

A Model Based Approach for Policy Tool Generation and Policy Analysis

Keara Barrett¹, Steven Davy¹, John Strassner², Brendan Jennings¹
Sven van der Meer¹ and William Donnelly¹

¹*Telecommunications Software & Systems Group*

Waterford Institute of Technology

Cork Road, Waterford, Ireland

{kbarrett, sdavy, bjennings, vdmeer, wdonnelly}@tssg.org

²*Motorola Labs*

Schaumburg, IL, USA

john.strassner@motorola.com

Abstract— We outline an approach to policy specification and analysis in which an information model is used as the starting point for semi-automated generation of an integrated suite of languages, tools and an ontology. The suite includes separate domain-specific languages for the specification of systems structure and policies respectively, editors and checkers for these languages, and a baseline ontology that can be augmented with semantic information to support policy analyses processes. We describe a prototypical realisation of the approach, showing how the languages, tools and ontology are used to support policy transformation and conflict detection processes.

Index Terms— Network Management, Policy Based Management, Model Driven Development, Ontologies

I. INTRODUCTION

Policy based management (PBM) systems have been considered for a wide range of application domains, including security management, pervasive systems management and communications network management. The PBM paradigm is generally seen as a means of reducing the level of manual intervention required to effectively manage communications networks, a goal that is becoming increasingly important as networks become increasingly heterogeneous, dynamic and interconnected. PBM systems consist of several key components, including a policy language editor, a policy analyser and a policy deployment infrastructure [1].

We contend that leveraging an information model in the development of these components allows for more efficient and flexible PBM system development. Specifically, we show how Domain Specific Languages (DSLs), semi-automatically generated from an information model, can draw on the data and relationships of the information model to enhance syntax checking and code completion. Moreover, policy analysis, including policy transformation and policy conflict detection,

is augmented by leveraging a baseline system ontology and policy language ontology.

Section 2 describes our approach to model based generation of policy languages and tools. These include both a generated structural DSL and policy DSL, along with their associated editors, and a system ontology. Section 3 discusses a prototype implementation of the approach that makes use of current Model Driven Development (MDD) techniques to aid in the generation of tools based on the system information model. Section 4 discusses and demonstrates the capabilities of the approach and highlights its advantages. Section 5 discusses related work whilst Section 6 draws conclusions and describes areas for future work.

II. MODEL-BASED APPROACH

An information model is a structured representation of the functionality of a system. This representation is typically described using object oriented diagrams that abstract the complexity of the system to a set of interrelated classes. The motivation for building an information model is traditionally for documentation used by the system developers to aid in the development of software tools. However, we contend that there exists extensive information within the model that can benefit not only as documentation but can be automatically leveraged to produce improved tool support and languages to aid in policy based management.

Applying a model-based approach in the development of PBM tools allows for additional resources to be spent on requirements analysis, as the PBM tools can be generated or at least semi-generated from the pertinent subset of the information model; meaning that fewer resource are needed for tool development. The benefit of rapid tool development afforded by the model-based approach is apparent when business/system requirements change. Tools must facilitate in achieving business/system requirement, therefore as business/system requirements change tools must be updated. The rapid generation of tools is again evident when a choice

This work has received support from the Science Foundation of Ireland under the Autonomic Management of Communications Networks and Services programme (grant no. 04/IN3/I404C).

