View metadata, citation and similar papers at core.ac.uk brought to you by CORE
provided by Repositório Institucional da Universidade de Aveiro

A graphical user interface for policy composition in CIM-SPL

Pedro Gonçalves[1], Carlos Figueira[1], Ricardo Azevedo[2], Rui Aguiar[1], José Luís Oliveira[1]
[1] Universidade de Aveiro, DETI/IT,
[2] Portugal Telecom Inovação, Applied Research and Knowledge Dissemination,
3810 Aveiro, Portugal
{pasg, cjcfigueira, ruilaa, jlo}@ua.pt, ricardo-a-pereira@ptinovacao.pt

Abstract

CIM-SPL is a declarative policy specification language proposed inside DMTF. SPL policies allow the specification of rules to govern the behavior of a system using a PBM approach. However, SPL requires thorough knowledge of the language syntax as well as full understanding of the management scenario and its available management features. This paper describes a graphical CIM-SPL editor application and the supporting policy edition metaphor. A graphical composition process of SPL policies is proposed, based on the use of drag and drop operations of the policy component items in a graphical interface. The editor includes policy creation wizards that guide the user in the policy specification process, in order to alleviate network administrators from the difficulties associated with the intricacies of SPL language. Additionally, a text-based SPL edition tool is provided as a complement for experienced SPL language operators.

1 Introduction

POLICY Based Management (PBM) [1] is a management paradigm that consists of the definition of rules that determine system behavior. Policies have long been known, with the first literature references occurring in the 60s. PBM separates the rules that define the behavior of a system from its functionality: rules determine the changes in system behavior according to the system internal state. Policies were proposed for many different areas, such as admission control, security and network management control. During the last decades, we have seen proposals for new PBM management technologies [2, 3], new data models for PBM information representation [4] and new policy languages [5-8] for its specification. More recently, PBM has been identified as the most appropriate approach to complex network management challenges, such as those appearing in Next Generation Network (NGN). NGN architectures include a framework developed by the 3GPP consortium [9] named IP Multimedia Subsystem [10] to deliver multimedia services in IP networks. PBNM methodology formed the base of the design of the IMS resource managing platform named Policy and Charging Control [9].

One of the most recently proposed policy languages is CIM (Common Information Model) Simple Policy Language (CIM-SPL) [11]. CIM-SPL is a policy specification language defined by Distributed Management Task Force (DMTF) that complies with the CIM Policy Data model and allows policy specification in a if(condition) then (action) model. Although appropriate for the majority of scenarios, CIM-SPL requires high technical expertise from the manager, since he has to know the correct syntax and semantics in order to be able to write the correct management rules.

The current paper proposes a graphical composition process of SPL policies, based on the use of drag and drop operations of the policy component items in a graphical interface. The editor is composed of several wizards and a graphical tool for the policy composition that supports human ruling of policy specification, alleviating the strain of the CIM-SPL learning curve. This graphical composition process liberates the manager from the semantic and syntactical details of policy specification language and consequently decreases the mistakes introduced by textual policy specification.

Section 2 discusses the state-of-art in policy languages and available policy editors. Section 3 presents our management architecture and describes the management technologies and the PBM data models used in current work. Section 4 presents our policy specification metaphor and describes the policy editor. Thereafter we present some examples of policies that can be defined with the help of the proposed editor and we analyze the developed application. Finally a conclusion and intended future work closes the paper.

2 Policy languages and editors

During the last decade policy-based management has raised interest as a powerful paradigm to tackle the challenges coming from the growing complexity of communication networks and services. In [12] Boutaba et al trace the history of policy-based management initiatives in very different areas (e.g. admission control, security, network management). This section describes some generic proposals and in particular describes the available policy editor applications.

2.1 *Policy Languages*

Ponder is a policy management framework developed at Imperial College London, which includes a policy specification language [5], a general architecture and policy deployment model. The framework also includes a policy editor and a domain browser tool. Ponder includes security policies (role-based access control) and management policies (management obligations), and supports object-oriented features like policy inheritance. There are four primitive policy types: authorizations, obligations, refrains and delegation.

