., Tabet, S., Grosof, B. & Dean, M. (2004). SWRL: A semantic web rule language combining OWL and RuleML. *W3C Member submission*, **21**. 66

Horrocks, I., Parsia, B., Patel-Schneider, P. & Hendler, J. (2005). Semantic web architecture: Stack or two towers? In *Principles and Practice of Semantic Web Reasoning*, 37–41, Springer. 64

Hu, H., Ahn, G. & Kulkarni, K. (2011). Ontology-based policy anomaly management for autonomic computing. In *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2011 7th International Conference on*, 487–494, IEEE. 48

Hull, R., Kumar, B. & Lieuwen, D. (2003). Towards federated policy management. In *IEEE 4th International Workshop on Policies for Distributed Systems and Networks, 2003. Proceedings. POLICY 2003*, 183–194. 54

Hwang, J., Xie, T., Chen, F. & Liu, A. (2008). Systematic structural testing of firewall policies. In *Reliable Distributed Systems, 2008. SRDS'08. IEEE Symposium on*, 105–114, IEEE. 44

IBM (2013). Beehive. http://www-01.ibm.com/ software/ucd/gallery/ beehive_research.html Last accessed: 03/04/2013. 7, 55

Jajodia, S., Samarati, P., Sapino, M.L. & Subrahmanian, V. (2001). Flexible support for multiple access control policies. *ACM Transactions on Database Systems (TODS)*, **26**, 214–260. 39

Jennings, B., Brennan, R., Donnelly, W., Foley, S., Lewis, D., O'Sullivan, D., Strassner, J. & van der Meer, S. (2009). Challenges for federated, autonomic network management in the future internet. In *Integrated Network Management-Workshops, 2009. IM'09. IFIP/IEEE International Symposium on*, 87–92, IEEE. 4, 68

Jennings, B., Feeney, K., Brennan, R., Balasubramaniam, S., Botvich, D. & van der Meer, S. (2010). Federating autonomic network management systems for flexible control of end-to-end communications services. *Autonomic Network Management Principles: From Concepts to Applications*, 101. 6

Kagal, L. & Rei, A. (2002). A policy language for the me-centric project. *HP Labs, accessible on*. 30, 32

Kagal, L., Finin, T. & Joshi, A. (2003). A policy language for a pervasive computing environment. In *Policies for Distributed Systems and Networks, 2003. Proceedings. POLICY 2003. IEEE 4th International Workshop on*, 63–74, IEEE. 41

Kempter, B. & Danciu, V.A. (2005). Generic policy conflict handling using a priori models. In *Ambient Networks*, 84–96, Springer. 43

Kolovski, V. & Hendler, J. (2007). Xacml policy analysis using description logics. In *Proceedings of the 15th International World Wide Web Conference*, vol. 44, 494–497. 47

Kolovski, V., Hendler, J. & Parsia, B. (2007). Analyzing web access control policies. In *Proceedings of the 16th international conference on World Wide Web*, 677–686, ACM New York, NY, USA. 47

Kruk, S., Grzonkowski, S., Gzella, A., Woroniecki, T. & Choi, H.C. (2006). D-foaf: Distributed identity management with access rights delegation. *The Semantic Web–ASWC 2006*, 140–154. 8, 56

Lamers, L., Piazza, J., Maier, A., Ericson, G., Davis, J. & Schopmeyer, K. (2010). Common information model (cim) infrastructure. Distributed Management Task Force. 25, 57, 130

Latré, S., van der Meer, S., De Turck, F., Strassner, J., Hong, W. *et al.* (2010). Ontological generation of filter rules for context exchange in autonomic multimedia networks. In *Network Operations and Management Symposium (NOMS), 2010 IEEE*, 575–582, IEEE. 60

Lepro, R. (2004). Cardea: Dynamic access control in distributed systems. *System*, **13**, 4–2. 34

Lin, D., Rao, P., Bertino, E. & Lobo, J. (2007). An approach to evaluate policy similarity. In *Proceedings of the 12th ACM symposium on Access control models and technologies*, 1–10, ACM. 47

Lin, D., Rao, P., Bertino, E., Li, N. & Lobo, J. (2010). Exam: a comprehensive environment for the analysis of access control policies. *International Journal of Information Security*, **9**, 253–273.

Liu, A., Chen, F., Hwang, J. & Xie, T. (2011). Designing fast and scalable xacml policy evaluation engines. *Computers, IEEE Transactions on*, **60**, 1802–1817. 8, 52, 53, 56, 70

Liu, A.X., Chen, F., Hwang, J. & Xie, T. (2008). Xengine: a fast and scalable xacml policy evaluation engine. In *ACM SIGMETRICS Performance Evaluation Review*, vol. 36, 265–276, ACM. 52

Lobo, J., Bhatia, R. & Naqvi, S. (1999). A policy description language. In *In Proc. of AAAI*. 30, 40

Lobo, J., Agrawal, D., Calo, S., Lee, K. & Westerinen, A. (2009). Common information model simplified policy language (cim-spl). Distributed Management Task Force, revision 1.0.0. 32

Lorch, M., Adams, D.B., Kafura, D., Koneni, M., Rathi, A. & Shah, S. (2003a). The prima system for privilege management, authorization and enforcement

in grid environments. In *Grid Computing, 2003. Proceedings. Fourth International Workshop on*, 109–116, IEEE. 34

Lorch, M., Proctor, S., Lepro, R., Kafura, D. & Shah, S. (2003b). First experiences using xacml for access control in distributed systems. In *Proceedings of the 2003 ACM workshop on XML security*, 25–37, ACM. 33

Lupu, E. & Sloman, M. (1997). Conflict analysis for management policies. In *Integrated Network Management V*, 430–443, Springer. 38

Lupu, E.C. & Sloman, M. (1999). Conflicts in policy-based distributed systems management. *Software Engineering, IEEE Transactions on*, **25**, 852–869. 38

Machiraju, V., Sahai, A. & Van Moorsel, A. (2003). Web services management network: An overlay network for federated service management. In *IFIP/IEEE Eighth International Symposium on Integrated Network Management*, vol. 3, 351–364, Citeseer. 54

Marouf, S., Shehab, M., Squicciarini, A. & Sundareswaran, S. (2009). Statistics & clustering based framework for efficient xacml policy evaluation. In *Policies for Distributed Systems and Networks, 2009. POLICY 2009. IEEE International Symposium on*, 118–125, IEEE. 51, 52

Marouf, S., Shehab, M., Squicciarini, A. & Sundareswaran, S. (2011). Adaptive reordering and clustering-based framework for efficient xacml policy evaluation. *Services Computing, IEEE Transactions on*, **4**, 300–313. 8, 51, 52, 56, 70, 171

Martin, E., Xie, T. & Yu, T. (2006). Defining and measuring policy coverage in testing access control policies. *Information and Communications Security*, 139–158. 44

McBride, B. (2002). Jena: A semantic web toolkit. *Internet Computing, IEEE*, **6**, 55–59. 159

Microsoft (2013). Yammer. https://www.yammer.com/ Last accessed: 03/04/2013. 7, 55

Miseldine, P.L. (2008). Automated xacml policy reconfiguration for evaluation optimisation. In *Proceedings of the fourth international workshop on Software engineering for secure systems*, 1–8, ACM. 52, 70

Moffett, J. & Sloman, M. (1994). Policy Conflict Analysis in Distributed System Management. *JOURNAL OF ORGANIZATIONAL COMPUTING*, **4**, 1–1. 37, 38

Moffett, J.D. & Sloman, M.S. (1991). The representation of policies as system objects. In *ACM SIGOIS Bulletin*, vol. 12, 171–184, ACM. 20

Moffett, J.D. & Sloman, M.S. (1993). Policy hierarchies for distributed systems management. *Selected Areas in Communications, IEEE Journal on*, **11**, 1404–1414. 20, 49

Moore, B., Ellesson, E., Strassner, J. & Westerinen, A. (2001). Policy core information model–version 1 specification. Tech. rep., RFC 3060, February. 25

Motik, B., Patel-Schneider, P., Parsia, B., Bock, C., Fokoue, A., Haase, P., Hoekstra, R., Horrocks, I., Ruttenberg, A., Sattler, U. *et al.* (2009). Owl 2 web ontology language: Structural specification and functional-style syntax. *W3C Recommendation*, **27**. 65, 159

Nardi, D. & Brachman, R. (2003). An introduction to description logics. *The description logic handbook: theory, implementation, and applications*, 1–40. 62

Nejdl, W., Olmedilla, D., Winslett, M. & Zhang, C.C. (2005). Ontology-based policy specification and management. In *The Semantic Web: Research and Applications*, 290–302, Springer. 35

O'Sullivan, D., Strassner, J. & van der Meer, S. (2009). A snapshot of ontological approaches for network and service management. *Journal for network and service management*, **17**, 231–233. 68

Pérez, J., Arenas, M. & Gutierrez, C. (2006). Semantics and complexity of sparql. In *The Semantic Web-ISWC 2006*, 30–43, Springer. 67

Proctor, S. (2006). Sun's xacml implementation apis. http://sunxacml.sourceforge.net/ Last accessed: 05/01/2013. 14, 128, 173

Prud, E., Seaborne, A. *et al.* (2008). Sparql query language for rdf. *W3C recommendation*, **15**. 66, 160

Rajan, R., Verma, D., Kamat, S., Felstaine, E. & Herzog, S. (1999). A policy framework for integrated and differentiated services in the Internet. *Network, IEEE*, **13**, 36–41. 23

Rajendran, S. & Huber, M. (2011). Autonomous identification, categorization and generalization of policies based on task type. In *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on*, 1333–1339, IEEE. 171

Realtime (2011). Openfire. http://www.igniterealtime.org/projects/openfire Last accessed: 15/12/2012. 14, 127

Rissanen, E. (2013). extensible access control markup language (xacml), version 3.0, oasis standard. *Internet http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html*. 33, 53, 110, 128, 154, 173, 174

Rubio-Loyola, J., Serrat, J., Charalambides, M., Flegkas, P., Pavlou, G. & Lafuente, A.L. (2005). Using linear temporal model checking for goal-oriented policy refinement frameworks. In *Policies for Distributed Systems and Networks, 2005. Sixth IEEE International Workshop on*, 181–190, IEEE. 50

Rubio-Loyola, J., Serrat, J., Charalambides, M., Flegkas, P. & Pavlou, G. (2006a). A functional solution for goal-ooriented policy refinement. In *Policies for Distributed Systems and Networks, 2006. Policy 2006. Seventh IEEE International Workshop on*, 133–144, IEEE. 50

Rubio-Loyola, J., Serrat, J., Charalambides, M., Flegkas, P. & Pavlou, G. (2006b). A methodological approach toward the refinement problem in policy-based management systems. *Communications Magazine, IEEE*, **44**, 60–68. 50

Russo, A., Miller, R., Nuseibeh, B. & Kramer, J. (2002). *An abductive approach for analysing event-based requirements specifications*. Springer. 50

Saint-Andre, P. (2011). Extensible messaging and presence protocol (xmpp): Core. http://xmpp.org/rfcs/rfc6120.html Last accessed: 05/06/2012. 126, 153

