

**UNIVERSIDADE FEDERAL DE SANTA MARIA  
CENTRO DE TECNOLOGIA  
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO**

**SGPCA – SISTEMA GERENCIADOR DE POLÍTICAS  
DE CONTROLE DE ACESSO**

**Dissertação de Mestrado**

**Paulo Ricardo Barbieri Dutra Lima**

**Santa Maria, RS, Brasil**

**2008**

# **SGPCA – SISTEMA GERENCIADOR DE POLÍTICAS DE CONTROLE DE ACESSO**

**por**

**Paulo Ricardo Barbieri Dutra Lima**

Dissertação apresentada ao Curso de Mestrado do Programa de Pós-Graduação em Engenharia de Produção, Área de Concentração em Tecnologia da Informação, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Mestre em Engenharia de Produção.**

**Orientador Prof. Dr. Raul Ceretta Nunes**

**Santa Maria, RS, Brasil**

**2008**

Dados Internacionais de Catalogação

L732 s

Lima, Paulo Ricardo Barbieri Dutra.

SGPCA – Sistema Gerenciador de Políticas de Controle de Acesso / Paulo Ricardo Barbieri Dutra Lima . – 2008.

91 f. : il.

Orientador: Raul Ceretta Nunes.

Dissertação (mestrado).

Universidade Federal de Santa Maria, Programa de Pós-graduação de Engenharia de Produção, 2008.

1. Tecnologia da Informação. 2. Controle de Acesso. 3. Políticas de Controle de Acesso. I. Lima, Paulo Ricardo Barbieri Dutra.

CDU 004. 78

Cristiane Rodrigues da Silva Crb 10/1802

---

© 2008

Todos os direitos autorais reservados a Paulo Ricardo Barbieri Dutra Lima. A reprodução de partes ou do todo deste trabalho só poderá ser feita com autorização por escrito do autor.

Endereço: Rua dos Andradas, n. 1730, bairro centro, Santa Maria, RS, 97010-032

Fone (55) 3222.6690; e-mail: [prbdl@yahoo.com.br](mailto:prbdl@yahoo.com.br)

---

**Universidade Federal de Santa Maria  
Centro de Tecnologia  
Programa de Pós-Graduação em Engenharia de Produção**

A Comissão Examinadora, abaixo assinada,  
aprova a Dissertação de Mestrado

**SGPCA – SISTEMA GERENCIADOR DE POLÍTICAS DE CONTROLE  
DE ACESSO**

elaborada por  
**Paulo Ricardo Barbieri Dutra Lima**

como requisito parcial para obtenção do grau de  
**Mestre em Engenharia de Produção**

**COMISSÃO EXAMINADORA:**

---

**Raul Ceretta Nunes, Dr. (UFSM)**  
(Presidente/Orientador)

---

**Roseclea Duarte Medina, Dra. - (UFSM)**

---

**Luis Felipe Dias Lopes, Dr. (UFSM)**

Santa Maria, 17 de Março de 2008.

## Agradecimentos

No decorrer destes dois anos de pesquisa no mestrado em engenharia de produção da UFSM, tenho muito a quem agradecer.

Primeiramente a Deus, pela vida.

A CAPES pelo fomento a pesquisa, fundamental para o desenvolvimento dos trabalhos.

A todos os colegas da GMICRO pela amizade e pelo companheirismo.

Ao meu amigo professor Dr. Raul Ceretta Nunes, pela excelente orientação, com certeza um exemplo de profissional.

Às Professoras Dra. Elena Ferrari e Dra. Bárbara Carminati pela acolhida na Itália e pelas orientações sobre gerenciamento baseado em políticas.

Em especial aos meus pais e minha namorada pelo apoio e incentivo ao longo do curso.

Enfim, agradeço a todos os demais que deram a sua contribuição para minha formação profissional.

## **RESUMO**

Dissertação de Mestrado  
Programa de Pós-Graduação em Engenharia de Produção  
Universidade Federal de Santa Maria

### **SGPCA – SISTEMA GERENCIADOR DE POLÍTICAS DE CONTROLE DE ACESSO**

AUTOR: PAULO RICARDO BARBIERI DUTRA LIMA

ORIENTADOR: DR. RAUL CERETTA NUNES

Data e Local da Defesa: 17 de março de 2008.

A informação é o bem mais valioso para as organizações, logo deve-se ter mecanismos para que ela possa ser bem protegida e que seja disponível somente para quem tem real permissão de utilizá-la. Dado esta necessidade de proteção da informação nas organizações, propõe-se neste trabalho um sistema de gerenciamento de políticas de controle de acesso, que possa ser utilizado de forma facilitada, ou seja, não requerendo conhecimento de linguagem de codificação de políticas. Adicionalmente, como a criação de novas políticas pode gerar conflitos com as já existentes, este trabalho propõe também algoritmos que gerenciam automaticamente, em tempo de criação das políticas, o controle de alguns tipos de conflitos, tais como conflitos de interesse. Como resultado tem-se um Sistema Gerenciador de Políticas de Controle de Acesso que possibilita que o processo de geração e edição de políticas ocorra de maneira facilitada e sem conflitos. O modelo de referência utilizado neste trabalho refere-se no âmbito de organizações da saúde, mas o estudo realizado pode ser utilizado em outras áreas

Palavras-chave: Gerenciamento baseado em políticas, Políticas de controle de acesso, XACML, Conflitos, segurança e autorização.

## **ABSTRACT**

**Master Dissertation  
Graduate Program of Production Engineering  
Federal University of Santa Maria**

**SGPCA – SISTEMA GERENCIADOR DE POLÍTICAS DE CONTROLE  
DE ACESSO**

**AUTHOR: PAULO RICARDO BARBIERI DUTRA LIMA**

**ADVISER: RAUL CERETTA NUNES, DR.**

**Date and Local: March, 17<sup>th</sup> of 2008, Santa Maria.**

Information is the most precious assets to organizations; hence it is necessary to have mechanisms to protect it and to make it available only to whom have factual permission to use it. Considering the need for protection of the information in organizations it is proposed in this work a system to manage access control policies which can be easily used, that is, it does not require any knowledge of policies codification language. Further, as the creation of new policies could generate conflicts with existent ones, this work also proposes algorithms which manage automatically, in a period of policies creation, the control of some kinds of conflicts, such as interest conflicts. As result, we have offer a Access Control Police Management System that enable that the process of generation and editing policies occurs easily and without conflicts. The reference model used in this work refers to health organizations; however this study can be applied in other fields.

Key-words: Police based Management, Access control policies, XACML, Conflicts, Security, Authorization

## LISTA DE FIGURAS

Figura 1 – Arquitetura IETF .....	21
Figura 2 – Arquitetura do PONDER e Editor de Políticas .....	22
Figura 3 – Larges – Seg .....	23
Figura 4 – Arquitetura do Gerenciador de Políticas para o Globus .....	24
Figura 5 – Sectet-Pl .....	25
Figura 6 – Exemplo de codificação de políticas em SPL .....	31
Figura 7 – Exemplo de Política em SPL que atribui ações a um perfil .....	31
Figura 8 – Codificação da Política em XML .....	32
Figura 9 – Codificação TOWER .....	33
Figura 10 – Codificação PONDER .....	34
Figura 11 – Política XACML parte 1 .....	36
Figura 12 – Política XACML parte 2 .....	37
Figura 13 – Separação de Deveres 1 .....	44
Figura 14 – Separação de Deveres 2 .....	45
Figura 15 – Políticas Duplicadas .....	45
Figura 16 – Verificação Dinâmica de Conflitos .....	46
Figura 17 – RBAC com DUA e DRA .....	50
Figura 18 – Hierarquia de perfis para o HUSM .....	52
Figura 19 – Unidades do HUSM .....	53
Figura 20 – Classificação dos Conflitos .....	54
Figura 21 – Algoritmo (Conflitos por Regras Negativas) .....	54
Figura 22 – Algoritmo (Conflitos de Interesse) .....	56
Figura 23 – Exemplo - Conflitos de Interesse .....	57
Figura 24 – Algoritmo (Políticas Duplicadas) .....	58
Figura 25 – Algoritmo (Conflito de Ações) .....	59
Figura 26 – Arquitetura da Interface .....	60
Figura 27 – Política gerada parte 1 .....	64
Figura 28 – Regras Negativas DUA .....	66
Figura 29 – Regras Negativas DRA .....	67
Figura 30 – Código Target 1 .....	67
Figura 31 – Código Target 2 .....	68



Figura 32 – Interface para Gerenciamento de PCAs Positivas .....	70
Figura 33 – Interface para Gerenciamento de PCA Negativas (DUA) .....	71
Figura 34 – Interface para Gerenciamento de PCA Negativas (DRA) .....	72
Figura 35 – Painel de Configuração de Conflitos de Interesse .....	73
Figura 36 – Política 1 / Interface .....	74
Figura 37 – Política 1 / Codificação .....	75
Figura 38 – Política 2 / Interface .....	76
Figura 39 – Política 2 / Codificação .....	76
Figura 40 – Política 3 / Interface .....	77
Figura 41 – Painel de Controle / Interface .....	78
Figura 42 – Política 5 / Interface .....	79
Figura 43 – Política 7 / Interface .....	80
Figura 44 – Política 9 / Interface .....	81
Figura 45 – Política 10 / Interface .....	82

## **LISTA DE TABELAS**

Tabela 1 – Resumo dos Gerenciadores de Políticas .....	26
--	----

## LISTA DE SIGLAS

CIBAC	<i>Contextual Information-Based Access Control</i>
DAC	<i>Discretionary Access Control</i>
DRA	<i>Denial Role Assignment</i>
DUA	<i>Denial User Assignment</i>
EPR	<i>Electronic Patient Record</i>
GMICRO	Grupo de Microeletrônica
HUSM	Hospital Universitário de Santa Maria
IETF	<i>Internet Engineering Task Force</i>
InCor	Instituto do Coração
LDAP	<i>Lightweight Directory Access Protocol</i>
MAC	<i>Mandatory Access Control</i>
O	Objeto
OASIS	<i>Open Architecture for Securely Interworking Services</i>
P	Política sendo gerada
PA	<i>Permission Assignment</i>
PCA	Política de Controle de Acesso
PEP	<i>Policy Enforcement Point</i>
PDP	Ponto de Decisão de Políticas
PR	Repositório de Políticas
PRMS	<i>Permission</i>
RBAC	<i>Role-Based Access Control</i>
SGPCA	Sistema Gerenciador de Políticas de Controle de Acesso
SPL	<i>Security Policy Language</i>
SR	Separação de Responsabilidades
SSI	<i>Static Separation of Interest</i>
TRBAC	<i>Temporal Role-Based Access Control</i>
UA	<i>User Assignment</i>
UFSC	Universidade Federal de Santa Maria
USR	Usuário
VPN	Virtual Private Network
XACML	<i>eXtensible Access Control Markup Language</i>
XML	<i>Extensible Markup Language</i>

## **LISTA DE ANEXOS**

ANEXO A – Publicações .....	91
-----------------------------	----

## SUMÁRIO

<b>INTRODUÇÃO</b> .....	15
<b>1.1 Objetivos</b> .....	16
<b>1.2 Motivação</b> .....	16
<b>1.3 Metodologia</b> .....	17
<b>1.4 Contribuições da dissertação</b> .....	18
<b>1.5 Organização da Dissertação</b> .....	18
<b>2 GERENCIAMENTO BASEADO EM POLÍTICAS</b> .....	20
2.1 Necessidade de Gerenciamento Baseado em Políticas .....	20
<b>2.2 Arquitetura para Gerenciamento baseado em Políticas</b> .....	21
<b>2.3 Gerenciadores de Políticas</b> .....	22
2.3.1 Ponder .....	22
2.3.2 Larges-Seg .....	23
2.3.3 Gerenciador de Políticas para o Globus .....	24
2.3.4 Sectet-Pl – Um protótipo gerador XACML .....	25
<b>2.4 Conclusões Parciais</b> .....	26
<b>3 POLÍTICAS DE CONTROLE DE ACESSO</b> .....	27
<b>3.1 Conceitos e Definições</b> .....	27
<b>3.2 Políticas Discrecionárias e Mandatórias</b> .....	28
<b>3.3 Políticas Baseadas em Perfis</b> .....	28
<b>3.4 Políticas com Regras Negativas</b> .....	29
<b>3.5 Linguagem de Codificação de Políticas</b> .....	30
3.5.1 SPL .....	30
3.5.2 XML .....	32
3.5.3 TOWER .....	33
3.5.4 PONDER .....	34
3.5.5 XACML .....	34
<b>3.6 Conclusões Parciais</b> .....	37
<b>4 CONFLITOS NAS POLÍTICAS</b> .....	38
<b>4.1 Conceitos Básicos</b> .....	38
<b>4.2 Classificações dos Conflitos</b> .....	39
4.2.1 Conflitos de Redundância .....	40
4.2.2 Conflitos de Negação .....	40

4.2.3 Conflitos entre políticas de obrigação e de autorização .....	41
4.2.4 Conflitos por Auto-Gerência .....	41
4.2.5 Conflitos de Múltiplos Gerentes .....	41
4.2.6 Conflitos de Interesse e de Deveres .....	42
4.2.7 Conflitos de Recursos .....	42
<b>4.3 Resoluções de Conflitos .....</b>	<b>43</b>
<b>4.4 Conclusões Parciais .....</b>	<b>46</b>
<b>5 ESPECIFICAÇÃO DO SGPCA .....</b>	<b>48</b>
<b>5.1 Definições .....</b>	<b>48</b>
<b>5.2 Requisitos do Sistema .....</b>	<b>51</b>
<b>5.3 Especificação dos Algoritmos .....</b>	<b>53</b>
5.3.1 Conflitos de Negação .....	54
5.3.2 Conflitos de Interesse .....	55
5.3.3 Conflitos de Redundância .....	57
<b>5.4 Projeto de Interface .....</b>	<b>59</b>
<b>5.5 Conclusões Parciais .....</b>	<b>61</b>
<b>6 IMPLEMENTAÇÃO DO SGPCA .....</b>	<b>62</b>
<b>6.1 Arquitetura do CIBAC .....</b>	<b>62</b>
<b>6.2 Implementação do SGPCA .....</b>	<b>63</b>
6.2.1 Representação de Políticas XACML no SGPCA .....	63
6.2.2 Representação de Políticas no código do SGPCA .....	67
<b>6.3 Dinâmica de Funcionamento do SGPCA .....</b>	<b>69</b>
6.3.1 Políticas Positivas .....	69
6.3.2 Geração de Políticas Negativas .....	71
6.3.3 Painel de Configuração de Conflitos de Interesse .....	72
<b>6.4 Testes no SGPCA .....</b>	<b>73</b>
6.4.1 Testes na Geração de Políticas .....	73
6.4.2 Testes de Conformidade .....	83
<b>6.5 Conclusões Parciais .....</b>	<b>84</b>
<b>7 CONCLUSÕES E PERSPECTIVAS .....</b>	<b>86</b>
<b>7.1 Conclusões .....</b>	<b>86</b>
<b>7.2 Trabalhos Futuros .....</b>	<b>86</b>
<b>REFERÊNCIAS .....</b>	<b>88</b>
<b>Anexo A – PUBLICAÇÕES .....</b>	<b>91</b>

## INTRODUÇÃO

Atualmente, as inovações presentes nas áreas de comunicação e computação vem proporcionando um aumento da utilização de sistemas computacionais em diversas áreas, como negócios, saúde e educação. Na área da saúde, a disponibilização de informações clínicas em redes de computadores, tal como o *Eletronic Patient Record* (EPR)<sup>1</sup>, necessita manter a privacidade dos pacientes, a confidencialidade dos dados e a integridade da informação (CFM, 2002).

A informação é o bem mais valioso para uma empresa, logo deve-se ter meios para que seja bem protegida e gerenciada. Deste modo, a segurança da informação é mantida com métodos fortes de controle de acesso e Políticas de Controle de Acesso<sup>2</sup> (PCA) bem elaboradas.

Uma PCA define diretrizes de alto nível que determinam como o acesso é controlado e como as decisões de autorização de acesso são estabelecidas (FERRAILOLO et al., 2001). Em outras palavras, estabelecem como um objeto<sup>3</sup> irá ser acessado em um meio. Um sistema aumentará seu nível de segurança na medida em que satisfaça a política de acesso aos seus dados. Por exemplo, em uma empresa se os dados confidenciais de seus clientes forem regulados por uma política que determina que somente os próprios clientes podem ter acesso a estes dados e os demais clientes e funcionários não, pode-se dizer que o sistema está minimizando as chances de ocorrência de uma falha de segurança.

Uma PCA deve estar de acordo com a política de segurança de uma organização, que tem como finalidade proteger os objetos contra ameaças que possam comprometer a confidencialidade, integridade e disponibilidade. A especificação precisa da implementação das políticas é importante com a finalidade de obter ou capturar os objetivos organizacionais da empresa e o correto uso das tecnologias (DAMIANOU, 2001).

A vantagem de gerenciar uma rede ou sistema distribuído usando políticas é que pode-se concentrar esforços na real necessidade de negócio. O gerenciamento baseado em políticas possibilita soluções que sejam rapidamente adaptáveis, uma vez que basta editar ou reeditar uma política.

---

<sup>1</sup> A razão da utilização da nomenclatura do Prontuário Eletrônico do Paciente em Inglês deve-se pelo fato de já existir a sigla (PEP) referenciando à Policy Enforcement Point.

<sup>2</sup> No decorrer do texto, pode-se utilizar o termo política para representar uma PCA, com a finalidade de facilitar a concordância do texto.

<sup>3</sup> Objeto refere-se a um recurso ou informação em um sistema ou empresa, no qual deseja-se proteger.

Geralmente uma PCA é definida por um administrador de redes ou uma pessoa que tenha um bom conhecimento técnico, pois essas políticas são especificadas normalmente em linguagens de marcação como, por exemplo, XACML (Extensible Access Control Markup Language) (OÁSIS, 2003) e PONDER (DAMIANOU, 2001), exigindo assim o conhecimento da linguagem. Adicionalmente, políticas podem conflitar em suas permissões se os relacionamentos entre as políticas não forem cuidadosamente verificados (SHANKAR, RANGANATHAN, CAMPBELL, 2005).

Conflitos em PCA podem acontecer quando os objetivos das políticas são os mesmos, por exemplo, um usuário não pode ser ao mesmo tempo permitido e negado para realizar uma determinada ação em um objeto. Duas políticas conflitantes não pode coexistirem juntas pois a resposta não será a apropriada, gerando assim uma vulnerabilidade no sistema.

Este trabalho procura minimizar a exigência de conhecimento técnico do administrador de rede em tempo de geração das políticas de controle de acesso.

## **1.1 Objetivos**

O objetivo geral deste trabalho é de possibilitar que a edição e criação de PCAs possa ser realizada de forma amigável e consistente, ou seja, de forma facilitada.

Os objetivos específicos são: especificar e implementar um sistema de edição de políticas, identificar conflitos em tempo de geração de políticas e fornecer uma interface amigável para eliminar a necessidade de conhecimento de linguagem de codificação de políticas.

## **1.2 Motivação**

Uma rede de computadores típica consiste em um largo número de equipamentos como: roteadores, servidores e uma variedade de aplicações oferecendo serviços a um grande número de usuários. A gerência de forma manual desses objetos e serviços é custosa e o



esforço e o tempo necessário aumentam exponencialmente à medida que o sistema expande-se (DAMIANOU, 2001).

Para gerar uma nova política, geralmente existe um formulário de solicitação de acesso, que o usuário interessado deve preencher a mão, onde é descrito os objetos que deseja-se ter direito de utilizar. Esta requisição é então enviada para um administrador de redes, que posteriormente adiciona a uma nova PCA que possibilita o acesso ao usuário.

Neste fluxo pode-se identificar alguns problemas como:

1) o criador da política deve ser um especialista, para poder descrevê-la em linguagem técnica e para poder avaliar se uma nova política não conflita com as já geradas;

2) um administrador de redes de uma empresa de médio porte tem inúmeras tarefas como por exemplo: configuração de firewall, roteadores, políticas, IIS entre outros, logo administrar estes inúmeros serviços é uma atividade exaustiva.

O gerenciamento baseado em política fornece ao administrador de redes a flexibilidade de não preocupar-se com detalhes técnicos da implementação das políticas, o qual seria uma tarefa custosa se fosse realizada de forma convencional, pois demandaria de conhecimento técnico da linguagem de políticas como: XACML, PONDER, entre outras.

Na literatura existem alguns gerenciadores de políticas (abordados no próximo capítulo) que em muitos casos são difíceis de serem utilizados e não realizam a verificação de conflitos. Diante desta exposição, surge a motivação para propor um gerenciador de PCA que seja de fácil administração e que forneça verificação automática de conflitos em tempo de criação ou edição de um PCA.

### **1.3 Metodologia**

A metodologia utilizada teve cunho teórico com bases no mapeamento e análise da literatura sobre: mecanismos de controle de acesso, políticas de controle de acesso, linguagens de políticas, gerenciadores de PCA. Conflitos e algoritmos para verificação de conflitos.

O processo de engenharia de software utilizado para a implementação do modelo e da arquitetura proposta foi o desenvolvimento evolucionário (SOMMERVILLE, 1996), que prevê que as atividades de especificação, projeto, implementação e validação ocorram juntas.

Com base em uma especificação inicial, projeta-se e implementa-se um protótipo inicial, para posteriormente ser validado.

A aplicação prática da implementação do gerenciador de políticas de controle de acesso se dá através da prototipação. Tendo por objetivo fornecer um gerenciamento facilitado de políticas de controle de acesso.

