Conflict Analysis During Authoring of Management Policies for Federations

Jason Barron, Steven Davy, and Brendan Jennings
FAME, Telecommunications Software & Systems Group
Waterford Institute of Technology, Ireland
{jbarron, sdavy, bjennings}@tssg.org

Abstract—We outline a policy conflict analysis process for the analysis of newly specified federation-level policies against previously deployed local/federation policies. The process is generic in the sense that it can be employed by each domain participating in a federation to help maintain the consistency of their local system with that of the federation. The conflict analysis process utilises an information model and associated ontology for representing both the static and dynamic application-specific aspects of the local operating environment and the federation to aid in the detection of potential inconsistencies. It employs two algorithms, one for the selection of previously deployed policies related to the federation-level policy and the other for analysing the returned policies against the federation-level policy. The selection algorithm reduces the number of deployed policies required to be retrieved for analysis against any newly specified federation-level policy, while the conflict analysis algorithm detects inconsistencies relating to the conditional element of a policy rule. We discuss a concrete example in the form of a federated XMPP communication scenario.

Index Terms—Federation, Policy Conflicts, Policy Conflict Analysis

I. INTRODUCTION

TODAY’S service environment is a rapidly changing one in which businesses require services that can be adapted to their operating needs. An example of one such service is Extensible Messaging Presence Protocol (XMPP) communication services that cross organisational boundaries. In order to allow multiple domains to join (or federate) their XMPP communication services, management processes need to be in place to allow control over the permitted communications between domains. Policies can be used to control the management of such services between federated domains. However, even though policies can be deployed to govern this form of federated communication, their use is still impeded by their heterogeneous nature in that each domain participating in a federation may be using different information models, policy based management systems and policy languages. Although these problems exist in single domain environments employing arbitrary levels of network management, they are exacerbated when dealing with federated domains.

“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-level policies have the potential to conflict with previously deployed local/federation policies at arbitrary levels of abstraction within each domain. This suggests the need to implement federated policy conflict analysis processes as part of an overall federated policy authoring process that includes federated policy specification. These processes would need to be generic in nature so that they can be used by each member of the federation 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.

To aid in the detection of policy conflicts, context information based on work carried out in [12] 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, device-level policy languages typically do not convey context data in their specification. This means that local context data would need to be retrieved from higher-level system policies that are related to the deployed policies to provide context information for the local domain. Federation-level policies will contain context data regarding the operating semantics of the federation such as the entities involved, time, location, activity and whether the entity/entities was a person or a group. This federation context data can be used to aid policy conflict analysis processes. 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 federation. Reasoning can then be performed on this overall context information to determine inconsistencies between proposed federation-level policies and deployed local/federation policies.

The DEN-ng federated domain model is an extension of the DEN-ng information model [11] that can facilitate the modelling of federated domain environments. The main advantage that DEN-ng has over other information models is that it is not only used to represent the structure of management information, but also defines a management methodology that determines how the management of large scale communications networks should be achieved. The DEN-ng information model can represent context information regarding federated and local domain environments. Unfortunately, information models are limited in the types of information they can represent and reason over and are more suited to modelling environments that are static. However, information models are
ideal for representing the structural aspects of domains and can be augmented with ontologies to represent the dynamic relationships (behaviour) that occur between the entities of domains as has been demonstrated by the work carried out in [5]. This work will further demonstrate that when used in conjunction with each other they can provide policy specification and conflict analysis processes more powerful modelling of domains as this allows structural and semantic information relevant to each operating environment within each domain to be represented.

This paper builds on our previous work [3] where the DEN-ng federated domain model extension was used to assist in the specification of federation-level policies. Assuming that federation-level policies have been specified using the process outlined in [3], they will need to be checked for conflicts during the deployment phase. This work describes how conflicts between newly deployed (or updated) federation-level policies and previously deployed local/federation policies can be detected. We have transformed the DEN-ng federated domain model into an OWL-DL representation that allows for additional domain specific semantics (entity relationships) to be added to the model. The conflict analysis process is a generic process where deployed policies are abstracted to a higher level and represented as instances in the ontology. The OWL-DL model can then be reasoned over to detect relationships and potential conflicts between policy pairs.