Shankar, C.S., Ranganathan, A. & Campbell, R. (2005). An eca-p policy-based framework for managing ubiquitous computing environments. In *Mobile and Ubiquitous Systems: Networking and Services, 2005. MobiQuitous 2005. The Second Annual International Conference on*, 33–42, IEEE. 43

Smith, M., Welty, C. & McGuinness, D. (2004). Owl web ontology language guide. *W3C recommendation*, **10**. 64

Smith, M., Hansen, D. & Gleave, E. (2009). Analyzing enterprise social media networks. In *Computational Science and Engineering, 2009. CSE'09. International Conference on*, vol. 4, 705–710, IEEE. 171

Strassner, J. (1999). *Directory enabled networks*. New Riders Publishing Thousand Oaks, CA, USA. 58

Strassner, J. (2003). *Policy-Based Network Management: Solutions for the Next Generation (The Morgan Kaufmann Series in Networking)*. Morgan Kaufmann. 2, 3, 21, 27, 37, 58, 60, 81, 94

Strassner, J. (2007). Knowledge Engineering Using Ontologies. *Handbook of Network and System Administration, chapter Elsevier Handbook*. 61

Strassner, J., Fleck, J., Huang, J., Faurer, C. & Richardson, T. (2004). Tmf white paper on ngoss and mda. In *TeleManagement Forum/Object Management Group, February*. 58

Strassner, J., Agoulmine, N. & Lehtihet, E. (2006). FOCALE–A Novel Autonomic Networking Architecture. *first Latin American Autonomic Computing Conference (LAACS), July*. 80

Strassner, J., Samudrala, S., Cox, G., Liu, Y., Jiang, M., Zhang, J., Meer, S.v.d., Foghlú, M.Ó. & Donnelly, W. (2008). The design of a new context-aware policy model for autonomic networking. In *Autonomic Computing, 2008. ICAC'08. International Conference on*, 119–128, IEEE. 100

Strassner, J., de Souza, J.N., van der Meer, S., Davy, S., Barrett, K., Raymer, D. & Samudrala, S. (2009). The design of a new policy model to support

ontology-driven reasoning for autonomic networking. *Journal of Network and Systems Management*, **17**, 5–32. 100

Twidle, K., Lupu, E., Dulay, N. & Sloman, M. (2008). Ponder2-a policy environment for autonomous pervasive systems. In *IEEE Workshop on Policies for Distributed Systems and Networks, 2008. POLICY 2008*, 245–246. 31

Twidle, K., Dulay, N., Lupu, E. & Sloman, M. (2009). Ponder2: A policy system for autonomous pervasive environments. In *International Conference on Autonomic and Autonomous Systems (ICAS)*. 31

Uschold, M. & Gruninger, M. (1996). Ontologies: Principles, methods and applications. *The Knowledge Engineering Review*, **11**, 93–136. 61

Uszok, A., Bradshaw, J., Jeffers, R., Suri, N., Hayes, P., Breedy, M., Bunch, L., Johnson, M., Kulkarni, S. & Lott, J. (2003). KAoS policy and domain services: Toward a description-logic approach to policy representation, deconfliction, and enforcement. *Policy*, **93**. 30, 32, 41

Van der Meer, S., Davy, A., Davy, S., Carroll, R., Jennings, B. & Strassner, J. (2006). Autonomic networking: Prototype implementation of the policy continuum. In *Broadband Convergence Networks, 2006. BcN 2006. The 1st International Workshop on*, 1–10, IEEE. 34

Verlaenen, K., De Win, B. & Joosen, W. (2007a). Policy analysis using a hybrid semantic reasoning engine. In *Policies for Distributed Systems and Networks, 2007. POLICY'07. Eighth IEEE International Workshop on*, 193–200, IEEE. 45

Verlaenen, K., De Win, B. & Joosen, W. (2007b). Towards simplified specification of policies in different domains. In *Integrated Network Management, 2007. IM'07. 10th IFIP/IEEE International Symposium on*, 20–29. 36

Verma, D., Center, I. & Heights, Y. (2002). Simplifying network administration using policy-based management. *Network, IEEE*, **16**, 20–26. 49

Viswanath, B., Mislove, A., Cha, M. & Gummadi, K.P. (2009). On the evolution of user interaction in facebook. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Social Networks (WOSN'09)*. 173

Wang, Y., Zhang, H., Dai, X. & Liu, J. (2010). Conflicts analysis and resolution for access control policies. In *Information Theory and Information Security (ICITIS), 2010 IEEE International Conference on*, 264–267, IEEE. 48

Westerinen, A., Schnizlein, J., Strassner, J., Scherling, M., Quinn, B., Herzog, S., Huynh, A., Carlson, M., Perry, J. & Waldbusser, S. (2001). Terminology for policy-based management. Tech. rep., RFC Editor. 21, 57

Wijesekera, D. & Jajodia, S. (2003). A propositional policy algebra for access control. *ACM Transactions on Information and System Security (TISSEC)*, **6**, 286–325. 39

Wong, A.K.Y., Ray, P., Parameswaran, N. & Strassner, J. (2005). Ontology mapping for the interoperability problem in network management. *Selected Areas in Communications, IEEE Journal on*, **23**, 2058–2068. 61, 93

Wu, Z., Liu, Y. & Wang, L. (2009). Dynamic Policy Conflict Analysis in Operational Intensive Trust Services for Cross-Domain Federations. In *Intensive Applications and Services, 2009. INTENSIVE'09. First International Conference on*, 1–6, IEEE. 47

Zhao, H., Lobo, J. & Bellovin, S. (2008). An algebra for integration and analysis of ponder2 policies. *POLICY. IEEE Computer Society*, 74–77. 31

Zhao, H., Lobo, J., Roy, A. & Bellovin, S.M. (2011). Policy refinement of network services for manets. In *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*, 113–120, IEEE. 51

Appendices

Appendix A

List of Acronyms

ACPL Autonomic Computing Policy Language

ABox Assertional Box

ARR Access Request Router

CA Condition-Action

COPS Common Open Policy Service

CIM Common Information Model

CIM-SPL CIM Simplified Policy Language

DAML DARPA Agent Markup Language

DL Description Logics

DMTF Distributed Management Task Force

DSL Domain Specific Language

EC Event Calculus

ECA Event-Condition-Action

ESN Enterprise Social Network

FAME Federated Autonomic Management of end-to-End communication services

FOCALE Foundation, Observation, Comparison, Action, Learning Environment

IETF Internet Engineering Task Force

JTP Java Theorem Prover

KB Knowledge Base

LDAP Lightweight Directory Access Protocol

LPDP Local Policy Decision Point

LTL Linear Temporal Logic

MDA Model Driven Architecture

MDD Model Driven Development

MOF Managed Object Format

NGOSS New Generation Operations Systems and Software

NIST National Institute of Standards and Technology

OCL Object Constraint Language

OWL Web Ontology Language

OWL-DL OWL-Description Logics

PBM Policy Based 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

QoE Quality of Experience

QoS Quality of Service

QPIM Quality of Service Policy Information Model

RBAC Role Based Access Control

RDF Resource Description Framework

RDFS Resource Description Framework Schema

SLA Service Level Agreement

SPARQL SPARQL Protocol and RDF Query Language

SWRL Semantic Web Rule Language

TMF TeleManagement Forum

TBox Terminological Box

UML Unified Modelling Language

W3C World Wide Web Consortium

WSMN Web Services Management Network

XACML eXtensible Access Control Markup Language

XML eXtensible Markup Language

XMPP eXtensible Messaging and Presence Protocol

XSF XMPP Standards Foundation

Appendix B

DEN-ng Federated Domain Model

This appendix highlights particular aspects of the existing DEN-ng model suitable for modelling management domains and extensions made to the DEN-ng model to cater for modelling federations.

B.1 DEN-ng Domain Model

Figure B.1 shows the existing DEN-ng Domain model. In DEN-ng Domains are containers whose elements are ManagedEntities. A DEN-ng Domain can have zero or more Contexts. Context is defined as an aggregate object containing different aspects (such as time, location, and communication method), where each aspect is defined as a ContextData object. This enables Context and ContextData to be richly described. Domains can be hierarchically organized; the composite pattern is used to do this. A special type of Domain, called a ManagementDomain, is defined which uses a set of PolicyRules to enforce the governance aspects defined for it on its constituent Entities. Each Context selects a set of PolicyRules that are used to govern behaviour appropriate for that context. Hence, as the Context of a Domain changes, the set of PolicyRules change in order to maintain the set of goals, business objectives, regulatory rules, and/or other governance constructs that are shared by each entity contained in the Domain.

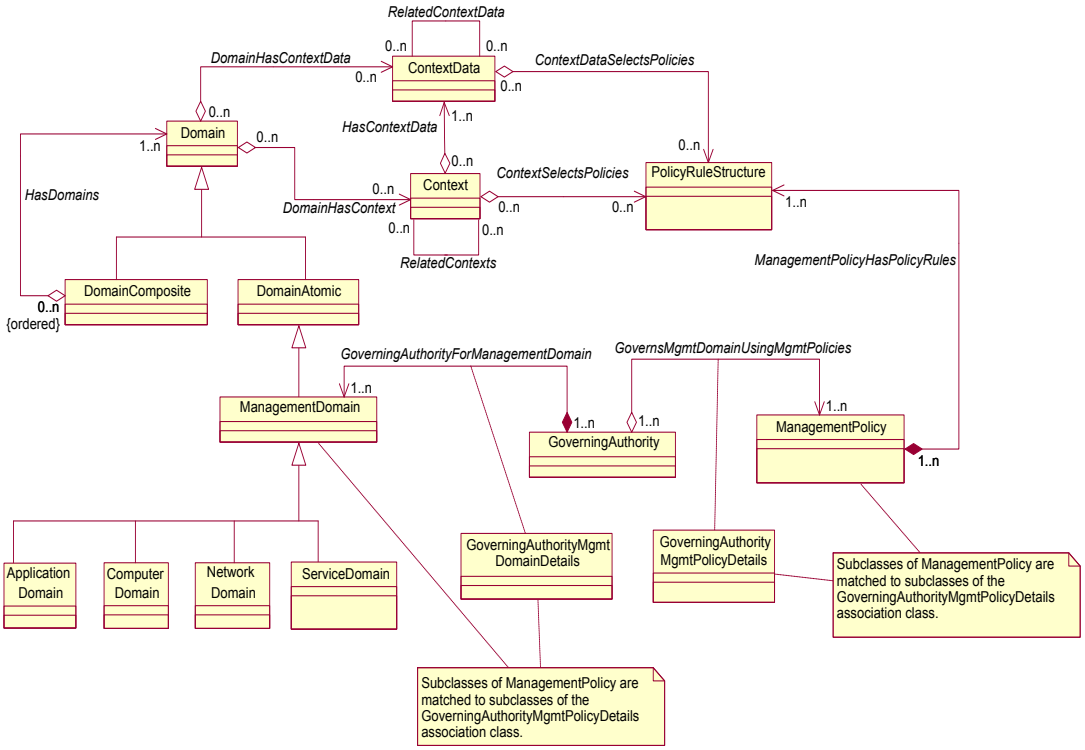

Figure B.1: Simplified View of the Existing DEN-ng Domain Model

The entities shown in Figure 3.1 that are critical to understanding this model are defined in the following subsections.