#### **1.4 Contribuições da dissertação**

O estudo realizado nesta pesquisa acerca da especificação e do gerenciamento baseado em políticas teve a finalidade de fornecer uma maior segurança tanto em sistemas distribuídos como em controle de acesso. As contribuições resultantes deste trabalho são:

- um modelo de controle de acesso que possibilita a geração de políticas negativas (sendo que políticas negativas expressão uma negação de um usuário ou perfil de acessar um determinado objeto) tanto para usuários como para perfis, garantindo assim uma maior flexibilidade;
- a especificação e implementação de um sistema de edição de políticas de controle de acesso que possibilite a geração e edição das políticas de maneira simples e livre de conflitos;
- uma interface gráfica pela qual um usuário pode realizar configurações de conflitos, conforme a estrutura organizacional da empresa.

#### **1.5 Organização da Dissertação**

O trabalho está dividido em sete capítulos. No capítulo 2 é apresentado o modelo arquitetural sobre gerenciamento baseado em políticas, bem como alguns dos principais gerenciadores já propostos na literatura. O capítulo 3 destaca conceitos sobre políticas e sobre linguagens para sua elaboração. No capítulo 4 é apresentado um estudo sobre conflitos entre políticas. No capítulo 5, são apresentadas as especificações do sistema junto com sua

modelagem. O capítulo 6 apresenta detalhes sobre a implementação do sistema junto com seus testes. No último capítulo, apresentam-se as conclusões, e os trabalhos futuros.

## **2 GERENCIAMENTO BASEADO EM POLÍTICAS**

Neste capítulo apresentam-se conceitos sobre o gerenciamento baseado em políticas, junto com sua aplicação e a real necessidade das empresas de realizar um gerenciamento automatizado. Será apresentado uma revisão dos Gerenciadores de Políticas já especificados e implementados, com a finalidade de verificar a real contribuição científica deste trabalho.

### **2.1 Necessidade de Gerenciamento Baseado em Políticas**

Uma organização real ou virtual pode ter um grande número de objetos e gerenciar estes objetos é uma tarefa complexa (LORCH, KAFURA e SHAH, 2003). Cada objeto em uma organização tem seu próprio proprietário com determinados mecanismos de controle de acesso.

O modelo de gestão de redes com a utilização de políticas traduz as diretrizes das empresas (políticas de alto nível) para o universo dos elementos de rede (políticas de baixo nível), permitindo que estas políticas sejam difundidas mais fáceis e rapidamente, sem a necessidade de gerência direta em equipamentos e arquivos, possibilitando desta forma, a reutilização de conhecimentos e processos.

A gerência baseada em PCA vem tornando-se uma solução muito empregada e prometedora para controlar as redes de computadores e sistemas distribuídos. Tais redes e sistemas são dirigidos pelas necessidades dos negócios, que requerem soluções de gerência auto-adaptáveis e que mudam dinamicamente o comportamento do sistema (DAMIANOU, 2001).

Adicionalmente, os Administradores de Redes devem ser isolados dos detalhes da implementação das políticas, o que pode ser alcançado com ferramentas que permitam a administração integrada e escondam os detalhes de implementação das políticas.

O principal benefício do uso do gerenciamento baseado em políticas é o aumento da escalabilidade e flexibilidade. A escalabilidade é aumentada porque a mesma política é aplicada a um grande número de objetos. A flexibilidade é ampliada pela separação da política da implementação, o que possibilita a realização de mudanças dinamicamente nas políticas, possibilitando mudar o comportamento do sistema sem modificar a implementação.

Entretanto, conseguir esta facilidade e obter uma completa consistência entre as políticas não é uma tarefa trivial, tendo em vista que pode existir centenas de políticas. A solução deve contemplar a detecção e resolução de conflitos.

## 2.2 Arquitetura para Gerenciamento baseado em Políticas

A Figura 1 mostra a arquitetura para gerenciamento baseado em políticas definido pela IETF (*Internet Engineering Task Force*) (STRASSNER et al., 2001) a qual é uma organização que reúne pesquisadores e projetistas de redes com a finalidade da evolução e padronização de arquiteturas. Este modelo é usado como base da maioria das ferramentas de gerência de políticas (*Policy management tool*).

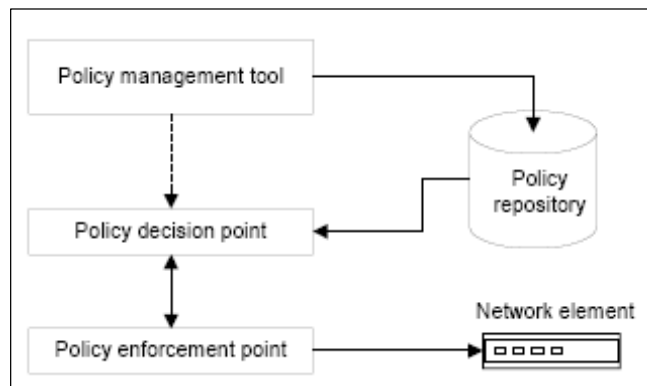


Figura 1 – Arquitetura IETF.

A arquitetura para gerenciamento de políticas da IETF, é composta de uma interface gráfica (equivale ao *Policy management tool* na Figura 1) para permitir que os administradores especifiquem as políticas; um repositório de políticas (*Policy repository*) que é usado para armazenar as políticas geradas pelo gerenciador de políticas; um *Policy Enforcement Point* (PEP), que realiza uma requisição baseada nos atributos do requisitante, nos objetos em questão e na ação; e um *Policy Decision Point* (PDP), que pega esses atributos e seleciona uma política que aplica-se a uma determinada requisição.

## 2.3 Gerenciadores de Políticas

### 2.3.1 Ponder

A arquitetura utilizada no PONDER (DAMIANOU, 2001) baseia-se na distribuição de políticas em domínios, ou seja, a política é gerada pelo Editor de Políticas conforme a ilustração da direita da Figura 2. Depois de gerada a PCA existe outra ferramenta chamada de Browser de domínio, que envia a política gerada para um determinado domínio. Existe também um compilador, que verifica a sintaxe de codificação das políticas e por fim temos o gerenciador de ferramentas o qual gerencia todos os atributos da política como pode-se observar na Figura 2.

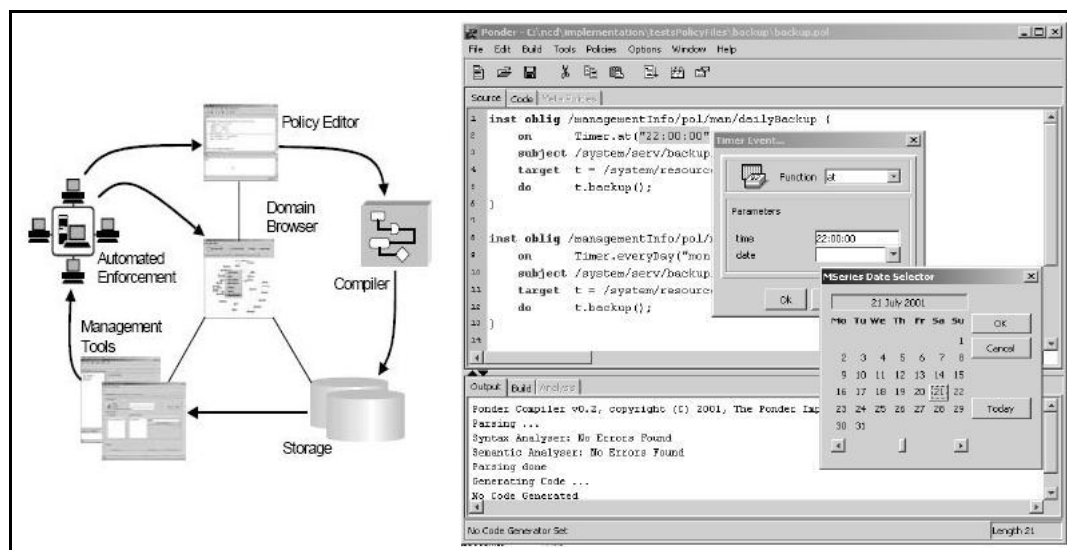


Figura 2 – Arquitetura do PONDER e Editor de Políticas

Esta arquitetura foi construída basicamente para fornecer um gerenciamento em servidores de backup, onde uma política é codificada e depois compilada. Foram propostos também uma linguagem chamada de PONDER, o qual será mais detalhada na sub-seção 3.4.4. A verificação de conflitos neste caso é realizada no momento da compilação, ou seja, quando uma política é gerada. Para a geração das políticas é necessário o conhecimento da

linguagem PONDER o qual torna-se de certa forma complexo, e não indicado para usuários convencionais.

### 2.3.2 Larges-Seg

O modelo de gerenciamento baseado em políticas Larges-Seg foi proposto em (FERREIRA, SANTOS e JÚNIOR, 2004) e utiliza a linguagem XACML para a geração das Políticas, sendo destinado para o gerenciamento de VPNs. A motivação para sua elaboração partiu do pressuposto do aumento do estabelecimento de VPNs, podendo comprometer a segurança de uma empresa se for realizada por pessoal não autorizado.

Na sua interface, são informados os seguintes campos para a geração/edição de PCAs: servidor de origem e destino, utilização ou não de criptografia, horário permitido de acesso e se o usuário é do tipo cliente ou funcionário. Estes dados servem para fornecer segurança no estabelecimento de VPNs, no entanto a verificação de conflitos nas PCAs não é tratada.

Na Figura 3 pode-se observar a arquitetura do Larges – Seg, onde temos o PEP que atua aqui entre dois pontos de uma rede, ou seja, no momento que a conexão é estabelecida. Pode-se observar também o PDP o qual atua entre as políticas e os objetos. Por fim existe um repositório de políticas LDAP.

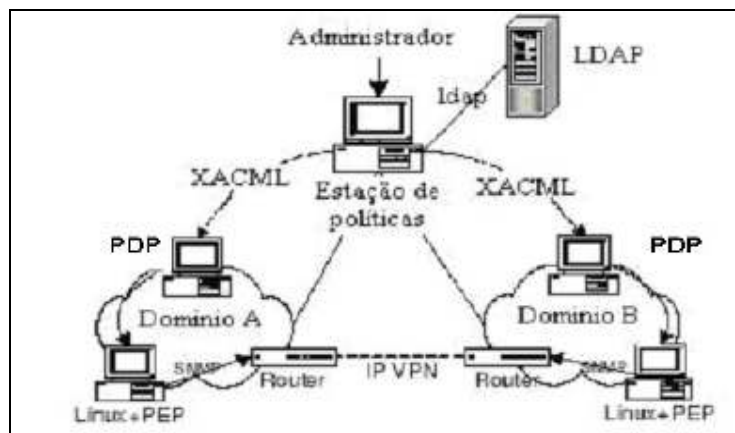


Figura 3 – Larges – Seg

Nos PDPs de cada domínio, serão validadas informações como: servidor de origem e destino, ao ser solicitado a VPN. Caso o acesso seja negado, será gerado um relatório de ocorrências que é disponibilizado para eventuais consultas do administrador, possibilitando, segundo os autores, a detecção da configuração errônea de políticas e/ou detecção de intrusão, ou mesmo tentativas de usuários burlarem as permissões.

### 2.3.3 Gerenciador de Políticas para o Globus

No Gerenciador de Políticas para o Globus (LORCH, KAFURA e SHAH, 2003), a gerência de políticas XACML é realizada por uma aplicação Java, onde um administrador de redes pode navegar entre os elementos das políticas selecionadas e especificar atributos presentes nas políticas como: objetos, usuários e ações<sup>4</sup>.

A aplicação tem em conjunto as funcionalidades para geração e modificação de políticas em XACML. Para manter a integridade, proteção e não repúdio, depois de gerar uma PCA o sistema utiliza uma assinatura digital antes do armazenamento e distribuição da política, possibilitando a criptografia da PCA gerada.

Esta aplicação utiliza a biblioteca Sun XACML (SUN, 2007), o qual fornece suporte para a construção de PCAs e também para requisições e respostas de acesso. Entretanto, o Globus não realiza verificação de conflitos nas PCAs que é um importante fator em gerenciamento de políticas.

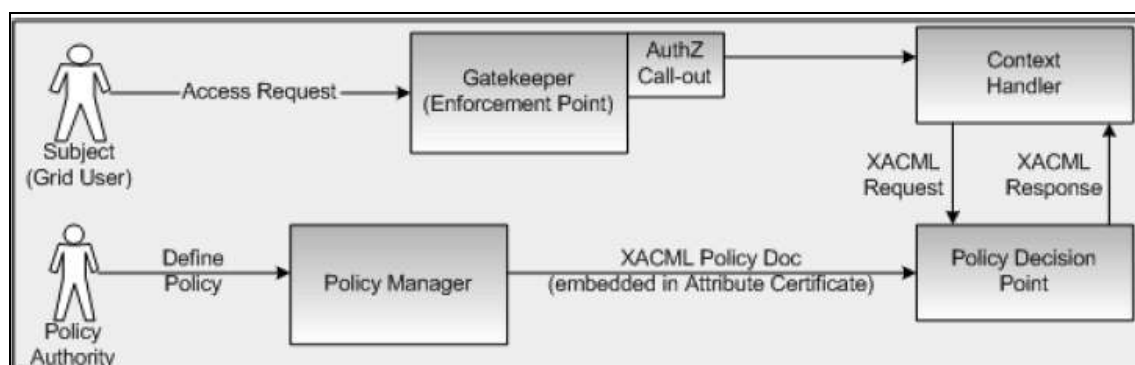


Figura 4 – Arquitetura do Gerenciador de Políticas para o Globus

<sup>4</sup> Ações são as operações que um determinado usuário ou perfil pode executar sobre um objeto.



Na Figura 4, pode-se observar o confronto entre uma requisição de acesso e uma política definida por um administrador de redes. Os elementos presentes nesta arquitetura são: o *Policy Manager*, que corresponde a uma interface onde uma PCA é especificada; o PDP; o *Context Handler*, que atua como um PEP; e o GateKeeper, que representa para o usuário o mecanismo de controle de acesso.

### 2.3.4 Sectet-PI – Um protótipo gerador XACML

O protótipo Sectet-PI (ALAM, BREU e HAFNER, 2006) foi desenvolvido para gerar políticas em XACML. E em conjunto com este gerador é apresentada uma linguagem de codificação própria. Na Figura 5, observa-se um exemplo de uma regra Sectet-PI junto com a interface de geração da PCA.



Figura 5 – Sectet-PI

No quadro esquerdo da Figura 5, observa-se uma regra dizendo que um Médico pode verificar o EPR todos os dias. Nota-se que abaixo esta mesma regra encontra-se codificada. Para a geração de uma política esta regra codificada deve ser colocada no protótipo para a geração de uma política em XACML, tal com ilustrado no lado direito da figura. Deve-se observar também que a verificação de conflitos não é tratada.

## 2.4 Conclusões Parciais

Neste capítulo foi salientado a real necessidade de implantação de gerenciadores baseados em políticas nas organizações, bem como foi realizada uma revisão bibliográfica sobre os principais gerenciadores de políticas já propostos na literatura.

Observa-se que um gerenciamento baseado em políticas tem como principal vantagem a melhora da escalabilidade e flexibilidade do sistema sobre gerenciamento, pois possibilita que uma mesma política seja aplicada a vários objetos e que o comportamento do sistema seja alterado sem alterar a implementação.

Outro importante fator a observar é que a estrutura básica dos gerenciadores de políticas segue o modelo arquitetural definido pela IETF. Na Tabela 1 pode-se observar uma tabela resumo dos gerenciadores de políticas.

**Tabela 1 – Resumo dos Gerenciadores de Políticas**

<b>GERENCIADORES DE POLÍTICAS</b>	<b>VERIFICAÇÃO DE CONFLITOS</b>	<b>OBJETIVO</b>
PONDER	SIM	GERENCIAMENTO EM SERVIDORES DE BACKUP
LARGES - SEG	NÃO	GERENCIAMENTO DE VPNS
GERENCIADOR DE POLÍTICAS PARA O GLOBUS	NÃO	GERENCIAMENTO PARA COMPUTAÇÃO PARALELA
SECTET - PL	NÃO	GERENCIA DE POLÍTICAS PARA UNIDADES HOSPITALARES

## **3 POLÍTICAS DE CONTROLE DE ACESSO**

Políticas de controle de acesso são um meio de restringir o acesso a objetos protegidos. Neste capítulo serão apresentados conceitos de políticas de segurança e de políticas de controle de acesso, junto com diferentes tipos de linguagens utilizadas para a codificação de políticas.

### **3.1 Conceitos e Definições**

Uma política de segurança é elaborada considerando-se o ambiente em que se está trabalhando, para que os critérios estabelecidos estejam de acordo com as práticas internas da empresa e com as práticas de segurança atualmente adotadas, a fim de buscar uma conformidade maior com critérios atualizados e reconhecidos em todo o mundo.

O principal propósito de uma política de segurança é informar aos usuários, equipe e gerentes, as suas obrigações para a proteção da tecnologia e do acesso à informação. A política deve especificar os mecanismos através dos quais estes requisitos podem ser alcançados. Outro propósito é oferecer um ponto de referência a partir do qual se possa adquirir, configurar e auditar sistemas computacionais e redes, para que sejam adequados aos requisitos propostos.

Para Lorch, Kafura e Shah (2003), políticas de controle de acesso definem como os serviços em sistemas computacionais podem ser usados, ou seja, são um meio de especificar ou modificar o comportamento da gerência dentro de um sistema.

Os conceitos de políticas de segurança e políticas de controle de acesso são parecidos, entretanto políticas de segurança referem-se mais precisamente a nível gerencial, logo pode ser expressa em uma linguagem natural, ou seja, sem codificação propriamente dita. Já políticas de controle de acesso, são de baixo nível e são codificadas.

### 3.2 Políticas Discricionárias e Mandatórias

Num controle de acesso discricionário (DAC - *Discretionary Access Control*) as políticas restringem o acesso a objetos baseado na identidade dos usuários ou grupos nos quais pertencem.

O DAC utiliza uma matriz de acesso definida pela tupla (S,O,A), onde “S” corresponde ao usuário, “O” objeto e “A” as ações. As ações são definidas através de uma matriz onde “S” refere-se as linhas e “O” as colunas e A[s,o] reporta as ações que “s” pode realizar em “o”.

No Controle de Acesso Mandatório (MAC - *Mandatory Access Control*) políticas garantem o controle de acesso baseando-se na classificação dos usuários e objetos do sistema. Para cada objeto e usuário do sistema é atribuído um nível de segurança. O nível de segurança associado ao objeto reflete o nível de importância da informação contida no objeto, classificando o objeto quanto ao dano potencial que um acesso não autorizado traria.

O MAC identifica as seguintes ações que usuários podem executar em objetos: Executar (sem observação e sem alteração), Ler (observação e sem alteração), Anexar (sem observação mas com alteração) e Escrever (observação e alteração).

### 3.3 Políticas Baseadas em Perfis

Perfis fornecem um grupo semântico de usuários em comum, pertencendo geralmente a uma posição dentro de uma organização tal como: gerente de departamento, gerente de projeto e analista. Especificar políticas organizacionais, por exemplo, para um grupo de gerentes, permite que caso tenha-se um novo gerente não necessite re-especificar os deveres e direitos para este novo usuário (DAMIANOU, 2001).

Em Lupu e Sloman (1999) usa-se perfis como um meio para agrupar políticas relacionadas a uma posição particular, por exemplo, criando um perfil de gerente, logo todos os gerentes podem ser atribuídos ou removidos desta posição sem mudar as políticas. Defini-se também relacionamentos entre perfis no que diz respeito ao uso dos objetos compartilhados ou na estrutura organizacional.

Políticas baseadas em perfis são definidas como uma associação de grupos de usuários com um grupo de ações. Existem dois grupos de políticas associadas dentro desta definição:

- 1) Grupo Homogêneo: grupo de usuários que são associados a um perfil;
- 2) Grupo Heterogêneo: grupo de ações que um perfil é associado.

Não deve-se confundir grupos de usuários com perfis, grupos de usuário são uma associação de usuários sem nenhuma obrigação específica, já perfis são grupos onde existem uma determinada tarefa ou ação relacionada. Usuários ganham ações dentro de um sistema sendo associados a perfis.

Políticas baseadas em perfis definem os direitos e deveres associados a uma posição dentro de uma organização que um determinado usuário tem. Por exemplo, a permissão de um gerente de vendas é diferente de um funcionário normal, pelo fato que a hierarquia de perfis é diferente uma da outra.

Um relacionamento hierárquico pode ser definido tendo como base a estrutura hierárquica das próprias organizações. Em uma hierarquia de perfis, um perfil pode conter outros perfis. No RBAC, (FERRAILOLO et al., 2001), (controle de acesso baseado em perfis) um perfil hierarquicamente superior pode conter um ou mais perfis hierarquicamente inferiores. Isto significa que um perfil superior tem todos os privilégios e pode executar todas as ações que os perfis de níveis inferiores contêm.

O modelo RBAC, fornece a característica de atribuição de usuários a perfis sendo este último atributo ligado diretamente as ações e objetos do sistema. Este modelo de controle de acesso abrange tanto autorizações como políticas de controle de acesso modelando as duas situações.

### **3.4 Políticas com Regras Negativas**

A especificação ou o uso de PCAs negativas complica uma autorização de acesso no sistema, pois pode gerar conflitos com políticas positivas. Entretanto existem algumas razões para um sistema suportar regras negativas como: para dar mais flexibilidade ao sistema, para fornecer uma maior segurança e para remover os direitos de acesso de um usuário temporariamente (DAMIANOU, 2001).

Segundo Bertino, Samarati e Jajodia (1997) uma política de negação, expressa uma proibição para um usuário de acessar um determinado objeto. Políticas negativas são mais

fortes que positivas, ou seja, caso um usuário tenha duas políticas associadas, uma positiva e outra negativa a que irá prevalecer será a de negação.

