Framework de Controle de Uso para Coalizões Dinâmicas de Negócios

Maicon Stihler, Altair Olivo Santin, Arlindo L. Marcon Jr.

Programa de Pós-graduação em Informática – PPGIa
Pontifícia Universidade Católica do Paraná – PUCPR
Curitiba, PR, CEP 80215-901, Brasil
{stihler,santin,almjr}@ppgia.pucpr.br
www home page: http://www.ppgia.pucpr.br

Resumo Os avanços tecnológicos fizeram surgir novas formas de colaborações multi-institucionais, as chamadas Coalizões de Negócios (CN). As CNs são dinâmicas e apresentam dificuldades na implementação de controles de acesso finos empregando as abordagens tradicionais. O gerenciamento, normalmente centralizado, torna o custo operacional proporcionalmente crescente com o número de participantes envolvidos. Para superar estas dificuldades este trabalho propõe uma framework baseada nos recentes avanços no campo de controle de uso, fornecendo controles capazes de refletir as mudanças do ambiente, dos recursos, e dos participantes – estendendo os controles para todo o período de utilização de um recurso. Além disso, optamos por utilizar uma divisão de responsabilidades na escrita das políticas, aumentando a flexibilidade do sistema. As políticas de controle fino são escritas numa linguagem de fácil compreensão por humanos, e posteriormente convertidas em XACML. Um protótipo baseado em serviços web foi desenvolvido e avaliado para demonstrar a viabilidade da proposta.

1 Introdução

As Coalizões de Negócios (CN) tem ganhado relevância com os recentes avanços em tecnologias de comunicação, e com a decorrente facilidade para interconexão de usuários e recursos geograficamente separados. Isto tem permitido que organizações com capacidades distintas colaborem, de modo a complementar suas capacidades, para atingir objetivos mais ambiciosos. As Coalizões de Negócios são caracterizadas por seu dinamismo, tanto com relação a usuários e recursos, quanto com as mudanças no número de organizações participantes.

Neste trabalho, consideramos uma CN composta por um intermediador (*Broker*), organizações consumidoras, e organizações provedoras. O *Broker* opera um conjunto de serviços administrativos que viabilizam a negociação de contratos entre consumidores e provedores. O consumidor é o cliente dos serviços oferecidos pela organização provedora.

As tarefas envolvidas no gerenciamento de usuários e políticas representam um custo operacional considerável, o que torna bastante desejável, para o provedor, que estas tarefas sejam delegadas seguramente a terceiros.

XXXIV Conferencia Latinoamericana de Informática (CLEI 2008), pp. 1315-1324, 2008.

Abordagens comuns para reduzir o custo gerencial consistem na redução da granularidade dos controles. Deste modo usuários são agrupados em conjuntos maiores (e.g. grupos ou papeis). Esta abordagem reduz os custos operacionais, mas inviabiliza um controle mais fino, comprometendo funções de auditoria, por exemplo.

Outro problema presente nas abordagens tradicionais é a baixa expressividade da linguagem de políticas, quando considerados ambientes dinâmicos. Controles tradicionais são basicamente estáticos, ou seja, os controles se aplicam somente no momento em que o acesso é requisitado, não havendo nenhum outro tipo de controle durante a utilização do recurso. Se houver uma alteração nas políticas durante um acesso, isto pode acarretar inconsistências entre essas e os direitos sendo exercidos.

O controle de acesso baseado em credenciais simples e direitos de acesso, carece de flexibilidade para refletir o dinamismo existente nos ambientes de CN. Em Coalizões de Negócios é necessário monitorar as alterações em atributos do ambiente, dos usuários, e dos recursos sendo utilizados constantemente.