Groups, roles, relationships and management structures also support the creation of composite policies. In contrast to what happens with CIM, Ponder supports the definition of events that trigger policy evaluation. Each policy rule contains the target and subjects to which the policy is to be applied, and it supports the definition of domains – a dynamic grouping scheme of managed objects.

XACML is a XML specification for expressing policies defined by OASIS [8]. The language permits access control rules to be defined for securely browsing XML documents and uses a XML schema to validate the defined policy components. XACML is used to specify a subject-target-action-condition oriented policy. The subject comprises identities, groups, and roles and the granularity of target objects is as fine as single elements within the document. It includes conditional authorization policies, as well as policies with external post-conditions to specify actions that must be executed prior to permitting access. The main advantage of XACML has to do with XML encoding, a widely adopted standard that eases policy specification and client development while still allowing human readability. The work on XACML includes an architecture for enforcing policies which extends the IETF policy architecture.

CIM-SPL is a policy specification language proposed by Agrawal et al. [6] inside DMTF. A CIM-SPL rule follows the schema represented in Figure 1. The import statement includes an object named as the anchor class, typically with the purpose of providing the policy action methods.

```
import <MOF Name>::<CIM Class Name>:<Object>
strategy <execution strategy>
policy{
    declaration { <constants and macros> }
    condition {<boolean expression>}
    decision {<action workflow>}
}:<priority>;
```

Figure 1 - SPL rule structure

The declaration section is optional and defines macros and constants for the sake of simplifying the remaining rule edition due to the typically long name of the CIM classes. The condition section contains a boolean expression allowing operators as the standard AND, OR and NOT, as well as arithmetic operators. SPL allows conditions to be composed of a set of sub-conditions, as long as the overall conditions can be evaluated as a boolean. Actions in the decision section are mostly the invocation of anchor class methods. CIM-SPL allows the composition of basic actions into more complex actions – basic actions can be executed in a serial execution or in a concurrent execution manner. The decision section allows execution of a policy decision action in a process as a cascade set of policies.

SPL implements CIM policy aggregation as Policy groups, with a syntax illustrated in Figure 2. Policy groups can contain policies and other groups in a nested group style and they import private anchor class. Each group has a priority implemented as an integer number that should be unique in the group scope. Groups of policies allow hierarchical organization of policies, grouping policies according to their functionality, importance, priority or just their scope.

Expressions in SPL can include all CIM data types, arrays, a set of built-in operators (logical and basic mathematical operators) as well as a set of built-in functions (e.g. max, ln, stringlength). Additionally SPL supports operations over arrays, referred to as collection operations. Collections, implemented through collect operator, create arrays of CIM data types or arrays of references to CIM instances. They allow, among other things, transverse of references of the CIM classes, from the CIM class imported as the policy anchor to reach the remaining classes, by means of CIM association classes that should be accessed in order to specify the policy rule.

```
import <MOF Name>::<CIM Class Name>:<Condition>
strategy <Execution Strategy>
declaration{...}
policy{...}:<Priority>
policy{...}:<Priority>
policyGroup: <Association Class>(<Property One>,
<Property Two>)
{...}:<Priority>
policyGroup: <Association Class>(<Property One>,
<Property Two>)
{...}:<Priority>
```

Figure 2 - Policy group syntax

2.2 *Policy editors*

The edition of policy rules has been mostly made through text files with direct use of the language semantics. Graphical edition tools are minimally explored and usually as simple property edition tools of the policy component items.

The Ponder toolkit [13], for instance, includes a console tool and a domain browser. The console tool is a graphical application that allows policy edition and parsing. The domain browser is a graphical application that provides a friendlier interface for the domain server.

