# JaCoWeb-ABC: Integração do Modelo de Controle de Acesso $UCON_{ABC}$ no CORBASec

**Marcelo Shinji Higashiyama[1], Lau Cheuk Lung[1],**

**Rafael R. Obelheiro[2] e Joni da S. Fraga[2]**

[1]Programa de Pós-Graduação em Informática Aplicada – PPGIA
Pontifícia Universidade Católica do Paraná (PUCPR) – Curitiba, PR
{shinji,lau}@ppgia.pucpr.br

[2]Departamento de Automação e Sistemas – DAS
Universidade Federal de Santa Catarina (UFSC) – Florianópolis, SC
{rro,fraga}@das.ufsc.br

*Resumo: A arquitetura JaCoWeb-ABC é uma extensão da especificação CORBASec que aplica o modelo de controle de acesso $UCON_{ABC}$ à sua camada de segurança. O JaCoWeb-ABC define um controle de acesso configurável, que trabalha com políticas de autorização, obrigação e condição definidas pelo administrador do sistema. Estas políticas de segurança podem ser definidas de duas formas, a primeira totalmente transparente para a aplicação e a segunda que possibilita trabalhar em conjunto com a aplicação. A combinação dessas duas funcionalidades irá definir um modelo de controle de acesso muito mais preciso e rigoroso sobre as ações dos usuários em um sistema.*

*Abstract: The JaCoWeb-ABC architecture is an extension of the CORBASec specification that applies the $UCON_{ABC}$ access control model to its security layer. JaCoWeb-ABC defines configurable access controls that deploy authorization, obligation and condition policies, based on the policies defined by the system administrator. These security policies can be defined in two different manners. The first one is totally transparent to the application and the second one makes it possible to work together with the application. The combination of these two functionalities will define a much more accurate and strict access control model for the actions of users within a system.*

## 1 Introdução

A popularização da Internet possibilitou que empresas de pequeno, médio e grande porte passassem a utilizar esta rede como um novo canal para seus negócios, oferecendo aos seus clientes serviços virtuais como e-commerce, Internet Banking, leilões, etc. Desta forma, as empresas passaram a direcionar o desenvolvimento de sistemas para aplicações do tipo cliente-servidor e buscar por soluções baseadas em sistemas distribuídos, como CORBA, RMI, DCOM e WebServices, que trabalham a partir de transações iniciadas pelo cliente e executadas pelo servidor. Esses serviços cada vez mais têm sido alvo de práticas indevidas por usuários da Internet. Inicialmente alguns

desses usuários tinham como objetivo apenas indisponibilizar os serviços, mas, nestes últimos tempos, esta prática tem sido direcionada para a realização de crimes digitais, levando empresas a terem enormes prejuízos e passar a investir fortemente na segurança de seus sistemas.

Muitos conceitos de segurança passaram a ser estudados e aplicados em sistemas, como os modelos de controle de acesso obrigatório (MAC) [1], discricionário (DAC) [2], baseado em papéis (RBAC) [3], entre outros. Mecanismos como SSL e certificados digitais também passaram a ser muito utilizados, principalmente para garantir a confidencialidade, integridade e autenticidade das informações. Mesmo assim, nem sempre esses conceitos são suficientes para impedir que usuários do sistema efetuem procedimentos abusivos ou até impróprios.

Focando esta necessidade, Sandhu e Park introduziram um novo modelo de controle de acesso chamado $UCON_{ABC}$ (*Usage Control*) [4, 5, 6], que define políticas de Autorizações, oBrigações e Condições, e trabalha com sujeitos e objetos que possuem atributos de controle mutáveis, ou seja, que podem ser alterados em função dos acessos. Uma grande vantagem deste modelo é a capacidade de adaptar diversos modelos de controle de acesso existentes em suas políticas de segurança, além de possibilitar um controle mais preciso sobre as ações de um usuário.

O modelo CORBA de Segurança (CORBASec) possui apenas uma especificação, padronizada pela OMG, que aplica o modelo de controle de acesso discricionário em sua camada de segurança. Consideramos que este modelo é muito restrito e não garante um total controle sobre as ações de um usuário aos recursos de um sistema. Desta forma, este artigo propõe a aplicação do modelo $UCON_{ABC}$ na especificação CORBASec, definindo um controle de acesso mais preciso e rigoroso sobre as ações de um usuário, possibilitando até a sua revogação em casos identificados como impróprios no sistema. Um grande desafio desta proposta é definir uma especificação capaz de implementar a política de segurança definida no modelo $UCON_{ABC}$ a trabalhar com os conceitos de autorização, obrigação, condição e também a mutabilidade de atributos, pois conforme descrito pelos próprios autores deste modelo [4], se trata de um modelo teórico que não pode ser implementado diretamente.

Este artigo está organizado da seguinte maneira: a seção 2 descreve o modelo $UCON_{ABC}$ proposto por Park e Sandhu, e a seção 3 apresenta o modelo CORBA de segurança. A seção 4 discute a proposta de implementação e arquitetura do modelo ABC sobre o CORBASec. Na seção 5 são mostrados os resultados obtidos através da implementação de um protótipo. A seção 6 mostra alguns trabalhos relacionados e a seção 7 apresenta as conclusões deste trabalho.

## 2 O MODELO $UCON_{ABC}$ – USAGE CONTROL MODEL

Os modelos tradicionais de controle de acesso (DAC, MAC, RBAC) baseiam-se na noção de sujeitos que acessam objetos em um sistema. Conceitualmente, isso pode ser representada por uma matriz que armazena os direitos de acesso que os sujeitos possuem sobre os objetos. Quando um sujeito deseja acessar um objeto, essa matriz é consultada para determinar se esse acesso deve ser autorizado ou não (a chamada decisão de autorização). Tipicamente, a decisão de autorização envolve atributos estáticos do sujeito e do objeto em questão.

