UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA

José Eduardo Angelin Ramos

Um estudo sobre especificação de políticas de Redes de Computadores utilizando um modelo genérico

Trabalho de Graduação

RECIFE, FEVEREIRO DE 2014

UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA

Um estudo sobre especificação de políticas de Redes de Computadores utilizando um modelo genérico

Trabalho apresentado como requisito parcial para obtenção do grau de Bacharel em Engenharia da Computação pelo Centro de Informática da Universidade Federal de Pernambuco.

Aluno: José Eduardo Angelin Ramos

Orientadora: Prof. Judith Kelner

RECIFE, FEVEREIRO DE 2014

Resumo

Gerenciamento de rede baseado em políticas (PBNM) usa automação orientada por políticas para gerir complexas redes de provedores de serviços e empresas. Existem diversos modelos clássicos de controle de acesso para políticas de segurança, tais como DAC, MAC e RBAC, porém estes não são totalmente satisfatórios quando as regras são definidas sobre um situação contextual. O modelo de controle de acesso baseado em organização (OrBAC) é um modelo genérico que é centralizado no conceito de organização e de regras contextuais, o que faz com que o modelo possua um alto poder de elaboração de políticas de segurança nos mais diversos cenários. O MotOrBAC é um editor de políticas de segurança desenvolvido sobre uma API em Java, baseado no modelo *OrBAC*. Através desta ferramenta é possível definir entidades relativas ao cenário em questão, definir regras de controle de acesso neste cenário e solucionar os potenciais conflitos entre as regras definidas. Neste trabalho é feito um estudo sobre a aplicabilidade do modelo de OrBAC, através da ferramenta MotOrBAC, em um cenário de redes de computadores. Através da elaboração do cenário descrito neste trabalho e da simulação deste cenário no MotOrBAC, foi possível detectar potenciais conflitos de políticas e realizar operações para soluciona-los, ratificando o poder do modelo OrBAC para especificar políticas de segurança.

Agradecimentos

ÍNDICE

1. Introdução 7
2. Conceitualização 9
   2.1. Gerenciamento de Redes de Computadores baseado em Políticas 9
   - Arquitetura 9
   2.2. Modelos de Controle de Acesso 11
   - DAC – Controle de Acesso Discricionário 12
   - MAC – Controle de Acesso Obrigatório 13
   - RBAC – Controle de Acesso baseado em Papeis 13
   - OrBAC – Controle de Acesso baseado em Organização 14
   - Discussão 15
3. OrBAC – Um modelo para controle de acesso 17
   3.1. Organização 17
   3.2. Entidades Abstratas e Concretas 17
   3.3. Subjects e Roles 18
   3.4. Objects e Views 19
   3.5. Actions e Activities 20
   3.6. Contexto 21
   3.7. Tipos de Contexto 21
   3.8. Políticas de Segurança 23
   3.9. Autorizações Concretas 24
   3.10. Restrições 24
   3.11. Gerenciamento de Conflito 24
4. MotOrBAC e OrBAC API 26
   - Apresentação da Ferramenta 27
5. Caso De Uso com o Modelo OrBAC 31
   - Descrição do Cenário de Caso de Uso 31
   - Simulação do Cenário no MotOrBAC 32
6. Conclusão e Trabalhos Futuros 38
7. Referências 40

ÍNDICE DE FIGURAS

Figura 1. O framework de políticas IETF definido pelo RAP. 10
Figura 2. Matriz de Controle de Acesso. 12
Figura 3. Representação da classificação com roles (papeis) no modelo RBAC. 14
Figura 4. Representação do modelo OrBAC. 15
Figura 5. Modelo da relação Empower. 19
Figura 6. Modelo da relação Use. 20
Figura 7. Modelo da relação Consider. 21
Figura 8. Classificação de contextos e suas informações. 23
Figura 9. Arquitetura do MotOrBAC 27
Figura 10. Roles associados à organização. 28
Figura 11. Resolução de Conflitos no MotOrBAC. 29
Figura 12. Exemplo de Simulação. 30
Figura 13. Modelagem dos Roles da Organization “Matriz”. 32
Figura 14. Modelagem dos Roles da Organization “Filial RN”. 33
Figura 15. Modelagem das Views. 33
Figura 16. Modelagem das Activies. 34
Figura 17. Contextos utilizados na simulação. 34
Figura 18. “Regra Geral” em “Filial RN”. 35
Figura 19. Permissões concedidas em “Filial RN”. 35
Figura 20. Conflitos acusados após a criação das regras. 36
Figura 21. Prioridades das regras conflitantes. 36
Figura 22. Simulação das regras do sistema. 37

1. INTRODUÇÃO

O gerenciamento de redes de computadores em grandes empresas é uma tarefa complexa. Alterações tanto na topologia da rede quanto na configuração dos seus componentes, como roteadores e servidores, podem aumentar ainda mais a complexidade do gerenciamento. Além disso, sistema computacionais em diversos ambientes requerem um controle de acesso por parte dos usuários sobre os recursos disponíveis. Em ambiente mais complexos é comum a necessidade de regras para controle de acesso específicas para determinadas situações, sendo necessário a avaliação de diversos fatores para decidir se será concedida a permissão no sistema.

Apesar da existência de diversos modelos para controle de acesso, o modelo OrBAC – Organization Based Access Control, ou Controle de Acesso baseado em Organização – surgiu como um modelo para especificação de políticas de segurança que pode ser completamente parametrizado por cada organização[1]. A principal vantagem do OrBAC para os três que serão abordados neste trabalho é a possibilidade de modelar cenários mais complexos utilizando uma quantidade de regras menores, já que as regras elaboradas no OrBAC são facilmente personalizáveis. Com isso, é possível a criação de regras mais complexos e que atinjam uma quantidade maior de situações. Esta característica se deve à utilização de alguns do conceitos que este trabalho aborda no capítulo 3, como hierarquia de entidades e contexto. Como exemplo, é possível que uma universidade proíba que os estudantes aluguem livros na biblioteca durante os últimos trinta dias letivos dos semestre. Neste caso, é possível especificar a restrição “durante os últimos trinta dias letivos dos semestre” através de um contexto, o que torna a regra mais eficiente.

Como será mais detalhado no capítulo 2, três dos principais modelos da literatura abordam apenas políticas de permissão e proibição de acesso. O OrBAC, por sua vez, suporta também políticas que impõem uma obrigação ou simplesmente uma recomendação de uma determinada ação. Como exemplo de uma política de obrigação, é possível especificar uma regra que obrigue o usuário a esperar 3 minutos até a próxima tentativa caso erre duas vezes a senha do seu *login*.

O OrBAC, por ser um modelo genérico, é possível utilizá-lo para modelar diversos cenários em diversas áreas de conhecimento, como por exemplo redes de computadores. A utilização de um modelo genérico para especificar políticas pode ser

adotada facilmente em diversos projetos. Este Trabalho de Graduação tem como objetivo realizar um estudo de caso de políticas de segurança aplicados sobre o contexto de redes de computadores utilizando o modelo OrBAC[2], pelos motivos citados anteriormente, comprovando a capacidade do modelo em especificar o cenário apresentado no caso de uso, utilizando o MotOrBAC[3], um editor para políticas de segurança no modelo OrBAC.

Os demais capítulos deste Trabalho de Graduação estão assim distribuídos, Capítulo 2 descreve o estado da arte em duas subseções: Gerenciamento de redes de computadores baseados em políticas e Modelos de Controle de Acesso. O Capítulo 3 introduz o conceito do modelo OrBAC e detalha as relações e entidades utilizadas pelo modelo. O Capítulo 4 apresenta a ferramenta MotOrBAC e a OrBAC API. O Capítulo 5 apresenta o estudo de caso desenvolvido no cenário de Redes de Computadores. O Capítulo 6 traz algumas considerações finais sobre o trabalho e detalha possíveis trabalhos futuros.

2. CONCEITUALIZAÇÃO

Neste Capítulo será abordado o processo de gerenciamento de políticas de controle de acesso. Ele será subdividido em 2 seções, na primeira será descrita resumidamente a arquitetura do sistema que fará o controle do gerenciamento das política em uma visão macro. A segunda seção deste capítulo é focada apenas nos modelos de controle de acesso que podem ser implementados para gerenciar as políticas definidas. Existem diversos modelos para controle de acesso na literatura, porém este trabalho irá descrever apenas três dos mais clássicos modelos encontrados na literatura, além do modelo OrBAC.

2.1. GERENCIAMENTO DE REDES DE COMPUTADORES BASEADO EM POLÍTICAS

O gerenciamento de redes de computadores em grandes empresas não é uma tarefa trivial. Alterações tanto na topologia da rede quanto na configuração dos seus componentes, como roteadores e servidores, podem aumentar ainda mais a complexidade do gerenciamento. Gerenciamento de redes de computadores baseado em políticas simplifica e automatiza o processo de administração de acesso a informações do sistema por parte de usuários e dispositivos da rede. A seguir será descrito um pouco sobre uma arquitetura desenvolvida para Gerenciamento de políticas no contexto de redes.

- **ARQUITETURA**

O IETF Policy Framework foi proposto pelo IETF Working Group como uma abordagem para gerenciamento de políticas na internet. O IETF Working Group criou um protocolo chamado Common Open Policity Service (COPS) [4] para integrar os serviços do framework. Um dos grupos que estudam gerenciamento de redes baseado em políticas, o Grupo de trabalho Resource Allocation Protocol(RAP) definiu um framework que consiste de interfaces para a definição das políticas, repositórios

para o armazenamento das políticas, um módulo para a interpretação das políticas contidas no repositório e um módulo para a aplicação destas políticas, conforme ilustrado na Figura 1. A seguir será descrito cada módulo resumidamente.

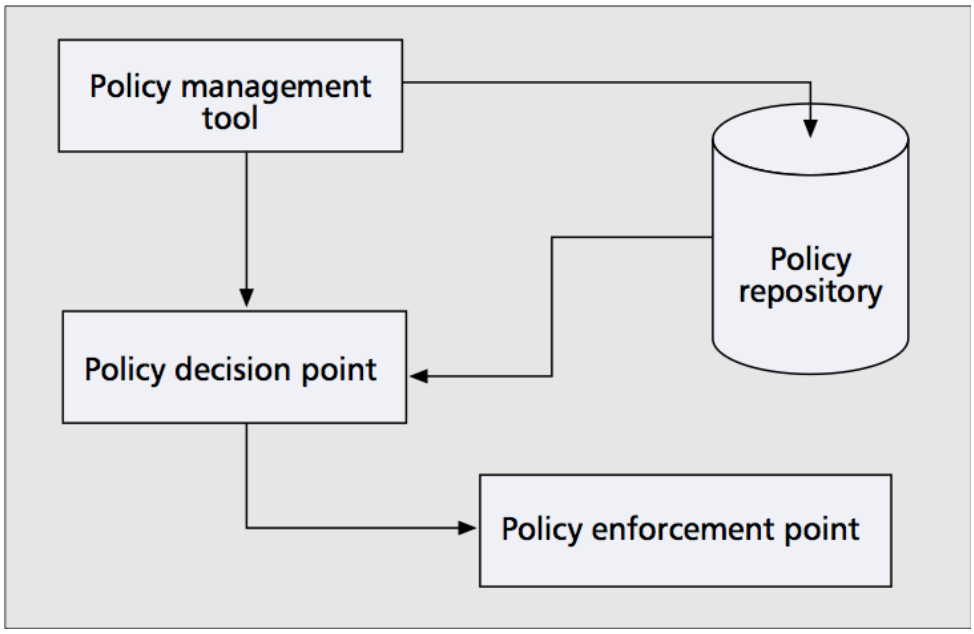


Figura 1. O framework de políticas IETF definido pelo RAP.