Neste trabalho propomos um framework que utiliza o modelo UCON$_{ABC}$[11] para permitir o controle durante toda a sessão de uso, permitindo a aplicação de controles tradicionais, e adicionando a possibilidade de monitoração de condições do ambiente, obrigações de usuários, e dinamismo para refletir alterações em atributos de ambiente, usuário, e recurso. Em nossa framework a escrita de políticas é feita pelo *Broker* (políticas de baixa granularidade) e pelo consumidor do recurso (políticas de controle fino), reduzindo os custos administrativos para o provedor. Nós divergimos da abordagem de Computação em *Grid* [16] por utilizarmos um gerenciamento descentralizado. A infra-estrutura de autorização favorece a autonomia do provedor sem perder o sincronismo com os parâmetros estabelecidos em contrato pré-estabelecidos. Um protótipo baseado em serviços web foi implementando, demonstrando a viabilidade da proposta, e promovendo o emprego de interfaces de baixo acoplamento entre os provedores e consumidores.

O artigo está organizado da seguinte maneira. A Seção 2 apresenta o modelo de controle de uso (UCON$_{ABC}$). A Seção 3 discute a arquitetura proposta. A Seção 4 apresenta um cenário motivador. A Seção 5 trata de detalhes de implementação do protótipo e sua avaliação. A Seção 6 mostra os trabalhos relacionados, e a Seção 7 apresenta as conclusões.

2 Controle de Uso

O controle de uso (UCON$_{ABC}$) pode ser entendido como uma extensão aos modelos tradicionais (e.g. DAC, MAC, RBAC), e inclui suporte a mutabilidade de atributos e continuidade de avaliação dos controles . A mutabilidade de atributos está relacionada a capacidade do controle de atualizar atributos do usuário ou recurso, enquanto que a continuidade dos controles implica na avaliação e aplicação dos controles durante todo o uso do recurso. O uso é dividido em dois momentos: antes da concessão do direito de acesso (*pre*); e durante o uso (*on-*

going). Estes dois momentos são utilizados para definir quais controles devem ser aplicados em um determinado momento.

As políticas são escritas como predicados que operam sobre os atributos do usuário, recurso, e do ambiente. Os controles são classificados em autorizações, obrigações, e condições. Todos podem ser subdivididos em *pre* ou *ongoing* (e.g. autorização *pre*). Autorizações avaliam se o usuário possui os direitos necessários para executar a ação desejada. Obrigações verificam se ações obrigatórias foram executadas por algum indivíduo de modo a conceder ou manter o acesso. Condições avaliam atributos de ambiente que independem de usuários e recursos.

3 Arquitetura

Na Figura 1 apresentamos uma visão geral da arquitetura proposta, composta por três partes principais, a infra-estrutura do *Broker*, a organização consumidora, e o provedor.

Na infra-estrutura do *Broker* (Figura 1) observa-se três elementos: O serviço de negociação, onde o consumidor pode procurar e contratar serviços de um provedor; O serviço de identificação interdomínio, para permitir a identificação dos usuários de forma única em todo o ambiente colaborativo; e o serviço de notificações, utilizado para alertar usuários e administradores sobre ocorrências que exigem sua atenção.

A organização provedora preocupa-se somente com a aplicação de controles de baixa granularidade (aplicados em nível de organização), enquanto que as organizações consumidoras tem liberdade para definir as políticas de controle fino para seus próprios usuários. O *Broker* monitora a conformidade contratual dos serviços providos.

Na Figura 1, vemos que ao contratar um serviço no *Broker*, o consumidor recebe um certificado de autorização contendo os limites de uso (evento 1), assim como permissão para criar políticas de controle fino no ponto de administração de políticas (PAP) do provedor (evento 2). Este certificado também permite a criação de certificados de identificação para os usuários do consumidor (evento 3). O usuário obtém sua identidade (evento 4), e faz sua inscrição obrigatória no serviço de notificações do *Broker* (evento 5). Feito isso, procede-se a requisição de uso (evento 6), que será interceptada pelo ponto de aplicação de políticas (PEP). O PEP verifica a identidade do usuário junto ao serviço de identificação interdomínio (evento 7). Se a identidade for autêntica e válida, uma requisição é enviada ao ponto de decisão de política (PDP) (evento 8), onde é feita a avaliação das políticas em nível de organização. Caso o resultado da avaliação no PDP seja positivo, faz-se a avaliação das políticas de controle fino. O resultado é enviado ao PEP, que fará a aplicação da decisão, liberando o acesso (evento 9) e a publicação de notificações no serviço de notificações (evento 10).