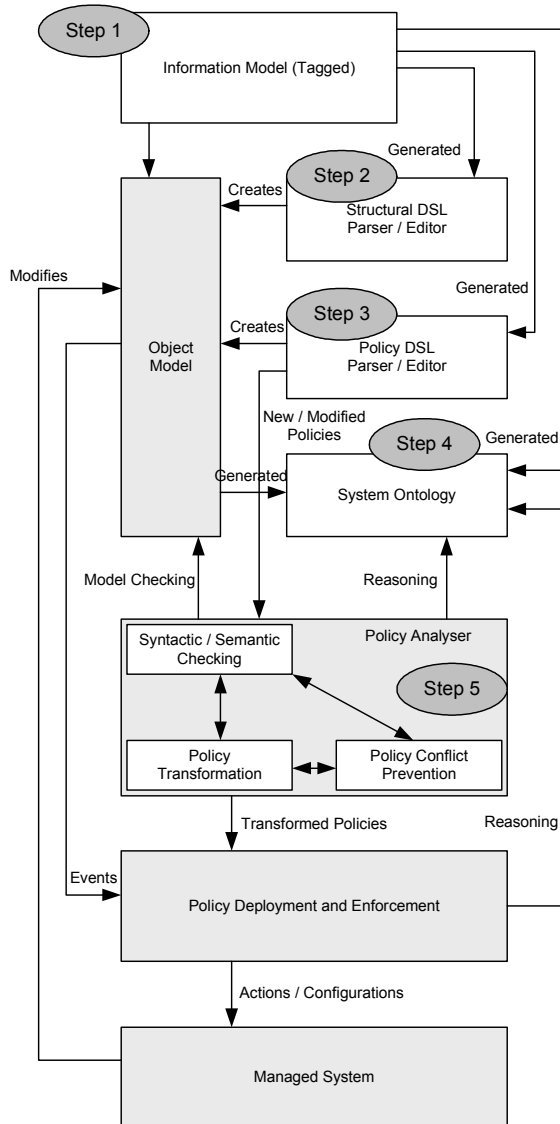


Fig 1. Steps of Model Based Approach for PBM

must be made between a general purpose tool and a domain specific tool. In many cases it is too expensive to develop a tool for each individual task/domain in spite of the advantages. Model based development alleviates some of the associated cost making domain specific tool development viable.

Most models, be it an information model such as DEN-ng, the TMF Shared Information/Data model (SID) [18] or the DMTF Common Information Model (CIM) [17], 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, we tag and extract only the relevant aspect of the model for the generation of a DSLs (domain specific language) and tools.

We intend to develop policy based management components that aid in the specification of the system object model, the policy language, and the policy analysis components. The process of policy enforcement is outside the scope of this paper, as we are mainly focused on demonstrating the advantages our approach during the specification phase of

policy based management. The following steps, depicted in Fig 1 outline an approach to use subsets of an information model for PBM system component development.

Step 1 of the approach requires the developer to identify the subsets of the information model for tool generation. Step 2 generates DSLs that can be used to describe object models of the structure of a managed system. Step 3 generates an associated policy DSL to aid in the definition of system behaviour. Step 4 generates an ontology that can be used to provide reasoning capabilities for management processes. Step 5 describes how policy analysis can take advantage of the information model and associated generated tools.

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

Step 2. Generate Structural DSL and Editor. A method is required to create an object model that represents the managed system (e.g. communications network) using concepts specified in the information model. This object model can then be shared across generated tools and languages, if required, so that a consistent view of the managed system is accessible. 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.

Multiple structural DSLs along with associated parsers and editors can be generated from identified and tagged subsets of an information model. This enables different object models to be defined that correspond to the disparate aspects of the managed system (e.g., security, routing and fault management).

Step 3. Generate Policy DSL and Editor. DSLs from step 2 can be used to represent the structure, but not the behaviour of a managed system. 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. For UML-based information models, marking the pertinent policies entities with a UML tag-value pair and subsequently generating a corresponding policy DSL will provide the user with a means of defining policy controlled behaviour.

For the policy DSL to be usable, an associated text or graphical editor and parser must be generated. If a text editor is preferred, it should ideally provide syntax highlighting and syntax checking. 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.

Step 4. Generate a Baseline Ontology. Automated reasoning is required for policy analysis and also for policy specification, to ensure that the policies are semantically correct. Information models that are based on or described with UML inherit the shortcomings of UML. UML was originally designed for human-to-human communication of models and therefore has no formal semantics, making automated reasoning difficult. On the other hand, ontologies facilitate enhanced representation, exchange, integration and querying of data by annotating data with formal semantics and thus allow for automated reasoning [2].

Defining an ontological model of a complex system from scratch is cumbersome and time consuming. The aim, therefore, is to leverage existing management models in defining ontologies. Generating a baseline system ontology from a subset of an information model provides policy processes such as policy analysis with capabilities to perform reasoning using the ontology. The baseline system ontology holds the semantic information currently available within the UML information model but in a form that can be reasoned over. This baseline model should then be further enhanced to incorporate semantic concepts that could not ordinarily be represented within a UML based information model to allow for more automated reasoning.