O *Policy Management Tool*, ou Ferramenta de Manipulação de Políticas, é o modulo responsável por receber as definições de políticas dos usuários ou administradores e converte-las em informações mais detalhadas e precisas para que seja realizado o controle das políticas[5]. Isto deve ser feito validando sintaticamente e semanticamente as políticas inseridas pelo usuário, certificando-se de que as políticas se aplicam, dadas as capacidades e restrições da rede. Este módulo também será responsável, no caso de um cenário em redes de computadores, pela definição de topologia da rede, dos usuários, das aplicações operacionais e outros fatores envolvidos no processo gerenciado pelas políticas. Ele também funciona como um monitor do estado da rede gerenciada pelas políticas.

A Ferramenta de Manipulação de Políticas armazena as políticas no *Policy Repository*, ou Repositório de Políticas, o qual provê uma localização central que orienta as operações na rede. Diversas abordagens para o armazenamento das políticas podem ser utilizadas, como o uso de um servidor de banco de dados ou servidores *Web*. Outra opção possível é utilizar um diretório LDAP, conforme sugerido pela IETF[6].

Policy Decision Points, ou pontos de decisão de políticas, são pontos que devem processar as políticas e informações relacionadas ao contexto do sistema para decidir qual política será aplicada na situação. PDP são construídos utilizando políticas de segurança através de modelos de controle de acesso, como os que serão descritos na próxima seção, e implementando mecanismos para processar requisições dos PEPs (*Policy Enforcement Points*, ou pontos de execução de políticas).

Depois da análise do PDP, os dados necessários para a configuração dos PEPs são recuperados do repositório e transformados no formato adequado, contendo as respostas sobre permissões de acesso, para que então possam ser aplicados nos PEPs.

Os PEPs são os componentes que são responsáveis pela aplicação das políticas definidas nos passos anteriores. Este módulo pode aparecer como um ou muitos componentes na rede, e deve controlar todos os dispositivos físicos os quais as políticas são aplicadas. PEPs também devem informar aos PDPs informações de estado da rede sempre que necessário, para que seja avaliada uma possível tomada de decisão. No caso de redes de computadores, componentes como *firewall* e roteadores frequentemente fazem o papel dos PEPs.

2.2. MODELOS DE CONTROLE DE ACESSO

O propósito de controle de acesso é limitar as ações ou operações que um usuário legítimo de um sistema computacional pode realizar. Controle de acesso é o processo de verificar todas as solicitações dos dados e recursos gerenciados pelo sistema e determinar em que circunstâncias as solicitações devem ser aceitas ou negadas. O mecanismo que fará a decisão para permitir ou negar o acesso é implementado através de um modelo de controle de acesso. A seguir será descrito alguns dos modelos mais conhecidos.

- **DAC – CONTROLE DE ACESSO DISCRICIONÁRIO**

O DAC foi um modelo de controle de acesso baseado na ideia de que cada usuário deve determinar quais os direitos de acesso que outros usuários ou aplicações do sistema possuem sobre as informações que são de suas responsabilidades. Diferentes modelos de DAC foram propostos na literatura[7], sendo um deles o *Access Matrix Model*, ou Matriz de Controle de Acesso, representado na Figura 2. Este modelo baseado em matriz foi proposto por Lampson[8] para proteção de recursos em sistemas operacionais, e posteriormente sofreu contribuições de Graham e Denning[9], até que foi formalizado por Harrison, Ruzzo, e Ullmann (*HRU model*) [10]. O modelo DAC é baseado na identificação do solicitante de acesso e nas regras explícitas de acesso que estabelecem quem pode ou não pode ter acesso a determinado recurso. Neste modelo, o proprietário(*own*) do recurso estabelecerá o direito de acesso, podendo inclusive haver uma transferência desta propriedade. Modelos discricionários são considerados inadequados para diversas aplicações, uma vez que são relativamente fracos e demasiadamente flexíveis: basta um equívoco por parte de um usuário inocente (ou um ato deliberado de um usuário malicioso) e informações importantes podem ser indevidamente reveladas, alteradas ou destruídas[11]. Outras informações sobre a vulnerabilidade do modelo DAC podem ser encontradas em [7].

	Arquivo 1	Arquivo 2	Arquivo 3	Programa 1
Maria		read		
João	own read write	read write		execute
Rafael		read	own read write	read

Figura 2. Matriz de Controle de Acesso.

- **MAC – CONTROLE DE ACESSO OBRIGATÓRIO**

O Modelo MAC[12], [13] foi criado através de pesquisas financiadas pelo Departamento de Defesa(DoD – *Department of Defence*) dos Estados Unidos e é baseado em práticas de segurança utilizadas no contexto de segurança nacional americana. A forma mais comum de políticas MAC é a *multilevel security policy*, ou política de segurança multinivel, baseado na classificação de *subjects* e *objects*. *Subject* são entidades ativas que solicitam acesso aos *objects*. Já os *objects* são entidades passivas que armazenam informações. Este modelo é caracterizado pela atribuição de *sensitivity label* (rótulos de sensitividade) aos *subjects* e *objects* do programa. O rótulo de segurança do usuário especifica o *sensitivity level* (nível de sensitividade) associado ao usuário. O nível de segurança aplicado ao *object* reflete o potencial dano que um acesso não autorizado à informação causará. O nível de segurança aplicado ao *subject* reflete a confiança que se aplica ao *subject* para disponibilizar dados críticos.

- **RBAC – CONTROLE DE ACESSO BASEADO EM PAPEIS**

RBAC é um mecanismo de modelo de controle de acesso não discricionário que permite e promove a administração central de uma política de segurança especifica de organização [14].

A principal motivação no modelo RBAC é que a política de segurança não concede permissões para usuários, e sim para papeis. Um usuário poderá herdar de um papel as permissões associadas ao mesmo se o usuário estiver associado ao papel, e então o usuário terá os mesmos privilégios de todos os usuários que compõem aquele *role (papel)*, conforme representado na Figura 3. O conceito de hierarquia de papeis também é introduzido neste modelo, onde é possível que papeis herdem permissões de outros papeis na hierarquia. Neste modelo também é possível especificação de restrições. Neste caso, é possível especificar que nenhum usuário poderá exercer dois determinados papeis.

De fato, para controle de acesso é mais importante saber qual as responsabilidades do usuário na organização em relação a sua identidade. Logo, em relação ao modelo DAC convencional, onde o usuário individual possui a propriedade

do dado no sistema e concede a permissão para outros usuários, o modelo RBAC apresenta uma vantagem. Também é possível verificar a vantagem do RBAC em relação ao MAC, onde *subjects* e *objects* são classificados por níveis de segurança.

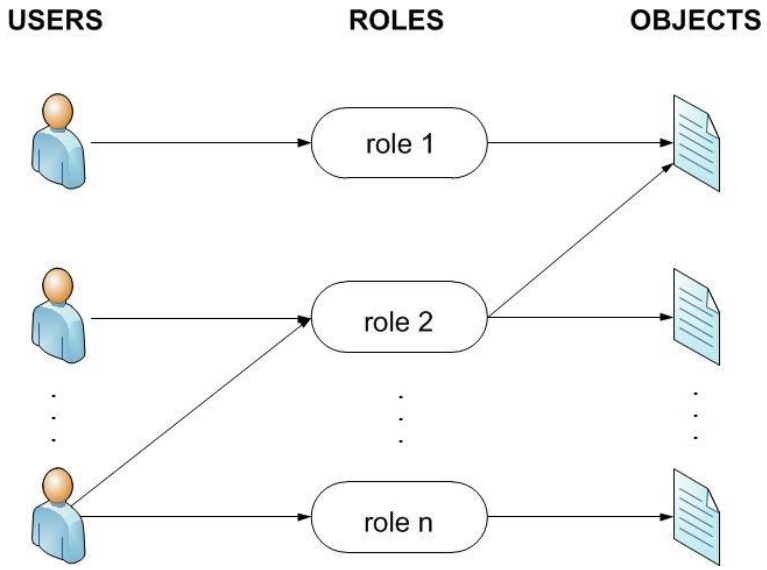


Figura 3. Representação da classificação com roles (papeis) no modelo RBAC.

- **ORBAC – CONTROLE DE ACESSO BASEADO EM ORGANIZAÇÃO**

O modelo OrBAC foi desenvolvido com o intuito de superar limitações dos modelos de controle de acesso anteriores introduzindo o conceito de **organizações** junto com o conceito de **contexto** [1]. OrBAC especifica política de segurança visando a definição ou o gerenciamento de organizações. Através deste modelo, não só empresas mas também componentes, como um *firewall* ou uma *Java Virtual Machine* (Máquina Virtual Java) podem ser modelados. O modelo OrBAC é representado através de um modelo entidade-relacionamento e da associação com atributos. O modelo OrBAC também utiliza o conceito de *role*, porém traz outros conceitos não utilizados antes, como o de *activity* (atividade) e *view* (conjunto de objetos). *Role*, *activity* e *view* são entidades classificadas como parte do *abstract level* (nível abstrato). Através do nível abstrato é possível inferir o *concrete leve*l (nível concreto), e as permissões definidas no nível abstrato serão atribuídas às entidades do nível concreto. No nível concreto são definidas as entidades s*ubject*, *action* e o*bject*, as quais são relacionadas com as entidades *role*, *activity* e *view*, respectivamente. A

Figura 4 mostra uma simples representação do modelo OrBAC, o qual será mais detalhado no próximo capítulo.

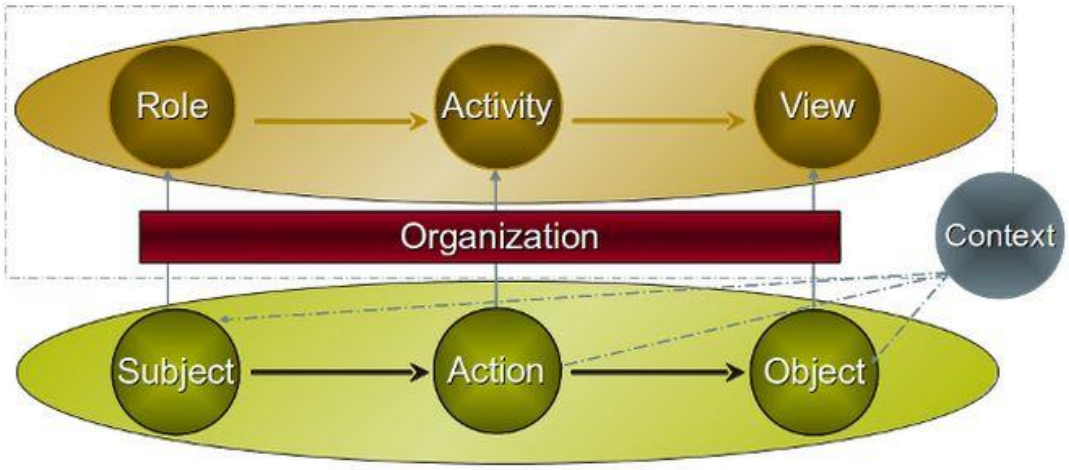


Figura 4. Representação do modelo OrBAC.