Apache Imperius is an open-source implementation of a SPL-based management platform that uses Java binding and includes a SPL policy editor [14]. Imperius SPL Policy editor was developed as an Eclipse plug-in and includes the features usually available in an integrated development environment (IDE). The editor simplifies policy edition with the use of policy templates, it performs statement auto-completion, error location indication, allows easy import and export of policies, syntax highlighting, policy compiler, policy execution and other features normally included in an IDE application. However, to have these features we must rely on a programming framework (Eclipse), which is not a suitable user interface for management. Furthermore our management architecture required policies to be executed in the management server, and not in the client interface, as happens in the eclipse platform. As such, we have developed a CIM-SPL editor that aims to create a better user interface to simplify the task of policy specification.

3 Management platform

Policies are rules that define the behavior of a system. Despite several policy notations, each rule usually includes at least two components: a condition and an action component. Implementations exist that include an event component in the policy rule in an event-condition-action policy format [15]. PBM platforms usually include a central policy server that receives policies through a user interface application. This interface is, in general, the single contact point between the human operator and the management platform.

This section describes our PBM management platform, as well as enabling technologies such as the communication protocol used in the entity communication, and the data model for management information.

3.1 Overview of the management solution

Our management solution consists of a policy server, a policy manager, several instrumentation providers and a policy repository, following traditional PBM systems.

The policy server is the central element of the management platform and was developed based on a WBEM open-source CIMOM [16]. WBEM [2] is a DMTF standard for system and network management. It relies on CIM to represent its information model and makes use of widely available web technologies for information encoding (XML) and information transport (HTTP). Several WBEM prototype and commercial implementations exist (e.g. [17]). WBEM solutions usually include four components: i) A CIM Object Manager (CIMOM), as the central element of the management scenario, ii) the human-computer interface, iii) instrumentation providers that connect the CIMOM to the management entities, and iv) a policy repository.

The providers adapt the policy rule format according to the managed elements they represent. The policy manager provides the user interface of the management platform, and implements several features associated with the configuration and visualization of services, topologies, events and policies. Since it is based on XML technologies, WBEM can be a rather resource-consuming solution in some management scenarios, when compared to binary protocols, like COPS or Diameter [18].

3.2 Data model for policy representation

The Common Information Model (CIM) is a standard developed inside DMTF for the representation of management information. CIM is an object-oriented model that tries to address all the IT components and includes expressions for common elements that must be presented to management applications (for example, object classes, properties, methods and associations). The elements of the model are Schemas, Classes, Properties and Methods. The model also supports Indications and Associations as Class types and References as Property types. Several management technologies reused the CIM information model [19-21], or were roughly based on the CIM information model [7].

The Policy Core Information Model (PCIM) [7] was a joint effort developed by IETF policy working group and DMTF for policy information representation. It is an object oriented policy information model that extends CIM to represent policy-related information. Another joint effort from the DMTF and the IETF Policy Framework Work Group is the CIM Policy model. In the CIM Policy data model, policies are represented as instances of the CIM Policy model classes in a *if(condition) then (action)* form. Policy components can be represented as instances of *CIM_PolicyCondition* and *CIM_PolicyAction*. Policies, conditions and actions can be aggregated and both conditions and actions can be associated with a policy through the use of association classes.

In current work, several extensions have been created over the CIM Policy model for the representation of policy information. Policies are represented as *CIM_PolicyRule* instances and policy components as instances of the classes derived from *CIM_PolicyCondition* and *CIM_PolicyAction*. Definition of the managed entities affected by each policy is specified in CIM by means of a *PolicyAppliesToElement* association class, and time / period policy component is specified as an instance of *PolicyTimePeriodCondition*.

4 The CIM-SPL editor

Although usable for the vast majority of management scenarios, CIM-SPL requires the operator to know the correct syntax and policy semantics in order to be able to write the management rules. Graphical policy editors are the best approach to reduce the policy language learning curve and configuration errors, since they avoid syntactical errors and help the policy specification person with the platform management details.