The outline of this paper is as follows: §II outlines related work; §III introduces our proposed federated policy conflict analysis approach. §IV presents an application of the federated policy conflict analysis process on a federated XMPP communication scenario. §V demonstrates an implementation of the selection and conflict analysis algorithms, while §VI evaluates the federated policy selection and policy conflict analysis processes. Finally, §VII summarises the paper and outlines directions for further work.

II. RELATED WORK

Most of the published work to-date with regards to policy selection and policy conflict analysis concerns their implementation within single domain environments. We provide a brief overview of this body of work.

There has been some work carried out to-date in the transformation of policies into ontologies and then reasoning over them. Campbell and Turner [4] specify policies in APPEL which are then transformed into OWL using POPPET before being reasoned over using PELLET. This work focuses on call control policies, where static and dynamic conflicts can be detected. However, this work is only concerned with dynamic aspects. Conflict detection is based on comparing the actions of a pair of policies to detect application-specific conflicts. Resolution is achieved through the use of resolution policies that specify the action to be taken when two policies conflict. Resolution policies are based on domain-specific knowledge and the use of applying policy priorities. This work is only applicable to a single-domain environment as it is grounded in the use of APPEL. 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.

In previous work [5] we 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 the conflict analysis algorithm using a conflict signature matrix. That work is similar to this work in respect of policy selection and conflict analyses. However, it has only been applied to a single-domain policy conflict analysis environment and has yet to be extended to a federated domain environment. This work aims to apply the policy selection and conflict analyses concepts to federated domain environments.

Fitzgerald and Foley [7] use ontologies to analyse firewall configurations associated with semantic web applications. They attempt to define an appropriate firewall configuration that is aligned with the type of semantic web application being hosted by the system. An ontology is used to represent the knowledge relating to a semantic web application. They develop ontologies to represent Linux Netfilter for filtering capabilities and TCP-Wrapper which is used to represent and reason over network access control configurations. This work is based on access control, but not the higher-level role-based access control, but the lower-level firewall filter rules used by applications such as Netfilter.

The definition of semantic conflicts is outlined by Calero et al. [1], who provide a taxonomy of semantic conflicts that can occur and outline some realistic scenarios based on the main types of semantic conflict. This includes conflict of authority, redundancy conflict, conflict of priorities, conflict of authority, multiple-managers conflict and self-management conflict. The CIM model which is specified in OWL is used to represent policies and SWRL rules are used to define the behaviour of the system.

Latré et al. [8] 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. Their work is similar to ours 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, it differs in that they introduce extensions to the DEN-ng information model to model how distributed entities use remote contextual data to optimise QoE services.

OWL and SWRL are used by Liu et al. [9] to describe context knowledge and use SWRL rules to represent rule-based inference rules for context reasoning. However, they do not use the modelling power available through the use of an information model. The work is also not aimed at policy conflict analysis.

The work carried out by Parreiras and Staab [10] utilises
ontologies with UML. They outline a framework using different concrete syntaxes for developing integrated models, while using a SPARQL-like approach for writing query operations. This work is based on the use of information models, but is aimed at specifying non-functional software requirements.

III. POLICY CONFLICT ANALYSIS APPROACH

In this section we outline our policy conflict analysis approach, which is depicted in Figure 1. An information model can be used to represent the structure of federated domains. The relevant aspects of the information model related to the application domain and the policy model used are transformed into ontologies, namely a domain ontology, a federation ontology and a policy ontology. The approach taken can be repeated in each domain to allow for their heterogeneity. Our federated policy conflict analysis 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 used by Barrett et al. [2]. The second phase involves the retrieval of currently deployed local policies that are related to the newly specified federation-level policy. The set of retrieved policies are then analysed to determine their semantic relationship and overall operating context relative to the federation-level policy. The third phase applies policy conflict analysis across a federated domain. The policy conflict analysis process will need to be employed under certain circumstances. The following cases have been identified as possible situations where inconsistency could be introduced into the system and so warrants policy conflict 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 conflict analysis process to ensure the consistency of the federation is maintained.