- **DISCUSSÃO**

As diferenças entre as necessidades governamentais e comerciais levaram ao desenvolvimento de dois mecanismos distintos de controle de acesso: Controle de Acesso Obrigatório (MAC) e Controle de Acesso Discricionário (DAC). MAC se concentra no controle da divulgação de informações através da atribuição de níveis de segurança de *objects* e *subjects*, limitando o acesso entre os diferentes níveis de segurança. Por outro lado, DAC se concentra no controle de acesso com granularidade fina de objetos. Isso se deve a um controle de acesso baseado em matrizes e modos de permissão a nível de objeto.

Tanto o MAC quanto o DAC possuem algumas limitações graves. A atribuição e aplicação de níveis de segurança do sistema de acordo com o modelo MAC coloca restrições sobre as ações do usuário que, ao aderir às políticas de segurança, impede a alteração dinâmica das políticas, dificultando substancialmente a tarefa de realizar alterações especificas em determinado *subject* ou *object*.

Já quanto aos sistemas DAC, são desconsiderados em muitas situações onde a segurança da informação é essencial, pois permitindo aos usuários controlar as permissões de acesso aos objetos tem um efeito colateral de abrir o sistema. Uma das vulnerabilidades mais tradicionais do DAC é o ataque Cavalo de Tróia[15]. Além disso, a manutenção dos sistemas e verificação das políticas de segurança é extremamente difícil em sistemas DAC, pois será necessário determinar regras especificas para cada usuário, além dos problemas quando o usuário possui a propriedade do objeto.

Em relação ao modelo RBAC, este ainda possui algumas desvantagens. Este modelo não permite a especificação de permissões atribuídas a contexto. Em outras palavras, se uma dada permissão for concedida a um determinado papel, todos os usuários que exercem este papel terão a permissão concedida. Logo, não é possível especificar que um médico poderá acessar os dados de um paciente a menos que o paciente seja atendido por este médico.

A escolha do modelo OrBAC para o estudo de caso realizado neste trabalho deve-se as vantagens para a elaboração de políticas de segurança, comparando com os três modelos mais tradicionais citados neste capítulo. Conforme abordado na seção deste capítulo que trata do OrBAC, a estrutura com a qual o modelo especifica as entidades possibilita um poder de representação de diversos cenários de uma forma mais flexível a potenciais manutenções das políticas de segurança.

Outra diferença notável do modelo OrBAC em relação aos outros modelos apresentados é a estrutura genérica com que o modelo representa não só permissões, mas também proibição e obrigação. Assim como o OrBAC, o modelo RBAC possui suporte para estruturar os cenários utilizando o conceito de Organização, porém não com muitas organizações como o OrBAC suporta.

3. ORBAC – UM MODELO PARA CONTROLE DE ACESSO

Neste Capítulo do trabalho será detalhado os principais conceitos do modelo teórico OrBAC.

3.1. ORGANIZAÇÃO

A entidade mais importante do modelo OrBAC é a entidade *organization*. No domínio de políticas de segurança para empresas de grande ou médio porte, podemos modelar como organização cada sede daquela empresa, podendo também aumentar a granularidade para cada departamento da empresa. Também é possível modelar a entidade *organization* através de uma estrutura hierárquica. Utilizando o mesmo domínio de aplicação, é possível estruturar hierarquicamente as sedes por suas posições geográficas na região.

3.2. ENTIDADES ABSTRATAS E CONCRETAS

O modelo OrBAC é dividido em dois níveis de entidades: Abstrato e Concreto.

3.2.1. NÍVEL ABSTRATO

No nível abstrato, o modelo OrBAC utiliza as entidades *role*, *activity* e *view* para modelar os predicados permissão, proibição ou obrigação sem especificar os atributos que serão contemplados pelos predicados. Políticas de Controle de acesso não são definidas sobre entidades concretas como *subject*, *action* ou *object*, mas sim sobre entidades abstratas. Com isso, o modelo OrBAC introduziu o conceito de *activity* e *view*. Logo, políticas de controle de acesso definem permissões (ou proibições ou obrigações) para controlar as atividades exercidas por um determinado grupo de usuários sobre um conjunto de recursos.

3.2.2. NÍVEL CONCRETO

O modelo OrBAC define o nível concreto através das entidades s*ubjects*, a*ctions* e *objects*. Através de predicados como *Empower*, *Consider* e *Use*, é possível relacionar *subjects* com *roles*, *action* com *activities* e *views* com o*bjects*, respectivamente. Com isso, os privilégios associados às entidades abstratas são atribuídos também às entidades concretas.

3.3. SUBJECTS E ROLES

Subject é uma entidade geralmente utilizada no modelo OrBAC para modelar entidades ativas como pessoas, usuários, grupos ou papeis dentro de uma organização. Já a entidade *role* é uma forma de vincular s*ubject* e *organization*. Um exemplo comum é criar cargos dentro de uma empresa como *roles* e associar os funcionários aos s*ubjects* de acordo com seus cargos. Embora não seja usual, é possível utilizar s*ubjects* e *roles* em outros cenários, como modelar papeis dos servidores e servidores específicos que executam estes papeis. Como exemplo do cenário anterior, “Servidor de Banco de Dados” e “Servidor de Aplicação X “podem ser associados a *roles*. “SRV_BD1“ e “SRV_BD2“ podem ser modelados como um s*ubject* relacionado ao papel “Servidor de Banco de Dados”.

Para que haja uma associação entre o nível concreto e abstrato o modelo define uma *relationship* (relação) chamada *Empower*. Assim, dada uma *organization* org, um *subject* s e um r*ole* r, então *Empower (org, s, r)*, significa que a organização org delega o papel r ao indivíduo s. A relação *Empower* é representada na Figura 5. Como um exemplo deste relacionamento, entende-se como *Empower (UFPE, João, professor adjunto)*: João é professor adjunto na UFPE.

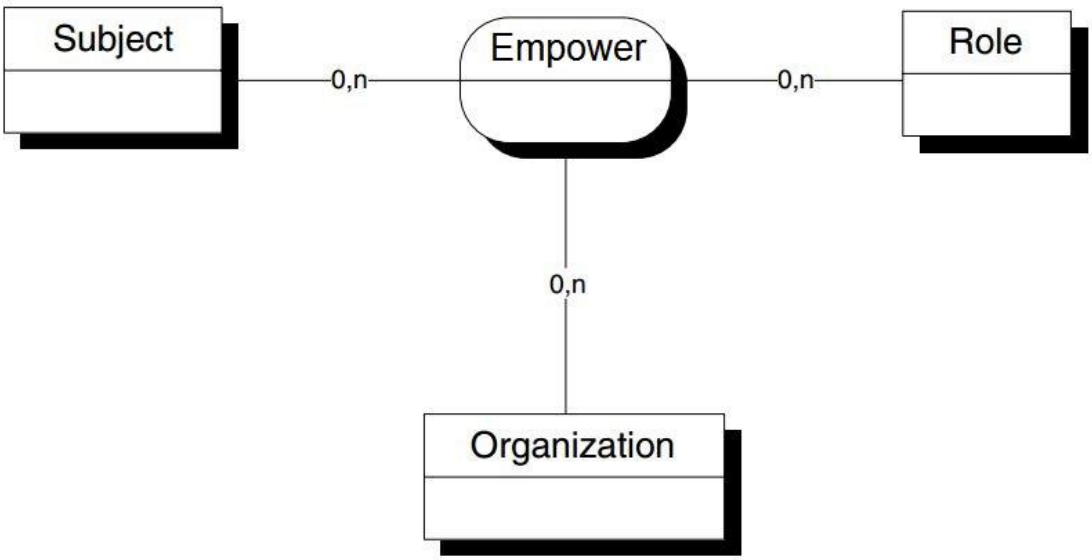


Figura 5. Modelo da relação Empower.

3.4. OBJECTS E VIEWS

Para modelar entidades inativas, como arquivos, programas executáveis e dispositivos da rede, o modelo OrBAC utiliza uma entidade denominada *object*. Assim como a entidade abstrata *role* foi definida sobre a entidade concreta *subject*, a entidade *view* é um forma de vincular *objects* e *organization*. Principalmente no cenário empresarial, *views* podem ser utilizados para modelar conjuntos de objetos com propriedades em comum, como por exemplo: “arquivos digitais”, “dispositivos eletrônicos” e “computadores da sala 4”. Neste mesmo cenário, *objects* modelam as instâncias das *views* especificadas e têm como exemplos: “Computador S4C2”, “Computador S4C8“ e “Relatório Mensal da Diretoria“.

Assim como para *roles* e *subjects*, *objects* e *views* devem ser associadas para que seja possível identificar a relação concreta e abstrata. Neste caso, foi definida uma relação chamada *Use*. Similar ao *Empower*, para uma dada *organization* org, *object* o e *view* v, então *Use(org, o, v)* significa que a organização org classifica um objeto o como pertencente ao conjunto v. A relação *Use* é representada na Figura 6. Assim, uma relação *Use(UFPE, Computador S4C2, computadores da sala 4)* entende-se: o Computador S4C2 pertence aos computadores da sala 4 na UFPE.

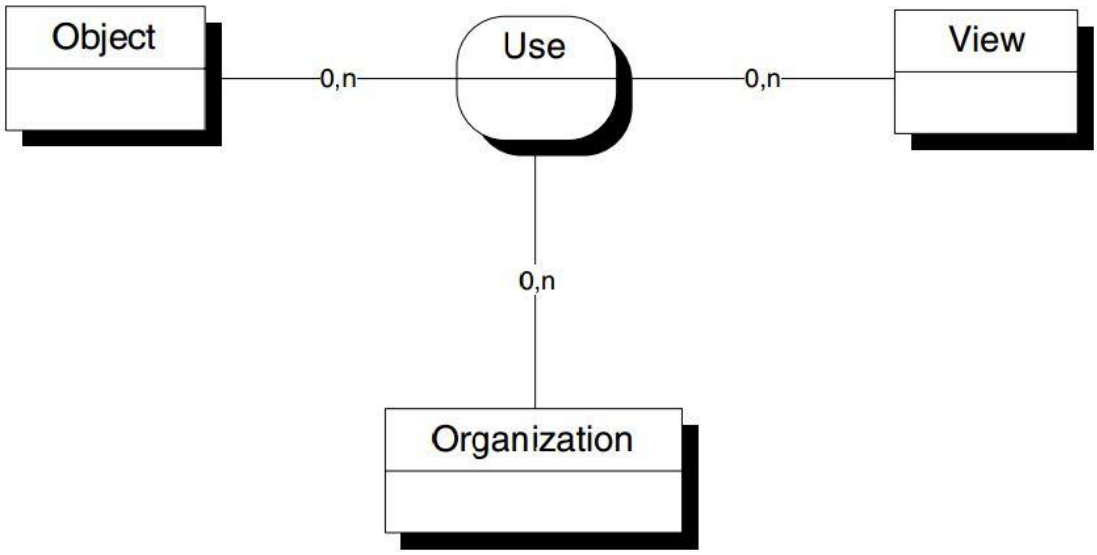


Figura 6. Modelo da relação Use.