Step 5. Using Generated Ontology for Policy Analysis. Policy analysis encompasses a set of processes ranging from policy transformation, and refinement, to policy conflict detection and resolution. Due to the flexibility offered to policy authors in defining the behaviour of the target system, strict processes are required to ensure the proper use and coordination of policy. Two specific functions of the policy analysis that take advantage of automated reasoning using a system ontology are policy transformation and policy conflict detection.

Policy transformation is the mapping between different representations of policy. Many languages for policy specification exist today. When two systems using two different languages need to communicate or merge, mappings between the two languages may be necessary. Since the different representations of policies have different semantics and expressive power, semantic integration and mapping techniques are required in the policy transformation process. Automated reasoning is an integral part of semantic mapping; it is used for example to infer relationships that have not been explicitly identified, and is therefore an essential capability for

policy transformation.

A policy conflict occurs when two or more policies are activated simultaneously enforcing contradictory management operations on the system [1]. Semantic information about management operations provided by the system ontology enables the policy analyser to reason over this information to detect occurrences of conflicting management operations. This step builds on work outlined in [4], where the information model is used to augment the policy conflict detection process. However that work did not consider using semantic information derived from the information model to enhance policy conflict analysis.

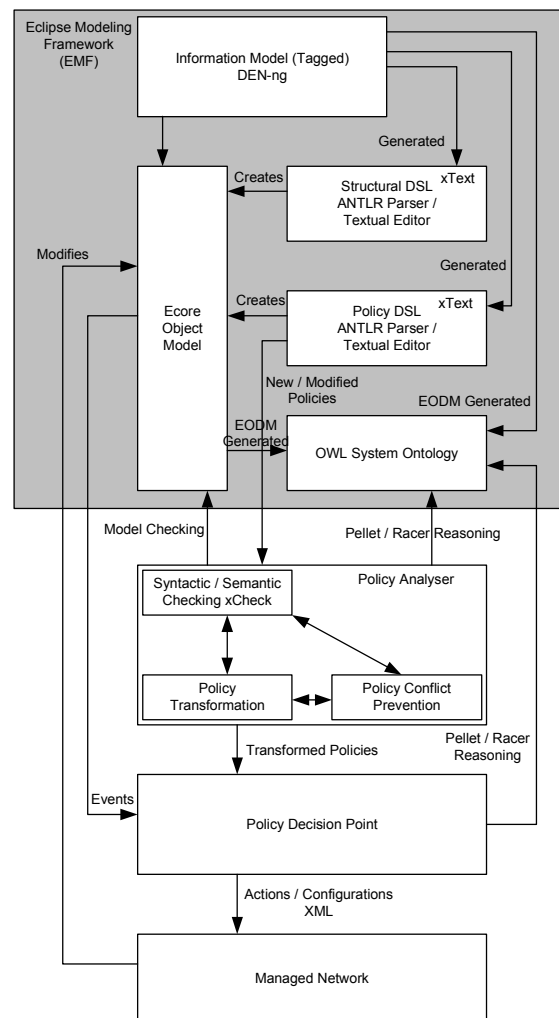


Fig 2 Implementation of PBM System

III. IMPLEMENTATION

In order to validate our approach, we built a prototype implementation of model-based tools and languages targeted at managing resources in a communications network using PBM. This prototype is depicted in Fig 2. We now describe how the model-based approach was followed by identifying an information model subset, generating a number of DSLs for

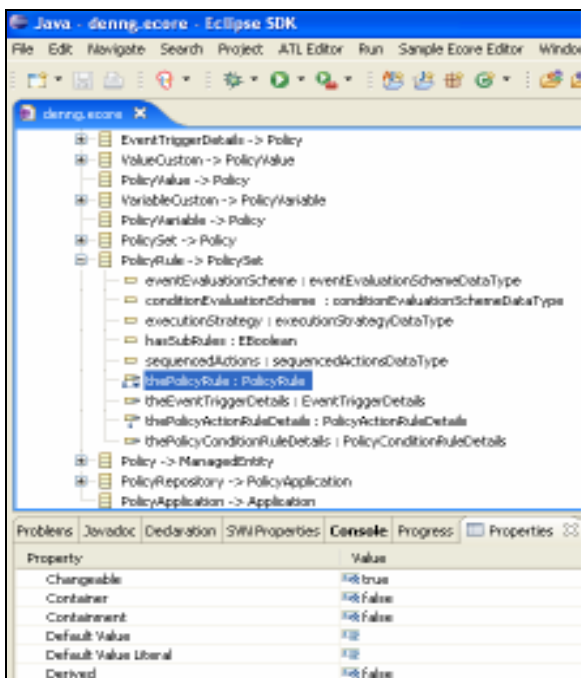


Fig 3 Part of the DEN-ng Ecore Model

representing structure and policy, and developing analysis tools.