Para ilustrar este cenário, suponha que deseja-se dar autorização a todos os membros de um grupo de usuários, exceto a um membro específico referenciado por “Pedro”. Na ausência de políticas negativas deve-se especificar autorizações positivas para todos os membros do grupo exceto “Pedro”, se autorizações negativas fossem consideradas bastaria dar autorização negativa para “Pedro”, fornecendo assim uma maior rapidez e segurança.

Regras negativas fornecem segurança, pois um usuário será negado de realizar um acesso no sistema, diferentemente de sistemas baseados somente em regras positivas que evitam o acesso apenas não fornecendo uma autorização positiva, o que dão margem para brechas na segurança.

### **3.5 Linguagem de Codificação de Políticas**

Existem várias linguagens com as quais pode-se elaborar políticas de controle de acesso como: XML (eXtensible Markup Language), XACML, PONDER, SPL, TOWER (HITCHENS et al., 2001) entre outras. Nas próximas seções será dado enfoque nas principais linguagens para a codificação de PCA, junto com seu referencial teórico.

#### **3.5.1 SPL**

A linguagem SPL (*Security Policy Language*) (RIBEIRO, ZUQUETE e FERREIRA, 2001), é baseada em eventos, ou seja, a cada autorização recebida é enviada uma das seguintes respostas: permitir, não permitir e ignorar. Na Figura 6 são exibidos alguns exemplos de codificação de políticas na linguagem SPL.

No item 1 pode-se verificar o código em SPL para a seguinte regra “Todo o evento em um objeto de propriedade do próprio usuário será liberado” com sua respectiva codificação logo abaixo. Pode-se observar também a regra do item 2 que evita aprovações de ordem de pagamento, pelo próprio proprietário da ordem de pagamento.

```

1 - Todo o evento em um objeto de propriedade do próprio
usuário será liberado.

Código: OwnerRule: ce.target.owner = ce.author :: true;

2 - Aprovação de ordem de pagamento não pode ser realizada
pelo proprietário da ordem de pagamento.

Código: DutySep: ce.target.type = "paymentOrder" &
ce.action.name = "approve" :: ce.author != ce.target.owner;

```

Figura 6 – Exemplo de codificação de políticas em SPL

No SPL, uma política pode herdar especificações de outras políticas, ou negar determinadas regras. Políticas podem ser usadas para modelar perfis, definindo ações que um determinado perfil pode realizar. O SPL suporta a composição de regras Hierárquicas dentro das políticas.

No código da Figura 7 pode-se verificar um exemplo no qual um perfil do tipo “clerks” pode executar todas as ações em objetos do tipo “invoice”. Esta política atribui ações a um determinado perfil, ou seja, defini quais ações poderão ser realizadas sobre um determinado objeto, caso deseja-se atribuir ações a outro perfil a estrutura de codificação da política permanece a mesma.

```

policy InvoiceManagement
{
team clerks ; // perfil
collection invoices =
AllObjects@{ .doctype = "invoice" }; // objetos
DoInvoices: new ACL(clerks, invoices, AllActions); // ações
?usingACL: DoInvoices;

```

Figura 7 – Exemplo de Política em SPL que atribui ações a um perfil.

### 3.5.2 XML

A linguagem de marcação XML (*Extended Markup Language*) (W3 CONSORTIUM, 2000), provê uma representação estruturada dos dados (metadados) e não possui elementos nem marcas pré-definidas, logo não é especificado como os autores devem utilizar esses metadados. Assim, existe total liberdade para utilizar qualquer método disponível, desde simples atributos, até a implementação de padrões mais complexos (ALMEIDA, 2002), o que possibilita o relacionamento facilitado das definições propostas.

O XML possibilita ao autor especificar a forma dos dados no documento, além de permitir definições e semânticas. Um arquivo eletrônico XML pode conter, simultaneamente, dados e a descrição da estrutura do documento, através do DTD-*Data Type Definitions* (gramáticas que conferem estrutura ao documento XML).

Em Kudo e Hada (2000), pode-se observar a utilização da linguagem XML para a codificação de políticas. No exemplo da Figura 8 tem-se uma política em XML, onde a identificação do usuário aparece na tag `<entry>` e as ações na tag `<policy>`.

```
<contents>
  <entry>
    <name>Alice</name>
    <officeTel>111-1111</officeTel>
    <homeTel>123-4567</homeTel>
  </entry>
</contents>
<policy>
  <xacl>
    <object href="/contents"/>
    <rule>
      <acl>
        <subject>
          <uid>Alice</uid>
        </subject>
        <action name="read" permission="grant"/>
        <action name="write" permission="deny"/>
      </acl>
    </rule>
  </xacl>
</policy>
```

Figura 8 – Codificação da Política em XML

No exemplo da Figura 8, a funcionária “Alice” pode executar a ação de leitura do objeto “contents”, no entanto não pode realizar nenhuma modificação neste objeto, visto que



existe uma regra na política que diz que ela está negada “deny” de escrever no objeto “contents”.

### 3.5.3 TOWER

Tower (HITCHENS et al., 2001) é uma linguagem utilizada para especificação e implementação de políticas do modelo RBAC. O Tower fornece estruturas para definição de: objetos, ações, usuários e perfis, e permite a definição de múltiplas estruturas dentro de blocos de Begin e End.

Variáveis podem ser definidas dentro de um bloco, possibilitando o sistema referenciar, por exemplo, o id de um usuário ou o tempo de acesso em um objeto. A especificação do TOWER possibilita também a restrição de validade de ações que devem ser executadas quando algum método é invocado.

Para ilustrar seu uso considere o exemplo de codificação da Figura 9. O objeto “cheque” pode ser acessado por membros do perfil “cliente”, mas um usuário não pode editar e autorizar um determinado “cheque” ao mesmo tempo.

```

begin
  issuing_user* : userid
  issue_privilege := privilege
    issuing_user := user
    {issue}
  end_privilege
  authorise_privilege := privilege
    issuing_user <> user
    {authorise}
  end_privilege
  check_permission := permission
    cheque_class
    privilege {issue_privilege, authorise_privilege}
  end_permission
  accountant := role
    permissions {check_permission}
  end_role
end

```

Figura 9 – Codificação TOWER

No exemplo, pode-se observar que a variável “issuing\_user” no primeiro bloco de instruções corresponde ao editor de cheque. No segundo bloco, onde refere-se à autorização, é definido através da expressão “issuing\_user < > user”, que caso o usuário seja o editor do cheque o mesmo não poderá ter a autorização. Este exemplo refere-se à separação de deveres no qual um usuário não pode legislar em causa própria.

### 3.5.4 PONDER

O PONDER (DAMIANOU, 2001) foi desenvolvido originalmente para gerenciar um sistema de backup. Esta linguagem é baseada em domínios, ou seja, a codificação da política é realizada especificando domínios sobre objetos, logo segue na Figura 10 um exemplo de política implementada na linguagem PONDER. Vamos supor a seguinte situação: Deseja-se possibilitar que o perfil de “Administrador de Segurança” tenha a autorização de adicionar e apagar usuários no servidor de domínio e de modificar preferências de backup de usuários. O elemento “inst auth+” refere-se ao local aonde a política será armazenada e o tipo de autorização, neste caso positiva, já o elemento “subject” diz respeito ao perfil, o objeto a ser gerenciado é referenciado pelo elemento “target”, por fim o elemento “action” trata das ações sobre um determinado objeto.

```
inst auth+ /managementInfo/pol/ac/userBackupMgmtAC {
subject /staff/admin/sec;
target bs = /system/servers/backup;
action bs.createUserAcc(), bs.deleteUserAcc(), bs.setPreferences();
}
```

Figura 10 – Codificação PONDER

### 3.5.5 XACML

O XACML padronizado pela OASIS, tem por objetivo descrever em XML, políticas de controle de acesso. As regras definidas em XACML permitem um aprimorado controle de

acesso, onde é possível determinar o modo de acesso, por exemplo, um documento somente pode ser acessado se provar que possui as ações necessárias para tal.

#### 3.5.5.1 Definições

O XACML define o formato de uma requisição que contém informações sobre o usuário, objeto, ação e unidade<sup>5</sup>. Isto facilita a especificação de algoritmos para encontrar políticas que se apliquem as requisições, facilitando também a tomada de decisões e a geração de respostas.

A linguagem para definição de políticas do XACML é usada para descrever requisitos de controle de acesso e, por se tratar de um padrão aberto, possuem pontos para extensão, possibilitando que um desenvolvedor defina novas funções, tipos de dados, combinações lógicas, etc. A resposta sempre inclui um retorno sobre se a requisição foi permitida ou negada usando quatro valores: Permitido, Negado, Indeterminado (erro na interpretação ou falta de informação sobre algum valor requerido) ou não Aplicável (a ação solicitada não é válida) (HIGASHIYAMA, 2005).

Um uso habitual da utilização do XACML é quando deseja-se gerenciar o acesso a um determinado objeto no sistema (arquivo, transação, etc.). Nesta situação utiliza-se o PEP. O PEP recebe da aplicação os dados do usuário e forma uma requisição em XACML contendo estes dados que são: recurso em questão, a ação e outras informações pertinentes a requisição. O PEP então envia a requisição para o PDP que irá mapear a requisição e identificar qual política será aplicada a uma determinada requisição.

#### 3.5.5.2 Elementos

Segundo Lorch, Kafura e Shah (2003) políticas XACML, consistem em uma árvore de sub-políticas, onde cada árvore é representada pelo elemento “target” (Cabeçalho de todas as políticas em XACML), enquanto as folhas são representadas pelo elemento “rule” (Regras de

---

<sup>6</sup> Setor de uma determinada instituição.

Acesso). O “target” tem como função avaliar se uma política é aplicada a determinada requisição, enquanto o elemento “rule” contém a ação e uma condição para uma determinada Política.

O XACML, define um padrão de dados como: string, boolean, integer, time, email, etc. e um padrão de funções como: igualdade, comparações aritméticas, etc. Estes padrões de tipos de dados e funções podem expressar diversas PCAs. O XACML permite que uma política contenha referências para outras políticas.

No exemplo da Figura 11, pode-se observar a estrutura do elemento “target” de uma política XACML, onde os primeiros elementos que deve-se observar são o “Subjects” e “Subject”, que são os usuários ou perfis presentes na política, neste exemplo o usuário é “Julius Hibbert”. A tag Resources, representa o objeto neste caso referido a um determinado EPR. O último elemento presente no “target” é o elemento “action” o qual se refere à ação que deseja-se realizar no objeto que neste caso é somente leitura (“read”).

```

<target>
  <subjects>
    <subject>
      <name>
        <value>Julius Hibbert</value>
      </name>
      <category>
        <value>access-subject</value>
      </category>
      <subject-id>
        <value>http://www.w3.org/2001/XMLSchema#string</value>
      </subject-id>
    </subject>
  </subjects>
  <resources>
    <resource>
      <value>http://www.w3.org/2001/XMLSchema#anyURI</value>
      <resource-id>
        <value>http://medica.com/record/patient/BartSimpson</value>
      </resource-id>
    </resource>
  </resources>
  <action>
    <value>read</value>
    <action-id>
      <value>http://www.w3.org/2001/XMLSchema#string</value>
    </action-id>
  </action>
</target>

```

Figura 11 – Política XACML parte 1

Em uma política XACML, as regras são expressas pela tag Rule que valida ou não um determinado target conforme a regra é elaborada. Na Figura 12, observa-se a tag Rule que coloca um restrição temporal para validar o elemento target o qual é “12:00:00” até as

“13:00:00”, sendo que somente é permitido o acesso a um determinado objeto se o usuário estiver acessando neste horário.

```

- <Rule RuleId="Access" Effect="Permit">
- <Condition FunctionId="http://research.sun.com/projects/xacml/names/function#time-in-range">
- <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
  <EnvironmentAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#time"
    AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time" />
  </Apply>
  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">12:00:00</AttributeValue>
  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">13:00:00</AttributeValue>
</Condition>

```

Figura 12 – Política XACML parte 2.

### 3.6 Conclusões Parciais

Neste capítulo, é apresentado conceitos de políticas, tendo em vista diferentes abordagens. Foram descritos também as principais linguagens de codificação de PCA junto com seu referencial teórico e exemplos de políticas já implementadas, com a finalidade de verificar a estruturação de diferentes tipos de PCA.

O conhecimento de diferentes tipos de linguagens de políticas serve para se ter uma visão mais detalhada de qual linguagem se adapta melhor a determinadas situações. Por exemplo, XACML é baseado em XML o qual é uma linguagem amplamente utilizada e compatível com diversas linguagens de programação o que facilita a sua especificação e implementação. A linguagem de codificação de políticas XACML é destinada a modelos de controle de acesso e consta com uma grande quantidade de referencial teórico, por estes motivos que foi a linguagem escolhida para a utilização neste trabalho.

## **4 CONFLITOS NAS POLÍTICAS**

É quase impossível para um administrador de redes verificar todas as políticas que podem ocasionar conflitos em um sistema. A verificação de conflitos em PCA, quando realizada de forma manual, ou seja, pela intervenção humana para apontar uma política conflitante, pode gerar erros e incompatibilidades. Já, quando se tem um mecanismo automatizado que verifique a existência de conflitos, a probabilidade de que ocorra algum erro é menor, pois é o sistema que realizará esta verificação.

Em sistemas distribuídos e de controle de acesso, freqüentemente existem múltiplos administradores que especificam políticas. Conflitos nestas políticas podem surgir devido a erros e omissões na hora em que a política está sendo criada. Neste capítulo será apresentada uma classificação de conflitos e algoritmos utilizados para o tratamento destes conflitos.

### **4.1 Conceitos Básicos**

A definição da palavra conflito no dicionário Michaelis (WEISZFLOG, 2007) é: oposição, embate e desacordo. A verificação de conflitos em PCA normalmente é realizada pelo administrador de redes o qual reconhece e evita conflitos de forma intuitiva utilizando a experiência adquirida no tratamento de políticas conflitantes.

Segundo Nyanchama e Osborn (1999), em ambientes críticos, onde determinadas ações de um usuário podem causar enormes danos, é muito importante conhecermos as combinações de operações que são permitidas e também as que os usuários não podem realizar. Por exemplo: em uma aplicação de venda um usuário não pode modificar ou estipular o valor de um determinado produto e ser o consumidor do mesmo, pois ele pode diminuir o valor ou realizar outros danos.

Para que ocorram conflitos em políticas de controle de acesso a seguinte condição conflitante deve ser satisfeita, deve ocorrer pelo menos uma sobreposição de uma regra dentro de duas políticas, por exemplo: a sobreposição de dois perfis, ou seja, o perfil da política 1 é o mesmo da política 2, dentro deste cenário um provável conflito pode acontecer.

Um conflito em uma política pode aparecer quando múltiplos eventos ou condições conflitantes atuam, tornando-se difícil para o sistema decidir que ação executar em um determinado momento (SHANKAR, RANGANATHAN e CAMPBELL, 2005).

## 4.2 Classificações dos Conflitos

Existem trabalhos que tratam conflitos relacionados a controle de acesso como, em (FERRAILOLO et al., 2001) e (AL-KAHTANI e SANDHU, 2004) e existem trabalhos os quais são direcionados a conflitos em sistemas distribuídos como em (MOFFETT e SLOMAN, 1993) e (LUPU e SLOMAN, 1999).

A classificação de conflitos é referenciada por conflito de modalidade (sintático) e de objetivos (semântico), sendo que conflitos de modalidade podem acontecer quando existem em um repositório, políticas positivas e negativas de autorização e de obrigação sendo políticas de obrigação relacionada a uma ordem, normalmente utilizado nas regras das “ações” que um usuário tem logo políticas de autorização refere-se à permissão de acesso sobre um determinado objeto, ou seja, se é permitido ou negado o acesso, tanto as políticas de obrigação como de autorização são dependentes uma da outra. Conflitos de Objetivos ocorrem quando são envolvidos critérios externos, para a ocorrência de um conflito, por exemplo: limitação de objetos que um usuário pode acessar.

A classificação de conflitos de modalidade é subdividida em duas partes: por conflitos de negação e entre políticas imperativas e de autoridade (autorização e de obrigação). Entretanto a classificação de conflitos de objetivos é subdividida em cinco partes: conflitos de interesse, múltiplos gerentes, deveres, de recursos e de auto-gerência.

É realizada uma classificação de conflitos por (CHARALAMBIDES et al., 2005), no qual trata conflitos relacionados a qualidade de serviços na rede. Os conflitos modelados são conflitos de redundância, de exclusão mútua, por alocação de banda e roteamento. Nesta classificação de conflitos, os conflitos de redundância e de exclusão mútua são aplicados em diversos casos, ou seja, são utilizados independentemente do domínio. Já os demais conflitos referem-se a casos específicos do domínio de aplicação proposto que são serviços de rede, por exemplo: tráfego de rede, alocação de banda etc.

Conflitos de redundância são classificados em duas modalidades: conflitos de modalidade e de objetivo, esta ação justifica-se devido o mesmo ser sintático e semântico, ou

seja, dentro de conflitos de redundância existem duas hipóteses de conflitos: por PCA duplicadas (sintático) e por PCA com conflitos de ações (semântico). Na subseção 4.2.1 será apresentado maiores detalhes sobre esta diferenciação. A seguir são detalhados os conflitos que podem ocorrer em PCA.

#### 4.2.1 Conflitos de Redundância

Conflitos de Redundância podem ocorrer por PCAs duplicadas ou por políticas com conflitos de ações. Se duas políticas são caracterizadas com o mesmo usuário, objeto e ações pode-se dizer que está duplicada, logo não deve existir (CHARALAMBIDES et al., 2005).

Políticas com ações conflitantes dentro do espoco de conflitos de redundância aplicam-se quando existem algumas regras chaves com inconsistência, ou seja, entre uma ou mais PCA, por exemplo: se as regras referentes a um usuário e objeto forem iguais, mas alguma outra regra determinística for conflitante pode-se dizer que ocorre uma inconsistência.

Para ilustrar políticas com conflitos de ações suponha que exista uma política que diz que um usuário pode acessar o EPR das 12:00:00 às 18:00:00. Entretanto outra política diz que esse mesmo usuário deseja acessar o EPR das 13:00:00 às 15:00:00, esta é uma situação de conflitos de ações para um determinado caso.

#### 4.2.2 Conflitos de Negação

Esta modalidade de conflito ocorre quando um usuário é ao mesmo tempo autorizado e negado para realizar um acesso em um determinado objeto, por exemplo, em uma política um supervisor é permitido assinar cheques de pagamento e em outra política ele é negado assinar cheques de pagamento (MOFFETT e SLOMAN, 1993).

Este conflito é muito relevante, pois não há meios para decidir se ação deve ser permitida ou negada.

Conflitos com regras contraditórias podem ocorrer pela simultânea presença de regras positivas e negativas em diferentes políticas levando-se em conta perfis e ações. Para Lupu e



Sloman (1999) conflitos com regras negativas entre PCAs podem existir quando duas ou mais políticas como modalidades opostas referem-se ao mesmo usuário, ação e objeto.

#### 4.2.3 Conflitos entre políticas de obrigação e de autorização

Esta modalidade de conflitos é uma especialização da apresentada acima, ou seja, dos conflitos por negação. Entretanto neste caso o relacionamento em questão que pode ocasionar um conflito são políticas de obrigação e política de autorização.

Com a finalidade de ilustrar esta modalidade de conflito suponha a seguinte situação: o usuário Paulo é obrigado a realizar a ação de “leitura” sobre o objeto “impressora 1” no entanto em outra política o mesmo usuário é negado a realizar a ação de “leitura” nesse mesmo objeto "impressora 1". O caso é o mesmo de conflitos de negação, mas neste caso leva-se em conta políticas de obrigação.

#### 4.2.4 Conflitos por Auto-Gerência

Esta situação é levantada quando um administrador de redes gerencia políticas as quais faz parte, e tem como decorrência manipular ou gerenciar ele mesmo. Para Dursun e Orencik, (2004), não deveria existir PCA autorizando gerentes para modificar políticas dos quais são pertencentes ou atores. Políticas são executadas sobre a supervisão dos Administradores de Redes, então estes usuários não podem ter ações e nem mudar PCA as quais tiver deveres relacionados a eles mesmos.

#### 4.2.5 Conflitos de Múltiplos Gerentes

Para Lupu e Sloman (1999) acessando um objeto em comum, múltiplos usuários podem constituir um conflito de múltiplos gerentes quando as operações de gerencia executadas neste objeto são dependentes uma da outra. Por exemplo, uma política diz que o

usuário “Pedro” somente pode realizar uma determinada ação no objeto (arquivo 1) se um determinado serviço estiver fora do ar, mas e contraste disso existe uma outra política para o usuário “Rodrigo” e que necessita que o serviço esteja no ar para que o mesmo possa acessar o objeto (arquivo 2). Resumindo neste caso os usuários “Pedro” e “Rodrigo” não poderão acessar o sistema simultaneamente, ocasionando um conflito de múltiplos gerentes (MOFFETT e SLOMAN, 1993).

#### 4.2.6 Conflitos de Interesse e de Deveres

Conflitos de interesse podem acontecer quando um usuário associa-se a perfis conflitantes, ganhando assim permissões que podem comprometer a segurança do sistema. Quando um único usuário pertence a dois perfis conflitantes pode haver um acúmulo de ações pelo usuário, conflitando com os interesses da organização.

Conflito de deveres refere-se especificamente a ações conflitantes, por exemplo, um determinado usuário não pode assinar e autorizar um determinado cheque de cobrança.

Os conceitos de conflitos de deveres e de interesses são parecidos, por exemplo, nos modelos baseados no RBAC as ações são atribuídas a perfis logo a única diferença significativa é que conflitos de deveres refere-se as ações conflitantes e conflitos de interesse a perfis conflitantes.