- **Re-negotiation of federation-level policies**
  A network operator as a member of the federation may wish to re-negotiate certain aspects of the federation. This may lead to new federation-level policies being specified or the updating of previously deployed federation-level 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-level policies. However, there also exists the case where another federation member wishes to implement a change to its local or federation-level policies. This will also require analysis of any changed local or federation-level policies.

A. DEN-ng Federated Domain model Ontology

The DEN-ng federated domain model depicted in Figure 2 is an extension of the DEN-ng information model [11] and includes support for representing the structure of federated domains. Most importantly, the DEN-ng federated domain model can represent context pertaining to a federated domain. This context data is crucial as input to any policy conflict analysis processes and can be obtained from each domain to give an overall view of context data for a federation of domains. The context data related to federation-level policies can be used to assist policy conflict analysis processes to determine situations in which deployed policies may conflict with the federation-level policy if that federation policy were to be deployed. Unfortunately, information models are limited by the fact that they can only represent the structure of entities and not the behaviour between entities. This has a negative impact on the types of reasoning that can be performed using information models on their own. However, they can be augmented with ontologies to provide a more detailed representation of the semantic relationships between domain entities and more importantly allow more powerful reasoning to be performed over the model. As part of this work, 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 structural aspects of federated domains to be represented. The OWL-DL model can then be enhanced by defining the dynamic relationships (behaviour) that occurs between entities of the federated domains.

The steps in phase one of the process which is the transformation of a subset of the DEN-ng information model to an ontology representation are outlined below:

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

Medium to large scale networks will contain a large number of deployed local policies. Returning all these deployed policies for analysis would lead to inefficiency of the conflict analysis algorithm as not all deployed policies will be related to the newly specified federation-level policy and hence will not need to be checked for consistency against the federation-level policy. A selection algorithm can be used to return only those policies that are related to the newly specified federation-level policy for analysis. SPARQL is a query language that can be used to query RDF graphs and only return the queried data. However, SWRL an alternative query language can be used to retrieve the queried data and also infer relationships between the queried data, so for this reason SWRL was chosen for the query language. SWRL rules can be created and used for the selection of related previously deployed local policies based on policy rule entity attributes such as subject, target, and action from the federation-level policy. The subject, target or action can be a superset, subset or equivalent to the policy rule entities from the local deployed policies. The policies need to be related by at least one of these relationships in order for them to be selected. The execution of the SWRL rule is less costly in terms of algorithmic efficiency than performing conflict analysis on all deployed policies. The steps in the second phase of the process which employs the policy selection algorithm process are outlined as:

- **PSA 1**: Instance of newly specified federation-level policy is added to the ontology
- **PSA 2**: SWRL rules are created, related to the newly specified federation-level policy
- **PSA 3**: Similar previously deployed policies are searched using SWRL rule and returned
- **PSA 4**: Returned policies are added as instances to the ontology

C. Policy Conflict Analysis Algorithm

Once the relevant set of related policies have been selected and returned. Higher-level system policies related to these selected low-level deployed policies will need to be retrieved. This is because lower-level policies lack semantic information pertaining to their implementation. The higher-level policies can be used to obtain context data relating to the operating context of the deployed local policies. This semantic information can assist the conflict analysis algorithm to detect inconsistencies with the proposed federation-level policy if the federation-level policy were to be deployed. The selected policies are checked for overlap based on policy rule condition relationships. The types of policy rule condition relationships that can be checked for are condition subset, superset, overlap and equality. The constraints on the algorithm are that it needs to be generic enough to be deployed across multiple domains and applicable to multiple applications. The steps in the third phase of the process which employs the policy conflict analysis algorithm process are outlined as:

- **PCA 1**: Policy instances are reasoned over based on policy rule condition relationships
- **PCA 2**: Additional reasoning can be applied to the instances
- **PCA 3**: Policy author is notified of reasoning results

D. Federated Policy Conflict Analysis

The policy conflict analysis process 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 the FRM [6]) 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-level policy needs to take place. The steps in the federated policy conflict analysis process are:

- **FPCA 1**: Service provider A (SPA) specifies a new federation-level policy
- **FPCA 2**: Policy selection algorithm applied
- **FPCA 3**: Conflict detection algorithm applied
- **FPCA 4**: SPA creates policy deployment request
- **FPCA 5**: Policy request sent through FRM
- **FPCA 6**: SPB/SPN specifies a new federation-level policy
- **FPCA 7**: Each service provider repeats Step 2 and Step 3
- **FPCA 8**: Status report sent back to SPA (if policy not capable of being deployed, re-negotiation of policy occurs (i.e. step4))
- **FPCA 9**: Each service provider either commits the federation policy or performs a rollback

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

IV. SCENARIO

We now describe an application of the federated policy conflict analysis process based on an inter-enterprise XMPP communication scenario. The scenario depicted in Figure 3 is based on three enterprises, AAA Consultancy, a software consultancy company that has been contracted to provide consultancy services to ISBank, a financial institution and DBDataStore, a database company that has been contracted to implement database services within ISBank. Each enterprise has a hierarchical XMPP grouping structure based on its internal organisational hierarchy.

A member of the Java consultants group from the consultancy company has been contracted to provide consultancy services and as such will be stationed at the bank’s premise for the majority of the contract. While stationed at the bank, a Java consultant from the consultancy firm and a database administrator (DBA) from the database services firm will be temporary members of the software development group (SD) and will require participation in XMPP communication services such as Instant Messaging (IM), multi-party chat and document transfer with the group leader of that particular SD group within the bank, and the Chief Technology Officer (CTO) of the bank. The Java consultant will require XMPP communication services with its in-house Java consultants group back in the consultancy company while stationed at the bank. Similarly, the DBA will also require XMPP communication services with its DBA group back in the database firm while stationed at the bank.

Furthermore, while stationed at the bank, neither the Java consultant nor the DBA will be permitted to participate in any XMPP communication services available with any other members of the SD group of which they are temporary members or any other XMPP groups or individuals within the bank. However, the Java consultant will require XMPP communication services with the DBA (and vice versa). This will require each enterprise to federate its XMPP communication services. In order to control the federation of the XMPP communication services, high-level federation policies will need to be specified. These federation policies can specify the structure and required behaviour of the federation, so as to manage XMPP communications traffic flowing between the three domains. A negotiation process will be required between the three domains in order to create and agree upon these federation-level policies. This negotiation process (FRM) is assumed to exist and is not part of this work. The federation policies can then be translated into system-level policies before being enforced on devices. In this scenario, XACML has been chosen for the deployment of system-level policies.

The Java developer and the Database Administrators are in separate software development groups. They require IM communication between each other as part of a federation contract. A conflicting situation occurs where more general local policies are deployed that apply to the groups such as a policy that prohibits temporary members of the software development groups such as the Java developer or DBA to IM any group members outside their own group. While a federation policy is enforced to permit the Java Developer and Database Administrator to IM each other. This type of conflict needs to be detected as it can have an indeterminate outcome depending on whether the local policy or the federation policy is enforced first. In XACML this could lead to the policy decision point reaching an indeterminate result and the communication would be prohibited by default.