3.5. ACTIONS E ACTIVITIES

No modelo OrBAC, as políticas de segurança são aplicadas para autorizar acesso em entidades inativas, *objects*, por entidades ativas, *subjects*, e regular a ação tomada pelo sistema. A entidade concreta *action* modela as ações que serão aplicadas aos *objects* pelos *subjects*. E assim como para *subject* e *objects*, *action* possui um nível abstrato denominado *activity*. A entidade abstrata *activity* busca categorizar *actions* que compartilham do mesmo princípio. Devido à maioria dos processos existente nas empresas atualmente serem informatizados, *activities* sobre o contexto de ações de computadores como “leitura“ e “escrita“ são exemplos mais comuns. Para *action*, “ler” e “consultar“ são exemplos de ações concretas em um cenário computacional.

Similar as relações *Use* e *Empower* descritas anteriormente, *Consider* é uma relação ternária para associar os níveis abstrato e concreto utilizando as entidades *organization*, *action* e *activity*. Logo, dada uma *organization* org, uma *activity* a e uma *action* α, a relação *Consider (org, α, a)* significa que a organização org categoriza a ação *α* como atividade *a*. A relação *Consider* é representada na Figura 7. Logo, como exemplo em um contexto de biblioteca universitária, *Consider (UFPE, empréstimo semanal, empréstimo)* entende-se: Empréstimo Semanal é uma atividade de empréstimo na UFPE.

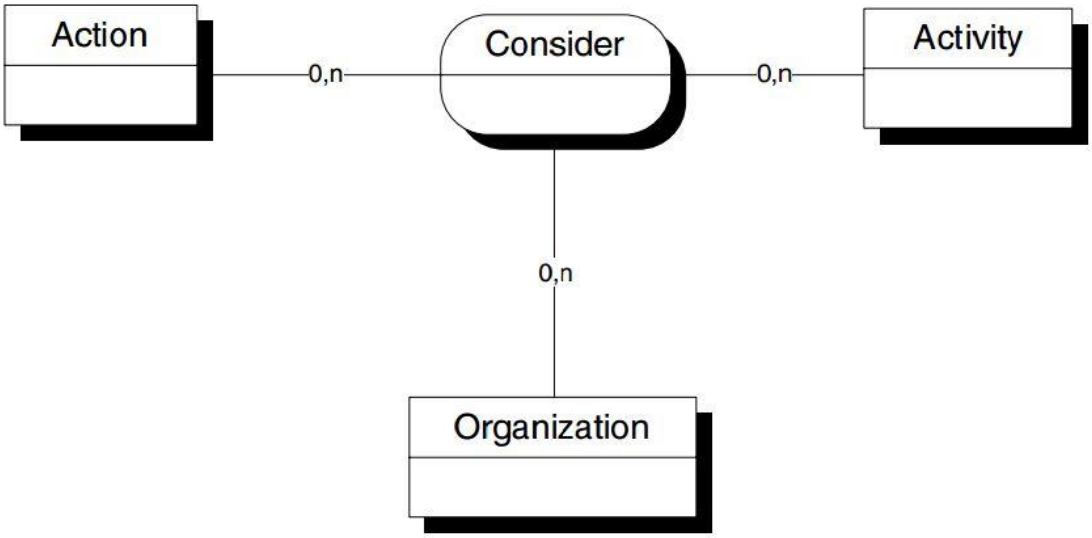


Figura 7. Modelo da relação Consider.

3.6. CONTEXTO

Context, ou contexto, é uma entidade criada para especificar condições que determinam se *roles* têm permissão para realizar *activities* em *views*. Está entidade será formada por uma expressão lógica que resulte em verdadeiro ou falso. Com isso, é possível modelar circunstâncias como “horário de trabalho“ e “dia útil“, e defini-las através de um Contexto.

Para a definição de *context* foi definido a relação *Hold*. Esta relação determina se o contexto está aplicado a determinado *subject*, *action* e o*bject*. Assim, dado que org é uma *organization*, sub é um *subject*, act é um *action*, obj é um o*bject* e ctx é um *context*, a relação *Hold(org, sub, act, obj, ctx)* significa que na organização org, o contexto ctx é aplicado na tupla (sub, act, obj).

3.7. TIPOS DE CONTEXTO

Em [16], são definidos cinco tipos de contextos (*temporal*, *spatial*, *user-declared*, *prerequisite*, *provisional*). Cada tipo de contexto é ideal para ser aplicado

em determinada situação. T*emporal context* é um tipo de contexto aplicado para definir políticas atribuídas durante um intervalo de tempo. Como continuação do exemplo mencionado anteriormente, “horário de trabalho“ pode ser definido como 07:00 até 19:00, e “dia útil“ como todos dias da semana exceto sábados, domingos e feriados.

Spatial context são usadas para definir restrições ligadas a localização onde o usuário faz a requisição de acesso. É possível exemplificar o *spatial context* em uma regra no domínio da universidade, onde um aluno pode ter acesso aos matérias virtuais da biblioteca apenas se ele se encontrar nas dependências da universidade. Outro exemplo válido para *spatial context* é modelar uma localização lógica. No domínio de redes de computadores, um recurso pode ter seu uso restrito caso o usuário não esteja em uma determinada sub-rede.

User-declared context é um tipo de contexto aplicado a situações em que é necessário analisar um objetivo ou propósito de um *subject*. No domínio de um hospital, é possível restringir o acesso dos assistentes aos relatórios dos pacientes apenas para casos de urgência. Um possível *user-declared context* neste caso é um contexto que verifica se o paciente foi declarado como caso de urgência para liberar o acesso.

Prerequisite context é um tipo de contexto quando for necessário o cumprimento de um requisito para obter o privilégio, e é necessário checar este pré-requisito em um banco de dados. No domínio empresarial, um auxiliar de escritório recebe o privilégio para acessar o relatório de produtos caso o gerente não esteja presente. Para isso, é assumido que se tenha um controle da presença dos funcionários no escritório através do sistema.

Finalmente, o *provisional context* é um tipo de contexto para modelar cenários onde será necessário realizar uma obrigação anteriormente ou posteriormente – dependendo da especificação do contexto – para que a regra principal seja ativada. No domínio hospitalar, um cenário de exemplo do *provisional context* é uma enfermeira que é obrigada a lavar as mãos para que tenha acesso as dependências onde se encontram os pacientes. A Figura 8 ilustra a classificação dos contextos e os tipos de dados que são representados.

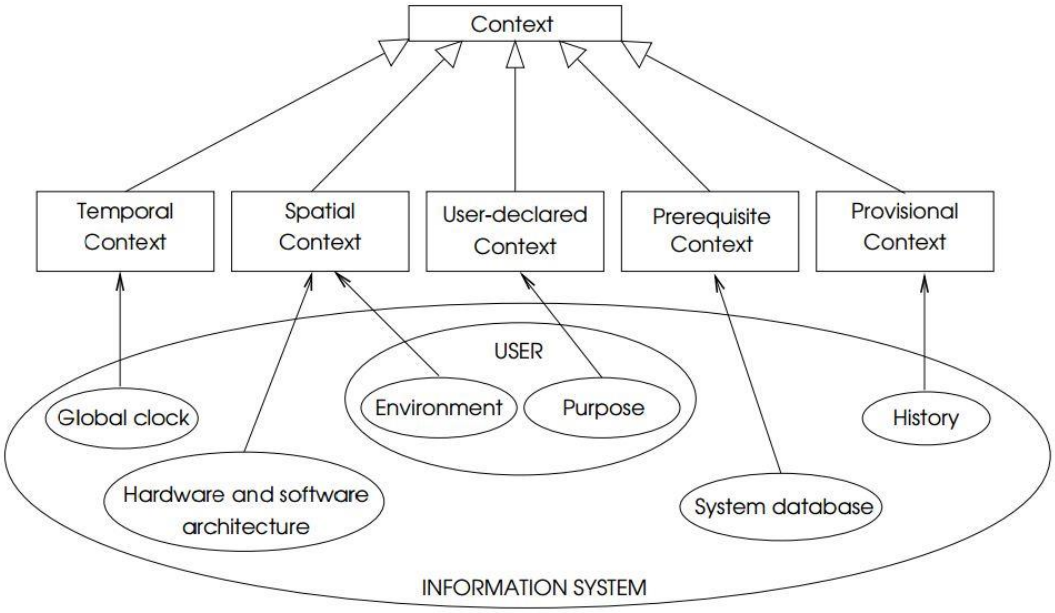


Figura 8. Classificação de contextos e suas informações.

3.8. POLÍTICAS DE SEGURANÇA

Durante as secções anteriores, foram definidas algumas entidades que compõem o modelo OrBAC. Será descrito então as relações utilizadas nos modelos para descrever as regras que cada cenário irá seguir. São definidas quatro formas de controle de acesso às entidades inativas por entidades ativas: *Permission*, *Prohibition*, *Obligation* e *Recommendation*. Se org é uma *organization*, r é um *role*, a é uma *activity*, v uma *view* e c um *context*, *Permission(org, r, a, v, c)* significa que a *organization* org concede permissão para o *role* r fazer a *activity* a em uma *view* v sobre o *context* c. Seguindo o mesmo princípio, *Prohibition(org, r, a, v, c)* significa que a *organization* org proíbe para o *role* r fazer a *activity* a em uma *view* v sobre o *context* c, *Obligation(org, r, a, v, c)* significa que a *organization* org obriga o *role* r fazer a *activity* a em uma *view* v sobre o *context* c e *Recommendation(org, r, a, v, c)* significa que a *organization* org recomenda o *role* r fazer a *activity* a em uma *view* v sobre o *context* c. Assim, uma relação *Permission(UFPE, Aluno, Alugar, Livro, Matriculado)* entende-se: na organização UFPE, alunos podem alugar livros desde que estejam matriculados.

3.9. AUTORIZAÇÕES CONCRETAS

Na seção anterior foi descrito os relacionamentos que definem as políticas de Segurança no modelo OrBAC. No entanto, estes relacionamentos são definidos no nível abstrato, e precisam ser derivados para o nível concreto. Assim, a partir de cada relação *Permission*, *Prohibition*, *Obligation* e *Recommendation* é gerado uma relação Is*_permitted*, *Is_prohibited*, *Is_obliged* e *Is_recommended*, respectivamente. Uma permissão concreta é dada por *Is_permitted(Subject, Action, Object)*, onde está relação será inferida das relações *permission(Org, Role, Activity, View, Context)*, *empower(Org, Subject, Role)*, *consider(Org, Action, Activity)*, *use(Org, Object, View)* e *hold(Org, Subject, Action, Object, Context)*. As inferências para *Is_prohibited*, *Is_obliged* e *Is_recommended* são similares ao *Is_permitted*.

3.10. RESTRIÇÕES

Contraints (restrições) foram definidas no modelo RBAC e são utilizadas com regras aplicadas às várias relações[1]. É possível criar restrições que impeçam que um mesmo *subject* tenha mais de um *role* ou que um *role* possua mais de um *subject*. Como exemplo, é possível representar o cenário onde apenas um usuário pode ser intitulado diretor executivo.