#### 4.2.7 Conflitos de Recursos

Quando duas ou mais PCA entre elas possibilitam o uso de mais objetos que os disponíveis, pode-se afirmar que existe um conflito de recursos. Para Lupu e Sloman (1999), conflitos de recursos ocorrem quando uma quantia de objetos disponíveis é limitada, e as políticas autorizando gerentes a usar um determinado objeto devem de esta forma ter um limitado número de objetos.

Esta modalidade de conflito refere-se quando os objetos a serem gerenciados são do tipo: dinheiro, espaço em disco etc., por exemplo, se uma política possibilitar o uso de

500MB (*Mega Bytes*), de espaço em disco e a quantidade disponível for de somente 300MB, ocorre um conflito de recursos.

### 4.3 Resoluções de Conflitos

Para a resolução de conflitos de interesse é fundamental conhecer as combinações de ações que não podem ser atribuídas a um mesmo usuário. Por exemplo, se política de segurança de uma empresa afirma que um funcionário não pode ser cliente, logo pode-se definir estes dois perfis como conflitantes (NYANCHAMA e OSBORN, 1999).

A separação de responsabilidades (SR) é um princípio da segurança da informação para impor políticas para redução de conflitos de interesse e tem larga aplicação em organizações comerciais, saúde, industriais e do governo. Tem como objetivo evitar que eventuais fraudes venham a ocorrer pela acumulação de poder em uma determinada pessoa, ou seja, previne que dois perfis ditos conflitantes venham a ser definidos para um único usuário.

Existem dois tipos de SR: estática e dinâmica. A SR estática aplica-se quando um determinado usuário é impedido de pertencer a dois perfis conflitantes. A SR dinâmica ocorre quando dois ou mais perfis associados a um mesmo usuário não podem ser ativados simultaneamente em suas seções correspondentes. Resumindo, um usuário pode neste caso ter dois perfis, mas somente pode logar no sistema com um deles. No exemplo abaixo pode-se compreender melhor o funcionamento desses dois atributos.

SR-Estática => 1 - {Médico Assistente, Prescreve, EPR}  
 2 - {Médico Adjunto, Valida Prescrição, EPR}  
 SR-Dinâmica => 1 - {Plantonista, Visualiza, EPR}  
 2 - {Médico Auditor, Edita, EPR}

Na SR-Estática, um Médico Assistente está proibido de vincular-se ao perfil de Médico Adjunto e vice-versa. Na SR-Dinâmica um usuário pode ter vinculado a si próprio o perfil de Plantonista e Médico Auditor, mas não pode utilizar ambos os perfis simultaneamente.

Segundo Zurko e Simon (1997) existem algumas variações da SR-Dinâmica, conhecidos como perfis restritos, que são: SR baseada em objeto, operacional e baseado em história. Entretanto não são relevantes para este trabalho.

Existem duas formas de resolução de conflitos em PCA, sendo a primeira de forma estática, neste caso a verificação de conflitos ocorre em tempo de compilação, ou seja, na hora em que a política for gerada. A verificação de conflitos de forma dinâmica realiza a verificação de conflitos em tempo de execução, ou seja, na hora em que uma política for utilizada.

Zurko e Simon (1997) propuseram alguns exemplos de separação de deveres, os quais são expostos através de algoritmos como pode-se observar um exemplo na Figura 13.

```

Users in Team <Teller>
may perform Action <TellerActions>
on Targets in Scope <TellerTargets>
if <Auditor NOT IN CurrentLabel>

Users in Team <Auditor>
may perform Action <AuditorActions>
on Targets in Scope <AuditorTargets>
if <Teller NOT IN CurrentLabel>

```

Figura 13 – Separação de Deveres 1

No primeiro trecho do algoritmo, um usuário é autorizado a executar ações do perfil de caixa somente se ele não tiver o perfil de Auditor ativo. O segundo trecho informa a mesma situação apresentada anteriormente, mas referindo-se a Auditor.

Outra resolução apresentada para a resolução de Conflitos de Deveres é relatada por Dursun e Orencik (2004), no algoritmo descrito na Figura 14, onde impede que um conflito de deveres aconteça, ou seja, detecta se o mesmo usuário submete e aprova um orçamento. Ele retorna verdadeiro caso exista um conflito e falso caso não for detectado nenhuma anomalia entre duas PCA.

```

boolean EC_DutySeperation(Target budget, Method
submit, Method approve){
    PolicySet policies = Node.getPolicies();
    for actor in Node.getAllActors() {
        if (//actor->budget.submit() or
            budget.approve()
            policies.isPolicyExist(actor,
            budget, submit)
            and
            policies.isPolicyExist(actor,
            budget, approve)){
                return true;
            }//if
        }//for
    return false; // no conflict found
}

```

Figura 14 – Separação de Deveres 2

Em Nyanchama e Osborn (1999) é proposto uma resolução de conflitos através de algoritmos que verificam através de grafos a existência de um determinado conflito. Esta metodologia baseia-se na presença ou não de determinadas arestas que são representadas por perfis. Por exemplo, pode-se observar na Figura 15 uma parte do algoritmo para PCAs com perfis duplicados, onde toma-se como base as arestas para verificar um provável conflito.

```

For all  $r_i, r_j \in \mathcal{R}$  Do
    If Effective( $r_i$ ) = Effective( $r_j$ )
        Then abort (message: Duplicate roles have been created.

```

Figura 15 – Políticas Duplicadas

Na Figura 15 pode-se verificar o algoritmo para detecção de perfis duplicados, onde é realizado uma comparação entre as arestas. Se o valor contido for igual, então é mostrado um aviso para o administrador e a política não é gerada.

Em Dunlop, Indulska e Raymond (2002) são propostos algoritmos para verificação de conflitos no momento em que a PCA é gerada. Para a ocorrência de conflitos a sobreposição de algumas regras é o primeiro indício para a ocorrência de conflitos em PCA. No algoritmo da Figura 16, realiza-se uma busca nas PCA existentes por políticas com sobreposição de regras, com a finalidade de não ser gerada uma PCA que venha a ocasionar algum tipo de conflito.

```

if there exists overlap {
  if policy_types of new_policy and existing_policy
    reflect (O+/O-), (P/F) or (O+/F) {
    with (temporal_classifier) of
      new_policy and existing_policy;
    with conflict profile of the policy_types
      (new_policy.temporal_classifier,
       existing_policy.temporal_classifier)
    from conflict database;
    if conflict profile reveals actual conflict {
      report the actual internal policy conflicts to
        a conflict resolution process;
    } end if – actual conflict
  }
}

```

Figura 16 – Verificação Dinâmica de Conflitos

Como pode-se observar no algoritmo da Figura 16 é elaborada uma comparação da nova política (*new\_policy*) com a política já existente (*existing\_policy*) em busca de políticas com regras sobrepostas, sendo verificados os elementos como: “O” sendo uma obrigação e “P” e “F” uma permissão e negação respectivamente. Verifica-se também se tanto a política que está sendo gerada como a que já existe têm atributos temporais conflitantes (*temporal\_classifier*) e se um perfil (*profile*) conflita com algum já presente no repositório de políticas.

Se ocorrer a sobreposição das seguintes regras: (*temporal\_classifier*), (*profile*) e se as regras forem contraditórias, por exemplo, (O+/O-) é acionado um evento de resolução de conflitos (*resolution process*) que remove a política que está sendo gerada.

#### 4.4 Conclusões Parciais

Neste capítulo foi abordado sobre conceitos de conflitos entre políticas, com enfoque na classificação e resolução de conflitos.

É de suma importância o conhecimento das modalidades de conflitos que possam ocorrer em tempo de geração das PCAs, para verificar quais aplicam-se melhor em determinados casos ou domínios de aplicação. O estudo dos algoritmos para a verificação de conflitos também é um importante fator para se entender melhor como pode-se evitar conflitos em políticas.

Pode-se observar que os conflitos que podem ser evitados em tempo de geração das políticas são: conflitos de interesse, conflitos de negação, conflitos de redundância e conflitos de auto-gerência. Entretanto conflitos de deveres também podem ser tratados em tempo de geração visto que seu conceito é similar ao de conflitos de interesse. As demais modalidades de conflitos aplicam-se em casos específicos de um determinado domínio de aplicação e seu tratamento não é normalmente realizado em tempo de geração das políticas, visto que requerem a análise dinâmica de requisitos externos.

## 5 ESPECIFICAÇÃO DO SGPCA

Neste capítulo, destaca-se a especificação do Sistema Gerenciador de Políticas de Controle de Acesso (SGPCA) proposto neste trabalho. A especificação abrange desde as definições e conceitos até os algoritmos para verificação de conflitos nas políticas. Embora o sistema seja parte integrante do CIBAC (Controle de Acesso Baseado em Informações Contextuais) (SOARES, NUNES e AMARAL, 2006) e seja direcionado para área de saúde, o SGPCA pode ser aplicado em outras áreas e pode ser vinculado a outros mecanismos de controle de acesso.

### 5.1 Definições

Segundo Kudo e Hada (2000), uma autorização deve apresentar pelo menos três parâmetros básicos: usuário, objeto e ação. Um **usuário** pode ser um identificador de um único usuário ou um grupo de usuários; um **objeto** pode ser um documento completo ou partes de um documento; e uma **ação** pode assumir a execução de escrita (gravação), leitura, criação, entre outras, como por exemplo, uma delegação.

Em mecanismos de controle de acesso baseado em contexto, tem-se a utilização de regras que determinam o ambiente em que uma requisição de acesso foi realizada, logo a política deve ser capaz de tratar informações contextuais. O CIBAC é um mecanismo de autorização baseado em informações contextuais, onde informação contextual é qualquer informação que possa ser usada para caracterizar a situação de uma entidade (DEY e ABOWD, 1999). Sendo o SGPCA parte do CIBAC, ele deve então ser capaz de realizar o gerenciamento de PCAs levando-se em conta as informações contextuais como, por exemplo, tempo e localidade.

A modelagem para o SGPCA é realizada com a finalidade de representar as PCAs e algoritmos propostos para a resolução dos conflitos. A terminologia aqui proposta serve para se compreender melhor a dinâmica do funcionamento do SGPCA e facilitar sua posterior implementação.

Esta terminologia baseia-se em modelos de controle de acesso, sendo que foram propostas algumas modificações nos modelos originais como no RBAC, como pode-se



observar nas seções abaixo descritas. Neste projeto a terminologia será utilizada para modelar os algoritmos referentes a controle de acesso.

A especificação deste modelo é baseada em dois modelos fundamentais: RBAC e TRBAC (*Temporal Role Based Access Control*) (BERTINO, BONATTI e FERRARI, 2001), onde o RBAC tradicional fornece a possibilidade de controlar o acesso de usuários baseado nos perfis que estes exercem em uma organização e o modelo TRBAC fornece a funcionalidade de tempo no modelo padrão RBAC.

Visando a especificação do SGPCA, nesta seção apresentam-se as definições e elementos que cercam o funcionamento do sistema:

- USERS: são os usuários que interagem com o sistema;
- ROLES: são uma coleção de perfis que um usuário ou grupo de usuários podem pertencer em uma organização;
- OPS: são as ações que podem ser executadas sobre os objetos;
- OBS: são os objetos do sistema;
- UNIT: fornece o atributo de localidade.

As propriedades do sistema “BEGIN” e “END” são responsáveis pelo intervalo temporal contido no sistema, o qual é baseado no modelo TRBAC e que introduz além do conceito de regras temporais a possibilidade de ativação e desativação de um determinado perfil em sua situação temporal, conforme pode-se observar na Definição 1 a seguir.

O relacionamento entre usuários e perfis é descrito no modelo RBAC pela propriedade “UA” (*User Assignment*) e é definido pelo relacionamento de muitos para muitos no qual é a atribuição de usuários a “perfis”. Já a propriedade “PA” (*Permission Assignment*), atua no relacionamento de perfis a permissões, onde o relacionamento é também de muitos para muitos.

**Definição 1:** Sendo “USERS” o conjunto de todos os usuários, “UNIT” o conjunto de todas as possíveis unidades e “ROLES-U” definida pela tupla (R,U), onde  $R \in \text{ROLES}$  e  $U \in \text{UNIT}$ , tem-se:

$\text{PRMS} \subseteq \text{OPS} \times \text{OBS}$ , onde “PRMS” é uma permissão em um objeto;

$T \subseteq \text{BEGIN} \times \text{END}$ , onde “T” é um período de tempo;

$\text{PA} \subseteq \text{ROLES-U} \times \text{PRMS} \times T$ , refere-se à atribuição de ações para um determinado perfil e unidade em um intervalo de tempo pré-determinado;

$\text{UA} \subseteq \text{USERS} \times \text{ROLES-U}$ , refere-se à atribuição de usuários a perfis em uma unidade.

Como exemplo assumi-se a seguinte situação, um usuário tem uma política associada ao perfil de “Doutor” o qual tem a ação de leitura no “arquivo1”, mas somente se for na unidade de “cardiologia” e das 12:00:00h até as 17:00:00 h.

(ROLES-U, PRMS, T):

PA = ((Doutor,Cardiologia), (arquivo1,ler), (12:00:00,17:00:00)).

No exemplo acima pode-se observar uma política positiva para um determinado perfil.

O modelo tradicional RBAC fornece a possibilidade de controlar o acesso levando-se em conta apenas regras positivas. Entretanto tem-se a necessidade de utilizar regras negativas em sistemas de controle de acesso. Para exemplificar, suponha que um usuário deve ser impedido de acessar a rede interna de uma empresa; a forma normal seria apenas retirar a permissão de acesso deste usuário, o qual poderia associar-se a outro perfil para tentar obter o acesso, mas com a utilização de regras negativas isto não adianta, pois o usuário pode ser negado a realizar um determinado acesso independente de perfil.

Em RB-RBAC-Ve (AL-KAHTANI e SANDHU, 2004) é introduzido o conceito de regras negativas no modelo tradicional do RBAC, mas é estritamente relacionado a perfis. Em nosso modelo, regras negativas podem ser utilizadas tanto para perfis como para usuários, ou seja, pode-se negar o acesso tanto de um usuário como de um perfil. Segue a definição que expressa estes dois casos.

**Definição 2:** Sendo “DUA” (*Denial User Assignment*) e “DRA” (*Denial Role Assignment*) duas propriedades relacionadas à negação de usuários e perfis, ambas associadas a regras temporais “T”, têm-se:

$$DUA \subseteq \text{USERS} \times U \times T;$$

$$DRA \subseteq \text{ROLES-U} \times T.$$

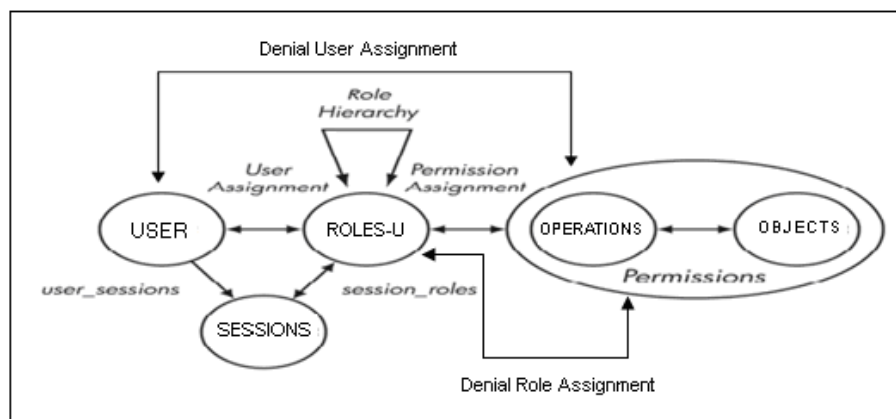


Figura 17 – RBAC com DUA e DRA

Na Figura 17, pode-se observar a estrutura do RBAC padrão, com as funcionalidades “DUA” e “DRA” já incluídas no modelo, onde estas propriedades atuam no relacionamento entre “USERS” – “PRMS” e “ROLES” – “PRMS”.

Com o objetivo de facilitar o entendimento deste modelo, considere as seguintes regras:

DUA: (Usuário1, Anestesia, 12:00:00,17:00:00)

DRA: (Doutor, Cardiologia, 15:00:00, 18:00:00)

A primeira estabelece que “Usuário 1” não está autorizado a executar ações na unidade de “Anestesia” das 12:00:00 h às 17:00:00 h, enquanto que a segunda estabelece que o perfil “Doutor” não está autorizado a executar ações na unidade de “Cardiologia” das 15:00:00 h às 18:00:00 h. Em outras palavras, DUA possibilita a negação de usuários e DRA de perfis. Entretanto, deve-se observar que o uso de regras negativas pode ocasionar conflitos entre as políticas.

## **5.2 Requisitos do Sistema**

Para utilizar o SGPCA no âmbito do HUSM, é necessário especificar alguns requisitos que cercam o funcionamento do hospital de maneira geral.

Existem dois requisitos que são necessários para o SGPCA, que são a hierarquia de perfis de acordo com a estrutura organizacional do HUSM e as suas diversas unidades existentes. A figura 18 apresenta a hierarquia de perfis do HUSM e a figura 19 suas unidades, ambas recuperadas da estrutura hospitalar do HUSM.

Usuário	Profissional	Médico	Residente				
			Cirurgião				
			Anestesista				
			Hematologista				
		Paramédico	Enfermeiro	Auxiliar de Enfermagem			
			Psicólogo				
			Fisioterapeuta				
			Biólogo				
			Fonoaudiólogo				
			Dentista				
			Assistente Social				
			Professor de Educação Física				
			Físico				
			Bioquímico				
			Biomédico				
			Farmacêutico	Auxiliar de Farmácia			
			Nutricionista	Auxiliar de Nutrição			
			Auxiliar Técnico				
		Administrador Executivo	Diretor				
			Técnico Administrativo	Auxiliar Administrativo	Escriturário		
					Secretário		
					Auxiliar de Registro de Saúde	Analista de Registro de Saúde	
				Técnico em Informações Médico Hospitalares	Analista de Informações Médico Hospitalares		
			Operador de Tele-atendimento				
			Administrador de RH		Analista de RH	Auxiliar de RH	
			Administrador Financeiro	Faturista	Analista de Faturamento		
				Técnico Contábil		Analista Contábil	
Técnico de Informática	Analista de Informática		Analista de Suporte				
			Analista Desenvolvedor				
	Administrador de Banco de Dados						
Pesquisador	Pesquisador Clínico						
Paciente							
Estudante	Graduando						
	Pós-Graduando						

Figura 18 – Hierarquia de perfis para o HUSM

Cabe ressaltar que a construção de uma hierarquia de perfis configura-se em um processo contínuo que evolui de acordo com as necessidades da organização e com as

circunstâncias encontradas no ambiente em que se insere, assim o SGPCA deve possibilitar a realização de modificações de forma dinâmica.

<ul style="list-style-type: none"> <li>• Unidade de Anestesia</li> <li>• Unidade de Angiologia e Cirurgia Vascular</li> <li>• Unidade de Cardiologia               <ul style="list-style-type: none"> <li>- Ambulatório</li> <li>- Internação</li> <li>- Unidade de Cardiologia Intensiva</li> <li>- Setor de Reabilitação Cardiológica</li> </ul> </li> <li>• Unidade de Cirurgia Cardíaca</li> <li>• Unidade de Cirurgia de Cabeça e Pescoço               <ul style="list-style-type: none"> <li>- Ambulatório de Cirurgia de Cabeça e Pescoço</li> <li>- Ambulatório de cirurgia do trauma buco-maxilo-facial</li> </ul> </li> <li>• Unidade de Cirurgia Geral               <ul style="list-style-type: none"> <li>- Ambulatório de cirurgia de hérnia e esôfago</li> <li>- Ambulatório de cirurgia de estômago e duodeno</li> <li>- Ambulatório de pós-operatório de cirurgia de trauma</li> <li>- Ambulatório de cirurgia do fígado, ves biliar e pâncreas</li> </ul> </li> <li>• Unidade de Cirurgia Torácica</li> <li>• Unidade de Dermatologia</li> <li>• Unidade de Endocrinologia e Nutrição</li> <li>• Unidade de Gastroenterologia</li> <li>• Unidade de Ginecologia               <ul style="list-style-type: none"> <li>- Ambulatórios de ginecologia geral</li> <li>- Ambulatório de colposcopia</li> <li>- Ambulatório de mastologia</li> <li>- Ambulatório de címetrio</li> <li>- Ambulatório de endócrino-ginecologia</li> <li>- Ambulatório de uro-ginecologia</li> <li>- Ambulatório de onco-ginecologia</li> <li>- Ambulatório de reconstrução de mama</li> </ul> </li> <li>• Unidade de Hemato – Oncologia</li> <li>• Unidade de Hemodinâmica</li> <li>• Unidade de Infectologia               <ul style="list-style-type: none"> <li>- Ambulatório</li> <li>- Internação</li> <li>- Hospital-dia</li> </ul> </li> <li>• Unidade de Nefrologia               <ul style="list-style-type: none"> <li>- Ambulatório</li> <li>- Unidade de diálise peritoneal</li> <li>- Unidade de hemodiálise</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>• Unidade de Neurologia</li> <li>• Unidade de Obstetrícia               <ul style="list-style-type: none"> <li>- Ambulatório de Pré-natal</li> <li>- Ambulatório de Pós-natal de alto risco</li> <li>- Internação</li> </ul> </li> <li>• Unidade de Oftalmologia</li> <li>• Unidade de Otorrinolaringologia               <ul style="list-style-type: none"> <li>- Ambulatório</li> </ul> </li> <li>• Unidade de Pediatria               <ul style="list-style-type: none"> <li>- Ambulatório geral</li> <li>- Ambulatório de pediatría</li> <li>- Ambulatório PASCAR</li> <li>- Ambulatório de diabetes infantil</li> <li>- Ambulatório de cirurgia pediátrica</li> <li>- Ambulatório de infectologia pediátrica</li> </ul> </li> <li>• Unidade de Pneumologia               <ul style="list-style-type: none"> <li>- Ambulatório</li> <li>- Internação</li> </ul> </li> <li>• Unidade de Proctologia               <ul style="list-style-type: none"> <li>- Ambulatório</li> <li>- Internação</li> <li>- Colonoscopia</li> </ul> </li> <li>• Unidade de Quimioterapia</li> <li>• Unidade de Radioterapia               <ul style="list-style-type: none"> <li>- Ambulatório de pacientes novos (inclui o planejamento do tratamento)</li> <li>- Ambulatório de pacientes em tratamento radioterápico</li> <li>- Ambulatório de revisão pós-tratamento radioterápico</li> </ul> </li> <li>• Unidade de Reumatologia</li> <li>• Unidade de Traumatologia</li> <li>• Unidade de Urologia               <ul style="list-style-type: none"> <li>- Ambulatório</li> <li>- Internação</li> <li>- Cirurgião</li> </ul> </li> <li>• Unidade de Terapia Intensiva para Adultos, Recém-nascidos, Pediatria e Embeologia</li> </ul>
---	--

Figura 19 – Unidades do HUSM

### 5.3 Especificação dos Algoritmos

As modalidades de conflitos selecionadas para serem abordadas no SGPCA foram: conflitos de Redundância, Negação e de Interesse. Estas modalidades de conflitos foram escolhidas devido poderem ser tratadas em tempo de geração das políticas, tendo em vista que o objetivo principal deste trabalho é a da geração de políticas livre de conflitos. Na Figura 20 segue um esquema dos conflitos implementados.