A. Federated Policy Conflict Analysis Process

Figure 4 shows the proposed federated policy conflict analysis process involving three federated domains. It depicts the sequence of steps required within and between domains in a federation in order to deploy federation-level policies and check for conflicts with previously deployed local/federation policies. The diagram depicts a situation where a domain (AAAConsultancy) wishes to deploy a set of federation policies. A newly specified or modified federation-level policy is added. A selection algorithm selects related previously deployed local/federation policies. The diagram depicts a situation where a domain (AAAConsultancy) wishes to deploy a set of federation policies. A newly specified or modified federation-level policy is added. A selection algorithm selects related previously deployed local/federation policies. The diagram depicts a situation where a domain (AAAConsultancy) wishes to deploy a set of federation policies. A newly specified or modified federation-level policy is added. A selection algorithm selects related previously deployed local/federation policies. Once these policies have been selected, they are checked for conflicts. An example of which would be AAAConsultancy selects policies, checks for conflicts, if no conflicts are detected or are detected,
but resolved AAAConsultancy notifies ISBank/DBDataStore of policies that need to be deployed (through FRM). ISBank/DBDataStore uses the selection algorithm to select policies and then employs the conflict analysis algorithm to check for conflicts, if no conflicts have been detected or have been detected, but resolved ISBank/DBDataStore notifies AAAConsultancy of status, so the federation policies are either committed or rolled back by each domain.

V. IMPLEMENTATION

For our implementation, we make use of the DEN-ng information model which involves the transformation of a subset of the information model into OWL representation and the instantiation of the model using XACML policies. We now provide an overview of the federated XMPP test-bed implementation which is depicted in Figure ?? and is in place within the three domains. It consists of an open-source XMPP server, an Intercepter and a XACML policy server.

Openfire\(^1\) 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 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.

An intercepter 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 tasks of the Intercepter are to forward XMPP packets to the XACML policy server, specifically the Policy Enforcement Point (PEP) component of the XACML policy server, process the response to the policy decision received from the PEP 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.

Sun’s XACML\(^2\), 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 interprets 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 Policy Decision Point (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 Lightweight Directory Access Protocol (LDAP) system.

The DEN-ng policy model can be used to model access control policies. The model has been extended to include access control aspects and more specifically a limited set of XACML concepts. The XACML policies deployed in the system can be viewed as using an event-condition-action sequence even though XACML doesn’t support an event object. The event in this case is the sending / receiving of an XMPP message. The process can be read as on receipt of an XMPP packet, check conditions (if applicable), apply the rule action. The XACML entities that need to be modelled are the subject entity, the target entity and action belonging to the XACML policy. These XACML entities can be mapped to the ECA class of the DEN-ng model. A subject in this case is the initiator of an XMPP communication request, the recipient of the XMPP communication request is the target and the action is the type of XMPP communication being sought. There are three types of XMPP communication available, instant messaging (IM), groupchat and file transfer. The XACML language also has an additional effect component which states whether the policy rule when fired should be permitted or denied.

An example federation-level policy is depicted in Figure 5. The federation-level policy has been specified using the process outlined in [3]. The policy contains context information

---

regarding the federation along with high-level federation rules that need to be deployed within each domain.

A SWRL rule is then defined to select deployed XACML policies stored in the test-bed policy repository. The SWRL rule compares the XACML and federation policy along the subject, resource and action properties to check that there is at least one match among the properties. If a policy is found to match at least one property, then it is returned for further analysis. An example of a typical SWRL rule is depicted in Algorithm 1.

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 conditional elements of these policy individuals. The types of conditional relationships between policy rule conditions 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 system policies. These system-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 the conditional element a second SWRL rule may be fired to retrieve the system-level policy associated with the related XACML policy.

By employing OWL and SWRL for modelling network entities where rule conflicts can occur provides different ways to detect conflicts in the knowledge base. The models can be specified in OWL which is based on Description Logic, a DL reasoner can be applied to detect occurrences of inconsistencies in the knowledge base. When contradictory facts are defined in the knowledge base, the knowledge base is deemed to be inconsistent. Reasoners are capable of detecting these inconsistencies. Hence, the consistency checking processes used by knowledge base reasoners can be used to detect occurrences of semantic conflicts between policies.