B.1.1 Domain Class

A Domain is a collection of Entities that share a common purpose. In addition, each constituent Entity in a Domain is both uniquely addressable and uniquely identifiable within that Domain. Note that a Domain is simply a container with metadata.

B.1.2 DomainAtomic Class

This is a concrete base class for representing Domains that can be individually managed. In addition, this Domain has characteristics and behaviour that can be externally accessed, so that this Domain can be viewed as a stand-alone Domain that can directly manage components that it contains.

B.1.3 DomainComposite Class

This is a concrete base class for representing Domains that contain one or more subordinate Domains. This Domain can be managed, and also aggregates a set of Domains, each of which can be individually managed. This type of Domain is primarily used to form groups of Domains, and hence manages the Domains that it contains as opposed to managing components in a Domain.

B.1.4 ManagementDomain Class

A ManagementDomain refines the notion of a Domain by adding two important behavioural features. First, it defines a common set of administrators that govern the managed entities that it contains. In other words, all constituent managed entities in this ManagementDomain are administered by the same user, group of users, and/or organisation(s). Second, it defines a set of applications that are responsible for different governance operations, such as monitoring, configuration, and so forth. Third, it defines a common set of management mechanisms, such as policy rules, that are used by the management mechanisms. Note that some management mechanisms may depend on particular management applications, and vice-versa.

B.1.5 ManagementPolicy Class

This is an abstract class that realises deontic actions (e.g., obligations and permissions) independent of the actual structure of the PolicyRule being used. That is, one set of ManagementPolicies using ECAPolicyRules can be defined for various deontic actions (e.g., authorization and obligation) by aggregating the ECAPolicyRules using the aggregation ManagementPolicyHasECAPolicyRules. A different set of ManagementPolicies using a different rule structure could be similarly defined by defining an aggregation between ManagementPolicy and the appropriate PolicyRuleStructure subclass. ManagementPolicy is the superclass for PolicyRules that manage a system. Its superclass is PolicyCategory. As such, ManagementPolicy has explicit associations with the subject and target of the ManagementPolicy, defined through the SubjectInteractsWith and TargetInteractsWith associations.

B.1.6 GoverningAuthority Class

A GoverningAuthority represents an individual or collection of ManagedEntities that are responsible for performing governance operations. Note that a GoverningAuthority can be either Human or Non-Human. The ManagementDomain represents the logical collection of ManagedEntities contained in a Domain, while the GoverningAuthority uses appropriate ManagementPolicies to govern both the ManagedEntities in the Domain as well as the management of the Domain itself. It is important to note that the Domain itself is not capable of management. Management actions are performed by the GoverningAuthority, and use ManagementPolicy instances for management actions (and other types of policy rules descended from PolicyRuleStructure for other types of policy actions).

A federation is a set of entities that are governed by a central authority but have a set of limited powers regarding their own local interests. DEN-ng modifies this to enable the federation to use either a single centralised or a set of distributed governing authorities, along with a continuum of governance mechanisms. The continuum is bounded on the one hand by autonomy, and on the other hand by local and/or global policy rules. This produces a range of possibilities, with the fundamental differentiating ones being:

1. A single central authority that dictates policy that all constituent domains must use (i.e., single central authority, subservient domains, only global policy rules)

2. A single central authority that coordinates the actions of otherwise autonomous domains (i.e., single central authority, autonomous domains, local and global policy)

3. A distributed set of authorities that themselves have no policy rules to enforce; rather, they arbitrate among their constituent components and use an agreed-upon mechanism, such as majority voting, to establish a limited set of domain-wide rules for each domain to follow, with the understanding that each domain is otherwise completely autonomous and can make its own rules (i.e., distributed authority, autonomous domains, local and global policy)

4. A distributed set of authorities that collectively dictate policy that is followed by each domain (i.e., distributed authority, subservient domains, global policy only) DEN-ng

B.1.7 GoverningAuthorityForManagementDomain Composition

This composition defines the set of GoverningAuthorities that are responsible for managing this ManagementDomain. The semantics of this composition are defined in the GoverningAuthorityMgmtDomainDetails association class. Appropriate subclasses of this association class are matched with their corresponding subclasses of ManagementPolicy, thereby preventing the needless proliferation of associations.

B.1.8 GovernsMgmtDomainUsingMgmtPolicies Aggregation

This aggregation defines the set of ManagementPolicies that are used to govern both the ManagedEntities in this particular ManagementDomain as well as the ManagementDomain itself. The semantics of this aggregation are defined by the GoverningAuthorityMgmtPoliciesDetails association class. Appropriate subclasses of this association class are matched with their corresponding subclasses of ManagementPolicy, thereby preventing the needless proliferation of associations.

B.1.9 GoverningAuthorityMgmtDomainDetails Association Class

This is an association class, and represents the semantics of the GoverningAuthorityForMgmtDomain composition. This class defines generic characteristics and restrictions on behaviour limiting which ManagementDomains can be controlled by which GoverningAuthority; these characteristics and behaviour can be refined by the subclasses of this class. In addition, subclasses of ManagementDomain are matched to subclasses of this association class in order to prevent association explosion.

B.1.10 GoverningAuthorityMgmtPolicyDetails Association Class

This is an association class, and represents the semantics of the GovernsMgmtDomainUsingMgmtPolicies aggregation. This class defines generic characteristics and restrictions on behaviour limiting which ManagementPolicies can be used by which GoverningAuthority; these characteristics and behaviour can be refined by the subclasses of this class. In addition, subclasses of ManagementPolicy are matched to subclasses of this association class in order to prevent association explosion.

B.1.11 Context Class

The Context of an Entity is a collection of knowledge and data that result from the set of all interrelated conditions in which an Entity exists. Events point out changing conditions that may affect that Entity; an appropriate governance mechanism, such as policy rules, then defines a set of actions in response to the Event(s) to change or maintain the state of the Entity according to these conditions and actions. Context can have multiple distinct sets of related data and knowledge that are used to adjust its state in accordance with the changes in the environment that it exists in. The DEN-ng model represents this as a set of aggregations to ContextData, where ContextData is a class that focuses on one specific type of data and/or knowledge that is aggregated by the Entity's Context.

B.1.12 DomainHasContext Aggregation

This aggregation defines the set of Contexts that are associated with this particular Domain. Note that each Context can have multiple aspects, which are modelled as ContextData objects.

B.1.13 DomainHasContextData Aggregation

This aggregation defines the set of ContextData objects that are associated with this particular Domain. Each ContextData represents one aspect of the overall Context of the Domain.

B.1.14 PolicyRuleStructure Class

This is an abstract class; it is used to represent the structure of a policy rule. Supported rule types include CA (condition-action, for backwards compatibility), ECA (event-condition- action, preferred over CA), Goal, and Utility policies. More formally, the purpose of this class is to define different subclasses that each formalise the semantics of different types of Policy Rules using a subsumption relationship. This enables a system (such as FOCALE) that uses DEN-ng to import different types of Policy Rules, each with their own specific structure, and represent how each is used. This provides extensibility, so that new Policy Rule types can be added without adversely affecting the overall design of the DEN-ng Policy Hierarchy. From an ontological perspective,

it is important to separate the semantics of the structural representation of the policy rule from other concepts that are required to use the Policy Rule, such as Policy Target and Policy Subject. This enables particular policy types, for example Management Policy, to add them as required. An example of a subclass that can be defined from PolicyRuleStructure is ECAPolicyRule, which formalises the semantics of a Policy Rule with an Event, Condition, Action structure. In essence, an ECAPolicyRule is a Policy Rule that has a Policy Event, a Policy Condition and a Policy Action.

B.1.15 ContextSelectsPolicies Aggregation

This aggregation selects a set of PolicyRules that are now applicable based on this particular set of Context information.

B.1.16 ContextDataSelectsPolicies Aggregation

This aggregation selects a set of PolicyRules that are now applicable based on this particular set of ContextData.

B.1.17 ManagementPolicyHasPolicyRules Aggregation

This aggregation defines the set of PolicyRules that are contained in this Management-Policy. Since the aggregation is between PolicyRuleStructure and ManagementPolicy, any or all types of PolicyRule (e.g., ECA, Goal, or Utility) can be used.

B.2 The New DEN-ng Federated Domain Model

The theory behind the DEN-ng FederatedDomain model is to retain the similarity to real-world federations and merge this with the context-aware nature of FOCALE and DEN-ng. Thus, a FederatedDomain is a set of (DEN-ng) Domains; the actual organisation of Domains in the Federation can be linear or hierarchical as necessary. The overall governance principles for the FederatedDomain are defined through Policy Rules (and optionally, state automata) but are related to the current Context of the Federation. Recall that in DEN-ng, a Context is an aggregate object, consisting of one or more aspects, called ContextData objects. Hence, in steady state, the FederatedDomain would have a particular Context, and each of the constituent Domains in the FederatedDomain would have a corresponding ContextData, which corresponds to the (local) context of

the Domain. Hence, each of the ContextData objects are part of the overall Context object, which means that each of the local contexts of each Domain contribute to the overall context of the Federation. This allows the different aspects of Context (the ContextData entities) to signal that they have a local change that may or may not affect the overall Context (and vice-versa). However, it is likely that not all contextual information from each Domain will be available for federation, due to privacy and other considerations. Hence, a new concept, called a FederatedContext, is introduced, which represents the set of contextual data that is allowed to be seen and used by the Federation. This is shown conceptually in Figure B.2.

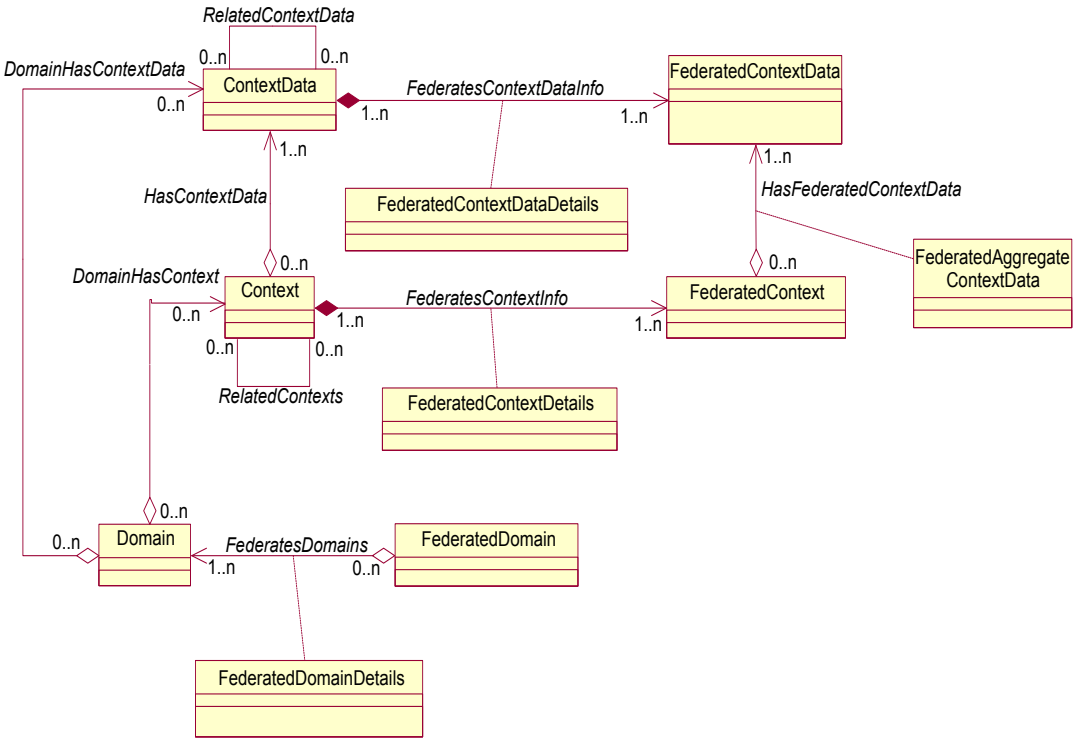

Figure B.2: Conceptual Relationship between Context, ContextData, FederatedDomain, and Domain

B.2.1 The FederatedContext Class

A FederatedContext represents the overall aggregate contextual information for a FederatedDomain. It collects local Context data from each local Domain in the Federation

and then filters the contextual information according to a set of Context-Aware Policy Rules. This is realised using the FederatedContextDetails association class of the FederatesContextInfo composition, and enables privacy and other rules governing the usage of contextual information to be enforced.