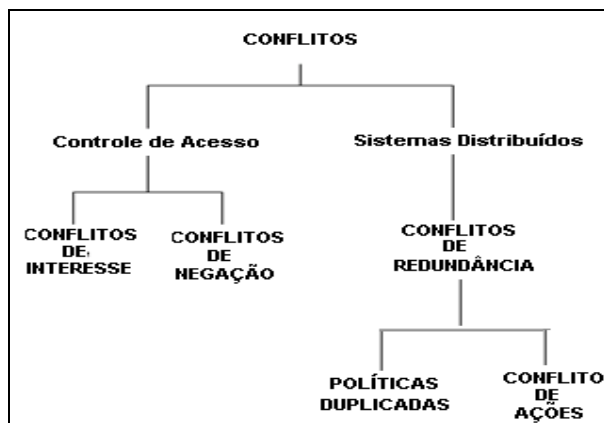


Figura 20 – Classificação dos Conflitos

Nas próximas seções é apresentado os detalhes de como cada conflito é tratado.

### 5.3.1 Conflitos de Negação

Com a finalidade de impedir que conflitos de negação venham a ocorrer entre as PCAs, propõe-se um algoritmo que além de impedir que uma política venha a ser gerada de forma conflitante, também aponte que política e regra está conflitando. A Figura 21 apresenta o algoritmo.

```

Algoritmo1 Controle de Políticas Negativas
Entrada: política sendo gerada P, repositório de políticas PR contendo PAs, DUAs e DRAs
Saída: Mensagem de conflito ou Política armazenada no Repositório.
1 For each PA ∈ PR
2   If (PA = Positiva) And (P = Negativa) Or (PA = Negativa) And (P = Positiva)
3     If (PA.R = P.R And PA.U = P.U And PA.T ∩ P.T ≠ ∅) Or
4       (PA.USR = P.USR And PA.U = P.U And PA.T ∩ P.T ≠ ∅)
5         then (P conflita com PA e a regra conflitante é, )
6         else PR ← P
7       EndIf
8     EndIF
9   End For
10 End Algoritmo1
  
```

Figura 21 – Algoritmo (Conflitos por Regras Negativas)

Este algoritmo atua sempre que uma PCA for gerada tanto com regras positivas como negativas, ou seja, se uma política positiva for gerada o SGPCA verifica se há alguma política com regras negativas para um mesmo usuário ou perfil e unidade, caso afirmativo a política não é gerada.

O algoritmo tem como entrada a política a ser gerada “P” e como saída uma mensagem de ocorrência de um conflito ou de política armazenada no repositório de políticas (PR), tanto para políticas positivas como políticas negativas.

Na linha 2 pode-se observar a existência de uma condição, no qual é verificado se as políticas podem conflitar, ou seja, se a política que esta sendo gerada é positiva e a que esta no repositório de políticas é negativa, ou vice-versa.

A condição das linhas 3 e 4, verifica se há interseção temporal em políticas referentes ao mesmo usuário/unidade ou perfil/unidade. Se houver conflito uma mensagem de conflito é apresentada apontando que política está conflitando com a que está sendo gerada e quais regras são conflitantes. Não havendo conflito a política é gerada e enviada para o repositório das políticas.

### 5.3.2 Conflitos de Interesse

Um caminho para evitar conflitos de interesse é pela Separação de Responsabilidades, que age no relacionamento entre usuários e perfis, o qual previne que um usuário venha a ter dois ou mais perfis ditos conflitantes. Para exemplificar esta situação vamos supor o seguinte caso: um Administrador de Redes está gerando uma PCA para definir que o usuário Pedro, o qual já tem atribuído o perfil de Enfermeiro na Cardiologia venha a ter o perfil de Médico na mesma unidade hospitalar. Deve-se evitar que isto ocorra, pois não deve ser permitido que um usuário venha a ter dois perfis conflitantes em uma mesma unidade.

O algoritmo proposto é definido, por exemplo, para evitar que um administrador que está gerando as PCAs venha a atribuir perfis conflitantes. Os perfis e unidades definidos como conflitantes são representadas pela propriedade SSI. (*Static Separation of Interest*), e são armazenados num arquivo.

Para o controle de conflitos de interesse é proposto um algoritmo que evita que um usuário seja atribuído a dois perfis conflitantes, como pode-se observar na Figura 22. Inicialmente o algoritmo verifica (linha 2) se tanto a política que esta sendo gerada como a

que está armazenada no repositório de políticas são positivas. Caso as políticas sejam positivas, a condição nas linhas 3 e 4 realiza as seguintes comparações: verifica se a PCA que está sendo gerada, contém o mesmo perfil e unidades do referenciado na propriedade SSI1.R1, o qual é o primeiro perfil e unidade conflitantes. Posteriormente verifica-se se o segundo perfil e a segunda unidade conflitantes representados pela propriedade SSI1.R2, são os mesmos da PCA que está no repositório de políticas; após a comparação dos perfis e unidades deve-se verificar o usuário, ou seja, para assegurar que o perfil da política que está sendo gerada é o mesmo da presente no repositório. O algoritmo aponta a política e o perfil que está conflitando entre as políticas.

```

Algoritmo 2 Controle de Conflitos de Interesse
Entrada: Política sendo gerada P, repositório de políticas PR.
Saída: Mensagem de conflito ou Política armazenada no Repositório
1 For each PA ∈ PR
2   If (PA = Positiva) And (P = Positiva)
3     If (SSI1.R1 = P.ROLE-U And SSI1.R2 = P.ROLE-U) And
4       (PA.USR = P.USR)
5       then P conflita com PA e o Perfil conflitante é R)
6       else PR ← P
7     EndIf
8   EndIf
9 EndFor
10 End Algoritmo 2

```

Figura 22 – Algoritmo (Conflitos de Interesse)

Na Figura 23 pode-se observar um exemplo prático do algoritmo, que na primeira linha verifica se o perfil de “Médico” e a unidade de “Cardiologia” são os mesmos do referenciado na política a ser gerada, posteriormente na linha 2 é verificado se o perfil e a unidade do repositório das políticas correspondem ao presente no arquivo de conflitos, por fim na linha 3 é realizado um controle para ver se o usuário da política que está sendo gerada é o mesmo do repositório das PCA. Se todas estas regras forem verificadas um conflito de interesse é indicado.



// ARQUIVO DE CONFLITOS		// POLÍTICA A SER GERADA
1 - MÉDICO, CARDIOLOGIA	=	PEDRO, MÉDICO, CARDIOLOGIA
// ARQUIVO DE CONFLITOS		// REPOSITÓRIO DE POLÍTICAS
2 - ENFERMEIRO, CARDIOLOGIA	=	PEDRO, ENFERMEIRO, CARDIOLOGIA
// REPOSITÓRIO DE POLÍTICAS		// POLÍTICA A SER GERADA
3 - PEDRO	=	PEDRO
4 - CONFLITO		

Figura 23 – Exemplo - Conflitos de Interesse

### 5.3.3 Conflitos de Redundância

Nesta seção será abordado sobre algoritmos para a verificação de conflitos de redundância, sendo este dividido em duas subcategorias: políticas duplicadas e por conflitos de ações.

#### 5.3.3.1 Políticas Duplicadas

Para prevenir esta modalidade de conflito, o algoritmo da Figura 24 controla o aparecimento desta modalidade de conflito de redundância, não deixando gerar PCAs duplicadas. As propriedades levadas em consideração neste caso são: usuários, objetos, perfis, unidades e ações.

```

Algoritmo3 Controle de Conflitos de Redundância (Políticas Duplicadas)
Entrada: política sendo gerada P, repositório de políticas PR
Saída: Mensagem de conflito ou Política armazenada no Repositório
1 For each PA ∈ PR
2   If ((PA = Negativa) And (P = Negativa)) And ((PA = Positiva) And (P = Positiva))
3     If (PA.USR = P.USR And PA.O = P.O And PA.R = P.R) And
4       (PA.U = P.U And PA.A = P.A And PA.TP = P.TP)
5         then (P conflita com PA)
6         else PR ← P
7     EndIf
8   EndIf
9 EndFor
10 End Algoritmo3

```

Figura 24 – Algoritmo (Políticas Duplicadas)

No algoritmo da Figura 24 a propriedade “USR” representa o usuário do sistema, “O” o objeto a ser protegido, “R” o perfil, “U” a propriedade relacionada à unidade, “T” o intervalo temporal e “A” a ação que este objeto pode ter. A etapa inicial do algoritmo assemelha-se a dos apresentados nas seções anteriores, tendo como entrada a PCA a ser gerada referenciada pela propriedade “P”. Este algoritmo atua tanto com políticas positivas como negativas, evitando duplicação em ambos os casos como pode-se observar na linha 2. Pode-se observar também a existência de uma condição nas linhas 3 e 4 onde deve-se observar a comparação das propriedades das PCA que estão sendo geradas com as presentes no repositório de políticas que já foram geradas. Caso as políticas sejam idênticas a política não é gerada e uma mensagem de conflito é mostrada, indicando qual a PCA está conflitando.

### 5.3.3.2 Conflito por Redundância de Ações

Conflito por redundância de ações no SGPCA ocorre quando duas políticas positivas, ou duas negativas, para um mesmo usuário/objeto apresentam intersecção temporal. O algoritmo para verificação de conflitos de ações da Figura 25 considera os seguintes elementos: usuário “USR”, objeto “O” e o intervalo temporal “T”.

```

Algoritmo 4 Controle de Conflitos de Redundância (Conflito de Ações)
Entrada: política sendo gerada P, repositório de políticas PR
Saída: Mensagem de conflito ou Política armazenada no Repositório
1 For each PA ∈ PR
2     If ((PA = Positiva) And (P = Positiva)) And ((PA = Negativa) And (P = Negativa))
3         If (PA.USR = P.USR And PA.O = P.O And PA.T ∩ P.T ≠ ∅)
4             then (P conflita com PA)
5             else PR ← P
6         EndIf
7     EndIf
8 EndFor
9 End Algoritmo 4

```

Figura 25 – Algoritmo (Conflito de Ações)

O algoritmo da Figura 25, inicialmente verifica se ambas políticas são positivas ou negativas. Em seguida, na linha 3, verifica se as propriedades usuário (USR) e objeto (O) são iguais tanto na política que vai ser gerada (P) como na presente no repositório das políticas (PA). Caso ocorra um conflito o algoritmo além de evitar que a política seja gerada, indica qual política conflita. O SGPCA pode resolver este conflito questionando o administrador de políticas sobre o desejo de realizar a união da política existente com a que está sendo criada, uma vez que elas apenas divergem no período de aplicação.

#### 5.4 Projeto de Interface

As informações contidas nas políticas servem como base para o projeto da interface do sistema. Nesta subseção serão definidos os detalhes das informações presentes nas políticas, bem como as regras que devem aparecer nas interfaces dos gerenciadores de políticas de controle de acesso.

Para Strassner et al. (2001) as interfaces dos gerenciadores de políticas devem estar condizente com o objetivo da empresa. Entretanto existe um padrão de regras que são indicadas que estejam presentes em interfaces gráficas de gerenciadores de políticas de controle de acesso, de acordo com a IETF que são: perfil, ação, condição (ex: condição temporal), conforme o objetivo da empresa, e um objeto.

Para que as informações das políticas satisfaçam os requisitos de uma determinada empresa ou instituição deve-se extrair o objetivo da empresa e transformar este objetivo em

regras de acesso que são aplicados nos objetos gerenciados (POLYRAKIS e BOUTABA, 2002).

**(Objetivos → Regras → Objetos gerenciados)**

De acordo com os modelos baseados no RBAC os elementos que devem ser levados em consideração em mecanismos de controle de acesso são: usuários, perfis, ações e objetos, logo estes elementos devem estar presentes na interface para a geração das políticas.

Em Ferreira, Santos e Júnior (2004), tem-se a definição de políticas em XACML para o estabelecimento de VPNs, onde as informações que estão presentes em sua interface são: servidor de origem e destino, utilização ou não de criptografia e horário permitido de acesso.

Na Figura 26 pode-se observar a representação da arquitetura da interface, onde são expostos a representação dos campos e componentes que constituem a interface do SGPCA.

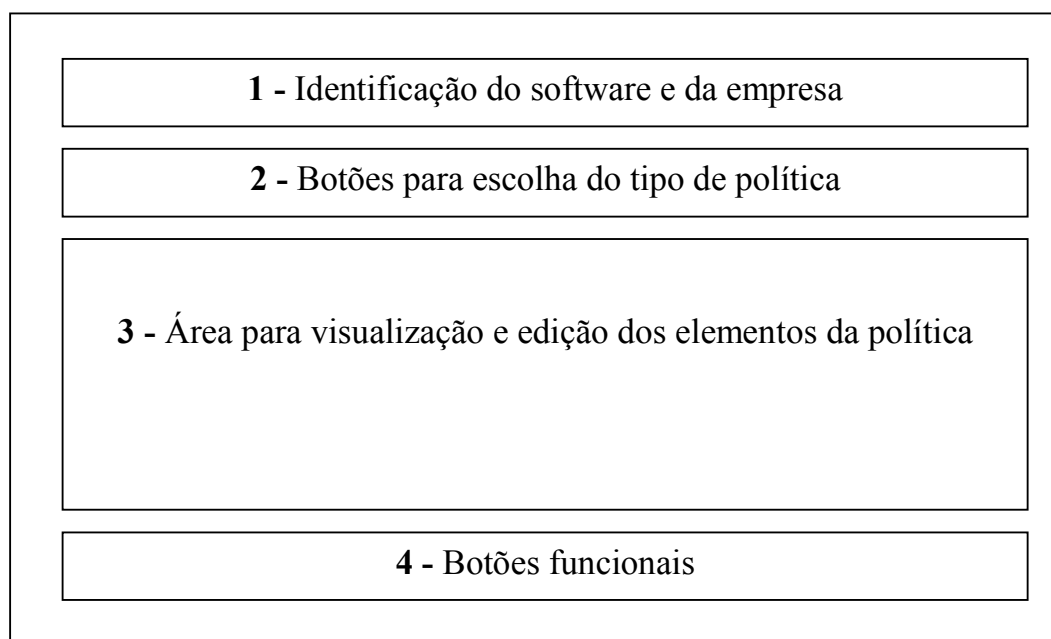


Figura 26 – Arquitetura da Interface

No item 1 da Figura 26 tem-se a identificação do software junto com as logomarcas da empresa a qual destina-se o software que é o HUSM, em conjunto encontra-se também a logomarca da GMICRO, laboratório onde foi realizado o desenvolvimento deste sistema. Os

botões que definem o tipo de política a ser gerada são abordados no item 2, os quais são nomeados por: “Negar um Usuário”, “Negar um Perfil” e “Configura Conflitos”.

O item 3 apresenta os campos os quais são destinados para a geração e edição das políticas. No caso de edição de uma política já criada este servirá também para a visualização das regras já cadastradas, possibilitando assim efetuar alterações. Neste item tem-se os seguintes elementos: “Nome do usuário”, “objeto”, “perfil”, “hora inicial”, “hora final”, “ação” e “unidade” sendo que os campos “perfil”, “objeto” e “unidade” são modelados de acordo com a estrutura organizacional do HUSM, o qual foi apresentado na seção 5.2.

No item 4 são definidos os botões, os quais realizaram ações sobre as regras definidas no item anterior. Neste item tem-se os seguintes botões: “Gerar”, “Limpar”, “Editar” e “Apagar”, o botão “Gerar” possibilita a geração das políticas em uma determinada pasta, o botão “Limpar” tem como função limpar os campos da interface, o botão “Editar” possibilita a edição de uma política já armazenada no repositório de políticas, por fim o botão “Apagar” deleta uma política presente no repositório de políticas.

Pode-se concluir que o projeto da interface para gerenciadores de PCAs devem refletir os interesses da organização no qual serão aplicados. Entretanto pode-se observar que existe uma padronização das regras existentes nas interfaces, as quais foram aplicadas neste trabalho, o qual será melhor explorado no próximo capítulo, nos testes de conformidade.

## **5.5 Conclusões Parciais**

Este capítulo apresentou algumas definições e requisitos necessários para a especificação do SGPCA. Definiu-se seus elementos essenciais e propriedades, bem como o seu modelo de controle de acesso, que se apresenta como uma extensão ao modelo RBAC. Os requisitos salientam que a hierarquia de perfis e o relacionamento de perfis com unidades (ROLE-U) são importantes no contexto de controle de acesso ao prontuário eletrônico do paciente.

Finalmente, foram propostos algoritmos de verificação de conflitos para regras negativas, conflito de interesse e conflitos de redundância. Tais algoritmos possibilitam que um editor de políticas verifique conflitos em tempo de criação das políticas.

## 6 IMPLEMENTAÇÃO DO SGPCA

Este capítulo trata sobre a implementação do SGPCA. Para a implementação utiliza-se a API `sun.xacml`, a qual é uma biblioteca destinada a dar suporte a linguagem XACML, mostra-se também aspectos de como é realizada a implementação de políticas em Java.

Este capítulo discute inicialmente a arquitetura do CIBAC (seção 6.1) e em seguida apresenta os aspectos conceituais da implementação do SGPCA (seção 6.2). Na (seção 6.3) é discutido a dinâmica do funcionamento do SGPCA. Os testes finais são apresentados na (seção 6.4).

### 6.1 Arquitetura do CIBAC

A arquitetura do CIBAC é baseada em serviços, ou seja, esta arquitetura é formada por três Web Services (Serviço de Administração, Serviço de Decisão e Serviço de Autorização), também por um banco de dados com informações contextuais e PCAs.

O Serviço de Administração implementado no CIBAC tem métodos para manipulação de informações contextuais no banco de dados. Já o Serviço de Decisão é o PDP, o qual tem a finalidade de escolher uma determinada política e fornecer uma resposta. O Serviço de Autorização é o PEP propriamente dito, onde tem a função de gerar uma requisição na linguagem de política.

Para a adaptação ao SGPCA foram modificados alguns serviços já propostos no CIBAC como: o PEP e o PDP. Uma das funções do PEP é montar uma requisição em XACML, para realizar esta montagem foi necessário realizar algumas alterações para que o mesmo contemple diferentes tipos de *targets*, visto que no SGPCA tem-se três tipos de políticas diferentes sendo elas: positivas, negativas para usuário e negativas para perfís, logo a requisição de acesso deve adaptar-se a estas políticas.

Também foram realizadas mudanças no PDP com a finalidade de que caso tenha-se duas políticas, sendo uma positiva e outra negativa, a mais forte sempre será a negativa, conforme descrito em (BERTINO, SAMARATI e JAJODIA, 1997).

## 6.2 Implementação do SGPCA

Este sistema, foi desenvolvido utilizando a linguagem de programação Java junto com a API `sun.xacml`, o qual fornece métodos para a manipulação de arquivos no formato XML com características XACML. Em conjunto foi utilizado o editor NetBeans 5.5, como ferramenta de desenvolvimento, pela sua facilidade de manipulação e criação das interfaces.

Nas próximas subseções serão apresentados aspectos técnicos sobre: a formatação das políticas em XACML para o SGPCA; de como é realizado o mapeamento das políticas utilizando a API `sun.xacml` e de como ela é realizada e tratada no JAVA. O conhecimento dos métodos desta API é fundamental para realizar a implementação dos conflitos.

### 6.2.1 Representação de Políticas XACML no SGPCA

O SGPCA utiliza XACML para a codificação das políticas de controle de acesso. A razão de sua escolha foi sua grande flexibilidade e extensibilidade, que permitem a definição de requisitos da aplicação com sintaxe e semânticas simples, além de ser largamente suportada por diversos equipamentos, já que é derivada de uma linguagem padronizada, no caso XML (FERREIRA, SANTOS e JÚNIOR, 2004).

Nesta seção será dado enfoque na formatação das políticas XACML para o SGPCA.

#### 6.2.1.1 Regras Positivas

A forma de estruturação de uma PCA na linguagem XACML, varia na sua codificação no SGPCA, ou seja, para PCAs positivas a estrutura é diferente das negativas, levando-se em conta os atributos os quais são definidos.