O modelo $UCON_{ABC}$ [4, 5, 6] estende o controle de acesso através do controle de uso (*usage control*), considerando não apenas os conceitos tradicionais de sujeitos, objetos, direitos e autorizações como também os conceitos de atributos mutáveis, obrigações e condições. Isso possibilita não apenas representar políticas de controle de acesso baseadas em modelos clássicos como uma nova gama de políticas mais precisas e rigorosas.

## 2.1 Componentes do modelo $UCON_{ABC}$

O modelo $UCON_{ABC}$ possui oito componentes: *sujeitos* (S), *atributos de sujeitos* (ATT(S)), *objetos* (O), *atributos de objetos* (ATT(O)), *direitos* (R), *autorizações* (A), *obrigações* (B) e *condições* (C), que estão representados na figura 1.

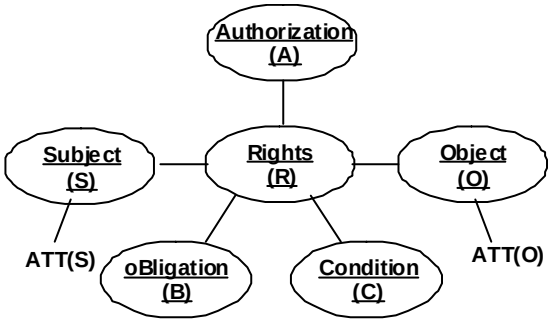


**Figura 1 – Modelo de Controle de Acesso ABC**

Os conceitos de sujeitos e objetos são os mesmos do modelo de controle de acesso tradicional. Direitos habilitam o acesso de um sujeito a um objeto em um modo particular, como leitura ou escrita. Uma característica importante do modelo $UCON_{ABC}$ é que neste os atributos são mutáveis, dependendo da atividade dos sujeitos no sistema. Isso significa que a decisão de autorização é efetuada em tempo de execução, no momento do acesso do sujeito ao objeto. Os processos de avaliação ABC estão definidos abaixo:

**Autorizações**: são exigências que devem ser satisfeitas antes ou durante a permissão do acesso de um sujeito a um objeto. As autorizações são baseadas nos atributos do sujeito e atributos do objeto, que especificam o direito em questão. Ex.: permitir o acesso a um arquivo apenas para um grupo de usuários de um sistema.

**oBrigações**: são exigências que um sujeito deve exercer antes (preB) ou durante (onB) o acesso a um determinado objeto. Na medida em que o sujeito cumpre suas obrigações no sistema, o acesso a novos objetos poderá ser concedido. Ex.: fornecer um *email* antes de baixar uma versão de demonstração de um *software*.

**Condições**: são fatores relacionados ao ambiente do sistema, que permitem verificar condições antes (preC) ou durante (onC) o acesso de um sujeito a um determinado objeto. Ex.: permitir o acesso de um sujeito a um objeto apenas em horário comercial.

## 2.2 A Família de Modelos ABC

Durante o processo de decisão do acesso, feito antes ou durante o acesso ao objeto, o modelo $UCON_{ABC}$ permite definir políticas de atualização de atributos do sujeito e objeto, que pode ser antes, durante ou depois da sua utilização, conforme apresentado na figura 2. As diferentes combinações de momento de decisão (antes do

acesso, durante o acesso) e de atualização de atributos (antes, durante ou após o acesso) geram uma família de 16 modelos possíveis. Essa variedade de modelos possibilita a definição de políticas bastante flexíveis, que vão de modelos clássicos de controle de acesso (DAC, MAC, RBAC) a DRM (*Digital Rights Management*) e *pay-per-use* [4].

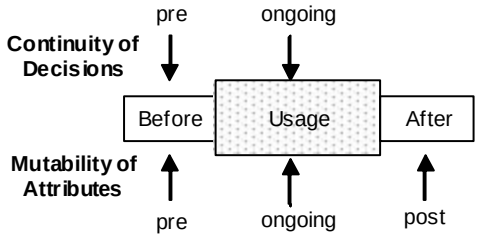


**Figure 2 - Continuity and Mutability Properties**

## 3 MODELO DE SEGURANÇA CORBA

O padrão CORBA, definido pela OMG (*Object Management Group*), especifica uma arquitetura de *software* que suporta aplicações distribuídas e garante a interoperabilidade entre diferentes plataformas de *hardware* e sistemas operacionais [7]. A especificação CORBA 1.1 definiu a linguagem de definição de interfaces IDL (*Interface Definition Language*) e as Interfaces de Programação de Aplicações - API (*Application Programming Interfaces*), que possibilitam a interação cliente-servidor via objetos com uma implementação específica de um ORB (*Object Request Broker*). A IDL possibilita definir a interface do método de um objeto implementado em um servidor, que será invocado a partir de uma aplicação cliente de uma forma transparente para a aplicação. A especificação CORBA 2.0 [7] passou por um grande processo de revisão, onde foram adicionadas diversas novas características, entre elas, a especificação do serviço de segurança CORBA (CORBASec) [8].

### 3.1 O Serviço de Segurança CORBA

O serviço de segurança CORBA (CORBASec) é uma especificação aberta de segurança em sistemas distribuídos [8, 9, 10]. O CORBAsec relaciona objetos e componentes em quatro níveis, numa estratificação de sistema: *o nível de aplicação*; o *nível de middleware*, formado por objetos de serviço (COSS), serviços ORB e o núcleo do ORB; o *nível de tecnologia de segurança*, formado pelos serviços de segurança subjacentes; e, por fim, o *nível de proteção básica*, composto por uma combinação de funcionalidades de sistemas operacionais e do hardware.

O CORBASec trabalha com serviços de segurança a nível de interceptadores, possibilitando proteger os objetos de serviço de forma transparente para a aplicação. Ele é composto por objetos de serviço (COSS), como o PrincipalAuthenticator, Credentials, AccessPolicy, RequiredRights, AccessDecision, SecurityManager, PolicyCurrent, Vault e SecurityContext. O objeto PrincipalAuthenticator é responsável por garantir o processo de autenticação de principais no CORBA, tendo como objetivo a aquisição de credenciais que serão utilizadas para a sua identificação no sistema. No CORBASec, as políticas são descritas na forma de *atributos de segurança* dos recursos do sistema (atributos de controle) e dos principais (atributos de privilégio). O CORBASec fornece apenas a família CORBA que contém quatro tipos de direitos: `g` *(get)*, `s` *(set)*, `m` *(manage)* e `u` *(use)*, embora permita a livre definição de outras famílias e direitos.