The DEN-ng UML-based information model was used to generate DSLs and tools. DEN-ng is a state-of-the-art information model that includes definitions of Resources, Services, Products, and Customers [1]. The Eclipse Modeling Framework (EMF) [14], a key open source initiative supporting the Model Driven Development (MDD) initiative [5], was used to generate DEN-ng based DSLs and tools. To reap the benefits of the MDD plug-ins provided by EMF, an Ecore representation (a meta model format akin to UML) of DEN-ng is needed. Fig 3 shows a sample snippet of the Ecore representation of DEN-ng.

Generating a Structural DSL. The structural DSL, as discussed in step 2 of section 1, is a tool that enables the creation of information model instances (i.e. object models) to represent the structure of the managed system. We use the textual DSL framework called xtext, created by open Architecture Ware (oAW), for generating the structural DSL [12]. The structural DSL is generated from the model elements in the DEN-ng model that were marked as relevant for describing the structure of the managed system. EAnnotations which are Ecore’s equivalent to tag-value pairs are used to identify the relevant model elements for a DSL to be generated. Fig 4 shows a snippet of the Structural DSL grammar.

The xtext framework also generates a corresponding ANTLR parser and a text editor plug-in for any xtext defined DSL; hence both a parser and an editor are available for the structural DSL. The structural DSL text editor provides syntax highlighting, a facility for code completion, an outline view and syntax checking. The code outlined in Fig 5 details the

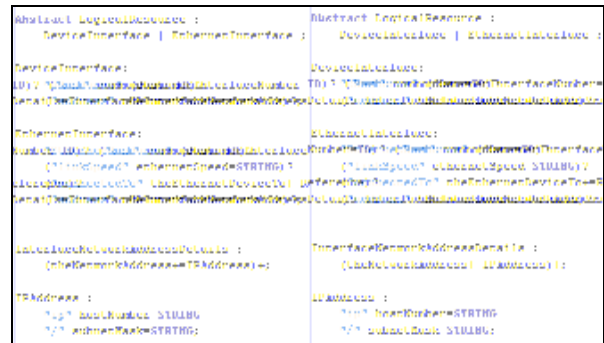


Fig 4 Structural DSL Grammar



Fig 5 Structural DSL Snippet

structural representation of two resources within the managed network specified using the structural DSL.

Generating a Policy DSL. The same technique used to create the structural DSL and its editor can be used to generate an event-condition-action (ECA) policy DSL that is based on the policy representation entities in the DEN-ng Ecore model. The semantics of the ECA policy DSL in terms of execution are as follows: on the occurrence of a set of events, if the condition clause evaluates to true, then execute the action clause. A subset of the xtext grammar for the policy DSL is shown in Fig 6. A policy rule defined with the policy DSL and its editor is shown in Fig 7.

Generating a Baseline System Ontology. The OMG’s Ontology Definition Metamodel (ODM) specification supports the use of legacy models as a foundation for ontology development and so includes mappings to and from UML and the World Wide Web Consortium (W3C) Web Ontology



Fig 6 Policy DSL Description



Fig 7 Policy Rule DSL Snippet

Language (OWL) [13].

IBM offers the EMF Ontology Definition Metamodel (EODM) as a component of their EMF-based Integrated Ontology Development Toolkit (IODT) for ontology-driven development [6]. EODM implements OWL parsing, reasoning and transformation between OWL and other modelling languages (e.g. Ecore). The EODM's org.eclipse.odm.owl.transformer.ecore package was used to produce an OWL representation of the identified subset of the DEN-ng information model. The resulting baseline ontology could then be viewed within Eclipse with the IODT plug-in or saved and opened with the Protégé OWL editing tool.