Na Figura 26, pode-se visualizar a PCA para o SGPCA, onde se tem o elemento *Target* que contém os elementos *Subjects* e *Resources*, nos quais possui os atributos “Diretor” como perfil e “Cirurgia Cardíaca” como unidade. Por fim o objeto “Prontuário”. Verifica-se

também a presença de um segundo *Target* dentro da tag *Rule*, onde se tem o *Subject* que representa o usuário, neste caso com o nome de “Jorge”.

```

- <Policy xmlns="urn:oasis:names:tc:xacml:1.0:policy" xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:ctx="urn:oasis:names:tc:xacml:1.0:context" PolicyId="9.xml" RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-
  combining-algorithm:permit-overrides">
- <Target>
- <Subjects>
- <Subject>
- <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:regexp-string-match">
  <SubjectAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string" AttributeId="_perfis" />
  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Diretor</AttributeValue>
</SubjectMatch>
- <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
  <SubjectAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string" AttributeId="_setor" />
  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Cirurgia Cardíaca</AttributeValue>
</SubjectMatch>
</Subject>
</Subjects>
- <Resources>
- <Resource>
- <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
  <ResourceAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string"
  AttributeId="_identificador" />
  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Prontuário</AttributeValue>
</ResourceMatch>
</Resource>
</Resources>
</Target>
- <Rule RuleId="Access" Effect="Permit">
- <Target>
- <Subjects>
- <Subject>
- <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
  <SubjectAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string" AttributeId="_nome" />
  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Jorge</AttributeValue>
</SubjectMatch>
</Subject>
</Subjects>
</Target>
- <Condition FunctionId="http://research.sun.com/projects/xacml/names/function#time-in-range">
- <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
  <EnvironmentAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#time"
  AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time" />
</Apply>
  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">11:00:00</AttributeValue>
  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">12:00:00</AttributeValue>
</Condition>
</Rule>
- <Rule RuleId="Access" Effect="Deny" />
- <Obligations>
- <Obligation ObligationId="Resposta" FulfillOn="Permit">
  <AttributeAssignment AttributeId="acoes"
  DataType="http://www.w3.org/2001/XMLSchema#string">Ler</AttributeAssignment>
</Obligation>
</Obligations>
</Policy>

```

Figura 27 – Política gerada parte 1

Também dentro da tag *Rule*, tem-se um atributo contendo os dados temporais, neste caso das 11:00:00h até às 12:00:00h. Por fim, a ação é verificada pelo elemento *Obligations* no qual tem o valor de “Ler”.



Com a análise geral de toda a PCA, pode-se dizer que um usuário com o perfil de “Diretor” estando na unidade de “Cirurgia Cardíaca”, pode ler o EPR somente no horário das 11:00:00h até as 12:00:00h.

A utilização de dois elementos *Target* justifica-se devido à propriedade DUA necessitar consultar o nome do usuário nas políticas para que a mesma possa ser funcional.

#### 6.2.1.2 Regras Negativas

Regras negativas no SGPCA são modeladas pelas propriedades “DUA” e “DRA”, sendo a primeira relacionada à negação de usuários e a última relacionada a negação de perfis. Estas propriedades adicionam atributos diferenciados das relacionadas nas regras positivas, considerando as necessidades do sistema.

A PCA para “DUA” contém quatro atributos: unidade, usuário, hora inicial e hora final, visto a utilização de dois elementos *Target*, uma referenciando a unidade e outra o usuário. Por fim tem-se o elemento *Condition* que carrega os atributos de tempo sendo uma condição necessária para a negação de um usuário.

A Figura 27 exemplifica uma PCA destinada a negação de usuários, onde, por exemplo, o usuário denominado de “Pedro” na unidade de “Anestesia” tem seu acesso negado no período das 03:00:00h às 10:00:00h. O elemento *Deny* presente na tag *RuleId* identifica se a regra é negativa.

```

<?xml version="1.0" encoding="ISO-8859-1" ?>
- <Policy xmlns="urn:oasis:names:tc:xacml:1.0:policy" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:ct="urn:oasis:names:tc:xacml:1.0:context" PolicyId="teste22.xml" RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:ru
  combining-algorithm:permit-overrides">
- <Target>
- <Subjects>
- <Subject>
- <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
- <SubjectAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string" AttributeId="_setor" />
- <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Anestesia</AttributeValue>
- </SubjectMatch>
- </Subject>
- </Subjects>
- </Target>
- <Rule RuleId="Access" Effect="Deny">
- <Target>
- <Subjects>
- <Subject>
- <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
- <SubjectAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string" AttributeId="_nome" />
- <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Pedro</AttributeValue>
- </SubjectMatch>
- </Subject>
- </Subjects>
- </Target>
- <Condition FunctionId="http://research.sun.com/projects/xacml/names/function#time-in-range">
- <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
- <EnvironmentAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#time"
  AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time" />
- </Apply>
- <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">03:00:00</AttributeValue>
- <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">10:00:00</AttributeValue>
- </Condition>

```

Figura 28 – Regras Negativas DUA.

A funcionalidade de negar um perfil é representada pela propriedade “DRA”, que contém em sua estrutura quatro atributos, que são usados para realizar uma negação de um determinado perfil, são eles: perfil, unidade, hora inicial e hora final. Neste caso existe somente um elemento *Target* que contém o perfil e unidade, e por fim o elemento *Condition* no qual os atributos de tempo estão presentes.

Pode-se observar na Figura 28, a PCA gerada representando a propriedade DRA, onde o perfil de “Diretor” tem seu acesso negado na unidade de “Anestesia” das 12:00:00h às 17:00:00 h.

```

- Policy xmlns:urn:oasis:names:tc:secml:1.0:policy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:cts="urn:oasis:names:tc:secml:1.0:common" PolicyId="t1.xml" RuleCombiningAlgId="urn:oasis:names:tc:secml:1.0:rule-
  combining-algorithm:permitt-overrides"
- Target
- Subjects
- Subject
- SubjectMatch MatchId="urn:oasis:names:tc:secml:1.0:function:regexp-string-match"
  SubjectAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string" AttributeId="_perfil"
  AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string" Director AttributeValue
  SubjectMatch
- SubjectMatch MatchId="urn:oasis:names:tc:secml:1.0:function:string-equal"
  SubjectAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string" AttributeId="_setor"
  AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string" Anestesia AttributeValue
  SubjectMatch
  Subject
  Subjects
  Target
- Rule RuleId="Access Effect Deny"
- Condition FunctionId="http://research.sun.com/projects/secml/nemias/function#time-in-range"
- Apply FunctionId="urn:oasis:names:tc:secml:1.0:function:time-one-and-only"
  EnvironmentAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#time"
  AttributeId="urn:oasis:names:tc:secml:1.0:environment:current-time"
  Apply
  AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time" 12:00:00 AttributeValue
  AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time" 17:00:00 AttributeValue
  Condition
  Rule
  Rule RuleId="Access Effect Permit"
  Policy

```

Figura 29 – Regras Negativas DRA

## 6.2.2 Representação de Políticas no código do SGPCA

No SGPCA, políticas são representadas como objetos Java, com a finalidade de realizar a manipulação de suas regras. Por exemplo, o elemento *Target* de uma política que contém os objetos e perfis devem ser mapeadas e lidas por uma variável para que estes dados possam ser usados para a edição e verificação de conflitos de uma PCA.

Na Figura 29, pode-se observar a leitura dos elementos perfil e unidade da PCA. Primeiramente cria-se um objeto no qual irá conter os atributos do *Subjects* do primeiro *Target* da PCA, posteriormente, estes elementos são armazenados nas variáveis perfil e unidade respectivamente.

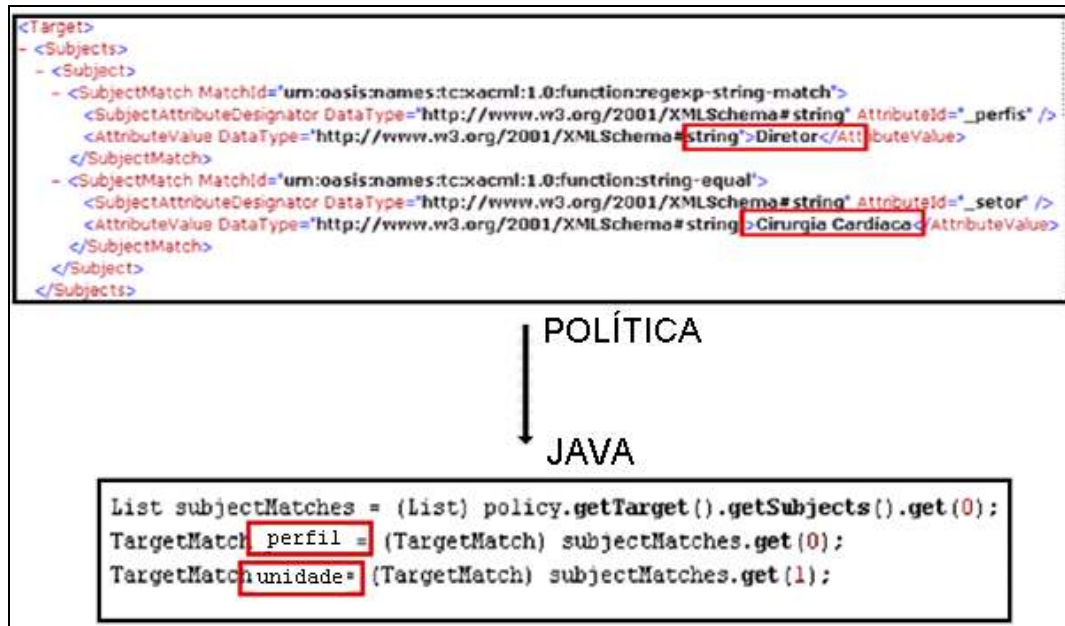


Figura 30 – Código Target 1.

Através das variáveis perfil e unidade como mostra a Figura 29 pode-se realizar qualquer atividade com os atributos das políticas referentes ao primeiro *Target*. O mapeamento dos outros elementos da política é realizado de forma parecida, como será mostrado a seguir.

Em uma PCA descrita em XACML, para dar a possibilidade de suportar a propriedade DUA deve-se criar outro elemento *Target* como visto na subseção 6.2.1.1. Pode-se observar na Figura 30, o procedimento para realizar o mapeamento do segundo *Target* que contém o nome do usuário, no qual deve-se navegar entre as tags da política através da linguagem Java até chegar ao elemento em questão.

```

Target target2 = rule0.getTarget();
if (target2 != null) { // SUJEITO
  subjectMatches = (List) target2.getSubjects().get(0);
  TargetMatch nome = (TargetMatch) subjectMatches.get(0);
}

```

Figura 31 – Código Target 2.

Observe o mapeamento do segundo *Target*, o qual contém o nome do usuário. Ele encontra-se dentro do elemento *Subjects* correspondente a regra descrita na política, sendo o primeiro elemento presente referenciado por `get(0)`.

Cada elemento chave na PCA é representado no java por um método fornecido pela API `sun.xacml`, dando assim a possibilidade de realizar a leitura dos elementos nos quais deseja-se realizar as manipulações. No SGPCA, a leitura dos elementos das PCAs serve para realizar a edição de PCAs realizando assim as alterações ou modificações nas políticas. Outra utilidade é para dar a possibilidade de realizar o controle de conflitos na geração de políticas.

### 6.3 Dinâmica de Funcionamento do SGPCA

Nas próximas subseções serão mostrados detalhes do funcionamento do SGPCA, junto com sua interface de geração de políticas e verificação de conflitos. Na subseção 6.3.1 será abordado sobre a dinâmica de funcionamento referente a políticas positivas. Na subseção 6.3.2 o foco será nas políticas negativas. Por fim a subseção 6.3.3 mostra o funcionamento do painel de configuração de conflitos.

#### 6.3.1 Políticas Positivas

O SGPCA permite que o usuário gere, edite e delete PCAs, utilizando para tal os dados presentes na sua interface. Estes dados são estruturados de forma que seja de fácil entendimento. A Figura 32 mostra a interface do SGPCA, direcionada a regras positivas. Alguns campos do SGPCA não são de preenchimento obrigatório, pois em alguns casos determinada instancia não se aplica. No campo “Política” da Figura 32 o usuário visualiza o nome da PCA, ou seja, do arquivo `.xml` onde estarão as informações sobre a PCA que está sendo editada. O campo “Nome do Usuário” é usado caso uma política seja aplicada a um determinado usuário.

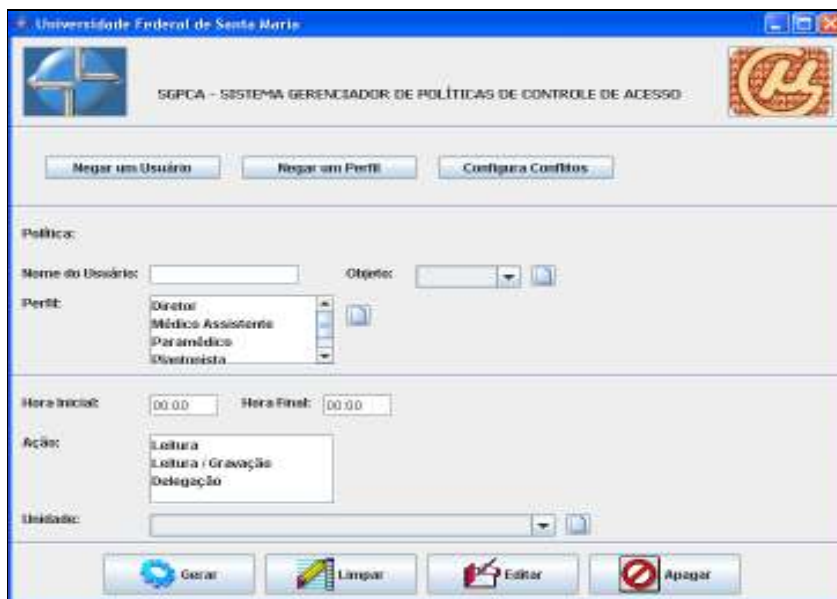


Figura 32 – Interface para Gerenciamento de PCAs Positivas.

Em “Perfil”, o usuário informa o perfil da PCA, por exemplo, Diretor, Médico Assistente, Paramédico, Plantonista, etc, e o “Objeto” indica o que a política irá proteger, que pode ser um EPR, impressora, etc.

Existe também no SGPCA a possibilidade de estabelecermos restrições temporais à um determinado recurso, por exemplo, pode-se dizer que um determinado perfil somente pode acessar o EPR se for no horário das 12:00:00h as 14:00:00h. Neste requisito temporal temos também um controle de inconsistências na interface, onde ele proíbe que seja digitada uma hora incompleta ou inválida. A finalidade é não ocorrer erros na geração da PCA decorrentes de falha na digitação.

O SGPCA também trata de regras contextuais, onde o usuário pode através da interface selecionar em que unidade uma determina política será aplicada, por exemplo: um médico que estiver realizando uma determinada consulta poderá ter acesso total ao EPR, mas se estiver na unidade de atendimento somente poderá ler o EPR.

Em determinadas instâncias presentes na interface tem-se a possibilidade de cadastro de novos elementos como em: objetos, perfis e unidades. Isto torna o sistema mais flexível e adaptável a diversas situações.

Uma das principais funcionalidades do SGPCA é de possibilitar a edição facilitada de uma política. Para isso os dados da política são disponibilizados na interface gráfica do SGPCA e apresentados para o usuário efetuar as devidas modificações.

### 6.3.2 Geração de Políticas Negativas

Para a geração de PCA negativas, existem duas interfaces: uma para o gerenciamento de políticas destinadas a negação de usuários e outra relativa aos perfis. Esta separação justifica-se pelo fato de não congestionar em uma só interface todas as funcionalidades do sistema e para facilitar o entendimento do usuário que está utilizando o sistema.

Na Figura 33, observa-se a interface para a negação de usuários, no qual existem os campos de “Nome do Usuário”, “Unidade”, “Hora Inicial” e “Hora Final”, sendo os botões “Gerar”, “Limpar”, “Editar” e “Apagar” os mesmos da interface de PCAs positivas.

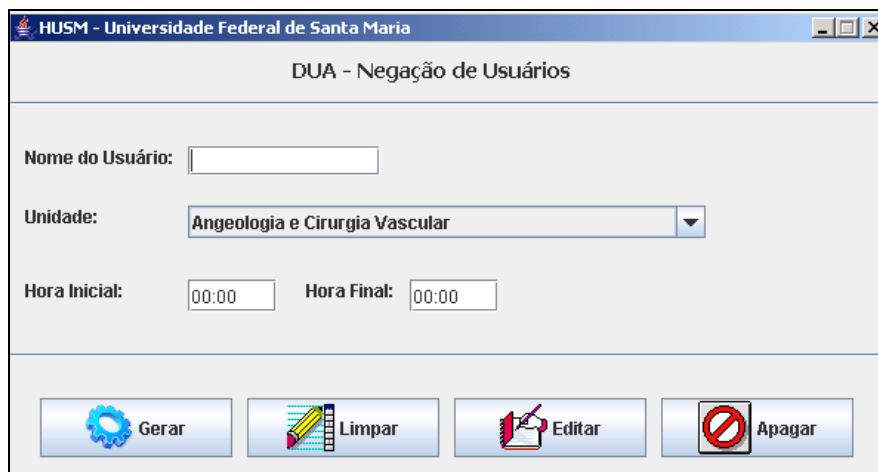


Figura 33 – Interface para Gerenciamento de PCA Negativas (DUA)

As interfaces para a geração de PCA negativas tanto para usuários como para perfis, são padronizadas, e os elementos que constituem as interfaces são na maioria os mesmos, somente diferenciando o campo “Nome” o qual na interface da DRA é trocado pelo campo “Perfil”. Pode-se observar na Figura 34 a interface do DRA, onde os elementos são: “Perfil”, “Unidade”, “Hora Inicial” e “Hora Final”.

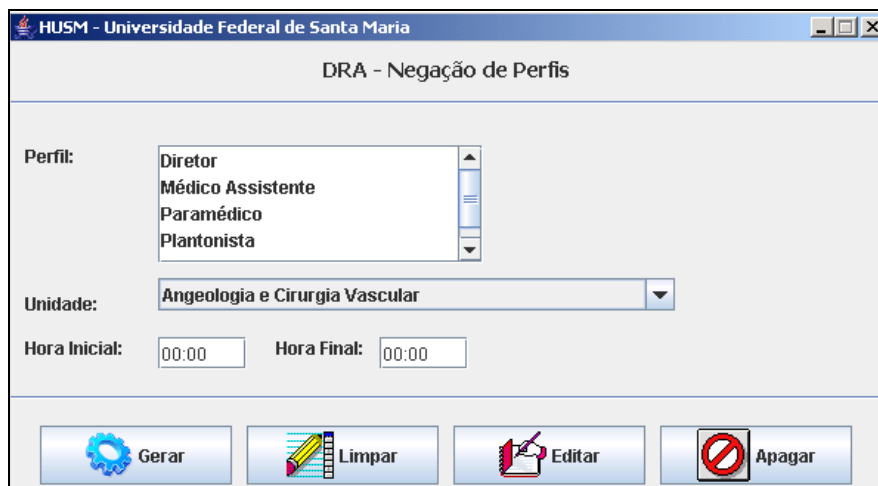


Figura 34 – Interface para Gerenciamento de PCA Negativas (DRA)

### 6.3.3 Painel de Configuração de Conflitos de Interesse

No SGPCA existe um painel de configuração de conflito de interesse qual um usuário responsável pode definir quais perfis e unidades hospitalares que são conflitantes, ou seja, que não podem ser atribuídos a um usuário de forma simultânea. Este painel de configuração tem como funcionalidade impedir conflitos de interesse entre PCAs. Entretanto, deve-se lembrar que um usuário pode ter dois perfis associados, desde que eles não sejam conflitantes com os interesses da organização.

Na Figura 35, pode-se observar o painel de configuração de conflitos, onde estão presentes os seguintes elementos: “Perfil1”, “Perfil2” e “Unidade”. Existe também um campo para a visualização das regras armazenadas, e dois botões sendo um para adicionar a regra e outro para removê-la.



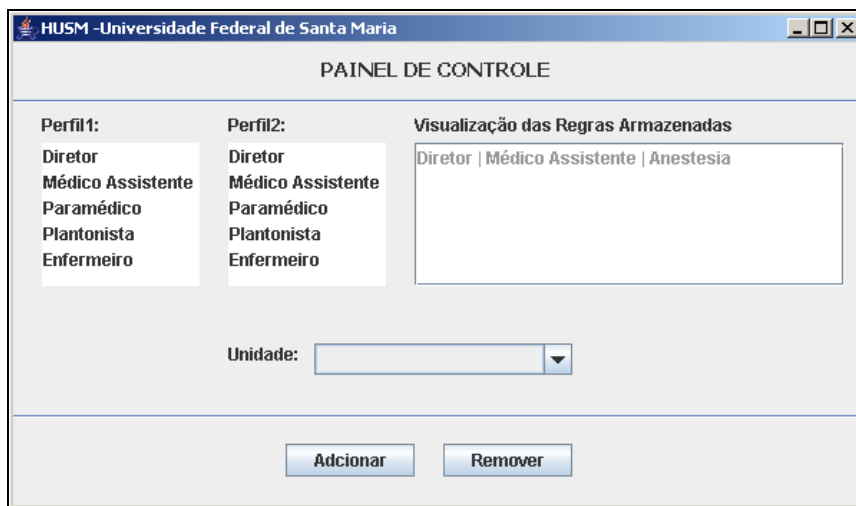


Figura 35 – Painel de Configuração de Conflitos de Interesse.

As regras armazenadas são verificadas sempre que for gerada uma nova PCA.

## 6.4 Testes no SGPCA

Com a finalidade de avaliar o SGPCA, foram realizados testes de funcionalidade na geração de políticas, algoritmos de verificação de conflitos e de conformidade com outros gerenciadores de políticas. Na seção 6.4.1, serão apresentados os testes na geração das políticas, bem como os algoritmos implementados, onde são simulados casos de erros. Os testes de conformidade serão vistos na seção 6.4.2.