O controle de acesso é implementado através dos objetos AccessPolicy e RequiredRights. O AccessPolicy contém um conjunto de principais identificados pelos seus atributos de privilégio, e, para cada um deles, uma lista de direitos de acesso para invocação dos métodos sobre os objetos (tabela 1). O RequiredRights, por sua vez, define os direitos requeridos para a invocação de uma operação sobre um objeto (atributos de controle), como mostra a tabela 2.

| Atributos de Privilégio | Direitos |
|---|---|
| Ana | corba: gs-- |
| Bob | corba: g–u |

**Tabela 1 - AccessPolicy**

| Direitos | Combinação | Operação | Interface |
|---|---|---|---|
| Corba: g–m- | All | Get_balance | Bank |
| Corba: gs-- | Any | Deposit | Bank |
| corba: gsu- | All | Open | Bank |

**Tabela 2 – RequiredRights**

O objeto AccessDecision é responsável pelo processo de autorização de acesso ao método de um objeto, a partir da sua invocação pelo cliente de uma aplicação. A sua intervenção é feita a partir do uso de interceptadores, que são objetos de serviço colocados no caminho da invocação entre o cliente e o servidor. Neste processo, um cliente efetua a invocação do método remoto que se encontra no servidor, passando as credenciais (objeto Credentials) obtidas na autenticação. A decisão de acesso avalia então os atributos de privilégio e de controle encontrados nos objetos AccessPolicy e RequiredRights (figura 3).

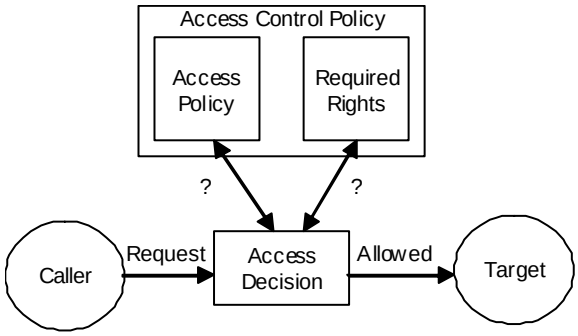


**Figura 3 – Modelo de Controle de Acesso do CORBASec**

Neste exemplo, o cliente Bob possui os direitos "gu". Se ele tentar invocar a operação "Get_balance", cujos direitos requeridos são "gm" com combinação All, ele terá o seu direito de acesso negado pelo objeto AccessDecision. Caso Bob tente executar a operação "Deposit", cujos direitos requeridos são "gs" mas a combinação é apenas Any, o objeto AccessDecision permitirá o acesso, já que Bob possui um dos direitos necessários ("g").

## 4 A Proposta JaCoWeb-ABC

O $UCON_{ABC}$ define um modelo muito completo que possibilita adaptar diversos conceitos de controle de acesso à sua funcionalidade. A especificação CORBA é muito conhecida e difundida no meio acadêmico. O fato do CORBASec definir uma camada de segurança totalmente transparente para a aplicação, que é controlada pelo ORB, pode ser vantajoso em diversos casos. Nem sempre, porém, esse controle transparente é suficiente, pois muitas das tomadas de decisão a serem definidas nas aplicações estão baseadas em ações que o usuário venha a efetuar posteriormente, após receber o acesso.

Para implementar o controle de acesso na aplicação seria necessário recorrer a ferramentas de mercado ou soluções proprietárias que possibilitassem integrar este segundo nível de controle de acesso diretamente na aplicação. Isso traria uma complexidade ainda maior na definição e administração das políticas de segurança, que teriam que ser definidas em dois pontos distintos, com conceitos de políticas totalmente diferentes, o que provavelmente inviabilizaria o uso do nível transparente de controle de acesso, pois seria mais fácil implementar toda a política no nível de aplicação.

Visando atacar este problema, a nossa proposta é integrar o modelo $UCON_{ABC}$ à especificação CORBASec, centralizando estes dois níveis de controle de acesso em sua camada de segurança. Isso possibilita definir políticas de segurança utilizadas pelo controle transparente para a aplicação, nos casos em que esta camada de segurança tenha todas as informações necessárias para uma decisão de acesso, bem como definir políticas a serem utilizadas pela aplicação, nos casos em que a decisão de acesso for baseada em informações disponíveis apenas na aplicação.

### 4.1 Mapeamento do modelo $UCON_{ABC}$ no CORBASec

Em termos de controle de acesso, um sujeito no modelo $UCON_{ABC}$ é equivalente a um principal no CORBASec. Cada principal possui atributos de privilégio, associados a ele através do objeto Credentials, e a esses atributos de privilégio são concedidos direitos, através do objeto AccessPolicy. Um objeto do $UCON_{ABC}$ corresponde a um método de uma interface CORBA; a cada método é associado um conjunto de direitos requeridos (juntamente com uma combinação), através do objeto RequiredRights. Uma autorização é implementada no CORBASec pelo objeto AccessDecision, que compara os direitos concedidos a um principal (AccessPolicy) e os direitos requeridos por um método (RequiredRights) para decidir se o principal pode invocar esse método.

A descrição acima resume a equivalência entre o modelo $UCON_{ABC}$ e o modelo de controle de acesso do CORBASec. Com base nessa descrição, fica evidente que vários componentes do modelo $UCON_{ABC}$ não encontram correspondente no CORBASec. Para a aplicação dos conceitos existentes no modelo $UCON_{ABC}$ no CORBASec, foram identificadas e mapeadas as seguintes diferenças entre os modelos, que terão que ser adaptadas para possibilitar a sua implementação:

1) Atributos: o modelo $UCON_{ABC}$ considera que atributos de sujeitos e objetos são arbitrários, ao passo que a especificação CORBASec, embora permita a definição de novos atributos de controle, exige que esses atributos sejam especificados através de uma IDL, o que torna essa definição um tanto quanto engessada. Portanto, torna-se necessário definir uma maneira para representar atributos de sujeitos e objetos de forma a aproveitar a flexibilidade oferecida pelo modelo $UCON_{ABC}$.