B.2.2 The FederatedContextData Class

A FederatedContextData represents the overall aggregate contextual information for an individual Domain in a Federation. Since each ContextData object represents a different aspect of contextual information, each ContextData object may have different visibility, access, and other rules that govern its usage. Hence, a FederatedContextData object collects local ContextData information from the local Domain in the Federation and then filters the contextual information according to a set of Context-Aware Policy Rules. This is realised using the FederatedContextDataDetails association class of the FederatesContextDataInfo composition, and enables privacy and other rules governing the usage of contextual information to be enforced.

B.2.3 The-FederatesContextInfo-Composition

This composition defines the set of Context information that is available to be used by a Federation. The semantics of choosing the subset of Context information that is made available to the Federation is defined by the FederatedContextDetails association class.

B.2.4 The FederatedContextDetails Association Class

This is an association class, and defines the semantics of the FederatesContextInfo composition. This class is designed to be a container, whose attributes and relationships are populated by a set of external applications; these attributes and relationships are then used to control the semantics of the aggregation, enabling which Context information can be used to produce a Federated Context.

B.2.5 The FederatesContextDataInfo Composition

This composition defines the set of ContextData that is available to be used by a Federation. The semantics of choosing the subset of ContextData that is made available to the Federation is defined by the FederatedContextDataDetails association class.

B.2.6 The FederatedContextDataDetails Association Class

This is an association class, and defines the semantics of the FederatesContextDataInfo composition. This class is designed to be a container, whose attributes and relationships are populated by a set of external applications; these attributes and relationships are then used to control the semantics of the aggregation, enabling which ContextData information can be used to produce a Federated ContextData object.

B.2.7 The HasFederatedContextData Aggregation

This aggregation defines the set of individual FederatedContextData elements that collectively determine the overall FederatedContext of the Entity. The semantics of this aggregation are represented by the FederatedAggregateContextDetails association class.

B.2.8 The FederatedAggregateContextDetails Association Class

This is an association class, and defines the semantics of the HasFederatedContextData composition. This class is designed to be a container, whose attributes and relationships are populated by a set of external applications; these attributes and relationships are then used to control the semantics of the aggregation, enabling which Context information can be used to produce a Federated Context.

B.2.9 The FederatedDomain Class

A FederatedDomain is defined as a collection of Domains in which each Domain in the Federation agrees to use zero or more global and zero or more local Policy Rules to govern the operation of the ManagedEntities that they contain. (Note that in principle, a different governance mechanism could be substituted for Policy Rules; this document simplifies this and limits the governance mechanism to Policy Rules for the time being. Note also that in FOCALE, the combination of Policy Rules and finite state automata are used, which has proven to be very effective; this design can also use the FOCALE approach.) The Federation is itself a ManagedEntity, and is typically logically centralised, but physically distributed. However, DEN-ng allows for logical distribution as well, as explained in the preceding Section. In a federation, if the governance model allows for autonomous or semi-autonomous constituent Domains, then the self-governing status of those Domains cannot be altered by the FederatedDomain that

contains them. The basis of the Federation may include social, political, geographical, and/or governance mechanisms that must be applied to all constituent Domains in order to govern behaviour that is of mutual interest. This is represented by appropriate Policy Rules (along with state automata to orchestrate the behaviour represented by the Policy Rules). Note, however, that each constituent Domain can act autonomously in other matters that are outside the governance provisions of the Federation.

B.2.10 The FederatesDomains Aggregation

This aggregation defines the set of individual Domains that are federated into this particular FederatedDomain. The semantics of this aggregation are defined by the FederatedDomainDetails association class.

B.2.11 The FederatedDomainDetails Association Class

This is an association class that implements the semantics of the FederatesDomains aggregation. This class serves as a container, whose attributes can be populated to suit the needs of the application(s) using this part of the model to restrict which policies can be used by which FederatedDomains for determining which Domains can participate in a given Federation.

B.3 Examples of Context-Aware Policy Governance

This section provides two different examples of how to use the DEN-ng context-aware policy rules to control how contextual information is used by Domains and FederatedDomains.

B.3.1 Applying Context-Aware Policy Rules to Govern Contextual Federation

Figure B.3 shows a customisable template for applying DEN-ng context-aware policy rules to manage the federation of contextual information. Recall that a ManagementPolicy provides deontic rules independent of the structure of the rule; hence, the actual policy rule content can be expressed as event-condition-action, goal, and/or utility function policy rules. Each of the three association classes (FederatedContextDataDetails, FederatedAggregateContextDetails, and FederatedContextDetails) has an aggregation

between itself and ManagementPolicy, which defines the set of ManagementPolicies used to govern the attributes and behaviour of each of the association classes. This indirectly governs the behaviour of the composition or aggregation that each association class represents the semantics of. Hence, external applications can use the designated ManagementPolicies to govern behaviour of how contextual information is federated.

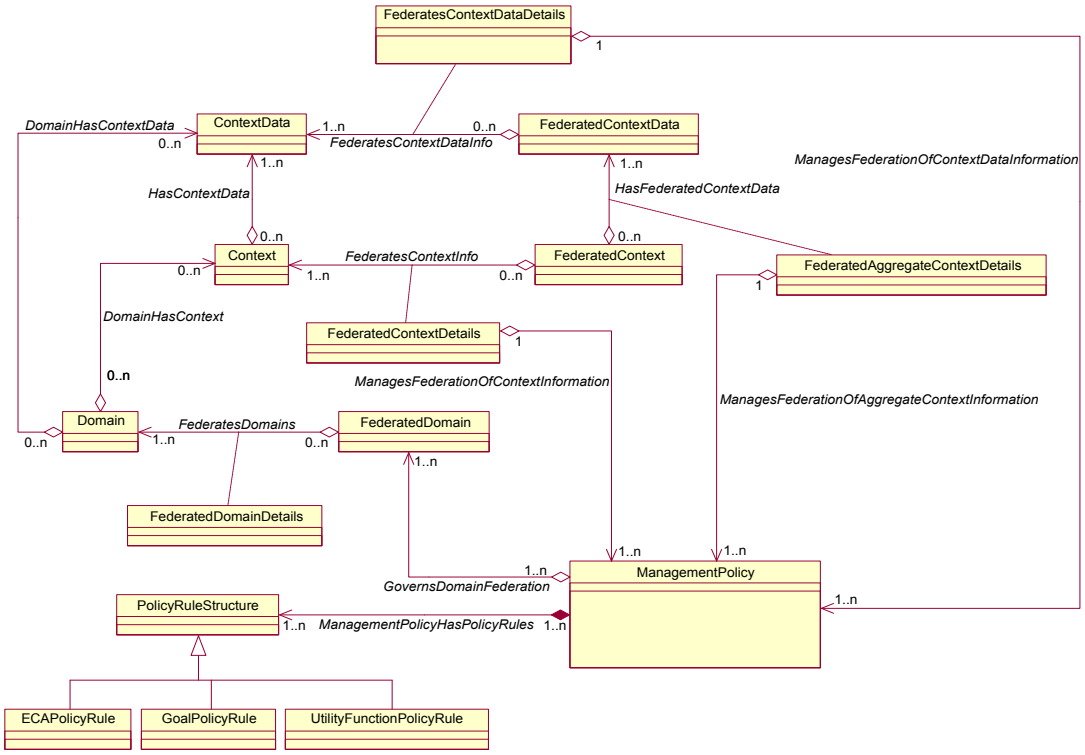

Figure B.3: Applying Context-Aware PolicyRules to FederatedContext

B.3.2 Applying Context-Aware Policy Rules to Govern Domain Federation

A similar approach to the above is shown in Figure B.4. In this figure, the two associations PoliciesGoverningFederationOfDomains and PoliciesGoverningFederatedDomainOperation define the set of ManagementPolicies that are used to control the attributes and behavior of the FederatedDomainDetails and GovernsFederatedDomainDetails association classes, respectively. This enables external applications to use these policies to indirectly control the associations governing how Domains are federated (e.g., which

Domains can be federated, for how long, etc.) and how the FederatedDomain itself is managed, respectively.

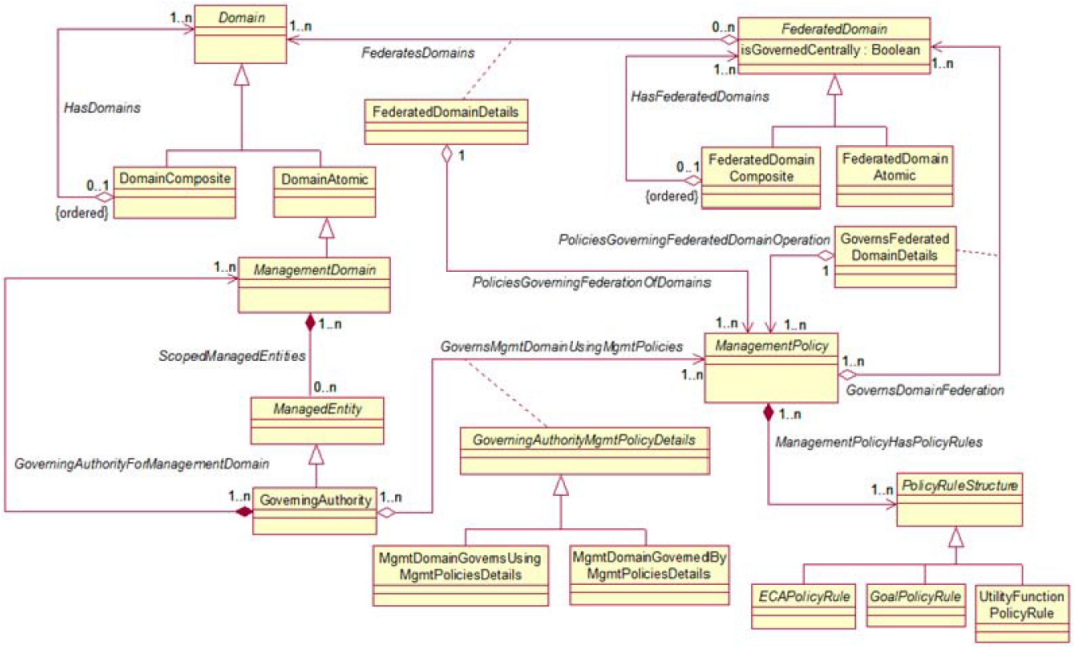

Figure B.4: Applying Context-Aware PolicyRules to FederatedDomains

B.4 The Overall DEN-ng FederatedDomain Model

Figure B.5 shows a simplified view of the DEN-ng FederatedDomain model.