Once the OWL representation is available, it is enriched with further semantics that were not possible to express with an object-oriented information model like Ecore. The ontology can be reasoned over using freely available reasoners such as Pellet [15], and Racer [16]. Fig 8 depicts in Protégé the DEN-ng policy DSL ontology model generated with EODM.

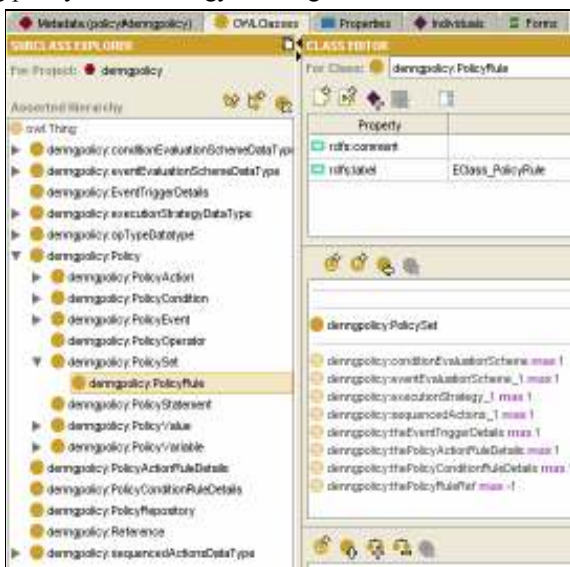


Fig 8 Policy DSL OWL Representation

IV. APPROACH ADVANTAGES AND APPLICATIONS

The primary objective of this work is to demonstrate the

advantages of a model-based approach for the generation of policy architecture components and for policy analysis. The following subsections outline the advantages of the approach from the perspective of policy analysis.

A. DSL Editor.

The advantage of using an information model to generate the DSL and policy editor is that it is capable of being made aware of constraints imposed by the information model. In a UML based information model the Object Constraint Language (OCL) is used to describe constraints. Constraints are used to validate object models specified within a DSL. We used oAW's xtext to generate the DSLs from an Ecore based information model, and oAW's xCheck language is used to define constraints over the DSLs that are mapped from the OCL constraints defined within the information model. Subsequently, object models instantiated using the DSLs conform to constraints imposed by the information model. Fig 9 is a simple constraint expressed in xCheck that prohibits a router interface from connecting to itself.

Context EthernetInterface

ERROR "Interface can not connect to itself":
!this.connectedTo.Equals (this)

Fig 9 xCheck constraint

Similarly, the specification of policy using the generated editor can be constrained to reference managed entities or entity types in the object model, which has been populated using structural DSLs. If a policy is specified for an object that does not exist, an error will be displayed to the author.

B. Policy Conflict.

The system ontology once generated from the information model can be enhanced with information that can aid in policy conflict detection. Management operations defined in the information model can be related to each other in specific ways, such that reasoning over these relationships can determine if there is a potential occurrence of policy conflict. For example, if two management operations associated with an ethernet interface of a router are Permit_IPPacket and Deny_IPPacket, then using an ontology we can assign a relationship between these operations to specify that they are disjoint/mutually exclusive. A policy conflict detection algorithm based on discovering disjoint management operations can be based solely on the relationships among operations, therefore reducing the need to analyse all unrelated policies.

This algorithm can detect all occurrences of policies where their simultaneous enforcement will produce both a Permit_IPPacket and Deny_IPPacket. Our policy conflict analysis approach can therefore detect conflicts when the modality of policies are related using an ontology, and conflicts when the semantics of management operations for a particular application are related using semantic concepts. The objective therefore is to devise algorithms that can take

advantage of ontological reasoning to reduce the complexity associated with policy conflict detection.

C. Policy Transformation

Policies may be expressed in a number of diverse policy languages, e.g. Ponder or Rei, and consequently mappings between the policies may be required for negotiation or deployment. If a policy language is not developed with a model based approach described in this paper, an ontology of the policy language must be created, perhaps manually, to benefit from our proposed policy transformation technique.

With an ontology of the policy language available policy language transformation can be achieved not just at a syntactic level but also at a semantic level. Moreover, with the semantics of the policy language expressed formally in an ontology the meaning of the policy language components are unambiguous. Such information will aid the policy transformation process (manual or automated) as the relationship between the different concepts in different policy languages can be identified. This ontology-based semantic comparison between policy languages can be used to augment any existing policy transformation process that concentrates solely on the syntactical mapping between languages. The expressive power of different policy languages will also become apparent with the availability of an ontology. This information can be used to determine if a policy language has the capacity to represent the required policies, e.g. obligation or authorisation.