Nesta arquitetura elimina-se a necessidade do provedor criar políticas de controle fino, sem perder o controle sobre seus serviços. Para isso foi empregada uma abordagem baseada em certificados de autorização. Quando o provedor publica um serviço para negociação, um certificado de autorização com os limites máxi-

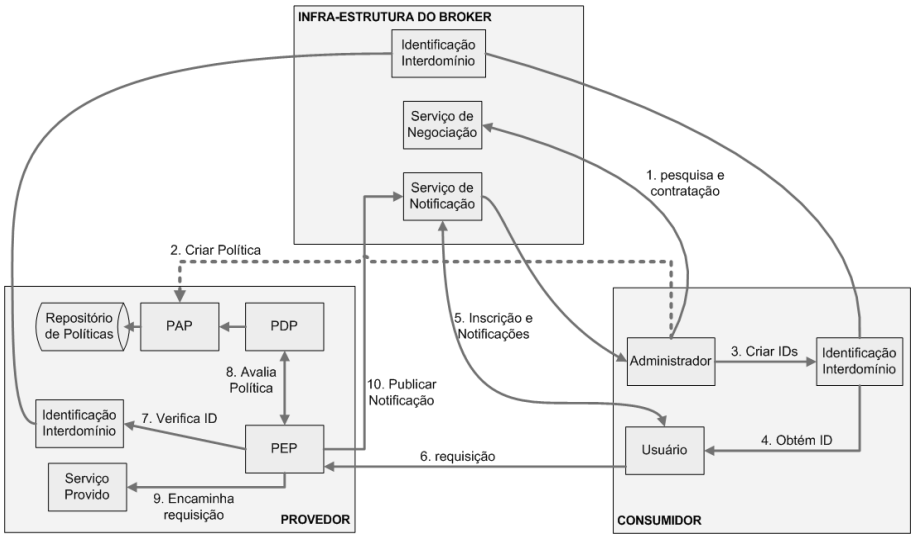

Figura 1. Visão Geral da Arquitetura

mos a serem negociados é fornecido ao *Broker*. A partir deste certificado, o *Broker* pode delegar parcialmente, ou totalmente, estes limites para o consumidor. Entretanto, se o *Broker* delegar limites além daqueles presentes no certificado delegado pelo provedor, quando o consumidor tentar acessar os recursos a operação será negada. Através de um processo de redução de cadeias (sumarização de direitos, com verificação de integridade de delegação) de certificados, o provedor poderá detectar a violação.

A estrutura dos certificados de autorização permite que as cadeias de certificados sejam utilizadas para identificar usuários. Com isso eliminou-se a necessidade de criação de contas de usuários em cada provedor, assim como a necessidade de utilização de serviços externos para executar a autenticação de usuários. Como o certificado de autorização dos usuários forma uma cadeia que se origina no certificado do provedor, a validade do mesmo pode ser facilmente comprovada.

O sistema de notificações é utilizado para promover uma melhor experiência ao usuário final. Ao transformar o sistema de notificações em uma obrigação para utilização do recurso, obtemos um canal de comunicação com a aplicação do usuário e dos administradores. Através das políticas podemos estabelecer ações que devem ser invocadas pelo PEP em determinadas situações como, por exemplo, alertar o usuário que a sua quota de uso está se esgotando. Um *token* com período de validade curto é fornecido como prova de inscrição no serviço de notificação, devendo esse ser revalidado regularmente pelo usuário, para que seu direito não seja revogado – isto equivale a uma obrigação UCON_{ABC}.

3.1 Suporte ao UCON$_{ABC}$

Os atributos utilizados na avaliação de políticas podem sofrer alterações durante uma sessão de uso, como efeito colateral do uso ou por alterações nas políticas de uso. O responsável por escrever as políticas de uso pode definir quais atributos devem ser atualizados. A atualização é efetuada por um serviço que abstrai as funções de atualização dos atributos, assim, o administrador não precisa conhecer detalhes de implementação dos possíveis repositórios de atributos. Atributos que não são atualizados diretamente pela política, mas que por um motivo ou outro venham a sofrer alterações, também podem afetar a avaliação da mesma.