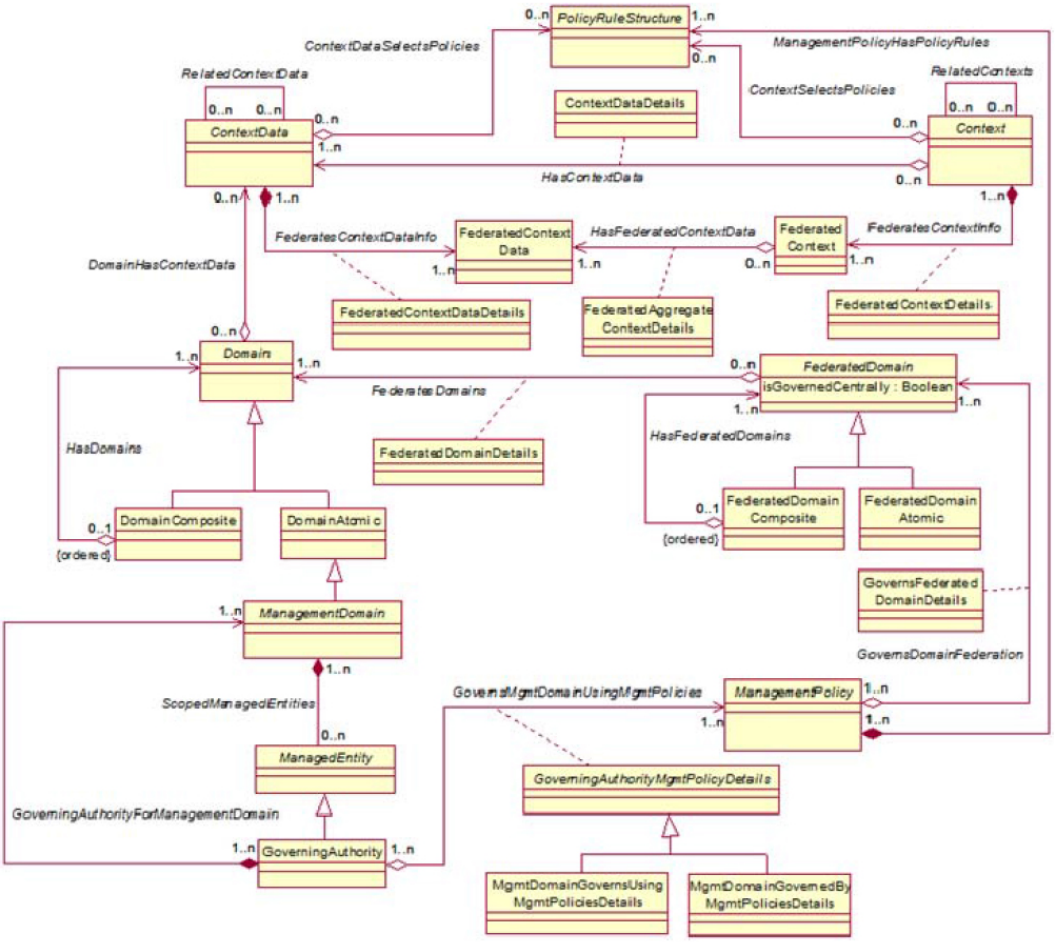

Figure B.5: Simplified FederatedDomain Model

B.4.1 The GovernsDomainFederation Aggregation

The GovernsDomainFederation aggregation defines the set of ManagementPolicies that govern this particular FederatedDomain. This includes operations such as access, management, and usage of ManagementPolicies to manage the FederatedDomain. The semantics of this aggregation are defined by the GovernsFederatedDomainDetails association class. It is important to note that a FederatedDomain itself is not capable of management. Management actions are performed by the GoverningAuthority, and use ManagementPolicy instances for management actions (and other types of policy rules descended from PolicyRuleStructure for other types of policy actions). This in turn means that policies are not themselves federated; rather, they are associated to a

federation. This ensures that policies local to a ManagementDomain are not exposed to other policies that should be globally used in the Federation. This decouples domain-internal governing aspects from global-domain federation governing aspects, which is required for scalability, among other reasons. For example, this approach enables cascaded federations to be created without compromising local domain management (for those situations in which the federation is designed to support semi-autonomous constituent domains).

B.4.2 The GovernsFederatedDomainDetails Association Class

This is an association class that implements the semantics of the GovernsDomainFederation aggregation. This class serves as a container, whose attributes can be populated to suit the needs of the application(s) using this part of the model to restrict which policies can be for managing the operation of the FederatedDomain itself (as opposed to the Domains contained in the FederatedDomain).

Appendix C

DEN-ng Federated Domain Model Axioms

This appendix presents both domain and policy axioms that were transformed into ontological format based on a subset of the overall DEN-ng information model.

C.1 DEN-ng Domain Ontology Axioms

This section provides a diagram and brief description of the most common domain axioms used by the policy consistency analysis processes.

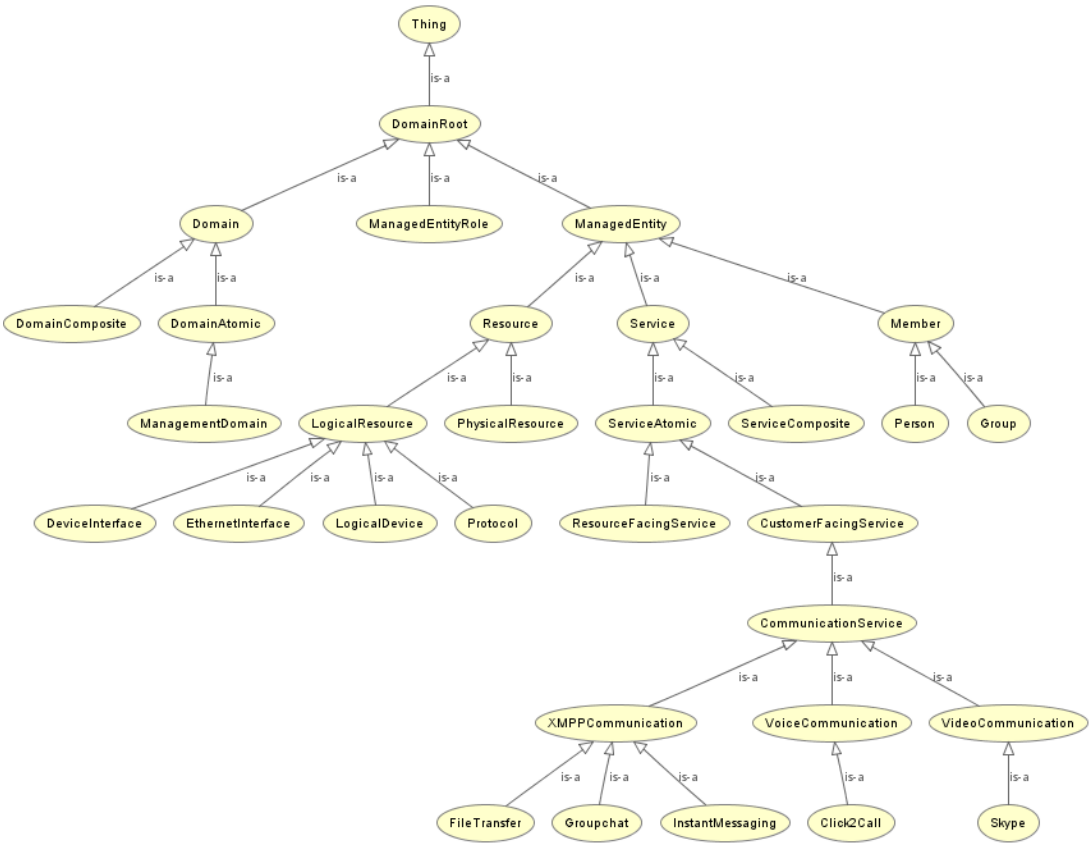

Figure C.1: Subset of overall DEN-ng Domain Model Axioms

A subset of the overall DEN-ng model used to represent typical aspects of a managed domain that were transformed into ontological format to create a domain model is depicted in Figure C.1.

C.1.1 Domain Root

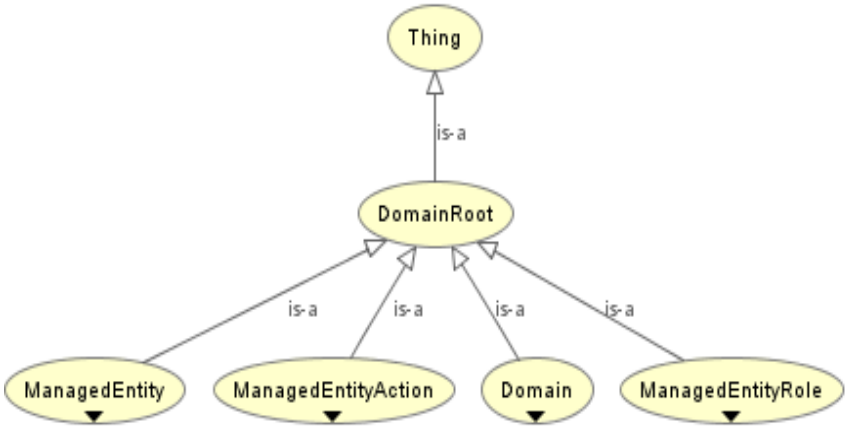

Figure C.2: Domain Root

A *DomainRoot* concept, depicted in Figure C.2 is an entity that is the superclass of all the concepts defined in the domain model.

C.1.2 Managed Entity

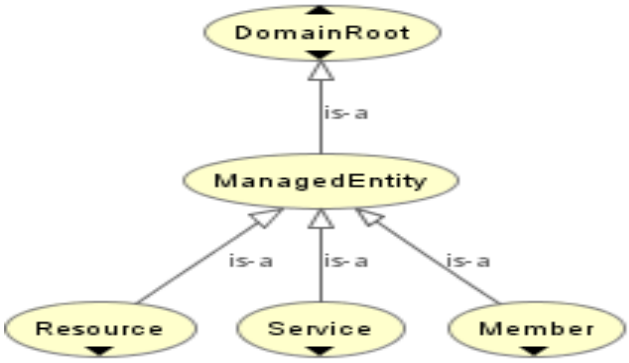

Figure C.3: Managed-entity

A *ManagedEntity* concept, depicted in Figure C.3 and defined by the axiom in Equation 3.8, is an entity that is uniquely addressable and manageable through the use of management policies. A ManagedEntity is something of interest that can be managed. Typical examples include Products (e.g., an application suite), Resources (e.g., network

devices and computers), and Services (e.g., VPNs and protocols). Any ManagedEntity can have Context or ContextData associated with it.