Our policy definition metaphor was based on [22] where Lopes et al. propose a policy composition process based on graphical operations over policy component items. We present a policy graphical editor composed of policy specification wizards, a graphical policy composition tool, a policy browser and a text-based SPL specification component. Graphical composition methods consist of the drag and drop operations of policy components present in the surrounding areas of the editor to the control present in the central area of the policy specification window where the policy component is represented.

The editor allows change in the edition method when the operator wishes: experienced users would prefer to use the SPL text edition methods; non- experienced users would prefer to finish the policy specification in the graphical composition window.

The graphical components of the policy editor enable a dual policy specification process:

i) The operator starts by creating a new blank policy, and then has to complete the suitable policy component, or

ii) He can just use a graphical wizard to guide him filling in the policy components.

The next sub-sections describe both the graphical edition features and the SPL edition features of the editor.

4.1 *Wizards for policy specification*

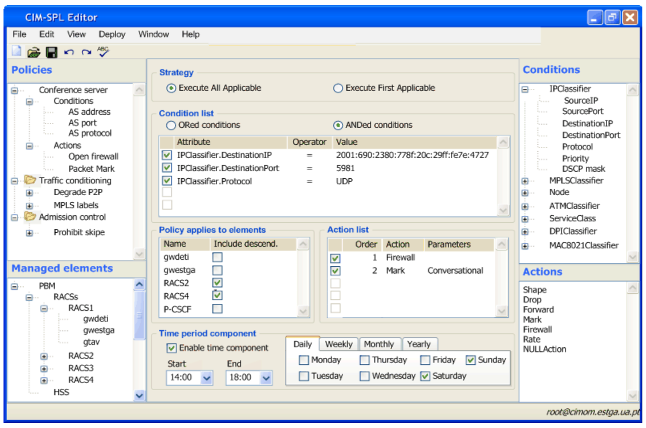
The new policy wizard process is made up of a set of dialog windows that ask for the policy component information, typically organized according to SPL policy components: the first step requests the general policy information (e.g. name, its ancestor name, implementation strategy and priority); the second step requests the condition component information, the third step requests the action component information, and the fourth step requests the managed elements the policy applies to. Once the policy definition wizard is finished, the policy components data is inserted in the CIMOM to be available for additional policy edition.

The new policy group is simpler than the new policy wizard: it is composed of a single step where the user is requested to define the group name, its ancestor group name, implementation strategy and priority. Since CIM-SPL does not allow empty policy groups, after successful group creation its information is temporarily stored in the memory until some descendent is created, and then it is created in the CIMOM repository.

4.2 *Graphical policy composition*

The graphical composition window, illustrated in Figure 3, is composed of several graphical components present in the central area of the window that represent each particular aspect of the policy rule.

The policy components include: the policy execution strategy represented by a radio button; the condition component containing a radio button where it can be defined how to compose the policy conditions (e.g. ORed or ANDed) and a list of simple conditions where condition usage can be suspended in the check box; the action component representing the actions executed by the policy in the form of a list control; the time period component specified in a set of common control windows representing the time validity of the policy assuming the policy's validity is periodical; and the elements from the management scenario affected by the policy rule.



Policies
- Conference server
  - Conditions
    - AS address
    - AS port
    - AS protocol
  - Actions
    - Open firewall
    - Packet Mark
- Traffic conditioning
  - Degrade P2P
  - MPLS labels
- Admission control
  - Prohibit skipe

Managed elements
- PBM
  - RACSs
    - RACS1
      - gwdeti
      - gwestga
      - gtav
    - RACS2
    - RACS3
    - RACS4
  - HSS

Strategy: [✓ Execute All Applicable] Execute First Applicable

Condition list: ORed conditions [✓ ANDed conditions]

	Attribute	Operator	Value
✓	IPClassifier.DestinationIP	=	2001:690:2380:778f:20c:29ff:fe7e:4727
✓	IPClassifier.DestinationPort	=	5981
✓	IPClassifier.Protocol	=	UDP