The types of 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.

- **Detection based on modelling conflictive individuals**

  New concepts related to a specific application domain that are deemed to be conflictive can be added to the knowledge base. If an individual is added to the knowledge base and is a member of this newly specified conflictive class then this individual can be assumed to give rise to a semantic conflict. A reasoner can be employed to retrieve such conflicting individuals in the knowledge base.

VI. Evaluation

Without the policy conflict analysis engine the proposed federation-level policy would need to be checked with all deployed XACML policies manually. Besides the fact that this is a highly error prone process, it is also extremely inefficient as the number of XACML policies required to be checked for conflict increases exponentially as the number of federated users increase in the system. The number of policies needed to be checked manually has been proven to decrease using the federated conflict analysis process.

The number of policies generated in total, based on each federation policy, for each domain is expected to increase dramatically as new federations are added to existing federations. A simple calculation shows that in a federation with three members, and another member joins, then two policies are added for each existing federation member, and the new member must add two policies for each existing member of the federation. The policies in this case are assumed to be simple access control policies controlling information flow IN and OUT of each federation member. Typically, these policies would be manually defined, and thus would need to be manually analysed. Our approach includes automated
analysis and can reduce the number of policies that require human attention to those abstracted to a high (or federation) level. To evaluate the numbers of policies that are expected to be required in the manual case versus policies automatically generated and analysed we defined a simple model. Take $N$ to equal the number of federations involved and $M$ to be the number of federation policies for a given member, we can define a function $p(n, m)$ that gives the number of policies generated locally for that federation policy. Thus the sum of $p(n, m)$ over all federations gives the total number of federation related policies. The function $p(n, m)$ can be defined as a normal distribution where the mean is twice the number of federations and the standard deviation is 1, assuming some efficiencies may exist. Thus, an analysis of the number of increasing policies as federations increase is given in Figure 6.

The results were produced in MATLAB. In our analysis, we assumed a normal distribution of federation level policies for each federation with mean 2 and standard deviation 1. Essentially, showing that each member of the federation needs only two federation level policies, one governing IN communication and the other governing OUT communication. But that these policies are elaborated per federation member into concrete XACML policies that need to be more specific. The results show that there is an added advantage of not only automatically generating the XACML policies, but in particular to have these policies automatically analysed during the authoring process.

VII. CONCLUSIONS

Federated domains will more than likely consist of heterogeneous information models, policy based management systems and policy languages, operating at arbitrary layers of network management. The policy conflict analysis approach presented in this paper takes such operating constraints into account and is a first step in supporting policy conflict analysis across federated domain environments. However, it does make some limited assumptions in that the information model used by each domain is required to possess the ability to represent context data to aid the policy conflict analysis process. SWRL rules are used for the selection of related previously deployed policies for conflict analysis where more sophisticated reasoning can be preformed using a semantic reasoning engine when the policies are expressed in an ontology language such as OWL. Unfortunately, another problem encountered is the fact that low-level deployed policies lack context data, hence the need to fire a second SWRL rule to retrieve those higher-level policies related to the deployed policies to retrieve the required context data for conflict analysis. Preliminary results show that using the policy selection and conflict analysis processes reduces the number of policies that need to be checked manually for possible occurrences of inconsistencies. Deployed policies returned for conflict analysis are added as instances to the ontology. It remains to be determined the effects in terms of reasoning efficiency when a large number of deployed policies are returned for analysis. Future work will investigate this along with methods for analysing context data with the aim of suggesting possible solutions to detected conflicts.

ACKNOWLEDGEMENT

This research activity is co-funded by Science Foundation Ireland via the FAME strategic research cluster (http://www.fame.ie), grant no. 08/SRC/11403, the Irish Research Council for Science, Engineering and Technology, and by Marie Curie Actions under FP7.

REFERENCES