C.1.3 Service

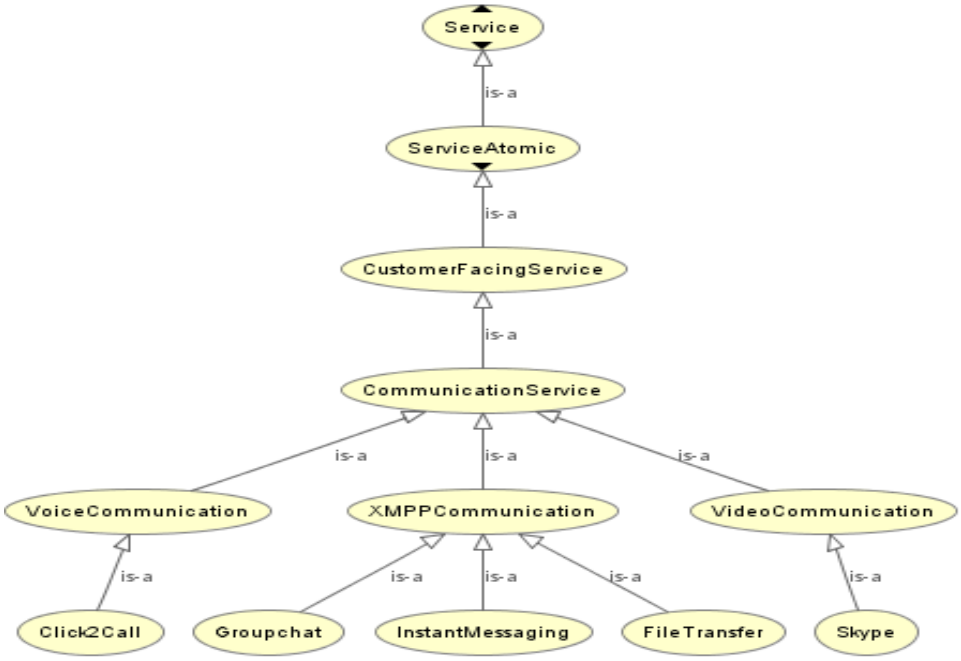

Figure C.4: Service

A *Service* concept, depicted in Figure C.4 and defined by the axiom in Equation 3.11, is either a customer facing service or a resource facing service. A customer facing service defines the characteristics and behaviour of a particular service as seen by the customer. This means that a customer purchases and/or is directly aware of the type of service and is in direct contrast to a resource facing service which support customer facing services, but are not seen or purchased directly by the customer. An example of a customer facing service is voice, video and/or instant messaging (IM) communication as it is seen and consumed by the customer, whereas, an example of a resource facing service may be the XMPP protocol used to provision the voice, video and/or instant messaging (IM) communication as it is utilised by resources, but not seen by the customer. It should be noted that this is an example of one type of service and that many other types of

services exist that can be extended from the service concept in the domain model.

C.1.4 Member

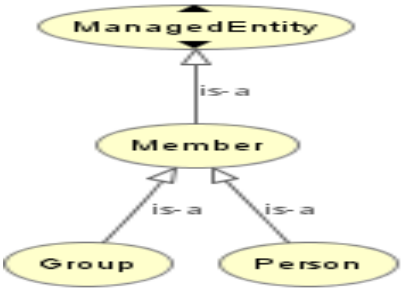

Figure C.5: Member

A *Member* concept, depicted in Figure C.5 is an individual person or a group of people belonging to a managed domain.

C.1.5 Managed Entity Role

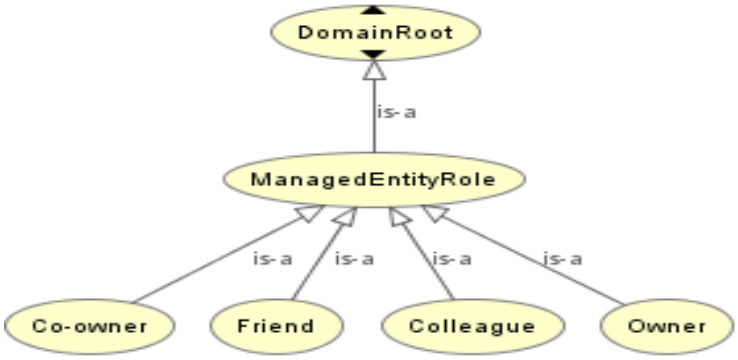

Figure C.6: Managed Entity Role

A *ManagedEntityRole* concept, depicted in Figure C.6 is an entity that takes on the role of either co-owner, friend, colleague, owner of a resource.

C.1.6 Resource

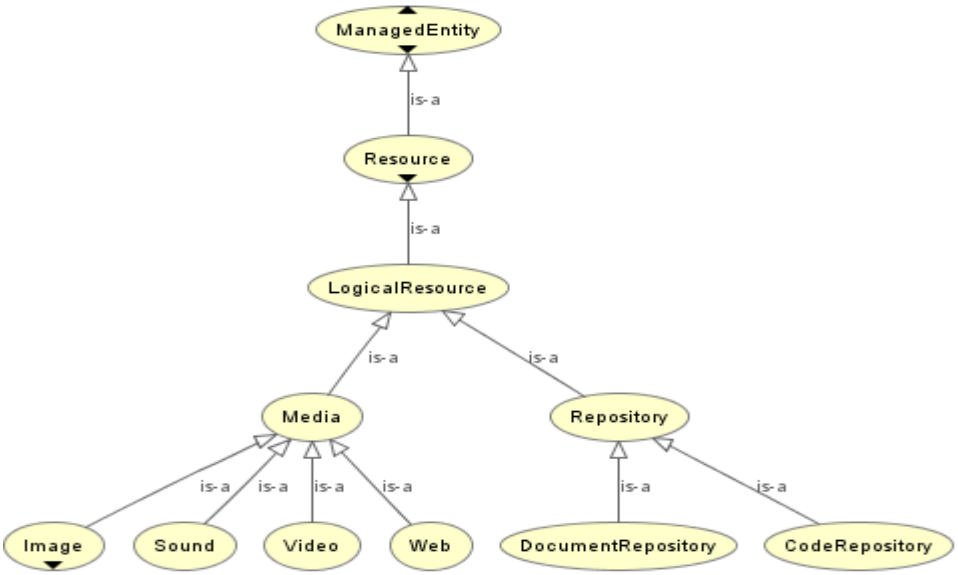

Figure C.7: Resource

A *Resource* concept, depicted in Figure C.7 and defined by the axiom in Equation 3.10, is defined in DEN-ng as either a physical or a logical resource. This enables a complex entity like a router to be split into its physical and logical components. An example of a physical component would be a piece of hardware such as a networking card, whereas a logical component may be a piece of software that runs on the device (e.g. protocols, ports, etc). Resource is the infrastructure that supports the provision of services and therefore the delivery of products to customers.

C.1.7 Managed Entity Action

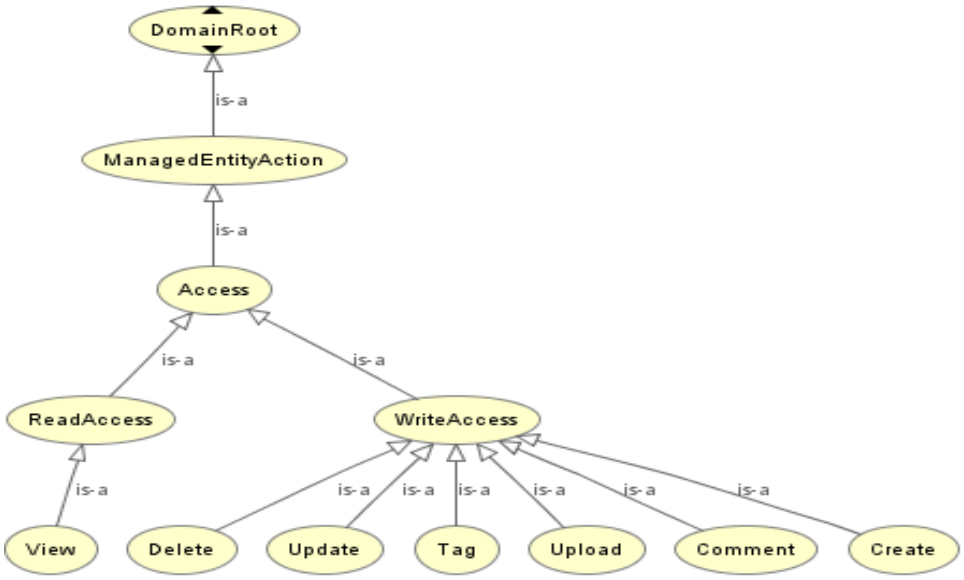

Figure C.8: Managed Entity Action

A *ManagedEntityAction* concept, depicted in Figure C.8 is an action performed by a managed entity and can be either a write access or a read access. A write access is either comment, delete, tag, update, upload or create. A read access is equated to view as no modification is made to the resource.

C.2 DEN-ng Policy Ontology Axioms

This section provides a diagram and brief description of the most common policy axioms used by the policy consistency analysis processes.