Policy applies to elements

Name	Include descend.
gwdeti	☐
gwestga	☐
RACS2	✓
RACS4	✓
P-CSCF	☐

Action list

	Order	Action	Parameters
✓	1	Firewall	
✓	2	Mark	Conversational

Time period component
✓ Enable time component
Start 14:00 End 18:00
Daily Weekly Monthly Yearly
☐ Monday ☐ Thursday ☐ Friday ✓ Sunday
☐ Tuesday ☐ Wednesday ✓ Saturday

Conditions
- IPClassifier
  - SourceIP
  - SourcePort
  - DestinationIP
  - DestinationPort
  - Protocol
  - Priority
  - DSCP mask
- MPLSClassifier
- Node
- ATMClassifier
- ServiceClass
- DPIClassifier
- MAC8021Classifier

Actions
Shape
Drop
Forward
Mark
Firewall
Rate
NULLAction

root@cimom.estga.ua.pt

Figure 3 - Graphical editor view

The surrounding areas present the items that can be used to compose the policy rules by means of dragging policy items to the policy representation area. Tree view controls represent elements hierarchically: policy groups, existing policies, CIMOM repository and their components (e.g. policy conditions and policy actions); the managed elements instantiated in the CIM server; and the policy condition components inside their policy condition. A list of policy actions the management platform is able to execute is also presented in the surrounding area.

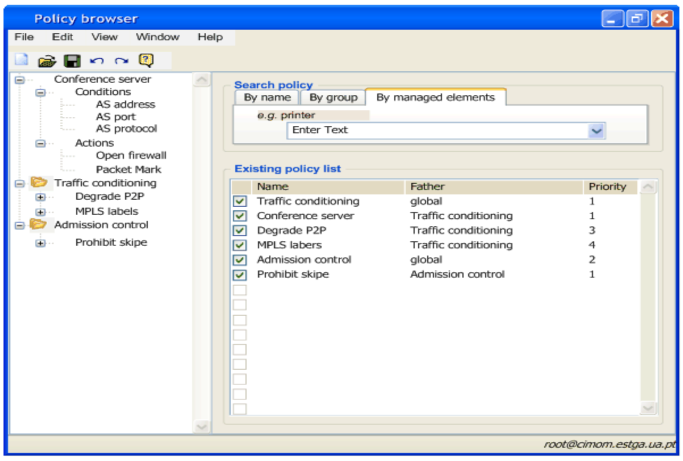
The specification of each aspect of the policies consists of dragging each element to the corresponding component of the policy present in the central area of the window. Interaction with the editor application follows the graphical applications metaphor with respect to the keyboard shortcuts and control behavior. Item enablement is allowed all over the application by use of check box controls: disabled elements (e.g. condition or action items, the managed elements the policy applies to, or the time period component elements) are inserted in the CIMOM repository, but disregarded during CIMOM operation.

4.3 *The policy browser*

The policy browser, illustrated in Figure 4, hierarchically lists all the policies, policy groups and policy components that exist in the management server, in the tree control present in the left hand window. In the right side window the browser lists all the policies that descend from the selected node in the tree control allowing the user to check and uncheck policies or groups, activating and deactivating the policy rule.

If a simple policy node is selected in the tree control, a panel listing all the components of the selected policy is shown (Figure 3). The policy browser, as well as the policy composition tool, allows a policy or a group to be dragged from one group to another, changing its ancestor property.

Since the policy list could include a large number of policies, the policy browser includes policy search facilities. Policy search process is allowed by one of three criteria: by the policy name, by the group the policy descends from and by the elements the policy manages. Search results are presented in the policy list control.