3.11. GERENCIAMENTO DE CONFLITO

A medida que as regras são definidas durante a construção do cenário no modelo OrBAC, é possível que apareçam regras que possuem uma potencial chance de conflito ou regras redundantes. Em [17] é sugerido o gerenciamento de conflitos usando prioridades e é definida uma abordagem para gerenciar potenciais conflitos. Desde que um *subject* pode potencialmente ser associado a dois *roles* diferentes, um *object* à duas *views* diferentes, um *action* à duas *activities* diferentes e dois *contexts* diferentes podem ser simultaneamente satisfeitos, todos os pares de Permission e Prohibition são potenciais conflitos. Assim, é possível eliminar potenciais conflitos através de *Separate Constraints*. Dado que exista uma *Separate Constraints* entre o

role r1 e o *role* r2, um mesmo *subject* não poderá ser associado à r1 e r2 simultaneamente. Logo, um par Permission-Prohibition onde uma das relações esteja associada a r1 e outra a r2 não apresentará mais um potencial conflito.

De forma similar, é possível criar *Separate Constraints* para *activities*, *views* e *context*. Em [17] também é definido uma estratégia para regras redundantes. Dado um cenário hospitalar, deseja-se que um enfermeiro seja proibido de ver a ficha dos pacientes, exceto em situação de emergência. Duas regras foram modeladas: Enfermeiros são proibidos de ler a ficha dos pacientes; e enfermeiros podem ler a ficha dos pacientes em uma situação de emergência. Neste cenário, a estratégia de *Separate Constraints* não se aplica, pois existe uma situação onde todas as entidades são semelhantes. Neste caso, deve-se indicar a prioridade de uma regra em relação a outra. Logo, caso a prioridade da segunda regra seja maior que a primeira, o modelo fica coerente com a proposta do cenário.

4. MOTORBAC E ORBAC API

Nesta seção será abordado o MotOrBAC version 2, uma ferramenta para definição e análise de políticas de segurança utilizando o modelo OrBAC, além da OrBAC application programming interface (API), utilizada na construção do MotOrBACv2. Para a implementação do caso de uso e ilustração do funcionamento da ferramenta foi utilizado o MotOrBAC versão 2.4.1.

Dada a crescente complexidade dos sistemas computacionais e a difícil tarefa de aplicar políticas de segurança neste contexto, o MotOrBAC[18] foi desenvolvido para especificação e administração de políticas no mesmo modelo. Ambos, MotOrBACv2 e OrBAC API, foram implementados em Java.

Muitos modelos de segurança possuem suporte para apenas um administrador escrever e manter as políticas de segurança do sistema. No entanto, devido aos sistemas computacionais se tornarem cada vez mais distribuídos, existe a necessidade de também distribuir a administração das políticas de segurança. A política de administração expressa que os usuários tenham privilégios para atualizar a política de segurança e as condições em que eles têm esses privilégios. O modelo de gestão do OrBAC é chamado AdOrBAC. AdOrBAC foi desenvolvido para expressar uma política de administração usando os mesmos conceitos introduzidos no modelo OrBAC. Isso faz com que o modelo OrBAC seja capaz de autoadministração.

Dentre as funcionalidades do MotOrBAC estão (1) a especificação de políticas baseado no modelo OrBAC, (2) a detecção de conflitos efetivos e conflitos em potencial, (3) a simulação de política de segurança e (4) a especificação de políticas de administração. Neste capítulo do trabalho serão descritas as três primeiras funcionalidades citadas. A quarta funcionalidade foi desenvolvida utilizando os conceitos do AdOrBAC[19], um modelo para administração do modelo OrBAC. Apesar do modelo AdOrBAC ser implementado no MotOrBAC, este trabalho não abordará a vertente de administração do modelo OrBAC, pois o trabalho propôs como objetivo apenas a especificação de um cenário de políticas de uma área de conhecimento especifica, sem o intuito de abordar a administração do mesmo.

O MotOrBAC foi desenvolvido para permitir a utilização do modelo OrBAC para administração de políticas. A arquitetura e a especificação inicial do MotOrBAC podem ser encontradas em [20]. MotOrBAC utiliza uma API, chamada OrBAC API, para

gerenciar as políticas e apresentar de forma amigável ao administrador através da GUI (Graphical User Interface).

O MotOrBAC é uma ferramenta de edição de políticas, que irá conter a interface necessária para o administrador de políticas realizar as operações de criação, edição e exclusão, tanto das entidades quanto dos relacionamentos do modelo OrBAC. Estas operações também podem ser realizadas através de programação utilizando a API em Java. O MotOrBAC também será responsável por apresentar ao administrador os conflitos de políticas detectados pelo *core* da aplicação, e será responsável por apresentar as alternativas que o administrador poderá escolher para eliminar conflitos de regras. Na Figura 9 é apresentada uma ilustração da arquitetura do MotOrBac e do *core* da ferramenta, o OrBAC API.

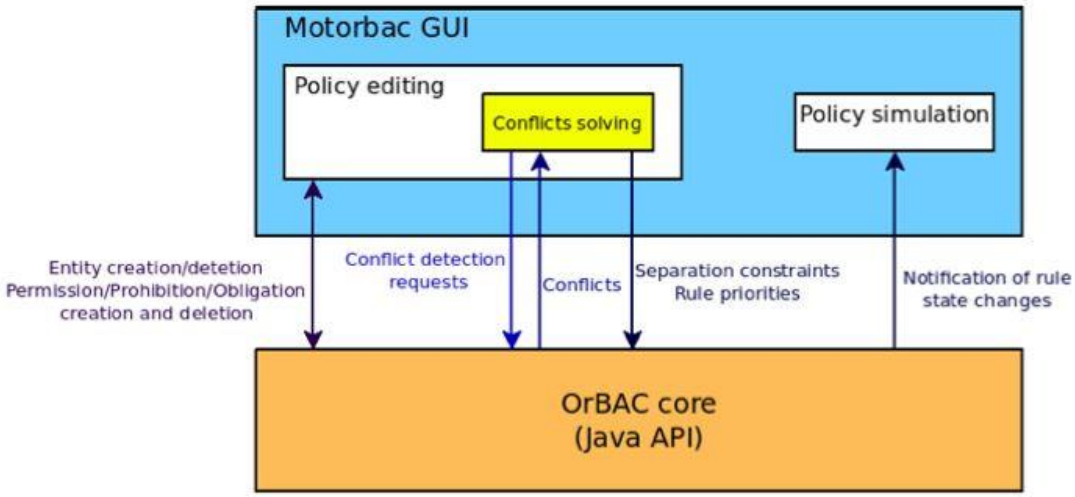


Figura 9. Arquitetura do MotOrBAC

- **APRESENTAÇÃO DA FERRAMENTA**

Conforme mencionado nos capítulos anteriores, o modelo OrBAC baseia sua estrutura em organizações. Cada organização possui um nível de independência com as outras. Entidades abstratas e as regras abstratas existem apenas sobre o domínio de uma determinada organização, podendo haver a herança de entidades e regras abstratas caso haja hierarquia de organizações. Na Figura 10 é mostrado um dos cenários disponíveis na ferramenta, a distribuição de *roles* de forma hierárquica, e sobre o domínio de uma organização especifica, “city_hospital”. Neste mesmo exemplo, podemos encontrar a distribuição hierárquica para *activies* e *views*. De

forma similar, regras abstratas também serão associadas apenas às organizações especificas onde elas foram definidas e as organizações que herdam da primeira.

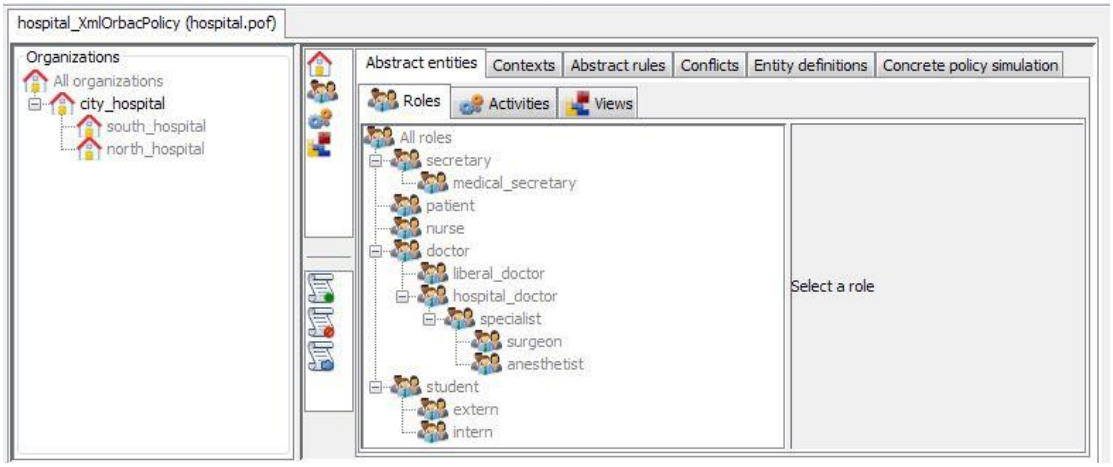


Figura 10. Roles associados à organização.

Conforme o modelo OrBAC, a ferramenta MotOrBAC também possibilita a especificação de contextos para as regras criadas. As políticas que atuam sobre determinado contexto só recebem o *status* de *active* quando o resultado lógico da definição do contexto for verdadeiro. Já quando o contexto tem falso como resultado de sua definição lógica, as regras recebem o *status* de *inactive*. Além disto, um único contexto pode ser representado em diversas organizações, e em cada uma delas o contexto pode ter uma definição diferente, no entanto apenas uma definição é permitida por organização. O MotOrBac na versão 2.4.1 possui 12 tipos de contextos[3], porém estes não serão detalhados neste trabalho.

Na aba *Abstract Rules* do MotOrBAC é possível definir as regras de permissão, proibição e obrigação entre as entidades. Seguindo o modelo OrBAC, para a definição dos relacionamentos (permissão, proibição e obrigação) é necessário a especificação do *role*, *activity*, *view* e contexto ao qual a regra se aplica.

A seguir nas abas do MotOrBAC é possível gerenciar os conflitos que podem surgir durante a elaboração das políticas. MotOrBAC distingue conflitos abstratos e concretos. Conflitos abstratos ocorrem entre regras, enquanto conflitos concretos envolvem instâncias concretas e refletem a concreta situação onde o conflito abstrato irá ocorrer[21].

Ainda na aba dos conflitos, o MotOrBAC apresenta duas soluções para o gerenciamento dos conflitos existentes. A primeira delas é a restrição de separação, ou *Separation Constraint*. Dependendo do tipo de conflito que ocorra, a inclusão de uma cláusula que restrinja duas instância diferentes de uma mesma entidade - como *role*, *activity*, *view* ou context - em uma mesma instância de entidade concreta poderá resolver o conflito. Como exemplo de uma situação onde é possível resolver um conflito utilizando *Separation Constraint*, é possível que em um hospital, diretores do hospital tenham acesso a sala do setor contábil, chamada “regra1”. No entanto, neste mesmo hospital, existe uma regra que proíbe que os médicos tenham acesso a sala do setor contábil, chamada “regra2”. No entanto, as duas regras em conjunto geram um conflito em potencial, pois caso exista algum diretor no hospital que também seja médico do hospital, ocorrerá um conflito de regras. Logo, para solucionar este conflito em potencial, é possível a inclusão de uma cláusula *Separation Constraint* entre diretor e médico no hospital. Na Figura 11 é mostrado como aplicar as funções de resolução de conflitos no MotOrBAC.