### 6.4.1 Testes na Geração de Políticas

Com a finalidade de testar a geração das políticas no SGPCA, foram propostos testes com o objetivo de verificar a funcionalidade do sistema. Nesta seção também será apresentado testes nos algoritmos que forcem casos de erros para avaliar a eficácia no controle de conflitos.

Como caso de testes vamos supor as seguintes situações:

**Caso 1)** Neste primeiro caso serão colocados na interface do SGPCA informações para a geração de uma política positiva para o SGPCA. Logo propõe-se o seguinte experimento: um usuário chamado “Roberto” o qual tem o perfil de “Médico Assistente” pode realizar a ação de “Leitura e Gravação” no objeto “EPR” se estiver na unidade da “Cardiologia” e for das 06:00:00h às 12:00:00h.

Política 1: o usuário “Roberto” com o perfil de “Médico Assistente” pode acessar o “EPR” com a ação de “Leitura /Gravação” na “Cardiologia” das 06:00:00h às 12:00:00h.



The screenshot displays the SGPCA (Sistema Gerenciador de Políticas de Controle de Acesso) interface. At the top, it shows the title 'Universidade Federal de Santa Maria' and the application name 'SGPCA - SISTEMA GERENCIADOR DE POLÍTICAS DE CONTROLE DE ACESSO'. Below the title bar, there are three buttons: 'Negar um Usuário', 'Negar um Perfil', and 'Configura Conflitos'. The main area is titled 'Política 1' and contains the following fields:

- Nome do Usuário:** Roberto
- Perfil:** Médico Assistente (selected from a list that also includes Diretor and Paramédico)
- Objeto:** Promédico
- Hora Inicial:** 06:00
- Hora Final:** 12:00
- Ação:** Leitura / Gravação (selected from a list that also includes Delegação)
- Unidade:** Cardiologia

At the bottom of the interface, there are four buttons: 'Gerar', 'Limpar', 'Editar', and 'Apagar'.

Figura 36 – Política 1 / Interface

Pode-se observar na Figura 36 a interface para políticas positivas do SGPCA, onde se tem a Política1 com os dados já preenchidos nos devidos campos. Na Figura 37 mostra-se a política já gerada e codificada na linguagem XACML o que ocorre após pressionar o botão “Gerar”.

```

<?xml version="1.0" encoding="ISO-8859-1" ?>
<Policy xmlns="urn:oasis:names:tc:xacml:1.0:policy" xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
xmlns:cts="urn:oasis:names:tc:xacml:1.0:context" PolicyId="Política 1.xml"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides">
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:regexp-string-match">
          <SubjectAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string" AttributeId="_perfil" />
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Médico Assistente</AttributeValue>
        </SubjectMatch>
      </Subject>
      <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
        <SubjectAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string" AttributeId="_setor" />
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Cardiologia</AttributeValue>
      </SubjectMatch>
    </Subjects>
    <Resources>
      <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
        <ResourceAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string"
AttributeId="id:entid:codex" />
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Prontuário</AttributeValue>
      </ResourceMatch>
    </Resources>
  </Target>
  <Rule RuleId="Access" Effect="Permit">
    <Target>
      <Subjects>
        <Subject>
          <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Roberto</AttributeValue>
          </SubjectMatch>
        </Subject>
      </Subjects>
    </Target>
    <Condition FunctionId="http://research.sun.com/projects/xacml/names/function#time-in-range">
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
        <EnvironmentAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#time"
AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time" />
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">06:00:00</AttributeValue>
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">12:00:00</AttributeValue>
      </Apply>
    </Condition>
  </Rule>
  <Rule RuleId="Access" Effect="Deny" />
  <Obligations>
    <Obligation ObligationId="Resposta" FulfillOn="Permit">
      <AttributeAssignment AttributeId="aces" DataType="http://www.w3.org/2001/XMLSchema#string">Leitura /
Gravacao</AttributeAssignment>
    </Obligation>
  </Obligations>
</Policy>

```

Figura 37 – Política 1 / Codificação

Neste primeiro teste (Figura 36 e 37) é possível observar que a política foi gerada de forma correta na linguagem XACML, onde pode-se observar os elementos: *Target*, *Resources*, *Obligation* e *Condition*. Deve-se salientar também que se ocorresse algum erro o SGPCA mostraria os erros e a política não seria exibida.

Pode-se concluir com este teste que o SGPCA gera políticas positivas de forma correta. Neste caso não havia outras políticas no repositório. Nos próximos casos de testes esta condição será melhor explorada, pois a política 1 permanecerá no repositório.

**Caso 2)** Neste segundo caso é proposto um teste para a geração de políticas negativas para um determinado Perfil (DRA). Para a propriedade DRA tem-se o seguinte experimento: o perfil de “Enfermeiro” não tem acesso na unidade de “Anestesia” das 12:00:00h às 18:00:00h.

Política 2 (Figura 38 e 39): o perfil de “Enfermeiro não pode ter acesso na “Anestesia” das 12:00:00h às 18:00:00h.

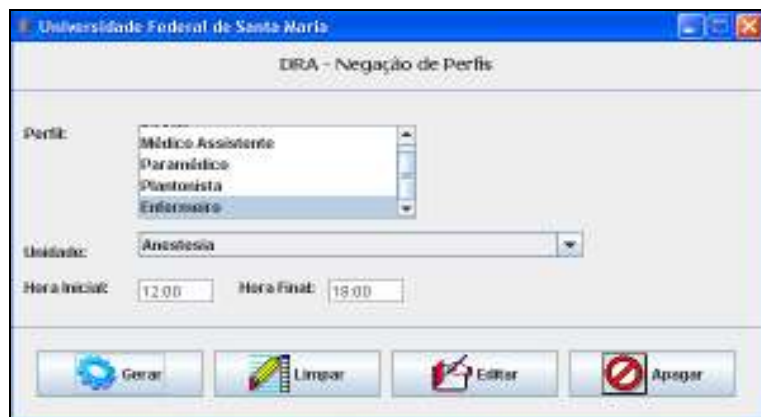


Figura 38 – Política 2 / Interface

Pode-se observar na Figura 38, a interface de geração de PCA negativas referente a perfis (DRA), onde tem como atributos selecionados: “Enfermeiro”, “Anestesia” e 12:00:00h às 18:00:00h respectivamente. A Figura 39 apresenta a política codificada para regras negativas onde observa-se a presença do elemento *Deny* responsável por negar uma política.

```

<?xml version="1.0" encoding="ISO-8859-1" ?>
- <Policy xmlns="urn:oasis:names:tc:xacml:1.0:policy" xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:ctx="urn:oasis:names:tc:xacml:1.0:context" PolicyId="Política 2.xml"
  RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides">
- <Target>
- <Subjects>
- <Subject>
- <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:regexp-string-match">
  <SubjectAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string" AttributeId="_perfil" />
  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Enfermeiro</AttributeValue>
</SubjectMatch>
- <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
  <SubjectAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string" AttributeId="_setor" />
  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Anestesia</AttributeValue>
</SubjectMatch>
</Subject>
</Subjects>
</Target>
- <Rule RuleId="Access" Effect="Deny">
- <Condition FunctionId="http://research.sun.com/projects/xacml/names/function#time-in-range">
- <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
  <EnvironmentAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#time"
    AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time" />
</Apply>
  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">12:00:00</AttributeValue>
  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">18:00:00</AttributeValue>
</Condition>
</Rule>
<Rule RuleId="Access" Effect="Permit" />
</Policy>

```

Figura 39 – Política 2 / Codificação.

Pode-se observar que o SGPCA gera políticas em XACML para DRA, e que as regras selecionadas na interface aparecem descritas na linguagem XACML como verifica-se na Figura 38.

Neste caso de teste pode-se observar a utilização da interface para a geração de políticas negativas para perfis. Como não houve conflitos com a política 1, a política 2 foi gerada corretamente.

**Caso 3)** O objetivo neste caso de teste é simular um conflito de negação. Logo propõe-se o seguinte experimento: o usuário “João” que é “Enfermeiro” pode acessar o “EPR” das 12:00:00h às 18:00:00h na unidade de “Anestesia”. Tal política conflita com a política 2 gerada no caso de teste 2.

Política 3 (Figura 40): O usuário “João” com o perfil de “Enfermeiro” pode acessar o “EPR” das 13:00:00h às 17:00:00h na unidade de “Anestesia”.

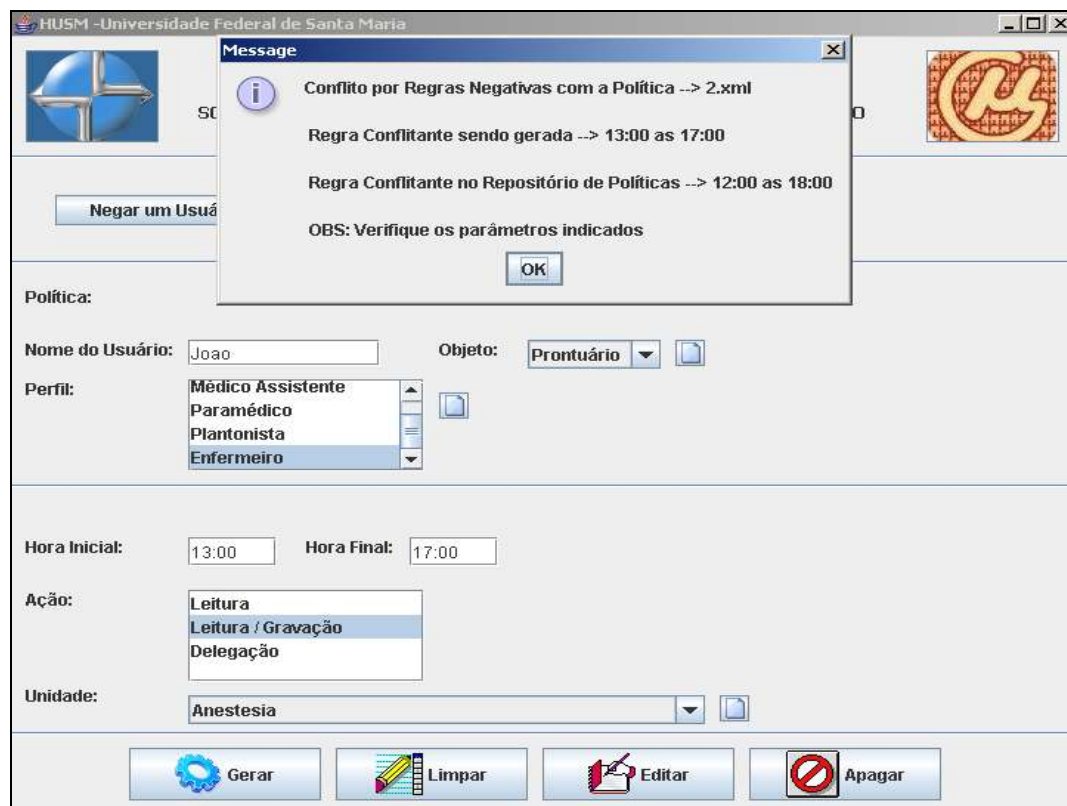


Figura 40 – Política 3 / Interface

Na Figura 40 são informados na interface do SGPCA os elementos nos quais deseja-se gerar a política, que nesse caso refere-se ao usuário “João” o qual tem acesso no objeto “Prontuário” na unidade de “Anestesia” das 13:00:00h às 17:00:00, mas pode-se observar a indicação da presença de um conflito por regras negativas com a Política 2 (Figura 38 e 39) gerada posteriormente, devido já conter no repositório de políticas uma PCA negando o perfil de “Enfermeiro” na unidade de “Anestesia” no horário das 12:00:00h às 18:00:00h. A mensagem de conflito é apresentada na tela relatando um conflito e mostrando quais regras da política estão conflitanto (vide figura 40).

Pode-se observar que o algoritmo indicou a presença de políticas com conflitos por regras negativas, indicando a política e as regras que estão conflitanto, logo pode-se concluir com este caso de teste que o SGPCA consegue verificar conflitos por negação.

**Caso 4)** O objetivo deste caso é testar o algoritmo de conflitos de interesse, logo propõe-se o seguinte experimento: adicionar uma regra no painel de controle de conflitos onde o usuário que tiver o perfil de “Diretor” não poderá ter o de “Médico Assistente” na unidade de “Anestesia”, conforme pode-se observar na Figura 41, e tentar gerar uma nova PCA, para que o mesmo usuário tenha o perfil de Médico Assistente na mesma unidade.

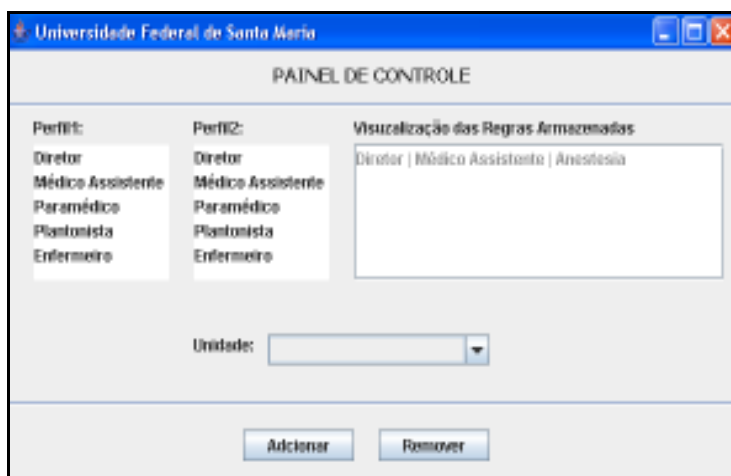


Figura 41 – Painel de Controle / Interface

Com a regra cadastrada no painel de conflitos um conflito de interesse deve ser indicado.

Política 4: o usuário “José” que tem o perfil de “Diretor” pode acessar o “EPR” das 12:00:00h às 18:00:00h na unidade de “Anestesia”.

Política 5 (a ser gerada): o usuário “José” o qual já tem o perfil de “Diretor” é associado ao de “Médico Assistente” na mesma unidade de “Anestesia” (vide Figura 41).

Pode-se observar na Figura 41 que a Política 5 não é gerada, pois o algoritmo de Conflitos de Interesse detecta que o usuário “José” já tem o perfil de “Diretor” e que não pode ser associado ao de “Médico Assistente”, visto que são perfis conflitantes e não devem ser atribuídos juntos em uma mesma unidade hospitalar. Além da política não ser gerada é exibida uma mensagem indicando a política conflitante e os perfis que estão conflitando: “Médico Assistente” e “Diretor”.

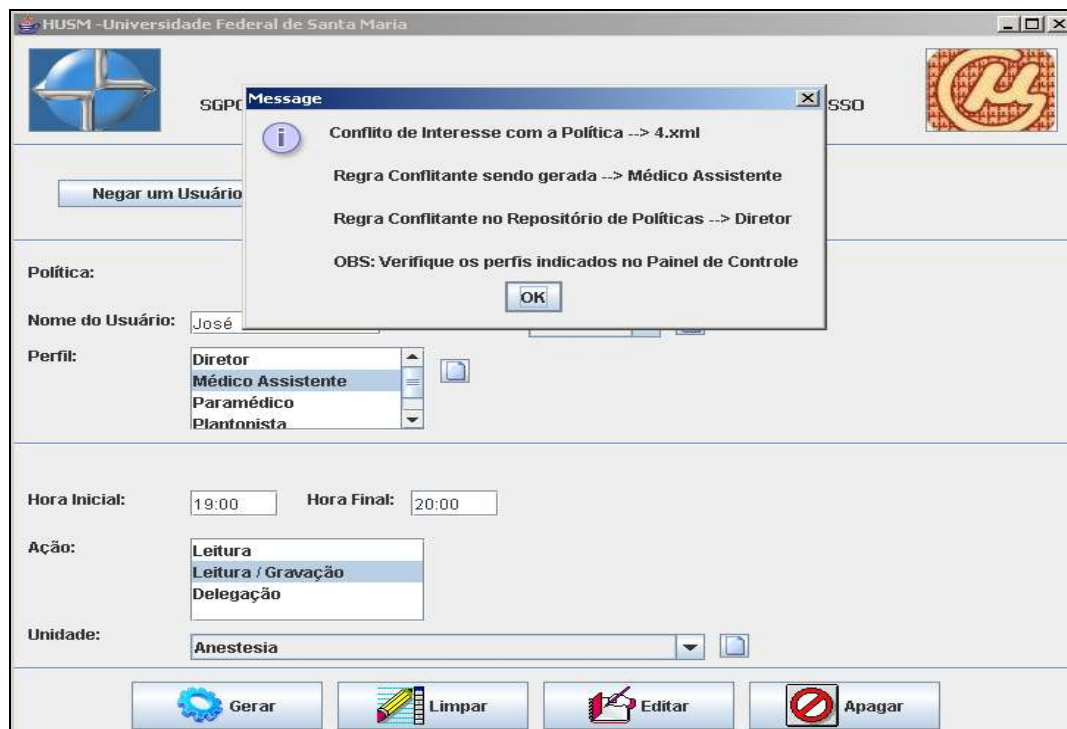


Figura 41 – Política 5 / Interface

Com este teste pode-se observar que o algoritmo para a verificação de conflitos de interesse comportou-se como o planejado, ou seja, não deixando que um usuário venha a ter dois perfis ditos conflitantes.

**Caso 5)** Neste caso de teste o objetivo é verificar e validar o algoritmo para verificação de conflitos de redundância por políticas duplicadas, logo propõe-se o seguinte experimento: com a Política 6 armazenada no repositório de políticas, é gerada uma política 7 definindo que o usuário “Rodrigo” com o Perfil de “Diretor” na unidade de “Cardiologia” pode “Ler” o objeto “EPR” das 18:00:00 às 22:00:00.

Política 6: o usuário “Rodrigo” que tem o perfil de “Diretor” na unidade de “Cardiologia” pode acessar o “EPR” das 18:00:00 às 22:00:00 com a ação de “Leitura”.

Política 7 (a ser gerada): o usuário “Rodrigo” que tem o perfil de “Diretor” na unidade de “Cardiologia” pode acessar o “EPR” das 18:00:00 às 22:00:00 com a ação de “Leitura”.

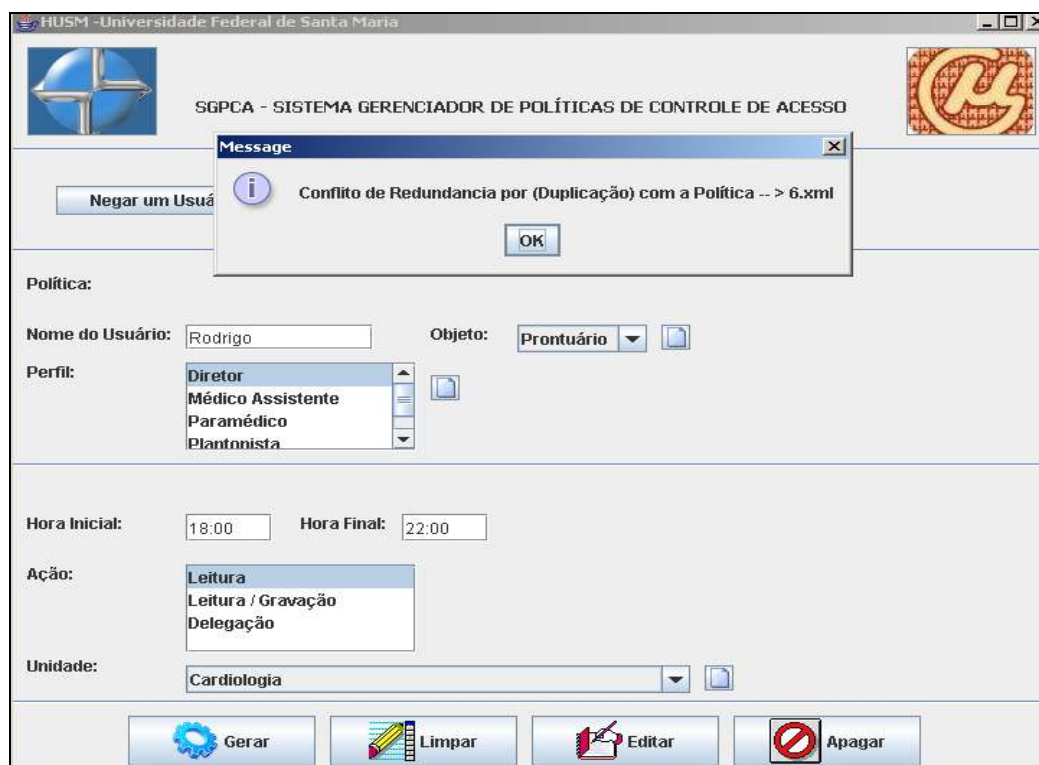


Figura 43 – Política 7 / Interface

Analisando a Figura 43, pode-se observar que um conflito de redundância por políticas duplicadas é levantado, devido já existir no repositório das políticas uma com as regras exatamente iguais. Logo pode-se concluir que o algoritmo proposto impede que políticas com conflitos de redundância por duplicação venham a ocorrer em tempo de geração das políticas.



**Caso 6)** Neste caso de teste o algoritmo a ser testado será o de verificação de conflitos por redundância de ações, logo propõe-se o seguinte experimento: com a Política 8 cadastrada no repositório de políticas, tenta-se gerar as políticas 9 e 10, conforme a Figura 44 e 45 respectivamente, onde a 9 não apresenta interseção temporal com a 8, mas a 10 apresenta interseção temporal com a 8 e 9.

Política 8: o usuário “Rodrigo”, que tem o perfil de “Paramédico”, pode acessar o “EPR” das 17:00:00 às 22:00:00.

Política 9 (a ser gerada): o usuário “Rodrigo” com o perfil de “Paramédico” pode acessar o “EPR” das 09:00:00h às 14:00:00h.

Política 10 (a ser gerada): o usuário “Rodrigo” com o perfil de “Paramédico” pode acessar o “EPR” das 12:00:00h às 18:00:00h.