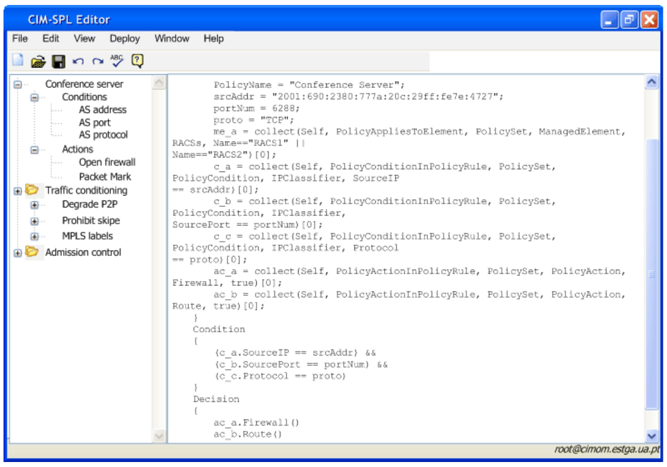
	Name	Father	Priority
✓	Traffic conditioning	global	1
✓	Conference server	Traffic conditioning	1
✓	Degrade P2P	Traffic conditioning	3
✓	MPLS labers	Traffic conditioning	4
✓	Admission control	global	2
✓	Prohibit skipe	Admission control	1

Figure 4 - Policy browser

4.4 SPL policy editor

Although very useful for non-expert users, the graphical edition methodology tends to require some time to fill in all the policy components. Experienced users would prefer to perform policy specification through SPL sentence writing methods, especially because it makes reuse of policy sentences easier. The editor application includes a SPL specification window, as illustrated in Figure 5. It is a text window that allows policy specification using SPL language. The editor includes copy-paste features, as well as allowing policy serialization to, and from, a text file. Policies can be exported and imported to SPL files. These features can be used in several scenarios: policies can be exported for backup or just for transport between different policy servers. They can also be exported to a text format in order to make use of script functionalities, to help distribution of policy rules in more complex management scenarios.

SPL editor window includes compile and run options available in the deploy menu. Compile action performs a policy syntax check with a policy compiler based on the Apache Imperius parser [14]. Correct policies can be stored in the CIMOM repository by means of the run option; otherwise syntactical errors are reported to the user. On the contrary to what happens with Imperius implementation, policies are not executed by the policy editor, but by the CIMOM. In our implementation, once the policy is transferred, the CIMOM starts to translate it to the managed elements through the appropriate providers. That difference in the implementation architecture made us restrict the policy components to CIM objects, avoiding the java-based properties and methods available in the Imperius policy platform.



```
        PolicyName = "Conference Server";
        srcAddr = "2001:690:2380:777a:20c:29ff:fe7e:4727";
        portNum = 6288;
        proto = "TCP";
        me_a = collect(Self, PolicyAppliesToElement, PolicySet, ManagedElement,
RACSs, Name=="RACS1" ||
Name=="RACS2")[0];
        c_a = collect(Self, PolicyConditionInPolicyRule, PolicySet,
PolicyCondition, IPClassifier, SourceIP
== srcAddr)[0];
        c_b = collect(Self, PolicyConditionInPolicyRule, PolicySet,
PolicyCondition, IPClassifier,
SourcePort == portNum)[0];
        c_c = collect(Self, PolicyConditionInPolicyRule, PolicySet,
PolicyCondition, IPClassifier, Protocol
== proto)[0];
        ac_a = collect(Self, PolicyActionInPolicyRule, PolicySet, PolicyAction,
Firewall, true)[0];
        ac_b = collect(Self, PolicyActionInPolicyRule, PolicySet, PolicyAction,
Route, true)[0];
    }
    Condition
    {
        (c_a.SourceIP == srcAddr) &&
        (c_b.SourcePort == portNum) &&
        (c_c.Protocol == proto)
    }
    Decision
    {
        ac_a.Firewall()
        ac_b.Route()
```

Figure 5 - CIM-SPL editor

Besides the graphical appearance of the SPL editor, Figure 5 also illustrates the SPL sentence obtained from the conference server policy. Import statement imports our anchor class responsible for CIMOM communication. The strategy statement defines that all policy actions should be executed. Our policy consists of three different sections: the declaration, condition and decision parts. The declaration section was used to create an alias for the complete IPv6 address. In the policy condition section the destination address is defined for the network packets that should be affected by the policy. The decision section defines the policy name and defines to which packets the "Drop" action should be applied.