2) Mutabilidade de atributos: no CORBASec, os atributos de sujeitos e objetos somente podem ser alterados a partir de ações administrativas de inclusão ou remoção de direitos. Já no modelo $UCON_{ABC}$, a alteração dos atributos pode ocorrer a partir do acesso de um sujeito sobre um objeto, baseado na definição de suas políticas de segurança.

3) Direitos: o CORBASec especifica apenas um modelo de controle de acesso discricionário, onde são avaliados os direitos de sujeitos e objetos. No modelo $UCON_{ABC}$, também são avaliados os atributos de sujeitos e objetos, mas com a possibilidade de adaptar diversos modelos de controle de acesso.

4) Autorização, Obrigação e Condição: o modelo CORBASec trabalha apenas com um processo de autorização. Desta forma, além de uma remodelagem do processo de autorização, os controles de obrigação e condição do modelo $UCON_{ABC}$ precisam ser totalmente definidos e implementados para o CORBASec.

5) Definição de políticas de controle de acesso: a definição de políticas no CORBASec é muito simples, baseada nos objetos AccessPolicy e RequiredRights, sendo que o processo de decisão é feito apenas com um cruzamento dos direitos concedidos e requeridos associados a uma combinação. Já no modelo $UCON_{ABC}$, as políticas são representadas a partir de expressões lógicas e de atribuição. O resultado dessas expressões irá definir o direito de acesso ao objeto. Para o cálculo das expressões são usados os atributos de um sujeito e objeto, além de variáveis externas como data/hora utilizadas pelo processo de Condição.

6) Níveis de avaliação de acesso: todo o processo de decisão de acesso no modelo CORBASec é feito apenas no nível do ORB. Desta forma, durante o acesso de um sujeito sobre o método de um objeto, este processo de decisão é feito de forma totalmente transparente para a aplicação. Caso um acesso seja negado pelo objeto AccessDecision, o método nem chega a ser executado. Já o modelo $UCON_{ABC}$ permite definir um controle de acesso transparente para a aplicação, em que os acessos podem ser decididos apenas com as informações dos atributos do sujeito e objeto, e também trabalhar em conjunto com a aplicação, caso em que o controle de acesso poderá ser baseado em fatores externos à camada, como por exemplo, obrigar um usuário a informar um *email* antes de continuar a acessar outras partes de um sistema. Nesta situação, o *email* deverá ser analisado, para certificar-se de que se trata de um *email* válido, cabendo à aplicação continuar o processo de decisão que deverá informar ao modelo $UCON_{ABC}$ que a obrigação foi cumprida pelo sujeito.

### 4.2 Alterações necessárias no CORBASec para adaptação do modelo $UCON_{ABC}$

A partir do mapeamento dos modelos, identificamos a necessidade das seguintes alterações sobre os componentes do CORBASec para adaptação do modelo $UCON_{ABC}$:

1) Os objetos AccessPolicy e RequiredRights deverão ser remodelados para lidarem com atributos genéricos. Desta forma, surgiu a necessidade de criarmos um objeto AttributeManager para gerenciar atributos que deve permitir a definição de atributos para um sujeito e objeto. Nesta implementação, permanecerão inalterados apenas os atributos de identificação do sujeito (atributos de privilégio) e objeto (interface e operação) definidos no CORBASec. O restante dos atributos (direitos e combinação) poderão passar a ser controlados pelo objeto AttributeManager.

2) O objeto AccessDecision deverá ser remodelado, de forma que possibilite efetuar o processo de decisão baseado em políticas de Autorização, Obrigação e Condição (decisão ABC). A definição destas políticas deve ser feita para cada objeto ABC (método de um objeto CORBA). Neste processo de decisão, podem ser definidas políticas de atualização de atributos para sujeitos e objetos.

3) Para avaliação das políticas ABC, deve ser implementado um interpretador de expressões lógicas e de atribuição capaz de calcular as expressões utilizadas para decisão e atualização baseadas nos atributos do sujeito e objeto e também variáveis externas ao CORBASec, como a data/hora.

4) Devido ao CORBA usar um modelo transacional, iniciado a partir da invocação de um método remoto, o processo de decisão ABC poderá ser efetuado somente antes (preA, preB, preC) do acesso de um sujeito sobre o objeto, pois em uma invocação de método não existe possibilidade de se efetuar o processo de decisão durante (*ongoing*) um acesso (onA, onB, onC).[1] Pelo mesmo motivo, o processo de atualização de atributos também poderá ser efetuado somente antes (preUpdate) ou após (posUpdate) o acesso de um sujeito sobre um objeto, não existindo o conceito de atualização durante o acesso (onUpdate).

5) Necessidade de se criar um processo de avaliação em dois níveis, sendo o primeiro de forma transparente para a aplicação e o segundo atuando em conjunto com a aplicação.

### 4.3 Níveis de Decisão e Verificação das Politicas ABC

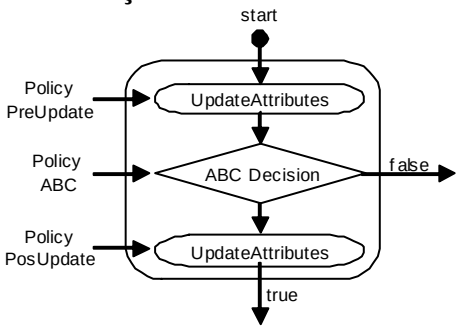


**Figura 4 – Objeto AccessDecisionABC**

Para possibilitar o processo de decisão de acesso e permitir também o processo para atualização de atributos definidos nas políticas do modelo $UCON_{ABC}$ (definido na figura 3) ao objeto AccessDecision do CORBASec, criamos o objeto AccessDecisionABC, representado na figura 4, que possibilita efetuar os processos de avaliação ABC juntamente com os processos de preUpdate e posUpdate. Esse objeto será utilizado na implementação do objeto AccessDecision do CORBASec, a partir da invocação de um sujeito a um objeto remoto do CORBA. Neste processo, o objeto AccessDecision identifica o Sujeito (principal) e Objeto (método remoto) da invocação, e repassa essas informações para o objeto AccessDecisionABC. A partir da identificação do Objeto, o AccessDecisionABC carrega as políticas de segurança ABC (policyABC) e as políticas de atualização de preUpdate e posUpdate. E conforme definido no modelo $UCON_{ABC}$, o processo de avaliação dessas políticas será baseado nos atributos do Sujeito e Objeto, que serão utilizados nos cálculos de expressões definidas nas políticas de decisão de acesso ABC e no processo de atualização de atributos.