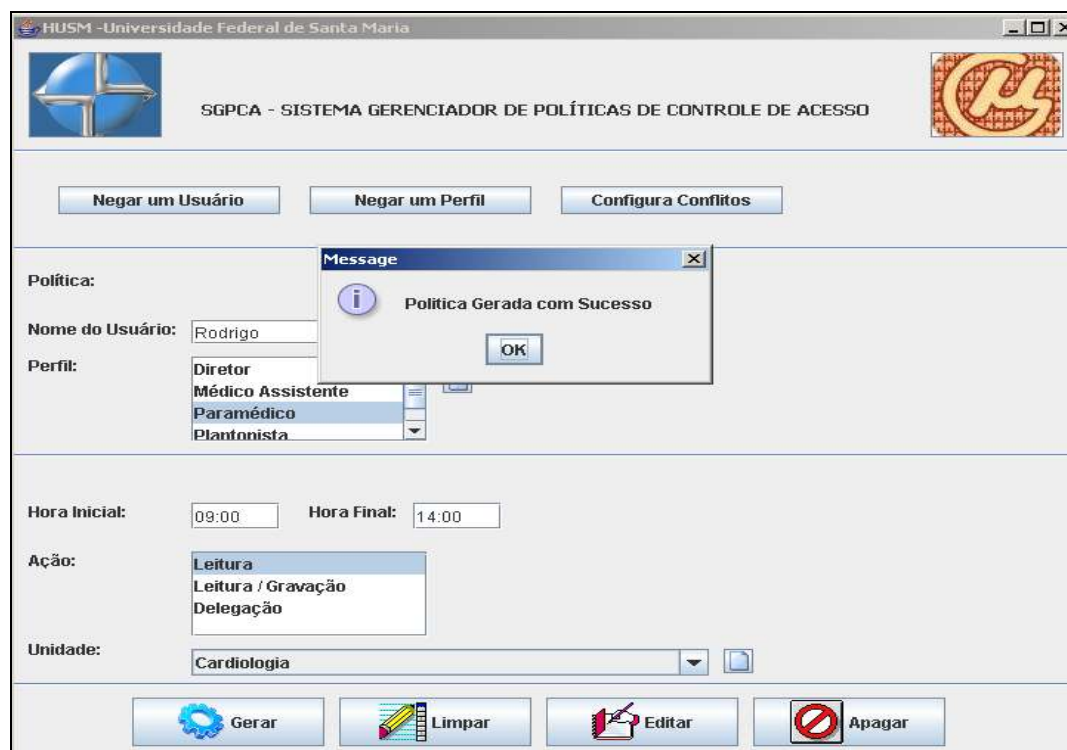


Figura 44 – Política 9 / Interface

Na Figura 44, pode-se observar que a política foi gerada normalmente, visto que a PCA não conflita com a política 8 no repositório das políticas, ou seja, não ocorre uma intersecção temporal para este caso específico.

A Figura 45 mostra a mensagem de conflito com a política 8 quando tenta-se gerar a política 10, momento em que é registrado um conflito de redundância por ações conflitantes. Na última linha da mensagem é sugerido que seja realizada uma troca no intervalo temporal. O SGPCA não indica o conflito com a política 10 porque pára ao encontrar o primeiro conflito.

Para resolver o conflito o administrador de políticas precisa redefinir a política 10, ou então reeditar a política 8, incluindo nesta o intervalo previsto por ambas políticas (8 e 10). Se isto ocorrer, ao tentar gerá-la, o SGPCA indicará o conflito com a política 9, que define o intervalo das 9:00:00h às 14:00:00h, o qual conflita com o intervalo das 12:00:00 às 22:00:00h. A solução para este caso é juntar as políticas 8, 9 e 10 numa única com intervalo das 9:00:00h às 22:00:00h, se este for realmente o desejo do administrador.

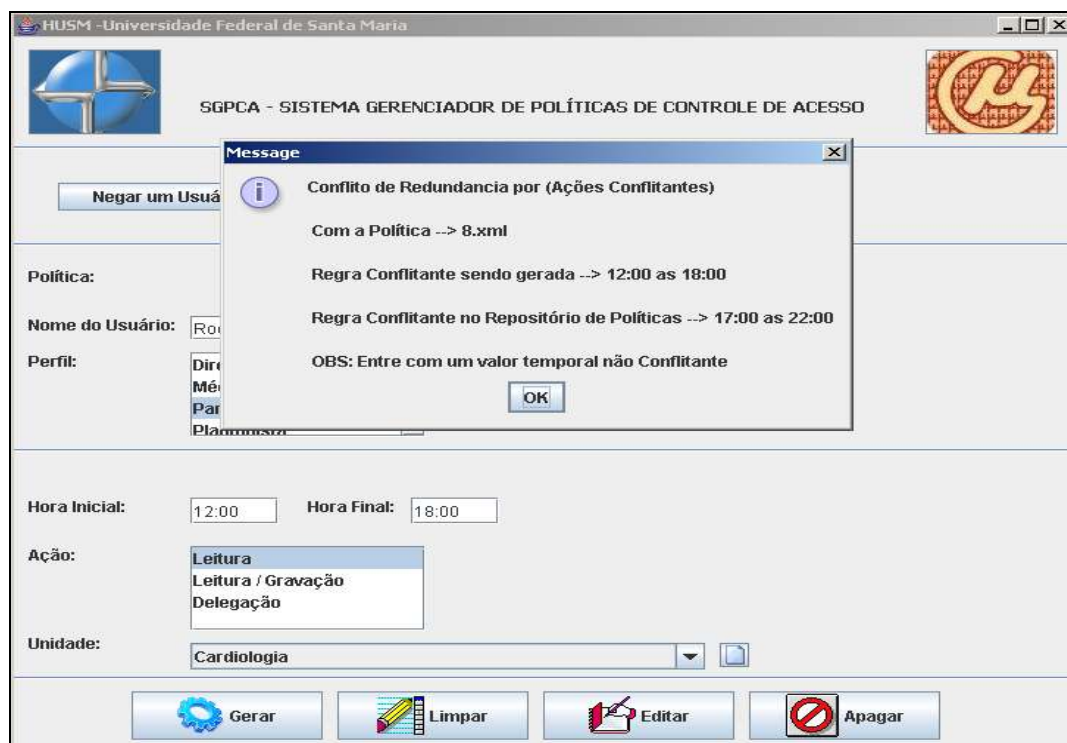


Figura 45 – Política 10 / Interface

Pode-se concluir com este caso de teste que em hipótese nenhuma uma política com conflitos de redundância de ações irá ser gerada sem que o algoritmo venha a identificar uma condição conflitante.

## 6.4.2 Testes de Conformidade

Nesta subseção serão realizadas comparações com outros gerenciadores de políticas, verificando se as funcionalidades apresentadas no SGPCA são válidas e se os dados na interface do sistema condizem com as apresentadas em outras ferramentas de gerenciamento de políticas.

O intuito de todos os gerenciadores de políticas é da proteção de um determinado objeto. Entretanto deve-se observar a finalidade o qual foi desenvolvido um determinado gerenciador que pode ser para: segurança de VPNs, servidores de backup, etc.

No SGPCA, apesar do foco ser unidades de saúde, o mesmo pode ser aplicado em diversas áreas, pois foram tomadas precauções para tornar o sistema flexível. Um grande cuidado que se deve ter é com as informações na interface do gerenciador, assunto que será abordado e testado nessa subseção.

Os gerenciadores de PCA abordados nesta seção já foram citados e comentados no capítulo 2. Nesta subseção o foco é nas informações presentes em suas interfaces bem como as funcionalidades.

A) Larges-Seg (FERREIRA, SANTOS e JÚNIOR, 2004): Esse gerenciador o qual tem por finalidade gerenciar o estabelecimento de VPNs tem em sua interface as seguintes regras: servidor de origem, horário de acesso e se o usuário é do tipo cliente ou funcionário. Neste cenário pode-se observar que o objeto são os servidores de origem e destino, o perfil é o cliente ou funcionário, hora de acesso refere-se ao intervalo temporal o qual um usuário ou perfil pode acessar ou estabelecer uma VPN. No caso do SGPCA pode-se observar que o mesmo está condizendo com as informações apresentadas pelo LARGES-Seg, pois nos dois casos tem-se nas suas interfaces um perfil, intervalo temporal e um objeto.

B) Gerenciador de Políticas para o Globus (LORCH, KAFURA e SHAH, 2003): Esta ferramenta tem como funcionalidade o gerenciamento de políticas para ambientes de computação em grid. Nesta ferramenta tem-se as seguintes regras: *person*, *allow* e *resource*, onde *person* refere-se a um determinado usuário, *allow* a ação e *resource* ao objeto em questão. Logo pode-se dizer que o SGPCA também está de acordo com as informações contidas nesta ferramenta, pois contempla todos os requisitos propostos mesmo o foco sendo diferente.

C) Sectet-PI (ALAM, BREU e HAFNER, 2006): Este gerenciador de políticas, foi implementado baseando-se em regras destinadas para hospitais. Em sua interface deve-se informar regras como: perfil, objetos, ação e condição. No entanto na regra de condição não é utilizado atributos temporais, mas sim, relacionados a outras regras, por exemplo: um Médico somente pode acessar o EPR de um paciente se ele for o médico titular deste paciente. Em modo geral o Sectet-PI não realiza a verificação de conflitos e exige o conhecimento de linguagem de codificação de políticas. Em comparação ao Sectet-PI o SGPCA contempla em sua interface todas a informações contidas no Sectet-PI e ainda realiza a verificação de conflitos nas políticas.

D) Ponder (DAMIANOU, 2001): Analisando os dados presentes na interface desse gerenciador de políticas, pode-se observar as seguintes regras: perfis, unidade temporal, objeto e as ações que podem ser executadas neste objeto. O Ponder, utiliza uma arquitetura baseada em domínios e é destinada para o gerenciamento de servidores de backup. Neste caso tem-se a necessidade do conhecimento e do estudo da linguagem do Ponder para a geração de políticas. No SGPCA, o conhecimento de linguagem de codificação de políticas não se faz necessário visto que o sistema se encarregará de codificá-la.

Na análise dos gerenciadores acima citados, deve-se observar que, exceto o Ponder, o LARGES-Seg, o Gerenciador de Políticas para o Globus e o Sectet-PI utilizam a linguagem XACML, o qual tem como propriedades: ações, perfis ou usuários, objetos e uma condição de acesso. No que se diz respeito às funcionalidades, todos os gerenciadores de políticas estudados têm as mesmas funções: geração e edição de políticas.

Com esta análise pode-se concluir que há uma padronização das regras e das funcionalidades apresentadas na interface entre os gerenciadores de políticas mesmo as linguagens de codificação de políticas sendo diferentes.

## **6.5 Conclusões Parciais**

Neste capítulo foi apresentado os aspectos sobre a implementação do SGPCA, abrangendo desde a utilização da API sun.xacml até os testes dos algoritmos para detecção de conflitos.

A utilização a API sun.xacml possibilita a leitura das políticas por programas Java, o que possibilita a implementação dos algoritmos de verificação de conflitos em Java. A

codificação das políticas em XACML e sua verificação em Java confere ao SGPCA a característica multiplataforma. Os testes nos algoritmos serviram para analisar a real eficácia dos algoritmos propostos para o tratamento de conflitos, bem como as funcionalidades oferecidas,, enquanto os testes de conformidade serviram para validar as informações contidas no SGPCA.

Nos algoritmos para a verificação de conflitos testados neste capítulo, pode-se observar que além de impedir que um determinado conflito ocorra o mesmo aponta o nome da política conflitante bem como as regras que estão conflitando. Tal informação, aliada a uma interface de fácil manuseio, torna o SGPCA uma ferramenta mais amigável para o gerenciamento de políticas.

## **7 CONCLUSÕES E PERSPECTIVAS**

Neste último capítulo são destacadas as principais contribuições obtidas neste trabalho, bem como são sugeridas algumas atividades que poderão ser exploradas em trabalhos futuros.

### **7.1 Conclusões**

Esta dissertação apresenta uma ferramenta de gerenciamento de políticas de controle de acesso, denominado SGPCA, o qual possibilita realizar um gerenciamento de políticas de controle de acesso de forma facilitada, ou seja, não exigindo o conhecimento de linguagens de codificação de políticas. O SGPCA agrega novas funcionalidades para o RBAC, como negação de usuários, sendo que os modelos propostos no RBAC contemplam somente a negação de perfis.

O processo de verificação de conflitos em tempo de geração das políticas de controle de acesso aliado ao processo de edição sem uso de linguagem de descrição de política, possibilita maximizar a produtividade na geração e edição de políticas livre de conflitos. Assim, o principal mérito do SGPCA é o de possibilitar a geração de políticas de forma simples e correta.

Nos testes realizados pôde-se observar a validação dos algoritmos e das informações e funcionalidades contidas no SGPCA. Como resultado conclui-se que o mesmo é funcional e que contempla todas as características dos gerenciadores de políticas de controle de acesso. A não exigência do conhecimento de uma linguagem específica de descrição de políticas de controle de acesso garantiu flexibilidade e eficiência ao sistema.

### **7.2 Trabalhos Futuros**

Para dar continuidade aos trabalhos realizados nesta dissertação, propõem-se algumas atividades que poderão ser realizadas futuramente:

- incluir mecanismos de autenticação no SGPCA, para possibilitar realizar o tratamento de outras modalidades de conflitos como o de Múltiplos Gerentes o qual necessita o conhecimento de quem esta utilizando o sistema para realizar o tratamento deste conflito;
- incluir no SGPCA a possibilidade de alteração das regras das políticas de forma manual, ou seja, dando a possibilidade do administrador modificar as políticas no código propriamente dito.

## REFERÊNCIAS

ALAM, M.; BREU, R. ; HAFNER, M. **Modeling permissions in a (U/X)ML world.** Proceedings of the First International Conference on Availability, Reliability and Security ARES, 2006.

ALMEIDA, M. B. **An introduction to XML, its use on the Internet and some complementary concepts.** may/ago. vol. 31, n.2, p.\5-13, 2005.

AL-KAHTANI, A. M. ; SANDHU, R. **Rule-Based RBAC with Negative Authorization.** Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC'04) .

BERTINO, E. ; BONATTI, P. A.; FERRARI, E. **TRBAC: A Temporal Role-Based Access Control Model.** ACM Transaction on Information and System Security, 4(3):191–233. 2001.

BERTINO,E. ; SAMARATI, P.; JAJODIA, S. **An Extended Authorization Model for Relational Databases.** Transactions on Knowledge and Data Engineering, Vol. 9, No. 1, January-february 1997 IEEE.

CFM. **Resolução 1.629/2002 do Conselho Federal de Medicina.** 2002. Disponível em <<http://www.arnaut.eti.br/ResoCFM.htm>>. Acesso em: fevereiro de 2007.

CHARALAMBIDES, M. ; FLEGKAS, P. ; BANDARA, K.A.; LUPU, C.E.; RUSSO, A. SLOMAN, M. ; DULAY, N. ; PAVLOU, G.; LOYOLA. J.R. **Policy Conflict Analysis for Quality of Service Management** Proceedings of the Sixth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'05).

DAMIANOU, N.; DULAY, N. ; LUPU, M. ; SLOMAN, M. **The Ponder Specification Language** IEEE, Workshop on Policies for Distributed Systems ans Networks (POLICY 2001), Bristol, UK, junho de 2001.

DEY, A. K. ; ABOWD, G. D. **Towards a Better Understanding of Context and Contextawareness.** Gvu technical report GIT-GVU-99-22, College of Computing, Georgia Institute of Technology, 1999.

DUNLOP, N.; INDULSKA, J.; RAYMOND, K. **Dynamic Conflict Detection in Policy-Based Management Systems.** p. 15, Sixth International Enterprise Distributed Object Computing Conference (EDOC'02), 2002.

DURSun, T. ; ORENCIK, B. **Police Distributed Conflict Detection Architecture.** IEEE Communications Society 2081-2085. 2004.



FERRAILOLO, D. F.; SANDHU, R. S.; GAVRILA, S. I. ; KUHN, D. R.; CHANDRAMOULI, R. **Proposed NIST Sstandard for Role-Based Access Control**. Information and System Security, 4(3):224–274. 2001.

FERREIRA, F. J. H. S. ; SANTOS, A.R.; JUNIOR, C. J. **LARCES\_Seg - Uma Arquitetura para Gerenciamento de VPNs Baseada em Políticas utilizando XACML**. WORKCOMP SUL, Florianópolis. WORKCOMP SUL, 2004.

HIGASHIYAMA, M. **Jacoweb-Abc: Integração do Modelo de Controle de Acesso Uconabs no Corbasec** Dissertação de Mestrado apresentada à Pontifícia Universidade Católica do Paraná, Curitiba-PR, Brasil. 2005.

HITCHEN, M.; V. VARADHARAJAN. **Tower: A Language for Role Based Access Control**. In Proceedings of the Policy Workshop 2001, HP Labs, Bristol, UK, Springer-Verlag, 29-31 January 2001.

LORCH, M. ; KAFURA, D. ; SHAH, S. **An XACML-based Policy Management and Authorization Service for Globus Resources**. Proceedings of the Fourth International Workshop on Grid Computing GRID 2003 IEEE.

LORCH, M. ; KAFURA, D. ; SHAH, S. ; PROCTOR, S. ; LEPRO, R. **First experiences using XACML for access control in distributed systems**. Proceedings of the ACM workshop on XML security. Workshop On XML Security, 2003.

LUPU, E ; SLOMAN, M. **Conflict Analysis for Management Policies**. Proceedings of the Vth International Symposium on Integrated Network Management IM (formerly know as ISINM), San-Diego (U.S.A.), Chapman&Hall May, 1999.

KUDO M. ; HADA S. **XML Access Control Language: Provisional Authorization for XML Documents**. Tokyo Research Laboratory, IBM Research, 2000.

MOFFETT, J. D.; SLOMAN, M. **Policy Conflict Analysis in Distributed System Management**. Journal of Organizational Computing, vol 4, n 1, pp 1-22, 1993.

NYANCHAMA, M. ; OSBORN, S. **The Role Graph Model and Conflict of Interest**. ACM Transactions on Information and System Security, Vol. 2, n. 1, February 1999, Pages 3–33.

OASIS. **eXtensible Access Control Markup Language (XACML) version 1.1**. Committee Specification, August. 2003.

POLYRAKIS, A.; BOUTABA, R., **The Meta Policy Information Base**, IEEE. Network, Março/Abril 2002.

RIBEIRO, C.; ZUQUETE, A.; FERREIRA, P. SPL: An access control language for security policies with complex constraints. In Proceedings of the Network and Distributed System Security Symposium, San Diego, California, February 2001.

SHANKAR, S.C. ; RANGANATHAN, A.; CAMPBELL, R. **An ECA-P Policy-based Framework for Managing Ubiquitous Computing Environments.** Proceedings of the Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services MobiQuitous 2005 IEEE.

SOARES, G. A ; NUNES, R.C. ; AMARAL, É M. H. do. **Um Modelo de Controle de Acesso Baseado em Contexto para Autorizações a Informações Médicas.** In: XXXII Conferência Latino-Americana de Informática, Santiago do Chile, 2006.

SOMMERVILLE, I. **Software engineering.** 5. ed. Addison-Wesley. 1996.

STRASSNER, J.; ELLESSON, E.; MOORE, B. ; WESTERINEN, A. **Policy Core Information Model - Version 1 Specification, RFC 3060,** Disponível em <http://www.ietf.org>, February 2001/. Acesso em Junho de 2007 ;

SUN MICROSYSTEM. **XACML Implementation.** Disponível em <http://sunxacml.sourceforge.net/>. Acesso em : Abril de 2007.

WEISZFLOG W. (Ed). **Moderno Dicionário da Língua Portuguesa** Disponível em <<http://michaelis.uol.com.br/moderno/portugues/index.php>> Acesso em: Dezembro de 2007.

W3 CONSORTIUM. **Extensible Markup Language (XML).** 2. ed., out. 2000 Disponível em <<http://www.w3.org/xml/>>. Acesso em: janeiro de 2007.

ZURKO, E.M. ; SIMON, R.T. **Separation of Duty in Role-Based Environments.** 10th Computer Security Foundations Workshop CSFW 1997 p. 183.

**ANEXO A - Publicações****PUBLICAÇÕES**

LIMA, P. R. B. D.; NUNES, R. C. **Sistema Gerenciador de Políticas de Controle de Acesso.** In: XV Jornada de Jovens Pesquisadores da AUGM, 2007, São Lorenzo / Paraguai. XV Jornada de Jovens Pesquisadores da AUGM, 2007.

CRISTO, F ; FERREIRA, L. ; MULLER, F.M ; LIMA, P. R. B. D. **Metaheurísticas para Solução do Problema da Árvore de Cobertura Mínima Generalizado.** In: XXXIX Simpósio Brasileiro de Pesquisa Operacional, 2007, Fortaleza. XXXIX Simpósio Brasileiro de Pesquisa Operacional, 2007.

LIMA, P. R. B. D. ; NUNES, R. C. ; SOARES, G.A . **SGPCA Sistema Gerenciador de Políticas de Controle de Acesso.** In: Encontro Nacional de Engenharia de Produção, 2007, Foz do Iguaçu. Encontro Nacional de Engenharia de Produção, 2007.

LIMA, P. R. B. D. ; NUNES, R. C. ; SOARES, G.A . **Verificação de Conflitos na Gerência de Políticas de Controle de Acesso.** In: V Escola Regional de Redes de Computadores, 2007, Santa Maria, 2007.

LIMA, P. R. B. D. ; NUNES, R. C. ; SOARES, G.A . **Um Editor de Políticas XACML para informações Médicas.** In: VI Simpósio de Sistemas de Informação, 2006, Carazinho. VI Simpósio de Sistemas de Informação, 2006.