5 Implemented policies and results evaluation

The set of policies that can be specified by this management platform is very broad. The examples present in the current paper are limited to QoS management issues, but the management solution could easily be extended to other areas. Policy components are supplied by the CIMOM, so addition of new policies to the management platform consists of the definition of a new CIM extension and development of the corresponding support in the CIMOM provider, without any change in the editor. Table 1 enumerates a few policies that were developed.

The first policy example consists of the definition of a traffic differentiation rule for a user. The user is identified by its IP address and its traffic packets are reshaped and forwarded in the network routers. In the second example a rule for traffic blocking is defined. The blocked addresses could represent a specific machine or an entire network. The third rule blocks access to a specific service. The service is identified through the transport protocols and port numbers. The last example consists of a rule for creating traffic degradation of a service after the user credit is exceeded.

After a successful compilation process, policies are translated to CIM-SPL and transferred into the CIMOM.

Table 1 - Policy examples summary.

Policy Name	Policy Description		
	Purpose	Conditions	Actions
Gold service	Define better service to a user	SrcAddress	Shape Forward
Block destinat.	Block access to destination	DestAddress	Drop
Block service	Prohibit access to a service	Port Numbers Protocol	Drop
Credit control	Degrade service after credit is exceeded	Credit value	Mark Route

The editor application makes extensive usage of the graphical controls in the policy creation wizards, in the graphical composition window and in the policy browser. Those components create a policy composition process that is much more user-friendly: the list controls represent items in a style that eases the reasoning of the policy component items; the combo box controls limit the values inserted by the user in the options offered by the CIMOM repository, thus creating semantic constraints for the policy items; the drag and drop support makes use of some well-known methods present in the graphical applications metaphor. Besides, since the graphical application components fill the SPL policy elements, syntactical errors are avoided and that is the reason why a policy compiler for the graphically specified policies is not needed.

Although the application presented in the current study allows a dual policy specification methodology, it does not implement direct translation of SPL sentences to graphical representation. Graphically specified policies are inserted in the CIMOM, or translated to SPL policy sentences. SPL policy sentences are syntactically verified and compiled (a process that ends with a CIMOM insertion), but they are not directly transposed to the graphical editor. Instead, they are imported in the graphical interface making use of a trick: the graphical interface window is refreshed after any change in the SPL policy interface, and thus it receives policy information from the CIMOM.

CIM-SPL does not fully comply with the CIM Policy data model: CIM_Policy_Rule inherits a property named "Enabled" from CIM_PolicySet that allows the existence of disabled policy rules in the CIM server, but CIM-SPL does not provide a means to specify disabled policies. Our choice was to not translate disabled policies to SPL language, but to allow the disabling of those policies in the policy browser.

Another known issue detected in the application and not yet resolved has to do with the cascade policy groups. Neither the CIM Policy data model nor SPL language allows the existence of empty policy groups; groups need to have subgroups or policies. Graphically created groups with no content other than disabled policies, if translated to SPL interface, cannot be created in the CIM server.

6 Conclusion

In this paper we have proposed a new policy edition metaphor that uses graphical tools to alleviate the CIM-SPL language learning curve. The editor also includes SPL text editor features for expert users and allows policy file serialization.

The graphical application includes policy definition assistants that create the policies, and policy groups, a policy browser and a policy composition tool. We propose a graphical composition method that makes use of graphical processes to compose policy elements. To illustrate the editor functionalities, several management rules were presented for QoS management.

The advantages of adopting graphical tools for management are dual: it releases the user from handling directly the language syntax and it reduces considerably the amount of errors due to mistyped rules.

In future work we plan to include network topology configuration support as well as service configuration features in our editor, in order to allow integrated configuration of the QoS issues of a real network operator.