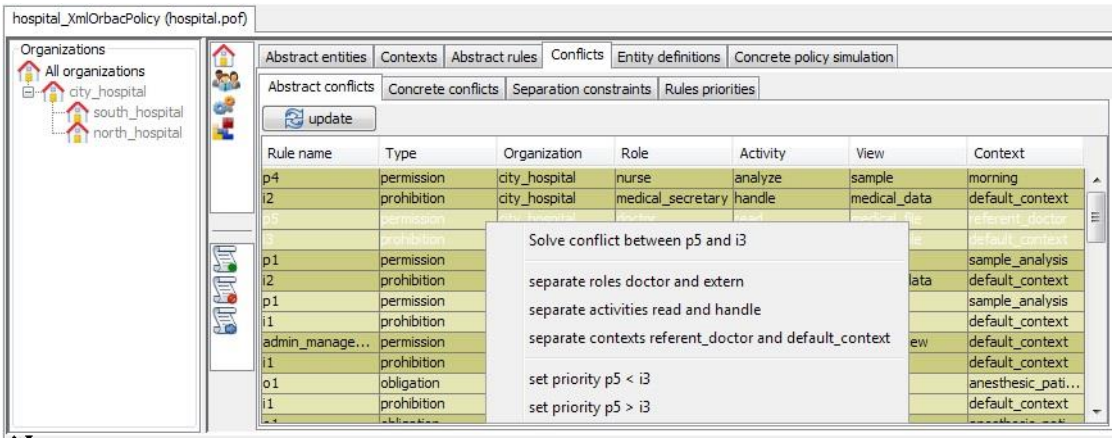


Figura 11. Resolução de Conflitos no MotOrBAC.

Outra solução possível para a solução de um problema de conflito é definir prioridades às regras. É possível através do MotOrBAC definir que uma regra especifica possui prioridade em relação a outra. Utilizando o mesmo exemplo do parágrafo anterior, é muito possível que a solução com *Separation Constraint*, embora válida sobre o ponto de vista do modelo OrBAC, não esteja adequada ao cenário exemplificado, pois possivelmente o diretor médico deverá continuar a exercer as duas funções. Neste caso, é possível determinar que a “regra1” possui maior prioridade que a “regra2”, e assim o conflito é resolvido.

Por fim temos a aba para simulação das políticas, conforme mostrado na Figura 12. Esta é a parte do MotOrBAC onde será possível visualizar quais políticas que estariam ativas e destacadas em verde, e quais estariam inativas e destacadas em vermelho no dado momento. Nesta simulação serão considerados os fatores que afetam diretamente as regras, como por exemplo, o aspecto temporal, caso exista em algumas das regras definidas.

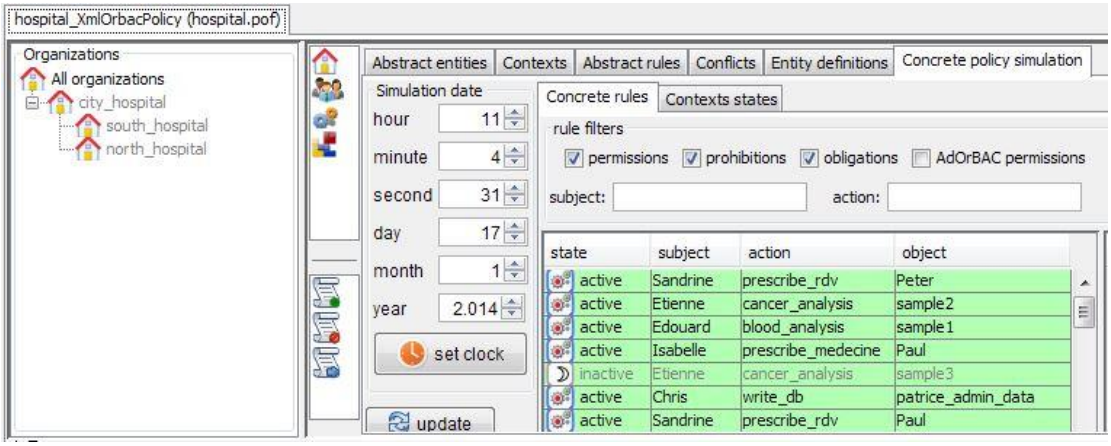


Figura 12. Exemplo de Simulação.

As políticas são estruturadas internamente pelo MotOrBAC em uma estrutura de grafos RDF, e são armazenadas em um arquivo XML, onde é possível a exportação de forma fácil para outras aplicações[22].

5. CASO DE USO COM O MODELO ORBAC

Neste Capítulo do trabalho será detalhado o cenário utilizado como caso de uso, e o processo de simulação do cenário utilizando a ferramenta MotOrBAC.

• DESCRIÇÃO DO CENÁRIO DE CASO DE USO

Conforme já mencionado anteriormente, o OrBAC é um modelo genérico, e por isso pode ser utilizado na especificação de cenários em diversas áreas. Nesta seção será apresentado um cenário na área de redes de computadores para a demonstração de caso de uso do modelo OrBAC. Este cenário foi inspirado no modelo funcional da ilha de automação da companhia CHESF, porém não reflete o ambiente real da companhia.

Neste cenário, o modelo organizacional da empresa é dividido em uma matriz principal e três filiais representando os estados da Bahia, da Paraíba e do Rio Grande do Norte. Em cada uma das filiais existem operadores e supervisores dos operadores. Um operador ou um supervisor de uma determinada filial possui privilégios de acesso apenas nesta filial, não sendo permitido o acesso do mesmo às outras filiais ou à matriz. Também existem Administradores do sistema, que são ligados diretamente à matriz e em decorrência disto têm acesso às filiais.

Cada filial acessa a rede corporativa da matriz, e a partir de então o usuário da filial é direcionado para a VLAN adequada para ele de acordo com as políticas de segurança da organização. O acesso aos dispositivos da rede corporativa pelas filiais é feita através de um acesso pelos terminais que existem na própria filial, ou por VPN. No caso do acesso via terminal, operadores têm um acesso restrito, enquanto supervisores e administradores possuem um acesso completo. Já para o acesso via VPN, apenas os administradores possuem privilégios nesta forma de acesso.

Neste cenário hipotético, a rede corporativa da empresa é dividida em três VLANs, denominadas VLAN1, VLAN2 e VLAN3. De acordo com as políticas da companhia, operadores podem ter acesso apenas a VLAN1, supervisores podem ter acesso à VLAN1 e à VLAN2, e os administradores têm total acesso à rede corporativa.

O acesso ao ambiente corporativo, ou seja, utilização dos terminais, é restrito ao horário de expediente, das 8 às 12 horas e das 14 às 18 horas. Outro requisito necessário é que seja proibido qualquer acesso que não seja estreitamente permitido nas políticas de segurança da companhia.

- **SIMULAÇÃO DO CENÁRIO NO MOTORBAC**

Nesta seção será demonstrado como o cenário descrito anteriormente foi modelado na ferramenta MotOrBAC. Inicialmente foram definidas quatro *organizations*, denominadas “Matriz”, “Filial RN”, “Filial PB” e “Filial BA”. Como é um requisito do cenário descrito anteriormente a separação lógica entre as filiais e a ligação da matriz com as filiais – dado que todos da matriz têm acesso as filiais – foi criada uma relação de hierarquia entre a matriz e as filiais. Foram criados quatro *roles* na ferramenta, denominados “Visitante”, “Administrador”, “Operador” e “Supervisor do Operador”. Os *roles* “Visitante” e “Administrador” foram associados à matriz, e por consequência às filiais. Os *roles* “Operador” e “Supervisor” foram criados apenas nas filiais. A Figura 13 apresenta a modelagem dos *roles* em relação à matriz e a Figura 14 em relação à uma das filiais.

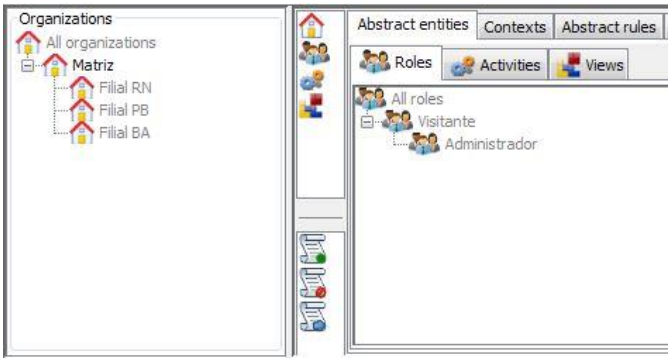


Figura 13. Modelagem dos roles da organization “Matriz”.

Figura 14. Modelagem dos roles da organization "Filial RN".

A rede corporativa da companhia do cenário hipotético foi representada através de *views*, onde foram criadas cinco *views*: "Dados", "Rede Corporativa", "VLAN1", "VLAN2" e "VLAN3". Todas as *views* relativas as VLANs foram definidas através de uma hierarquia sobre "Rede Corporativa".

As formas de acesso à rede corporativa foram modeladas através de quatro Activies: "Acesso", "VPN", "Terminal" e "Acesso Restrito". A entidade "Acesso Restrito" foi definida através de uma hierarquia sobre "Terminal". Neste caso, um acesso à "Terminal", implica no acesso completo ao terminal, enquanto "Acesso Restrito" implica em um acesso parcial aos recursos do terminal. Todas as *views* e *activities* foram criadas na organização "Matriz", e por consequência são herdadas por todas as filiais. A Figura 15 e a Figura 16 apresentam as modelagens para *activities* e *views* em relação à matriz.

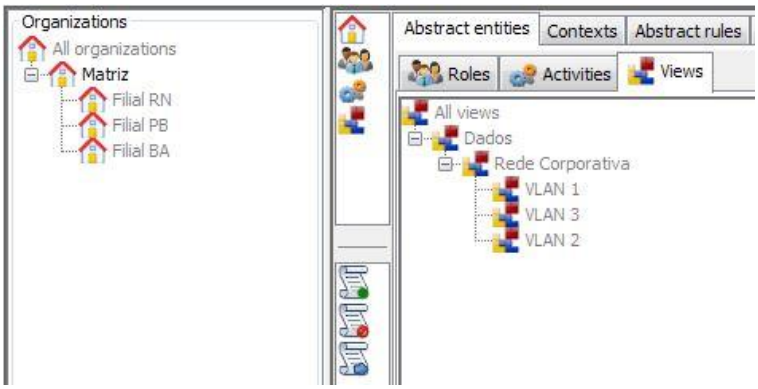


Figura 15. Modelagem das views.