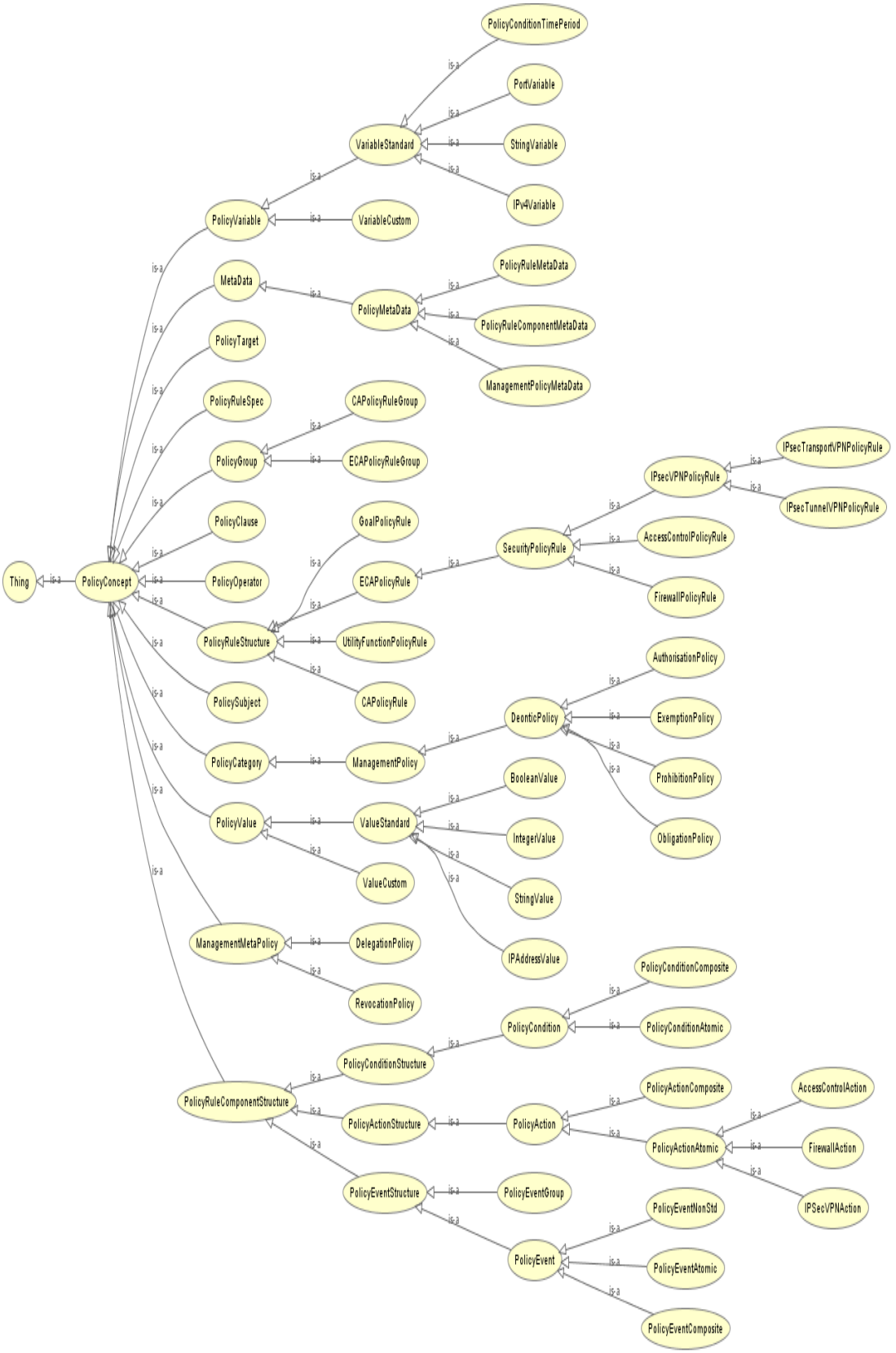

Figure C.9: Subset of overall DEN-ng Policy Model Axioms

A subset of the overall DEN-ng model used to represent typical aspects of management policies that were transformed into ontological format to create a policy model is depicted in Figure C.9.

C.2.1 Policy Concept

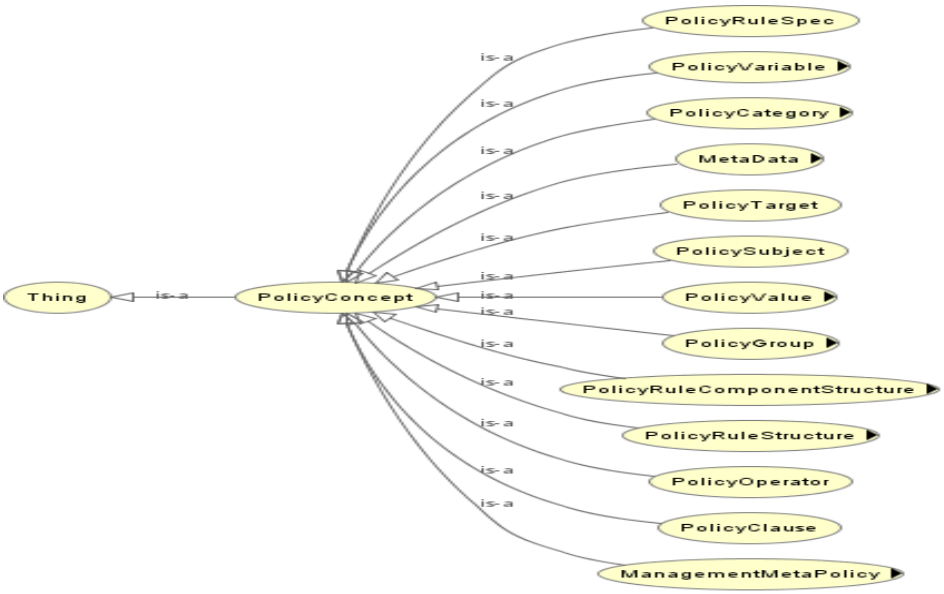

Figure C.10: Policy Concept

A *PolicyConcept* concept, depicted in Figure C.10 and defined by the axiom in Equation 3.12, is the superclass for the classes that together constitute the definition and representation of a Policy Rule and its components. A PolicyConcept is the root of the DEN-ng Policy model. As such, it defines common attributes, methods and relationships that all policy subclasses use and take part in. This class is named PolicyConcept because it does not define the characteristics and behaviour of a policy rule (or any of its components); it defines a part of the overall DEN-ng model that is concerned with modelling generic concepts related to policy.

C.2.2 Management Policy

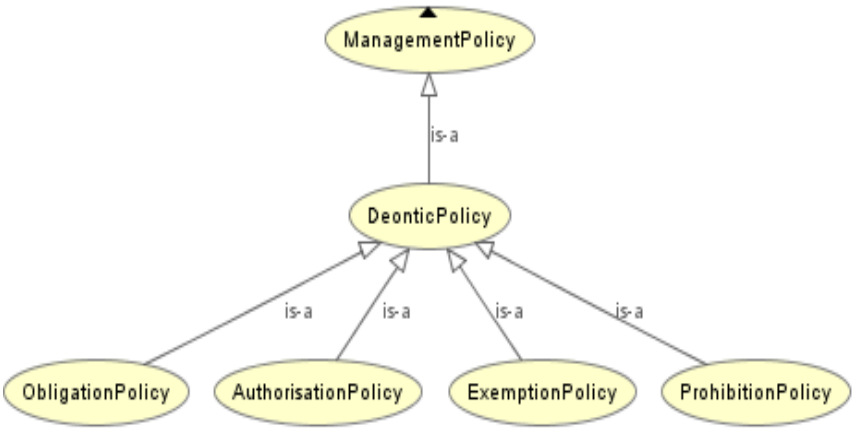

Figure C.11: Management Policy

A *ManagementPolicy* concept, depicted in Figure C.11 and defined by the axiom in Equation 3.15, is the superclass for PolicyRules that manage a system. As such, ManagementPolicy has explicit associations with PolicySubjects and PolicyTargets, defined through the SubjectInteractsWith and TargetInteractsWith relationships. ManagementPolicy aggregates one or more PolicyRuleStructure objects through the hasPolicyRules relationship to represent different types of policy rules (i.e ECA, CA, etc.) to realise many different forms of management directives. In order to represent popular policy types such as deontic policies, new ManagementPolicy subclasses are created.

C.2.3 Management Meta Policy

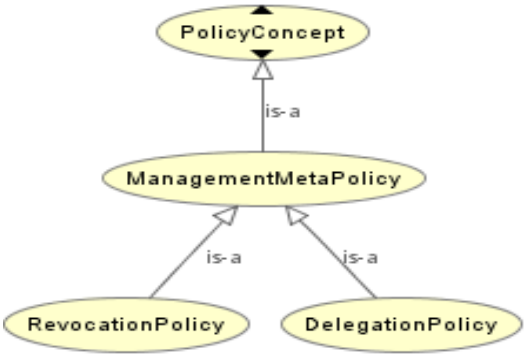

Figure C.12: Management Meta Policy

A *ManagementMetaPolicy* concept, depicted in Figure C.12 and defined by the axiom in Equation 3.18, is a meta policy that provides additional details regarding the semantics of a management policy. Exemption is, literally, immunity or release from an obligation. Deontic logicians assign the concept "need not" to authorization. A management meta policy is either a delegation policy or a revocation policy. A delegation policy grants some permissions from a delegator to a delegatee to access or interact with some resources. A revocation policy retracts the permissions granted from a delegator to a delegatee.

C.2.4 Meta Data

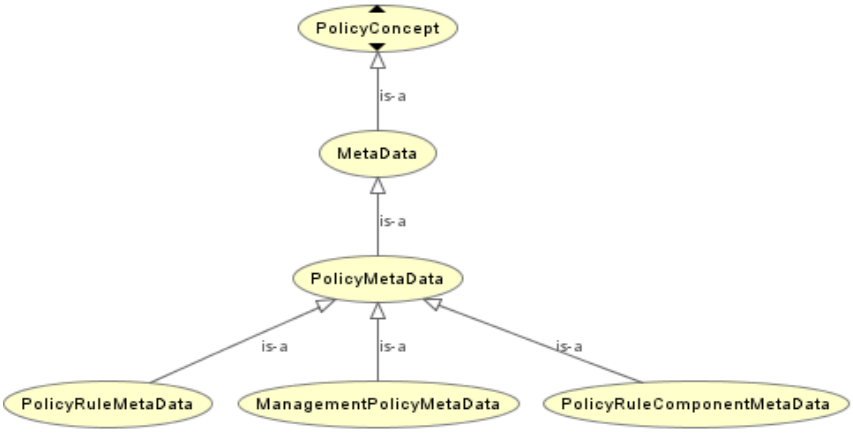

Figure C.13: Meta Data

A *MetaData* concept, depicted in Figure C.13 is a superclass for meta data pertaining to the management policy, the policy rule component and the policy rule.

C.2.5 Policy Rule Structure

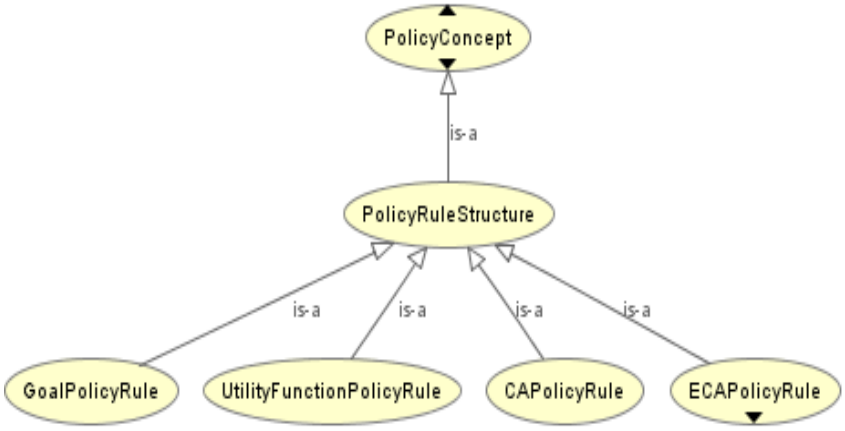

Figure C.14: Policy Rule Structure

A *PolicyRuleStructure* concept, depicted in Figure C.14 and defined by the axiom in Equation 3.19, is a superclass for a variety of policy rules defined in the policy model. This includes goal, condition-action (CA), event-condition-action (ECA) and utility

function policy rules. The most common types of policy rules are defined as being either an event-condition-action (ECA) rule or a condition-action (CA) rule. The difference being that with an ECA rule a particular event (or set of events) must occur in order for the rule's conditional entity to be evaluated. With the CA rule the event is considered implicit as is the case for access control policies where the event is a request to access some resource or service.

C.2.6 Security Policy Rule

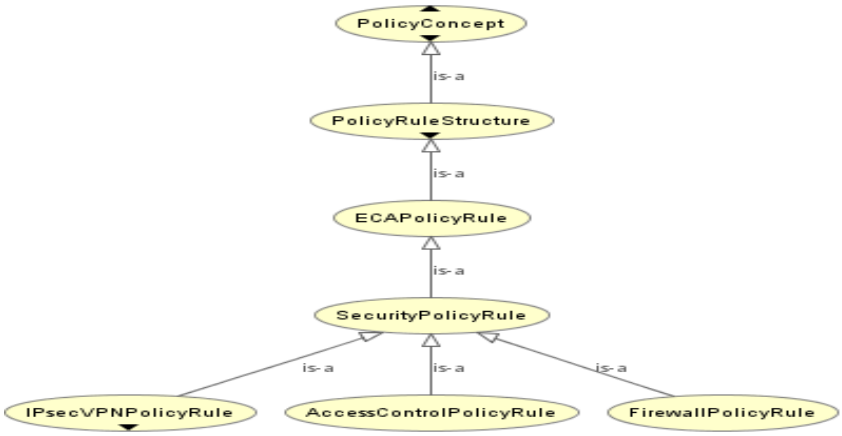

Figure C.15: Security Policy Rule

A *SecurityPolicyRule* concept, depicted in Figure C.15 is a superclass for the most common types of security policy rules such as firewall, access control and IPSec/VPN policy rules.