References

[1]. D.C. Robinson, and M.S. Sloman, "Domains: a new approach to distributed system management," *Distributed Computing Systems in the 1990s, 1988. Proceedings., Workshop on the Future Trends of*, pp. 154-163.

[2]. J.P. Thompson, "Web-based enterprise management architecture," *Communications Magazine, IEEE*, vol. 36, no. 3, 1998, pp. 80-86.

[3]. G.C.M. Moura, G. Silvestrin, R.N. Sanchez, L.P. Gaspary, and L.Z. Granville, "On the Performance of Web Services Management Standards - An Evaluation of MUWS and WS-Management for Network Management," *10th IFIP/IEEE International Symposium on Integrated Network Management (IM 2007)* pp. 459-468.

[4]. DMTF, Common Information Model (CIM) Specification - Version 2.9, 2005.

[5]. D. Nicodemos, D. Naranker, L. Emil, and S. Morris, "The Ponder Policy Specification Language," *Proceedings of the International Workshop on Policies for Distributed Systems and Networks*, Springer-Verlag, pp. 18 - 38.

[6]. D. Agrawal, S. Calo, L. Kang-Won, and J. Lobo, "Issues in Designing a Policy Language for Distributed Management of IT Infrastructures," *Integrated Network Management, 2007. IM '07. 10th IFIP/IEEE International Symposium on*, pp. 30-39.

[7]. B. Moore, RFC 3460 - Policy Core Information Model (PCIM) Extensions.

[8]. OASIS, eXtensible Access Control Markup Language 3 (XACML) Version 2.0, 2005.

[9]. J.J.P. Balbas, S. Rommer, and J. Stenfelt, "Policy and charging control in the evolved packet system," *Communications Magazine, IEEE*, vol. 47, no. 2, 2009, pp. 68-74.

[10]. 3GPP, IP Multimedia Subsystem (IMS) Stage 2, 2008.

[11]. DMTF, CIM Simplified Policy Language (CIM-SPL), 2007.

[12]. R. Boutaba, and I. Aib, "Policy Based Management: A Histotical Perspective," *Journal of Network and Systems Management*, vol. 15, no. 4, 2007, pp. 447-480.

[13]. A. Pilz, ""Policy-Maker": a toolkit for policy-based security management," *Network Operations and Management Symposium, 2004. NOMS 2004. IEEE/IFIP*, pp. 263-276 Vol.261.

[14]. A. incubator, “Apache Imperius”, http://incubator.apache.org/imperius/, 2008-02-12.
[15]. S. Chetan Shiva, R. Anand, and R. Campbell, “An ECA-P policy-based framework for managing ubiquitous computing environments,” *Mobile and Ubiquitous Systems: Networking and Services - MobiQuitous 2005.*,2005.
[16]. “OpenWBEM project”, www.openwbem.org, 2006-04-05.
[17]. P. Goncalves, J.L. Oliveira, and R.L. Aguiar, “A WBEM based solution for a 4G network integrated management,” *Joint International Conference on Autonomic and Autonomous Systems and International Conference on Networking and Services - AICT2005*,2005.
[18]. P. Gonçalves, J.L. Oliveira, and R.L. Aguiar, “An evaluation of network management protocols,” *11th IFIP/IEEE International Symposium on Integrated Network Management (IM 2009)*, IEEE.
[19]. DMTF, “Web-Based Enterprise Management (WBEM) Initiative”, http://www.dmtf.org/standards/wbem/, 2008-12-14.
[20]. DMTF, Web Services for Management (WS-Management), 2005.
[21]. S. Omari, R. Boutaba, and O. Cherkaoui, “Enterprise directory support for future SNMPv3 network management applications,” *Global Telecommunications Conference, 1999. GLOBECOM '99*, pp. 2010-2014 vol.2013.
[22]. R. Lopes, N. Raimundo, M. Varanda, J. Oliveira, and V. Roque, “Executable Graphics for PBNM ” *5th IEEE International Workshop on IP Operations and Management - IPOM 2005*, Springer, 2005, pp. 108-117.