A monitoração de atributos possibilita a continuidade dos controles. O PDP registra todos os atributos relacionados a cada sessão de uso. Ao receber um aviso de alteração de atributo, o PDP identifica as sessões afetadas pelo atributo, e reavalia as políticas aplicáveis. Quando uma violação de política é detectada, o PDP invoca o PEP, que pode fornecer um período de tempo para regularização da situação. Passado este tempo, o PEP pode suspender a sessão de uso e informar a violação ao administrador, ou revogar unilateralmente o acesso do usuário. Os controles de uso e atualizações de atributos podem ser aplicados antes (*pre*) do acesso e e durante (*ongoing*) o acesso.

Os mecanismos necessários para a avaliação de predicados de uso são ocultados do administrador de políticas, este somente vê tais predicados como se fossem atributos num sistema de monitoração oferecido pelo provedor. A utilização de repositórios de atributos é uma possibilidade, mas de pouca aplicabilidade em nosso cenário.

4 Cenário

Suponhamos um ambiente de CN no qual existem dois participantes. Um deles é um provedor de espaço de armazenamento (serviço de *storage*). O outro é uma organização que necessita do serviço de *storage* (consumidor). Consideramos que o consumidor deseja contratar 100 gigabytes do serviço de *storage*.

Após identificar a oferta de serviço no serviço de negociação do *Broker*, o consumidor estabelece um contrato, recebendo em troca um certificado de autorização contendo os limites de utilização do serviço, assim como permissões para criar políticas de controle de uso no repositório do provedor. O consumidor, ao apresentar o certificado ao provedor, faz com que o provedor configure os controles locais de baixa granularidade (controle fino) de acordo com os limites especificados no mesmo.

O consumidor obtém, então, permissão para criar políticas de controle de uso para seus usuários. Através da infra-estrutura de monitoração do provedor, o consumidor (organização) pode incluir uma variedade de atributos para serem avaliados por sua política, sem se preocupar com detalhes de implementação. Esta política, que é escrita na forma de predicados, é criada através de uma interface de administração. Janelas de diálogo correspondentes a cada estágio da sessão de uso (antes ou durante)permitem que o consumidor aplique os controles

desejados em todas as fases do uso. Após terminar a escrita das políticas, um *parser* converte esses predicados em regras de uma linguagem compreensível pelos mecanismos de controle.

O usuário obtém um certificado de autorização criado pelo administrador do seu domínio, e faz a inscrição no serviço de notificações. Sua aplicação envia requisições de uso para o PEP do provedor, apresentando um comprovante de que está inscrito no serviço de notificação (um *token* que deve ser revalidado regularmente) juntamente com seu certificado de autorização. O PEP invoca o serviço de validação do *token* e do certificado, enviando então as informações relevantes para o PDP para que este tome uma decisão. A decisão é remetida ao PEP que pode, entre outras coisas ações, permitir, negar, suspender, ou gerar notificações para o usuário.

5 Implementação do Protótipo

Nosso protótipo é composto por Serviços Web [2], objetivando baixo acoplamento, segurança, escalabilidade, extensibilidade, e interoperabilidade. A comunicação é feita através de *Simple Object Access Protocol* (SOAP) [6] sobre HTTP, o que facilita a integração com as infra-estruturas já existentes, pois a porta HTTP costuma estar aberta para outros serviços de Internet, reduzindo custos de administração de segurança e facilitando a interoperabilidade.

Para proteção das mensagens fim-a-fim adotou-se a WS-Security [7], especificação de segurança para Serviços Web. Aplicamos *timestamps*, assinaturas digitais, e cifragem de mensagens. Assinamos com chaves assimétricas, para identificar participantes sem ambigüidades. Empregamos a especificação WS-SecurityPolicy [8] para definir como aplicar os mecanismos da WS-Security para proteger as comunicações. Os *tokens* do serviço de notificação são assinados utilizando XML Signature [3].

O protótipo foi construído a partir da framework Apache Axis2 [14], por este possuir diversas classes que facilitam consideravelmente a criação de Serviços Web. Nesta framework também é fornecido, através do módulo Apache Rampart, suporte para WS-Security, WS-SecurityPolicy, e WS-Trust [9].