Para possibilitar a quebra do processo de decisão em dois níveis de acesso, o objeto AccessDecisionABC foi adaptado para que fosse chamado nestas duas situações. Abaixo estamos detalhando a especificação de cada um destes objetos, que são ilustrados na figura 5.

[1] O conceito *ongoing* pode ser adaptado no CORBASec a partir da invocação de dois métodos distintos, sendo o primeiro responsável por liberar um acesso e o segundo para um acesso periódico a um objeto (ex.: em um serviço disponibilizado em um site, um método pode servir para que um sujeito responda uma pergunta a cada 15 minutos, e o outro método para possibilitar o acesso a serviços deste site, que somente poderá ser acessado caso o usuário tenha respondido às perguntas em períodos inferiores a 5 minutos).

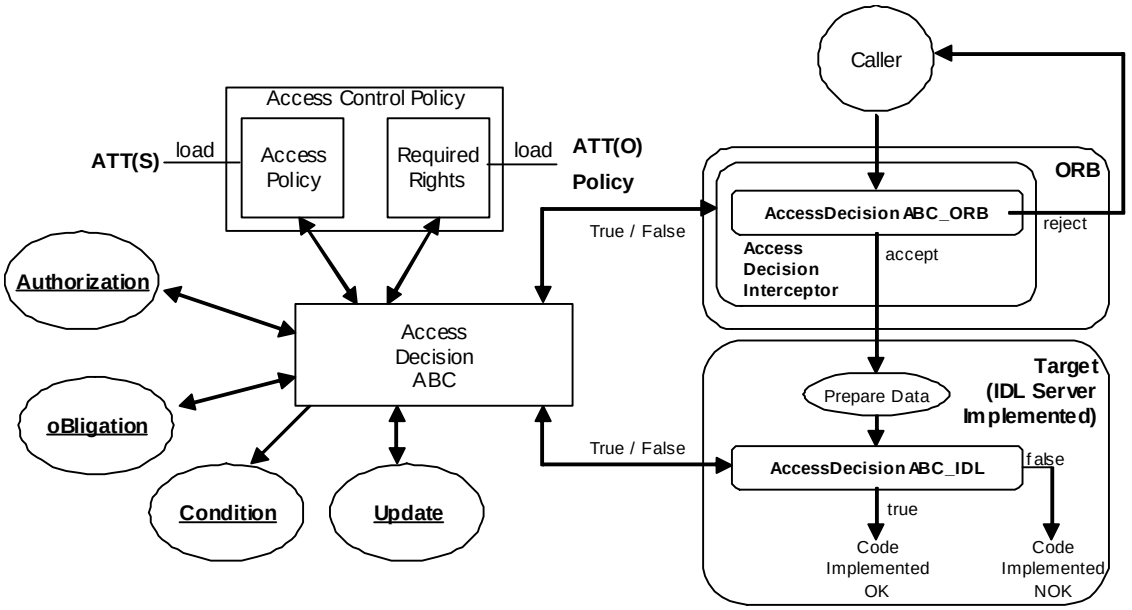


**Figure 5 - Processo de avaliação efetuado pelo objeto AccessDecisionABC**

**AccessDecisionABC_ORB**: controle de acesso ABC a nível de *middleware*, que funciona de forma transparente para a aplicação, sendo necessária a configuração das políticas de segurança que serão controladas pelo objeto. Nesta situação, o processo de decisão depende apenas das informações existentes nos atributos do sujeito e objeto. Durante a invocação é chamado o interceptador de controle de acesso que efetua a chamada ao método *access_allowed()* do objeto AccessDecision, que recebe como parâmetros as suas credenciais, que possuem os atributos de identificação do Sujeito, e também o nome da interface e a operação do objeto que está sendo invocado. O objeto AccessDecision chama o objeto AccessDecisionABC_ORB, passando estes os parâmetros de identificação do Sujeito e Objeto, e este em seguida invoca o objeto AccessDecisionABC. A partir das credenciais, o objeto AccessPolicy associa Sujeito (S) aos seus atributos (ATT(S)), e pelo nome da Interface e Operação, o objeto RequiredRights associa o Objeto (O) aos seus atributos (ATT(O)). A partir desses atributos e das políticas de decisão e atualização do $UCON_{ABC}$ carregadas, o objeto AccessDecisionABC realiza o processo de decisão (ABC) e atualização. Após este processo, caso o método *access_allowed* retorne o valor *true*, o método implementado a partir da IDL será invocado.

**AccessDecisionABC_IDL**: controle de acesso ABC a nível de aplicação, que deve trabalhar em conjunto com a aplicação para o processo de decisão. Para isso ser possível, a arquitetura JaCoWeb-ABC disponibiliza uma API que deve ser utilizada durante a implementação da IDL do objeto no servidor, para que ela possa trabalhar em conjunto com sua camada de decisão. Esta API, nomeada *AccessDecisionABC_IDL.access_allowed()*, deve receber quatro parâmetros que estão disponíveis pela especificação CORBASec durante a implementação da IDL no servidor: *CurrentObject*, para aquisição das credenciais do sujeito, *ObjectImplemented*, para identificação do nome da interface do objeto, *methodName*, que é o nome do método invocado (operação), e *Vector*. Vector é um objeto que possibilita adicionar uma lista de variáveis que serão utilizadas em políticas definidas pelo JaCoWeb-ABC. Nesta situação, a IDL implementada no servidor deve capturar informações externas a este objeto como,

por exemplo, parâmetros recebidos pelo método invocado, preparar estes dados para adicioná-los em uma variável Vector, chamar a API *access_allowed()*, e finalmente tratar a condição do acesso, que terá um retorno *true* ou *false*, continuando o processo de decisão. A partir desta chamada, o processo é idêntico à chamada do objeto AccessDecision_ORB já explicado anteriormente.