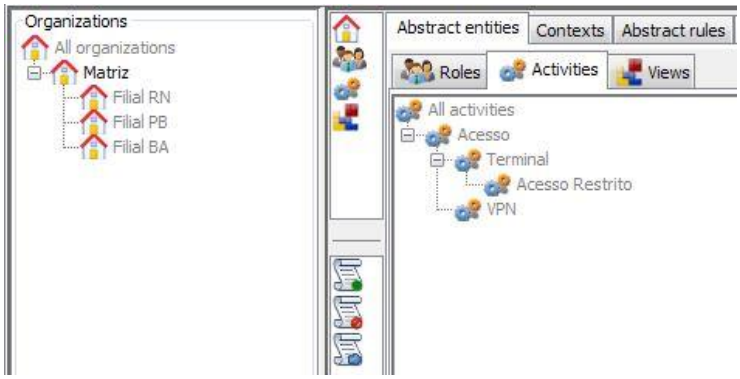


Figura 16. Modelagem das activies.

Durante a especificação dos requisitos do cenário trabalhado, é mencionado que o acesso ao ambiente corporativo, em alguns casos, é restrito ao horário de expediente. Para representar estes casos foi criado um novo *context*, além do já existente por padrão, chamado “expediente”. O *context* “default_context”, padrão do MotOrBAC, é definido como sempre verdade. O *context* “expediente”, definido nesta simulação, será verdade sempre no período das 8 às 12 horas e das 14 às 18 horas. A Figura 17 ilustra os *contexts* utilizados no cenário simulado.

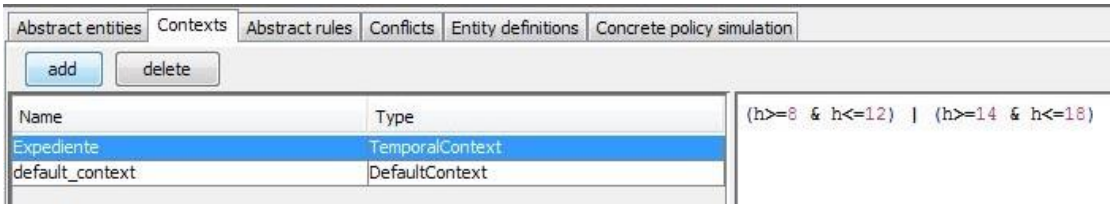


Figura 17. Contextos utilizados na simulação.

Após a criação das entidades, devem ser definidos os relacionamentos entre as mesmas. Estes relacionamentos serão definidos através de regras de permissão e proibição. A primeira regra definida foi uma proibição chamada “Regra Geral”, na *organization* “Matriz”, do *role* “Visitante” realizar a *activity* “Acesso” na *views* “Dados” sobre o *context* padrão. Esta regra, de início, proíbe qualquer tentativa de acesso ao sistema, seja na matriz ou nas filias. A Figura 18 mostra a “Regra Geral” derivada em “Filial RN”. A partir de então são definidas regras para conceder as permissões descritas na seção anterior, considerando as regras especificas de cada organização. A Figura 19 mostra as permissões na organização “Filial RN” como exemplo.

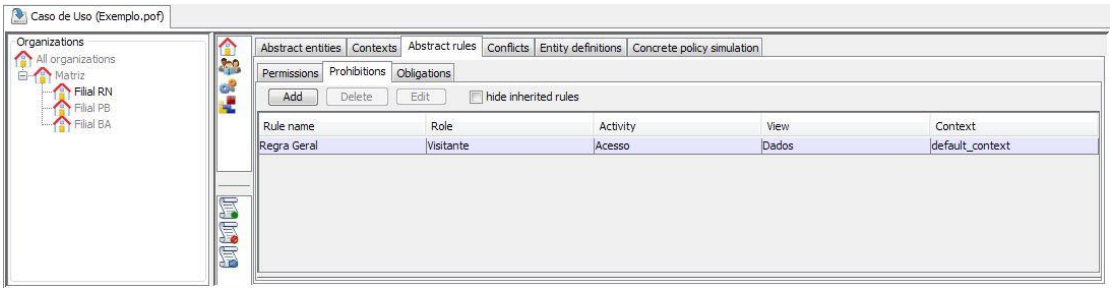


Figura 18. "Regra Geral" em "Filial RN".

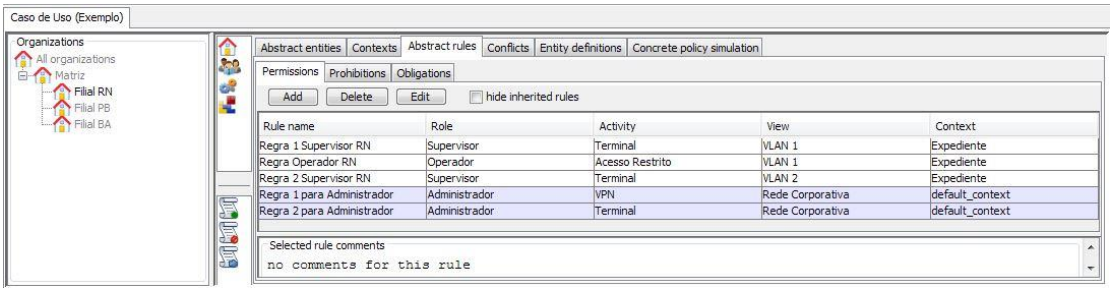
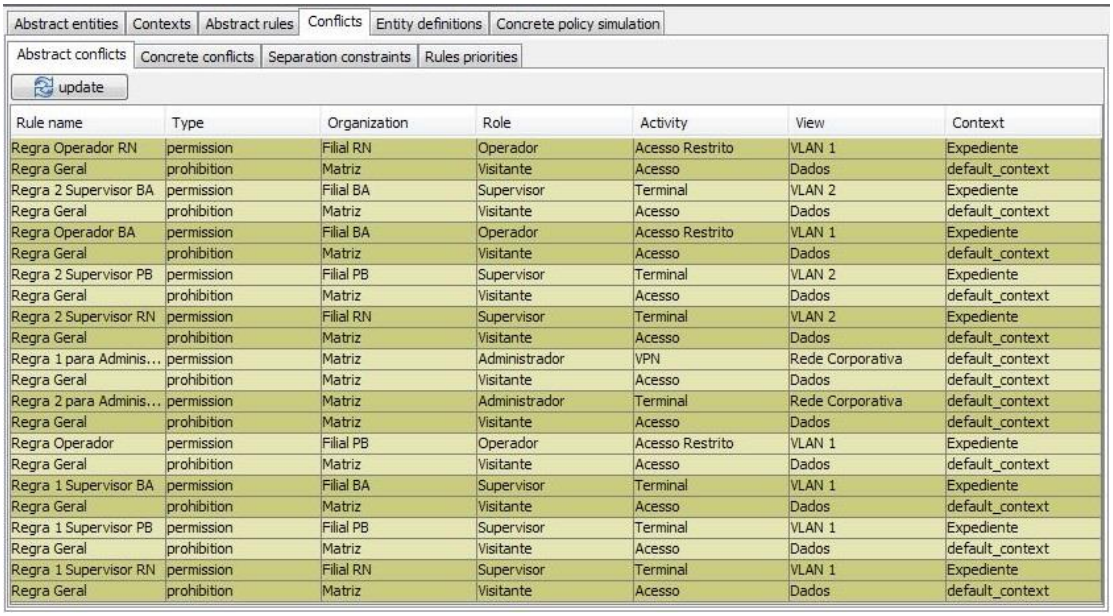


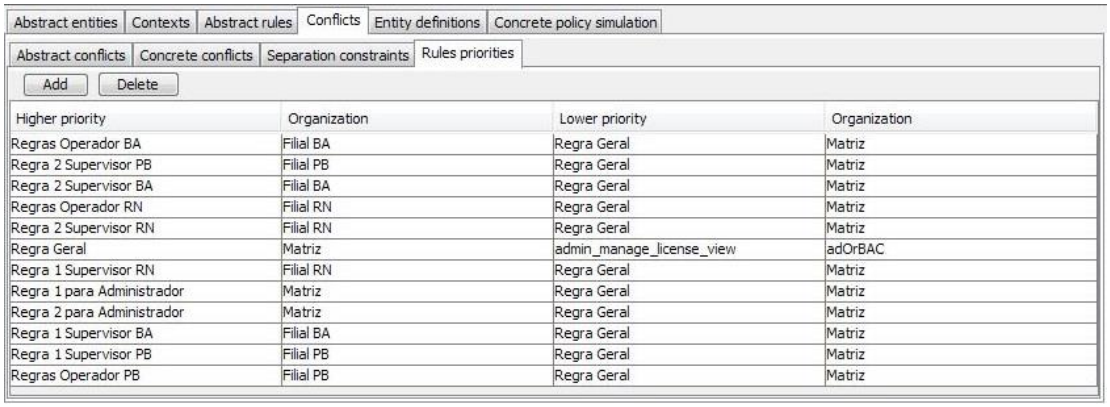
Figura 19. Permissões concedidas em "Filial RN".

Como foi explicado no capítulo anterior, regras de controle de acesso podem conflitar umas com as outras. No caso do cenário trabalhado nesta simulação foram apresentados alguns conflitos, todos eles relacionados com a regra "Regra Geral", conforme pode ser visto na Figura 20. Estes conflitos se devem ao fato de "Regra Geral" proibir qualquer acesso. Assim, todas as permissões necessárias aos *roles* do sistema conflitam com a "Regra Geral". Para resolver este empasse, foi definido que "Regra Geral" possui menos prioridade que qualquer outra regra criada no sistema. Com isso, sempre que seja satisfeita as condições necessárias para a concessão de uma permissão no sistema, esta permissão terá maior prioridade que "Regra Geral" e será a regra utilizada. Na Figura 21 são apresentadas as prioridades criadas para as regras conflitantes.



Rule name	Type	Organization	Role	Activity	View	Context
Regra Operador RN	permission	Filial RN	Operador	Acesso Restrito	VLAN 1	Expediente
Regra Geral	prohibition	Matriz	Visitante	Acesso	Dados	default_context
Regra 2 Supervisor BA	permission	Filial BA	Supervisor	Terminal	VLAN 2	Expediente
Regra Geral	prohibition	Matriz	Visitante	Acesso	Dados	default_context
Regra Operador BA	permission	Filial BA	Operador	Acesso Restrito	VLAN 1	Expediente
Regra Geral	prohibition	Matriz	Visitante	Acesso	Dados	default_context
Regra 2 Supervisor PB	permission	Filial PB	Supervisor	Terminal	VLAN 2	Expediente
Regra Geral	prohibition	Matriz	Visitante	Acesso	Dados	default_context
Regra 2 Supervisor RN	permission	Filial RN	Supervisor	Terminal	VLAN 2	Expediente
Regra Geral	prohibition	Matriz	Visitante	Acesso	Dados	default_context
Regra 1 para Adminis...	permission	Matriz	Administrador	VPN	Rede Corporativa	default_context
Regra Geral	prohibition	Matriz	Visitante	Acesso	Dados	default_context
Regra 2 para Adminis...	permission	Matriz	Administrador	Terminal	Rede Corporativa	default_context
Regra Geral	prohibition	Matriz	Visitante	Acesso	Dados	default_context
Regra Operador	permission	Filial PB	Operador	Acesso Restrito	VLAN 1	Expediente
Regra Geral	prohibition	Matriz	Visitante	Acesso	Dados	default_context
Regra 1 Supervisor BA	permission	Filial BA	Supervisor	Terminal	VLAN 1	Expediente
Regra Geral	prohibition	Matriz	Visitante	Acesso	Dados	default_context
Regra 1 Supervisor PB	permission	Filial PB	Supervisor	Terminal	VLAN 1	Expediente
Regra Geral	prohibition	Matriz	Visitante	Acesso	Dados	default_context
Regra 1 Supervisor RN	permission	Filial RN	Supervisor	Terminal	VLAN 1	Expediente
Regra Geral	prohibition	Matriz	Visitante	Acesso	Dados	default_context

Figura 20. Conflitos acusados após a criação das regras.