Para a representação de políticas em nível de mecanismo foi utilizada a especificação XACML [5], um padrão interoperável para definição de políticas de segurança. A interpretação de políticas em nível de mecanismo é realizada com a biblioteca XACML da Sun [13].

O sistema de identificação interdomínio e negociação é baseado em certificados SPKI (*Simple Public Key Infrastructure*) [4] e SDSI (*Simple Distributed Security Infrastrcuture*) [12]. SPKI/SDSI oferece uma infra-estrutura de chaves públicas flexível, e um modelo de autorização para construir sistemas distribuídos seguros e escaláveis. Os sujeitos são uma chave pública e qualquer um pode emitir certificados. Baseados em autorização, os certificados formam cadeias de autorização. Um serviço de *token* seguro (STS) usando XKMS (*XML Key Management Specification*) gerencia as cadeias de certificados e emite certificados para usuários.

5.1 Criação de Políticas UCON$_{ABC}$

Na Figura 2 é apresentado um exemplo de política baseada em predicados criada pelo consumidor. Na linha 01 existe um predicado que verifica se o grupo do usuário se chama *Developers*. Na linha 03 é verificado se o usuário possui o direito para executar a ação *write*. A inscrição no serviço de notificação é verificada na linha 05, a verificação do *token* é realizada pelo PEP.

```
****************** Controles Pre *****************
01 verifyGroup( user )
02   (user.group  eq ‘‘Developers’’)
03 verifyRight( user )
04   contains(user.permissions, ‘‘Write’’)
05 isSubscribed( user )
06   pep.isValid( user.token )
*************** Controles Ongoing ***************
07 verifyQuota( user )
08   ((service.quotaOrg(user.OrgID) le (50))
09     and
10    (service.quotaUser(user.ID)  lt 10)
11   )
12    or
13   (service.quotaUser(user.ID)  lt 5 )
14 verifyTimeShift( user )
15   ( env.now  ge user.startTS)
16    and
17   ( env.now  le user.endTS)
18 verifyToken( user )
19   pep.isValid( user.token )
```

Figura 2. Política baseada em predicados

Os controles aplicados durante o uso (*ongoing*) também estão na Figura 2. Na linha 07 um predicado composto (utilizando operadores lógicos *and*, *or*) verifica, primeiro, se a quota utilizada pela organização consumidora é menor ou igual a metade do limite contratado (linha 08). Em caso positivo, o usuário pode utilizar a sua quota extra se esta já não foi consumida (linha 10), caso contrário o usuário é limitado a sua quota padrão (linha 13). Na linha 14 é apresentado um predicado para verificar se o usuário está utilizando o recurso dentro de seu horário de trabalho. Por fim, na linha 18, verifica-se a conformidade com a renovação do *token*, o que pode ser considerado como uma obrigação UCON$_{ABC}$.

Como os mecanismos são incapazes de interpretar uma política como a apresentada na Figura 2, é necessário a utilização de um *parser* para gerar as políticas correspondentes em XACML. Entretanto, XACML não foi concebida com o intuito de representar controles de uso baseados em predicados. Isto não impede, no entanto, que a utilizemos para emular o comportamento esperado.

Para a geração das políticas XACML cada família de controle (e.g. autorizações *pre*, condições *ongoing*, etc.) é agrupada em uma política separada. Isto é necessário para que o avaliador UCON$_{ABC}$ consiga selecionar os controles corretos para cada estágio da avaliação. As regras XACML suportam expressões condicionais que permitem que o *parser* crie estruturas equivalentes a predicados UCON$_{ABC}$. Estas expressões condicionais ajudam a avaliar se o valor booleano de uma regra é verdadeiro ou falso.

Para exemplificar, uma regra XACML derivada do predicado verifyQuota da Figura 2 é gerada da seguinte maneira: o *parser* identifica que o operador lógico base do predicado verifyQuota é o operador *or* (linha 12), e gera o equivalente XACML *function:or*. Nas linhas 08 a 11 está o primeiro operando, e na linha 13 o segundo operando.