### 4.4 Processo de Cálculo de Expressões

Os processos de decisão definidos pelas políticas de segurança do $UCON_{ABC}$ são baseados em expressões lógicas e de atribuição. Desta forma, tivemos que definir e implementar um interpretador capaz de calcular estas expressões, trabalhando com os atributos do sujeito e objeto. Para isso, criamos dois objetos detalhados abaixo:

Objeto LogicExpression: foi criado para calcular expressões lógicas utilizadas pelos processos de decisão ABC.

Objeto AttribExpression: foi criado para o processo de atualização de atributos, sendo capaz de calcular as expressões baseadas no tipo de dado dos atributos para o processo de atribuição. Ele é capaz de trabalhar expressões com tipos matemáticos, strings, datas, vetores e matrizes.

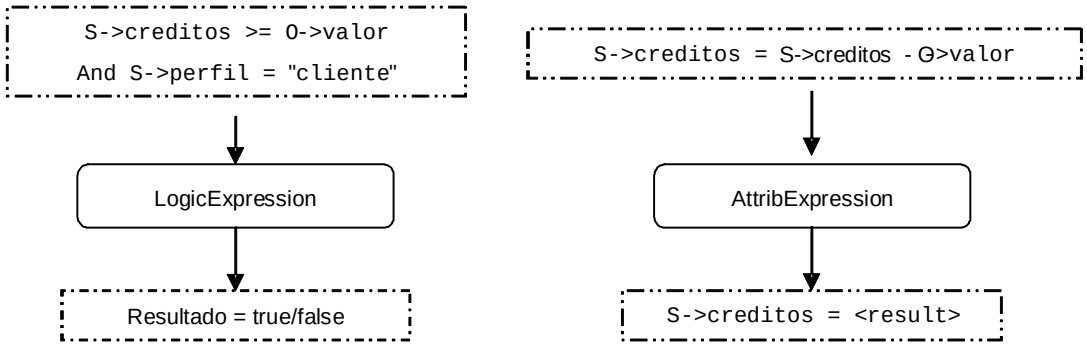


**Figura 6 - Objetos LogicExpression e AttribExpression**

A identificação dos atributos usada pelas expressões é definida da seguinte maneira: para referenciar atributos de sujeitos, a expressão deve usar "S->", enquanto que atributos de objetos devem ser referenciados por "O->". A figura 6 apresenta um exemplo de passagem de expressões para estes objetos. No objeto LogicExpression, uma expressão lógica é passada. Nesta expressão, é avaliado se os créditos de um sujeito são maiores que o valor de um objeto e se o perfil do sujeito é de um cliente, retornando um valor lógico *true* ou *false*. Para o objeto AttribExpression, é passada uma expressão de atribuição, onde o valor do objeto é subtraído dos créditos do cliente; no final, o atributo "creditos" é atualizado pelo resultado da expressão.

Além da identificação de atributos do sujeito e objeto, os objetos LogicExpression e AttribExpression também podem interpretar em suas expressões chamadas de funções específicas, como por exemplo, capturar informações como a data e hora do sistema. Para isso, a interpretação destas chamadas teve que ser especificada e implementada por estes objetos. [2]

[2] Mais detalhes sobre as funções interpretadas por estes objetos podem ser obtidas em http://www.ppgia.pucpr.br/~shinji/jacowebABC/relatorioTecnico.pdf.

### 4.5 Visão Geral do Modelo

A figura 7 apresenta uma visão geral da arquitetura JaCoWeb-ABC. Todas as políticas de controle de acesso ABC estarão definidas para cada objeto remoto do CORBASec, uma vez que todos os controles são efetuados sobre estes objetos. Para esta definição, foram criados os objetos PolicyObject., que possui todas as políticas de decisão e atualização do modelo $UCON_{ABC}$, o objeto PolicyABC que possui as políticas de autorização (policyA), obrigação (policyB) e condição (policyC), e também as políticas de atualização de atributos (objeto PolicyUpdate), que podem ser feitas antes (preUpdate) ou depois (posUpdate) do processo de decisão. A decisão de acesso ABC pode ser efetuada de forma transparente para a aplicação (PolicyABC_ORB) ou trabalhar em conjunto com a aplicação (PolicyABC_IDL), conforme discutido na seção 4.3. Para adaptar as políticas do modelo $UCON_{ABC}$ na arquitetura JaCoWeb-ABC, representadas nas figuras 5 e 7, utilizamos uma linguagem baseada em XML capaz de especificar os atributos do sujeito e objeto e também as políticas de autorização, obrigação, condição e atualização.[3]