For example, with an ontology of both the Rei and KAoS policy languages available one can determine, with a multitude of ontology mapping techniques, that the semantics of KAoS's negative authorization policies are semantically equal to Rei's prohibition policy. This information can be used to determine that a KAoS negative authorization policy, as described in Fig 10, should be represented as a prohibition policy in Rei and not a permission, obligation or dispensation policy. There are a number of different semantic mapping techniques that can be used for the transformation between the policy languages, see [19] [3].

```
<policy:NegAuthorizationPolicy rdf:ID="Test">
  <policy:controls rdf:resource="#TestAction" />
  <policy:hasSiteOfEnforcementrdf:resource "#ActorSite"/>
  <policy:hasPriority>5</policy:hasPriority>
  <policy:hasUpdateTimeStamp>101010107
  </policy:hasUpdateTimeStamp>
</policy:NegAuthorizationPolicy>
```

Fig 10 KAoS negative authorization policy

V. RELATED WORK

A. Policy Language and Tool Development

The DMTF is working on a CIM-compliant policy specification language called the CIM Simplified Policy Language (CIM-SPL) for expressing management policies,

however they did not use MDD techniques to generate the language and associated tools so the benefits of the model-based approach are not realised. CIM is defined in its own proprietary meta-model, and hence is not UML-compliant. Therefore it is difficult to use model based development approaches with CIM or CIM-derived models for the generation of policy languages and tools [7].

Likewise, the Positif Project [8] has defined their Security Policy Language (SPL) by extracting from the DMTF CIM model classes to represent different types of security policies. SPL is used to define desired security behaviour over systems and services described with the System Description Language (SDL). SDL is similar to our structural DSL in its objectives. Again, since the languages are based on CIM model based approaches can not be used for tool generation.

B. Policy Analysis

Kempton and Danciu [9] outline an approach to re-using invariants described in existing information models to aid in policy conflict detection. The approach describes an algorithm that analyses the information model defined constraints in order to build a set of incompatible management operations. Simultaneously triggered policies using these conflicting operations are then flagged for resolution. However this approach is limited in that it is concerned solely with constraints defined within the information model, and cannot reason over constraints described at a higher semantic level that can be achieved using ontologies. Our approach uses ontologies to represent constraints among management operations, thus allowing for more flexibility in detecting conflicts.

Bandara et al. [10] propose a policy conflict analysis approach for domain independent and application specific conflicts. However, their method of policy conflict detection is based on constraints only and the language is not tied directly to an information model / ontology. Instead they translate the policies into a logic program based on event calculus, and examine this to detect conflict.

Kaviani et al [11] propose the use of R2ML to map policy rules from a source policy language to a target policy language. R2ML is an intermediate rule language for exchanging information; which aims to comply with the requirement of the W3C's Rule Interchange Format (RIF). They demonstrate by transforming policies specified with PeerTrust and Rei. However, they do not consider the semantics of the policy languages and R2ML when performing the transformation. No ontology of the policy languages are proposed to ensure that the transformations between the diverse languages are semantically correct.

VI. CONCLUSION AND FUTURE WORK

This paper outlines our model based approach to policy tool and language generation. This approach makes development of domain specific policy tools viable, reduces the time to deployment of these tools and allows rapid updates to the tools to reflect changing business and/or system requirements. We

detail the tools used in the implementation of prototype policy tools, which includes domain specific languages, to illustrate a practical execution of the approach.

The second novel proposal described in the paper is the use of models to enhance policy analysis processes. The enhancements acquired by employing a semantically rich formal ontology model, such as one specified with OWL-DL, for detecting policy conflicts are outlined. In addition the benefits of augmenting policy transformation processes with ontological representations of the policy languages involved are explained.