Avançando na conversão do primeiro operando (linhas 08 a 12) observa-se que o operador base é um *and*, sendo então representado em XACML como *function:and*. O primeiro operador relacional (linha 08) é convertido para o equivalente XACML *function:integer-less-than-orequal* que recebe os respectivos parâmetros (atributo de usuário e valores codificados). A segunda função (linha 10) resulta em *function:integer-less-than*, também com seus respectivos parâmetros. A interpretação do predicado na linha 13 resulta em uma função *function:integer-less-than* tomando os atributos de usuário e valores codificados como parâmetros.

O estado a que se aplica um determinado controle é definido no cabeçalho das políticas XACML, deste modo, com base nas informações da sessão do usuário o avaliador UCON$_{ABC}$ é capaz de selecionar somente uma política aplicável. No caso de atualizações de atributos definidas em política, não existem regras de controle, mas sim expressões XACML chamadas <Obligation> que carregam um identificador de função e atributos como parâmetros, os quais podem ser utilizados para invocar os mecanismos de atualização. Estas *tags* XACML não devem ser confundidas com as obrigações UCON$_{ABC}$, pois são na verdade ações que o PEP deve executar ao receber uma decisão do PDP.

Como a biblioteca XACML não possui o conhecimento de conceitos de controle de uso, é necessária a criação de um outro módulo, que chamamos de avaliador UCON$_{ABC}$, responsável por identificar quais controles devem ser aplicados e então invocar o PDP com os parâmetros corretos. Assim, ao receber uma requisição, o avaliador, por exemplo, executa as atualizações *pre*, seguidas das autorizações *pre*, obrigações *pre*, e condições *pre*. Caso uma das avaliações falhe, as demais não são realizadas e o serviço de notificação é acionado.

5.2 Avaliação do Protótipo

Como prova de conceito, executamos testes de armazenamento de dados no *storage* com cargas de 1 mil, 10 mil, 50 mil, e 100 mil bytes. Também foram executados, com sucesso, ensaios para verificar se o avaliador (PDP) reagia corretamente quando havia alteração nas variáveis de ambiente, atributos, etc. Os testes de performance foram executados 100 vezes, assegurando-se um coeficiente de variação de menos de 5%. O ambiente de testes era composto por três hosts

conectados em uma rede Ethernet 100 Mbps. Os hosts eram AMD64 2800, com 1024 MB de memória RAM. Os testes foram executados em sistemas Linux 2.6, exceto os testes SPKI/SDSI, executados em Windows XP.

O processo decisório é pouco influenciado pela carga de dados utilizada. O tempo de resposta tende a aumentar para cargas maiores que 10 mil bytes devido à forma pouco eficiente de gerenciar de anexos SOAP nos serviços web. O gerenciamento de cadeias de certificados SPKI/SDSI foi avaliado com uma cadeia de três certificados e quatro chaves, executando-se de 25 a 200 requisições ao sistema de identificação, para validar as requisições. Observamos que o tempo de resposta fica estável com requisições seqüenciais, para requisições em paralelo o serviço apresenta falhas em torno de 10% das requisições, suportando atender satisfatoriamente 150 requisições simultâneas.

6 Trabalhos Relacionados

Um sistema de autorização baseado em UCON$_{ABC}$ é proposto em [17]. O sistema proposto tem dificuldades para suportar os requerimentos de nosso cenário, especialmente por ter uma administração de usuários e políticas totalmente centralizada, podendo causar problemas de desempenho e segurança. O sistema não suporta UCON$_{ABC}$ em todo seu espectro, além de utilizar algumas extensões não padronizadas na especificação XACML. Em [10] é analisado o controle de uso em máquinas virtuais, para ambientes de *grid* – este sistema pode ser interessante para aplicações clássicas *standalone*, para colaborações de grande escala seu custo para gerenciar políticas é proibitivo.

O Virtual Organization Membership Service (VOMS)[1] é um serviço de *membership* que gerencia informação de autorização em ambientes de *grid* de modo estruturado. As informações sobre os papéis dos usuários dentro da colaboração ficam num servidor central e as políticas no provedor. VOMS não aborda controle de uso.