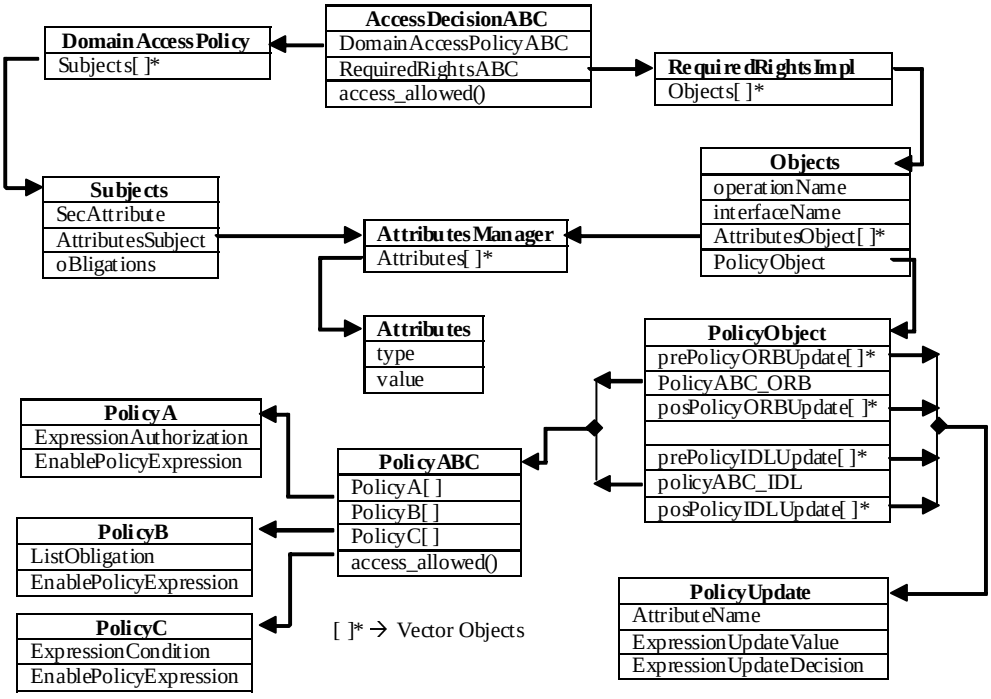


Figura 7 - Componentes da Arquitetura JaCoWeb-ABC

## 5 Aspectos da Implementação e Medidas de Desempenho

Para a aplicação do modelo JaCoWeb-ABC, utilizamos o JacORB 2.2 [12], que é um ORB livre desenvolvido em Java pela Freie Universität Berlin/XTRADYNE Technologies AG da Alemanha. Foi utilizada a versão do Java 1.5.0_01. Para o processo de autenticação de principais, foram utilizados os recursos disponíveis pelo JacORB, a partir da autenticação SSL, baseado em chaves públicas e privadas, garantindo também a confidencialidade das informações por um canal cifrado.

[3] O modelo de políticas utilizadas pelo modelo JaCoWeb-ABC baseada em XML também podem ser obtidas em http://www.ppgia.pucpr.br/~shinji/jacowebABC/relatorioTecnico.pdf, que poderão ser melhor compreendidos a partir dos exemplos apresentados.

Os componentes do JaCoWeb-ABC mostrados na figura 7 tiveram que ser implementados. Para o processo de decisão e atualização, tivemos que criar os objetos LogicExpression e AttribExpression, capazes de interpretar e calcular expressões a partir dos atributos de sujeitos e objetos, conforme mostrados na figura 6. Estes objetos são os componentes mais complexos deste projeto, pois tivemos que desenvolvê-los, de forma que eles pudessem calcular as expressões lógicas, aritméticas, atribuição e funções definidas e suportadas pelo JaCoWeb-ABC. O carregamento das políticas ABC foi implementada no construtor do objeto AccessDecisionABC, em que é chamado quando este objeto é instanciado pela primeira vez, que efetua a leitura das políticas que estão definidas dentro de um arquivo. O objeto AccessDecisionInterceptor foi configurado de forma que as invocações de métodos de objetos CORBA, o objeto AccessDecisionABC_ORB fosse utilizado. O objeto AccessDecisionABC_IDL também foi criado para disponibilizar o controle de acesso a ser implementado a nível de aplicação, durante a implementação de uma IDL.

Para execução de testes de desempenho do modelo, utilizamos o exemplo jaco.demo.benchmark, que vem junto com o pacote do JacORB, tendo como base três avaliações sobre a invocação de um objeto CORBA, apresentadas na figura 8:

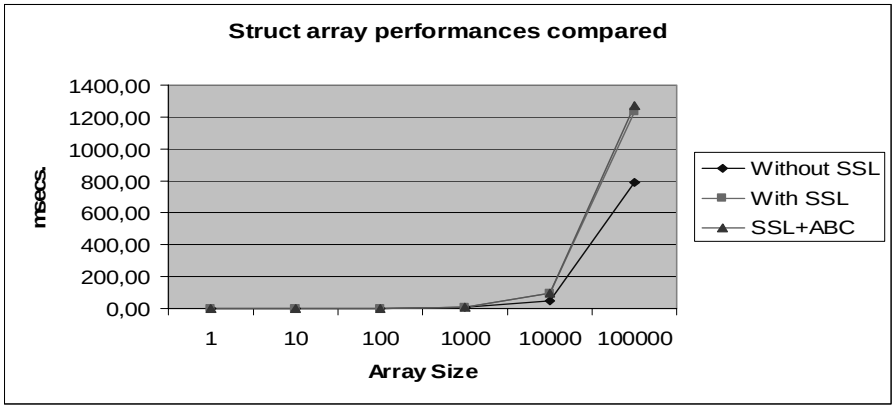


**Figure 8 – Performance of the JaCoWeb-ABC Model**

1) Utilização normal: invocações de métodos sem políticas de segurança.
2) Com SSL: invocações de métodos com o uso de criptografia baseada em SSL.
3) políticas do modelo JaCoWeb-ABC (SSL+ABC): invocações de métodos com a utilização de criptografia e a intervenção das políticas de segurança definidas para o objeto AccessDecisionABC.

Nesta avaliação foram definidas políticas de autorização e de atualização para os métodos do exemplo, em que utilizamos uma adaptação de uma política discricionária do CORBASec dentro das especificações do JaCoWeb-ABC, e também um atributo "contador", que era incrementado a cada invocação dos métodos. Essas medidas visam verificar o custo de desempenho do modelo em relação ao CORBASec convencional.

## 6 Trabalhos Relacionados

Existem alguns trabalhos que implementam modelos de controle de acesso sobre a especificação CORBASec. Em [11], o modelo obrigatório Bell e Lapadula é aplicado ao CORBASec, possibilitando definir políticas baseadas em níveis de segurança da informação, como NÃO-CLASSIFICADO, CONFIDENCIAL, SECRETO e ULTRA-SECRETO. Em

[10], é implementado o modelo RBAC, que possibilita a atribuição de um ou mais papéis a um principal, que são ativados conforme as requisições de seus acessos. Em ambos os casos, a implementação destes controles foram efetuados ao nível do ORB, de forma transparente para a aplicação.