Higher priority	Organization	Lower priority	Organization
Regras Operador BA	Filial BA	Regra Geral	Matriz
Regra 2 Supervisor PB	Filial PB	Regra Geral	Matriz
Regra 2 Supervisor BA	Filial BA	Regra Geral	Matriz
Regras Operador RN	Filial RN	Regra Geral	Matriz
Regra 2 Supervisor RN	Filial RN	Regra Geral	Matriz
Regra Geral	Matriz	admin_manage_license_view	adOrBAC
Regra 1 Supervisor RN	Filial RN	Regra Geral	Matriz
Regra 1 para Administrador	Matriz	Regra Geral	Matriz
Regra 2 para Administrador	Matriz	Regra Geral	Matriz
Regra 1 Supervisor BA	Filial BA	Regra Geral	Matriz
Regra 1 Supervisor PB	Filial PB	Regra Geral	Matriz
Regras Operador PB	Filial PB	Regra Geral	Matriz

Figura 21. Prioridades das regras conflitantes.

Depois de resolvido os potenciais conflitos das políticas de segurança, o sistema já está apto a gerenciar o acesso dos usuários aos recursos. Para isso, primeiramente foram criados *subjects*, *actions* e *objects*, e foram atribuídos *roles*, *activities* e *views*, respectivamente. Como exemplo pontual, ao *subject* criado "Administrador" foi atribuído o *role* "Administrador"; à *action* criada "Acesso Terminal Restrito" foi atribuído a *activity* "Acesso Retrito"; e ao *object* criado "192.168.1.0/27" foi atribuído a *view* "VLAN1".

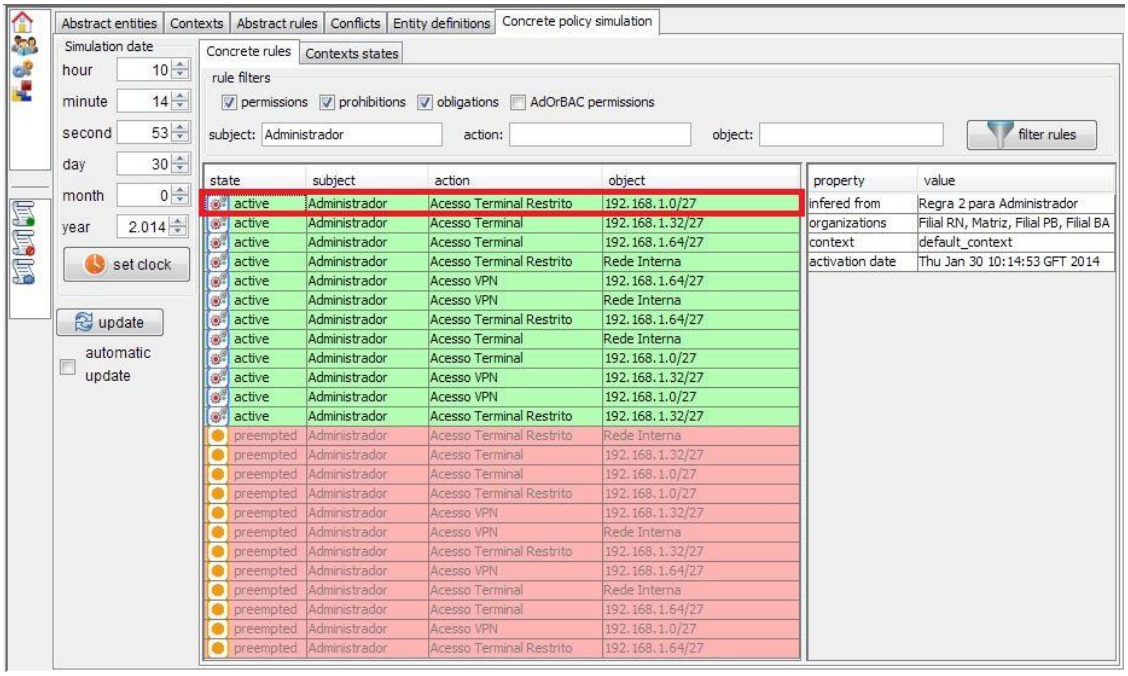


Figura 22. Simulação das regras do sistema.

A partir de então, regras concretas, inferidas a partir das regras definidas na forma abstrata, começam a ser criadas pelo sistema. A Figura 22 ilustra as regras inferidas para o *subject* “Administrador”. Ainda na Figura 22 é possível perceber que a regra concreta destacada em vermelho permite ao *subject* “Administrador” a *action* “Acesso Terminal Restrito” ao *object* “192.168.1.0/27” foi inferida a partir da regra abstrata “Regra 2 para Administrador”. É possível perceber também que as regras de proibição estão marcadas como “*preempted*” e inativas, pois as mesmas foram inferidas através da “Regra Geral” e possuem menor prioridade no sistema.

6. CONCLUSÃO E TRABALHOS FUTUROS

O objetivo deste Trabalho de Graduação consistiu em avaliar a viabilidade de utilizar um modelo de controle de acesso genérico para gerenciar políticas de segurança em um ambiente específico. Neste trabalho, utilizou-se como ambiente específico um cenário em redes de computadores e como modelo de controle de acesso genérico, o OrBAC.

Para atingir este objetivo, um estudo de caso foi desenvolvido simulando um cenário hipotético, onde foi necessário estruturar na ferramenta as entidades associadas ao cenário e definir as regras necessárias para reproduzir as políticas de controle de acesso à rede interna de uma companhia elétrica. Finalmente, foi necessária a verificação das regras que possuíam conflitos entre si, e a resolução dos mesmos através de eleição de prioridade entre regras.

O cenário utilizado para a simulação do caso de uso foi baseado na Companhia Hidro Elétrica do São Francisco – CHESF, porém não refletindo o ambiente real da companhia. Por se tratar de um cenário especifico de redes de computadores e uma organização empresarial, foi possível perceber a eficácia da representação das entidades de forma hierárquica e organizacional (representação da matriz e das filiais). Utilizando regras sobre contexto específico foi criado um contexto para regular o acesso apenas no horário do expediente, restrição do cenário utilizado neste trabalho, e que é comum de se encontrar em ambientes corporativos.

Através do cenário criado e da sua simulação no capítulo 5, é possível concluir que o modelo OrBAC obteve êxito na especificação das políticas de segurança. Todos os requisitos do ambiente proposto no caso de uso deste trabalho foram modelados sem maiores dificuldades, ratificando a capacidade de especificação de políticas do OrBAC.

Durante a realização deste trabalho foram encontradas algumas dificuldades em relação a usabilidade do editor de políticas MotOrBAC. Algumas dificuldades como falhas de leitura do arquivo de armazenamento do cenário criado no MotOrBAC, extensão ".pof", e falhas no *refresh* do editor de políticas aconteceram, porém ainda sim foi possível realizar com eficiência a modelagem das políticas propostas utilizando a ferramenta.

Este trabalho pode ser estendido através do desenvolvimento de um framework de propósito genérico para integrar os módulos do PDP e dos PEP, buscando aumentar a flexibilidade e o reuso da ferramenta de manipulação e tomada de decisão, no entanto mantendo uma interface padronizada para a implementação dos PEPs.

Outro trabalho futuro é o desenvolvimento de uma nova ferramenta de interface com o usuário, onde o administrador de segurança consiga desenvolver um cenário de regras simples sem a necessidade de uma maior entendimento do modelo de controle de acesso utilizado.

7. REFERÊNCIAS

[1] A. Abou, E. Kalam, A. Mi, R. El Baida, and C. Saurel, “Organization based access control,” 2003.

[2] “OrBAC Organization Based Access Control The official OrBAC model website.” [Online]. Available: http://orbac.org/.

[3] “MotOrBAC an OrBAC policy editor.” [Online]. Available: http://motorbac.sourceforge.net/.

[4] E. D. Durham, J. Boyle, R. Cohen, S. Herzog, R. Rajan, and A. Sastry, “The COPS (Common Open Policy Service) Protocol.” D. Durham, Ed., 2000.

[5] D. C. Verma, “Simplifying Network Administration Using Policy-Based Management,” no. April, pp. 20–26, 2002.

[6] J. Hodges and R. Morgan, “Lightweight Directory Access Protocol (v3): Technical Specification.” 2002.

[7] P. Samarati and S. De Capitani, “Access Control : Policies , Models , and Mechanisms.”

[8] B. W. Lampson, “Protection,” *5th Princet. Sym- posium Inf. Sci. Syst.*, pp. 437–443.

[9] G. S. Graham and P. J. Denning, “Protection — Principles and practice,” pp. 417–430.

[10] M. a. Harrison, W. L. Ruzzo, and J. D. Ullman, “Protection in operating systems,” *Commun. ACM*, vol. 19, no. 8, pp. 461–471, Aug. 1976.

[11] L. O. B. Lento, J. da S. Fraga, and L. C. Lung, “A Nova Geração de Modelos de Controle de Acesso em Sistemas Computacionais.”

[12] D. E. Bell and L. J. LaPadula, “Secure computer systems: Unified exposition and multics interpretation,” 1976.

[13] K. J. Biba, “Integrity Considerations for Secure Computer Systems,” 1975.

[14] D. F. Ferraiolo and D. R. Kuhn, “Role-Based Access Controls,” pp. 554–563, 1992.

[15] R. Ausanka-crues, “Methods for Access Control : Advances and Limitations.”

[16] F. Cuppens and a. Miege, “Modelling contexts in the Or-BAC model,” *19th Annu. Comput. Secur. Appl. Conf. 2003. Proceedings.*, 2003.

[17] F. Cuppens, N. Cuppens-Boulahia, and M. Ben Ghorbel, “High Level Conflict Management Strategies in Advanced Access Control Models,” *Electron. Notes Theor. Comput. Sci.*, vol. 186, pp. 3–26, Jul. 2007.

[18] F. Autrel, F. Cuppens, and N. Cuppens, “MotOrBAC 2 : a security policy tool,” no. 1.

[19] A. Mi, “Administration Model for Or-BAC,” pp. 1–15.

[20] N. C. C, “Fr´ ed´ eric Cuppens, Nora Cuppens-Boulahia et C´ eline Coma,” 2006.

[21] G. Z. Michele Bezzi, Penny Duquenoy, Simone Fischer-Hübner, Marit Hansen, “Privacy and Identity Management for Life,” *5th IFIP WG 9.2, 9.6/11.7, 11.4, 11.6/PrimeLife Int. Summer Sch.*

[22] “RDF Test Cases.” [Online]. Available: http://www.w3.org/TR/rdf-testcases/.