Akenti[15] aborda o problema de identificação de usuário e especificação de políticas por múltiplos proprietários. Utilizando infra-estrutura de chave pública Akenti resolve o problema de unificação de identidades, porém os proprietários definem as políticas de autorização em certificados, que são coletados pelo PDP para a tomada de decisão. Assim como o VOMS, no Akenti não há suporte para controle de uso e nem é explorado o poder e flexibilidade de SPKI/SDSI.

7 Conclusões

Este trabalho mostrou a viabilidade de aplicação de controle de uso em coalizões dinâmicas de negócios, distribuindo os controles tanto para o provedor (delegando para o *Broker*) como para o consumidor. Utilizando para isso um processo gerencial flexível simples. A utilização de SPKI/SDSI simplificou o sistema de identificação e autorização, resultando numa redução de custos e facilitando colaborações *ad hoc*.

A utilização de especificações para serviços web, especificação de políticas, segurança, etc., reduziram consideravelmente a dependência de tecnologias proprietárias, assim como promoveram o baixo acoplamento – já favorecido pelo modelo arquitetural adotado. O serviço de notificação ajudou a promover uma melhor interação entre os usuários e o serviço provido, além de servir como exemplo de obrigação UCON_{ABC} em nossa proposta. Ficou demonstrado que é possível implementar controles UCON_{ABC} sem a necessidade de violar as especificações (e.g. XACML).

Por fim, a avaliação do protótipo mostrou que a integração das tecnologias não impactou significativamente o desempenho geral do sistema. Nós concluímos que o presente trabalho trás um conjunto de vantagens qualitativas que não foram identificadas em outros trabalhos da literatura técnica.

Referências

1. R. Alfieri, R. Cecchini, L. Dell'Agnello, . Frohner, A. Gianoli, K. Lrentey, and F. Spataro. Voms, an authorization system for virtual organizations. *Lecture Notes In Computer Science*, pages 33–40, 2004.
2. D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, and D. Orchard. Web services architecture, 2004.
3. D. Eastlake, J. Reagle, and D. Solo. Xml-signature syntax and processing., 2002.
4. C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. SPKI Certificate Theory. RFC 2693 (Experimental), Sept. 1999.
5. S. Godik and T. Moses. Oasis extensible access control markup language (xacml) specification 1.1., 2002.
6. M. Gudgin, M. Hadley, N. Mendelsohn, J. Moreau, H. F. Nielsen, A. Karmarkar, and Y. Lafon. Soap version 1.2 part 1: Messaging framework (second edition), 2007.
7. K. Lawrence and C. Kaler. Web services security: Soap message security 1.1 (ws-security 2004), 2004.
8. K. Lawrence and C. Kaler. Ws-securitypolicy 1.2, 2007.
9. K. Lawrence and C. Kaler. Ws-trust 1.3, 2007.
10. F. Martinelli, P. Mori, and A. Vaccarelli. Towards continuous usage control on grid computational services. *Proceedings of the The International Conference on GRID Networking and Services 2005*, 2005.
11. J. Park and R. Sandhu. The ucon_{abc} usage control model. *ACM Trans. Inf. Syst. Secur.*, 7(1):128–174, 2004.
12. R. L. Rivest and B. Lampson. Sdsi - a simple distributed security infrastructure, 2007.
13. Sun Microsystems, Inc. Sun xacml implementation, 2007.
14. The Apache Foundation. Apache axis2 web services engine, 2007.
15. M. R. Thompson, A. Essiari, and S. Mudumbai. Certificate-based authorization policy in a pki environment. *ACM Trans. Inf. Syst. Secur.*, 6(4):566–588, 2003.
16. S. Venugopal, S. Venugopal, R. Buyya, and K. Ramamohanarao. A taxonomy of data grids for distributed data sharing, management, and processing. *ACM Comput. Surv.*, 38(1):3, 2006.
17. X. Zhang, M. Nakae, M. J. Covington, and R. Sandhu. Toward a usage-based security framework for collaborative computing systems. *ACM Trans. Inf. Syst. Secur.*, 11(1):1–36, 2008.