C.2.7 Policy Rule Component Structure

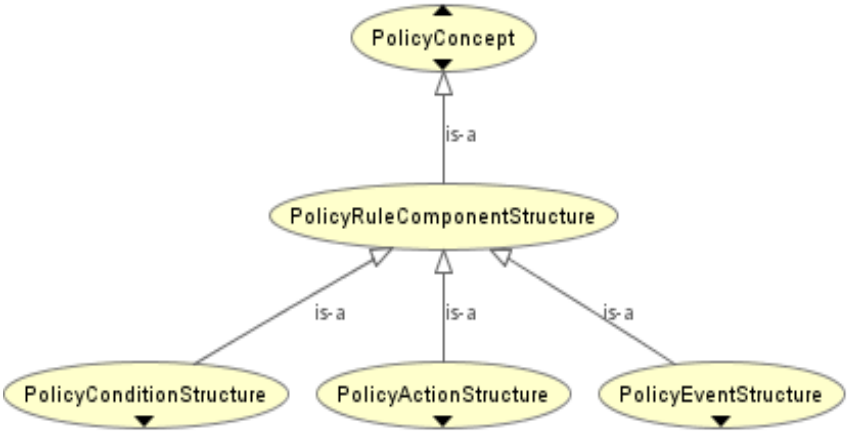

Figure C.16: Policy Rule Component Structure

A *PolicyRuleComponentStructure* concept, depicted in Figure C.16 and defined by the axiom in Equation 3.22, is a superclass for the policy condition structure, the policy action structure and the policy event structure classes. Since different types of Policy Rules have different structural components, the PolicyRuleComponentStructure class is used to represent the different types of Policy Rule Components that can be used in a PolicyRule. Typical example subclasses of this class include PolicyEvent, PolicyCondition, and PolicyAction (which are used together to define an ECAPolicyRule).

C.2.8 Policy Event

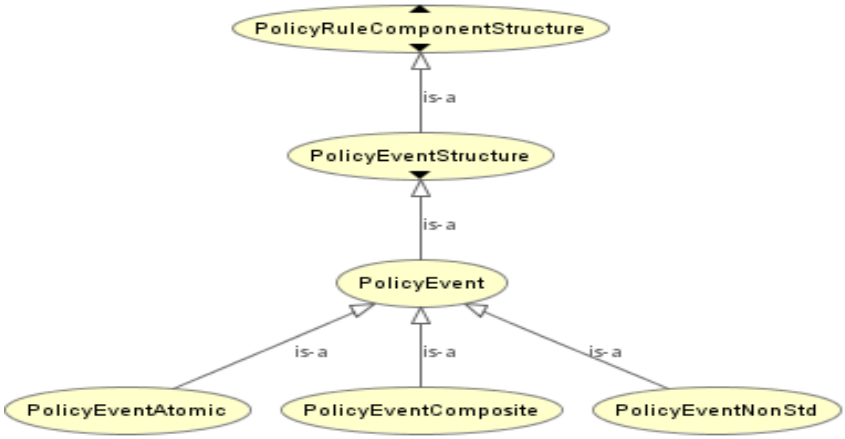

Figure C.17: Policy Event Structure

A *PolicyEvent* concept, depicted in Figure C.17 and defined by the axiom in Equation 3.23, is a superclass for atomic policy events, composite policy events, and non-standard policy events. PolicyEvent represents the occurrence of something significant and whose detection is used to trigger the evaluation of PolicyRule conditions. There are three types of PolicyEvent namely, PolicyEventAtomic, PolicyEventComposite, and PolicyEventNonStd. PolicyEventAtomic represents the happening of a single atomic occurrence. PolicyEventComposite represents the happening of multiple occurrences while PolicyEventNonStd is a generic extension mechanism for representing event instances that have not been modelled with the attributes specified in the DEN-ng model. PolicyEvent is similar to the concept Event in the policy description language and also the concept Event in the Ponder policy language.

C.2.9 Policy Condition

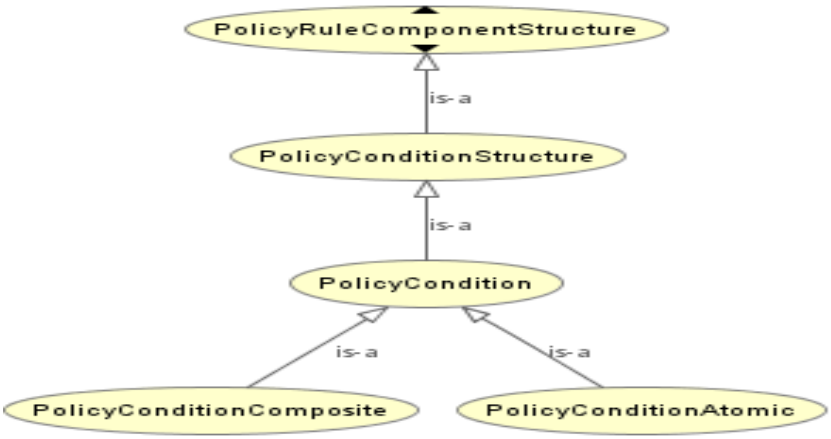

Figure C.18: Policy Condition Structure

A *PolicyCondition* concept, depicted in Figure C.18 and defined by the axiom in Equation 3.24, is a superclass for composite policy conditions, atomic policy conditions and non-standard policy conditions. PolicyCondition represents constraints that must hold true before the action of a PolicyRule can be executed. There are three types of PolicyCondition namely, PolicyConditionAtomic, PolicyConditionComposite and PolicyConditionNonStd. PolicyConditionAtomic represents a singular constraint that must hold true before the action of a PolicyRule can be executed. While a PolicyConditionComposite represents multiple constraints, some or all of which must hold true before the action of a PolicyRule can be executed. PolicyConditionNonStd is a generic extension mechanism for representing event instances that have not been modelled with the attributes specified in the DEN-ng model. A concept analogous to PolicyCondition can be represented by many policy languages.

C.2.10 Policy Action

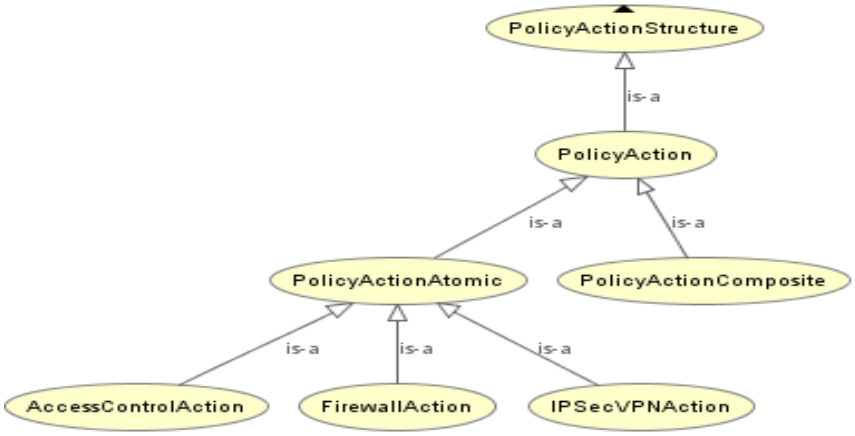

Figure C.19: Policy Action Structure

A *PolicyAction* concept, depicted in Figure C.19 and defined by the axiom in Equation 3.25, is a superclass for atomic policy actions, composite policy actions, and non-standard policy actions. PolicyAction represents some undertaking on a resource. There are three types of PolicyAction namely, PolicyActionAtomic, PolicyActionComposite and PolicyActionNonStd. PolicyActionAtomic represents the execution of one singular action while PolicyActionComposite represents multiple actions. PolicyActionNonStd is a generic extension mechanism for representing event instances that have not been modelled with the attributes specified in the DEN-ng model. A concept analogous to PolicyCondition can be represented by many policy languages.

C.2.11 Policy Variable

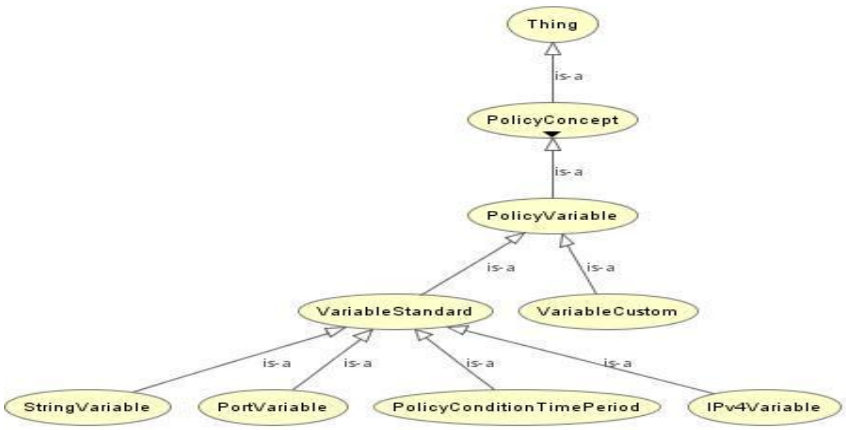

Figure C.20: Policy Variable

A *PolicyVariable* concept, depicted in Figure C.20 is a superclass for standard policy variables and customised policy variables.

C.2.12 Policy Value

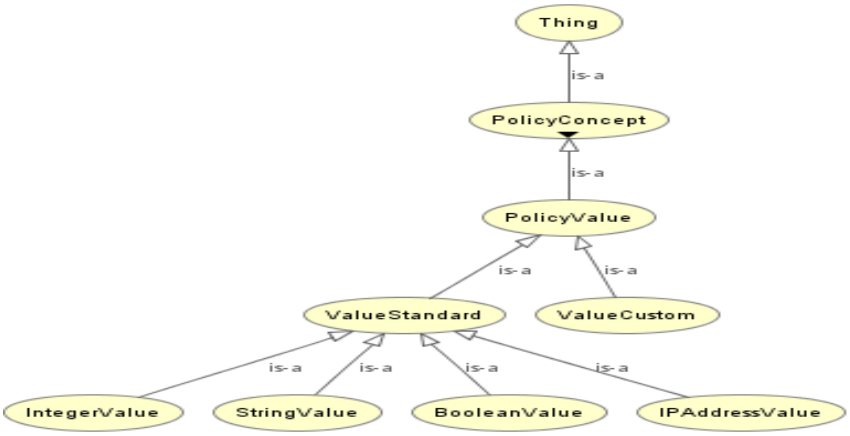

Figure C.21: Policy Value

A *PolicyValue* concept, depicted in Figure C.21 is a superclass for standard policy values and customised policy values.