Our future work entails extending the prototype implementation described in this paper in order to demonstrate the advantages of applying ontology reasoning for policy conflict detection and policy transformation. While policy transformation can be done on a policy language to policy language basis, an upper ontology can be used to reduce the number of transformations required as the number of languages increase. We intend to define an upper ontology to act as the interchange format for mapping between policy language ontologies. With this upper ontology for policy languages, existing ontology mapping techniques can be used to aid in the transformation of policy. Existing policy languages (e.g. Rei, Kaos, and Ponder) and the Rule Interchange Format (RIF) will be considered when developing the upper ontology. Future work concerning policy conflict detection is to leverage the system ontology for detecting conflicts among different views of a policy language continuum. The policy continuum introduces intricate complexities into policy conflict detection process due to the nature of the associations among policies across different views. Thorough investigation into these complexities is left for future work. We also intend to enhance processes for policy conflict detection so that they can harness a wider variety of information from the information model such as finite state machines.

REFERENCES

- [1] J. Strassner, "Policy-Based Network Management", *Morgan Kaufmann Publishers*, (1st ed.), ISBN 1-55860-859-1, 2004.
- [2] I. Horrocks, B. Parsia, P. Patel-Schneider and J. Hendler, "Semantic web architecture: Stack or two towers?," in Proc. *Principles and Practice of Semantic Web Reasoning (PPSWR 2005)*, pp. 37-41, Sept 2005.
- [3] A. Wong, P. Ray, N. Parameswaran, J. Strassner, "Ontology mapping for the interoperability problem in network management," *Journal on Selected Areas in Communications*, vol. 23, issue 10, pp. 2058- 2068, Oct. 2005.
- [4] S. Davy, B. Jennings, J. Strassner, "Policy Conflict Prevention via Model-driven Policy Refinement," in Proc 17th *IFIP/IEEE Distributed Systems: Operations and Management (DSOM)*, pp. 209-220, Oct. 2006.
- [5] T. Büchner, F. Matthes, "Introspective Model-Driven Development," in Proc. *3rd European Workshop on Software Architectures*, pp.33-49, Nantes, France, 4-5 Sept. 2006.
- [6] Y. Pan, G. Xie, L. Ma, Y. Yang, Z. Qiu, and J. Lee, "An MDA-Based System for Ontology Engineering," IBM Research Report RC23795, China Research Lab and Thomas J. Watson Research Center, 2005.
- [7] DMTF CIM Simplified Policy Language (*CIM-SPL*), Document Number DSP0231, version 1.0.0a, DMTF, 2007
- [8] A. Lioy, "Policy Based Security Tools and Framework", FP6 POSITIF project, IST-2002-002314, Available: <http://www.positif.org/ispl.html>
- [9] B. Kempter, A.V. Danciu, "Generic Policy Conflict Handling Using apriori Models," in Proc. *16th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM 2005)*, pp.84-96, Barcelona, Spain, October 24-26, 2005.
- [10] A. K. Bandara, E. C. Lupu and A. Russo, "Using Event Calculus to formalize policy specification and analysis", In *4th IEEE Workshop on Policies for Distributed Systems and Networks*, 2003
- [11] N. Kaviani, D. Gasevic, M. Hatala, D. Clement, and D. Wagner, "Towards Unifying Rules and Policies for Semantic Web Services," in Proc. *3rd Annual Lornet Conference*, Canada, November 7-10, 2006 (CD Proceedings)
- [12] Open Architecture Ware (Oaw), <http://www.openarchitectureware.org/>
- [13] The OMG Ontology Definition Metamodel, submitted by IBM and Sandpiper Software, Inc., 5th June 2006
- [14] EMF Development team, The Eclipse Modeling Framework website, <http://www.eclipse.org/emf>
- [15] E. Sirin, B. Parsia, B. Cuenca Grau, A. Kalyanpur, Y. Katz, "Pellet: A Practical OWL-DL Reasoner", *Journal of Web Semantics*, 2006.
- [16] RACER/RacerPro: www.sts.tu-harburg.de/~r.f.moeller/racer/
- [17] Distributed Management Task Force, Common Information Model (CIM), version 2.15, <http://www.dmtf.org/standards/cim>, April 17th 2007
- [18] TeleManagement Forum, NGOSS Release 4.5 Shared Information/Data model (SID) Model Suite (Phase 5), www.tmfforum.org, Available 24th May 2005.
- [19] Y. Kalfoglou and M. Schorlemmer, "Ontology mapping: the state of the art", *The Knowledge Engineering Review*, Volume 18, Issue 1, pages 1-31, 2003