O Resource Access Decision Facility (RAD) [13] é um modelo de controle de acesso que está especificado pela OMG. Ele define políticas de administração e controle de acesso a nível de aplicação, que é necessário quando o processo de decisão deve ser baseado a partir dos parâmetros ou informações externas que não podem ser interceptadas pelo objeto AccessDecision durante a invocação do método.

O Ponder [14] define uma linguagem declarativa orientada a objetos para distribuição de políticas de controle de acesso a recursos de rede e sistemas distribuídos. Ele oferece suporte para definição de políticas de autorização e obrigação, oferecendo ao usuário uma interface simples para especificação de políticas abstratas. Diferente do $UCON_{ABC}$, o conceito de obrigação no modelo Ponder corresponde a uma condição que, se for válida, é seguida por uma ação que pode ser o registro de um evento de auditoria ou a emissão de um alerta de segurança.

Devido ao $UCON_{ABC}$ se tratar de um novo modelo, que é chamado por Park e Sadhu de "a próxima geração de controle de acesso" [5], não identificamos referências no meio acadêmico sobre implementações do modelo $UCON_{ABC}$ em aplicações.

## 7 Conclusão

O modelo $UCON_{ABC}$ permite um controle mais preciso sobre a utilização de usuários aos objetos de um sistema. O JaCoWeb-ABC não efetou mudanças na especificação CORBASec relacionadas à parte cliente e nem foram tratadas questões sobre a administração das políticas de segurança. Assim, novos trabalhos necessitarão dar continuidade a este para poder englobar todas estas mudanças

A combinação do processo de decisão dos 2 níveis de acesso do JaCoWeb-ABC (abcORB e abcIDL) permite que um sistema tenha um controle poderoso e flexível sobre as ações de um usuário, possibilitando segregar as funcionalidades de segurança da lógica de negócios, o que facilita não só o desenvolvimento da aplicação como a administração das políticas de segurança, simplificando ambos os processos.

Devido ao CORBA trabalhar com invocações de métodos remotos, em um formato transacional, acreditamos que os mesmos princípios aplicados neste artigo poderão também ser utilizados em camadas de segurança de outras tecnologias similares, como RMI, DCOM e WebServices, além de WebServers, como o WebSphere da IBM ou IIS da Microsoft. Para essa implementação ser realizada, é necessário analisar a arquitetura da tecnologia em questão, para identificar e associar os seus componentes ao modelo $UCON_{ABC}$. Feito isso, deve ser avaliado como será efetuada a segregação das políticas de segurança que se encontram pela camada de invocação do método remoto (abcORB) e disponibilizar APIs às aplicações para que elas possam atuar em conjunto com as políticas ABC definidas no servidor (abcIDL).

Em [15], é avaliado que o $UCON_{ABC}$, por se tratar simplesmente de um modelo teórico que ainda não possui protótipos implementados, seria significativamente caro e complexo de ser implementado em sistemas reais. Entretanto, com base nos resultados

teóricos e de implementação obtidos neste trabalho, acreditamos que o $UCON_{ABC}$ será cada vez mais adotado como referência para a implementação de um nível mais elevado de controle de acesso.

## Referências


[1] R. Sandhu. "Lattice-based access control models". IEEE Computer, 26(11) 9-19, November 1993

[2] D. D. Downs, J. R. Rub, K. C. Kung, and C. S. Jordan. "Issues in Discretionary Access Control". In: Proceedings of the 1985 IEEE Symposium on Security and Privacy, pp. 208-218, April 1985.

[3] R. Sandhu, "Role-Based Access Control," Advances in Computer Science, vol. 46, Academic Press, 1998.

[4] J. Park and R. Sandhu, "The $UCON_{ABC}$ Usage Control Model". ACM Transactions on Information and System Security (TISSEC), Feb. 2004

[5] R. Sandhu and J. Park. "Usage Control: A Vision for Next Generation Access Control". 2nd International Workshop Mathematical Methods, Models and Architectures for Computer Networks Security (MMM-ACNS), 2003, St. Petersburg, Russia.

[6] X. Zhang, J. Park, F. Parisi-Presicce and R. Sandhu. "A Logical Specification for Usage Control". 9th ACM Symposium on Access Control Models and Technologies, 2004.

[7] Object Management Group. The Common Object Request Broker 2.0/IIOP Specification, Revision 2.0. OMG Document 96-08-04, 1996.

[8] OMG. Security Service Specification, v1.8. OMG Doc. 02-03-11, Mar. 2002

[9] C. M. Westphall e J. S. Fraga. "A Large-Scale System Authorization Scheme Proposal Integrating Java, CORBA and Web Security Models and a Discretionary Prototype". Proc. IEEE LANOMS'99, p. 14–25, dezembro de 1999.

[10] Obelheiro, R. R. e Fraga, J. S. Role-Based Access Control for CORBA Distributed Object Systems 7th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS´2002), 2002, San Diego, CA.

[11] C. M. Westphall, J. S. Fraga, C. B. Westphall e S. C. S. Bianchi. "Políticas de Segurança Obrigatórias: Bell e Lapadula no CORBAsec". XX SBRC, p. 846–861. Búzios, RJ, 2002.

[12] G. Brose. "JacORB: Implementation and Design of a Java ORB". Proc. IFIP DAIS'97, Cottbus, Germany, Chapman & Hall, Sep. 1997.

[13] OMG. Resource Access Decision Facility, v1.0. OMG Doc. 99-03-02, Mar. 1999

[14] Damianou et al. "The Ponder Policy Specification Language". Proceedings of the Workshop on Policies for Distributed Systems and Networks. 2001

[15] Q. He. Privacy Enforcement with an Extended Role-Based Access Control Model. North Carolina State University Computer Science - TR-2003-09. 2003.