UFSM

Trabalho de Graduação

Implementação do CIBAC no SIE usando SOA

Cesar Augusto Guerra de Souza

Curso de Ciência da Computação

Santa Maria, RS, Brasil

2006

Implementação do CIBAC no SIE usando SOA

por

Cesar Augusto Guerra de Souza

Trabalho de Graduação apresentado ao Curso de Ciência da Computação – Bacharelado, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de
Bacharel em Ciência da Computação.

Curso de Ciência da Computação

Trabalho de Graduação nº 213

Santa Maria, RS, Brasil

2006

Universidade Federal de Santa Maria
Centro de Tecnologia
Curso de Ciência da Computação

A Comissão Examinadora, abaixo assinada, aprova o Trabalho de Graduação

Implementação do CIBAC no SIE usando SOA

elaborado por

Cesar Augusto Guerra de Souza

como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Comissão Examinadora:

Prof. Dr. Raul Ceretta Nunes
(Orientador)

Prof^a. Me. Oni Reasilvia Sichonany
(Co-orientadora)

Prof^a. Dr^a. Iara Augustin

Prof^a. Dr^a. Márcia Pasin

Santa Maria, 15 de setembro de 2006.

Sumário

Lista de Figuras

RESUMO

Trabalho de Graduação
Ciência da Computação
Universidade Federal de Santa Maria

Implementação do CIBAC no SIE usando SOA

Autor: Cesar Augusto Guerra de Souza

Orientador: Prof. Dr. Raul Ceretta Nunes

Local e Data da Defesa: Santa Maria, 15 de setembro de 2006.

Atualmente, os assuntos confidencialidade e privacidade, assim como heterogeneidade e mudança, são preocupações inevitáveis na área de Tecnologia e Informação, em especial na área médica. O primeiro requer mecanismos de controle de acesso enquanto que o segundo requer arquiteturas com baixo acoplamento, tal como as orientadas a serviço.

Afim de atender requisitos de confidencialidade e privacidade na gerência de informações do Hospital Universitário da UFSM, foi proposto um modelo de controle de acesso baseado em informações contextuais - CIBAC - que realiza o controle de acesso ao Sistema de Informações para o Ensino - SIE, software responsável por essa tarefa.

Este trabalho descreve a implementação do modelo CIBAC junto ao SIE, utilizando uma Arquitetura Orientada a Serviços - SOA - e a tecnologia de Web Services, a fim de facilitar a sua utilização em ambientes heterogêneos e sujeitos a mudanças. A versão SIE/CIBAC foi denominada SIE-Saúde.

ABSTRACT

Final Undergraduate Work
Computer Science
Universidade Federal de Santa Maria

SOA based CIBAC Implementation on SIE

Author: Cesar Augusto Guerra de Souza

Advisor: Dr. Raul Ceretta Nunes

Local e Data da Defesa: Santa Maria, 15 de setembro de 2006.

Nowadays, the subjects confidentiality and privacy, as well as heterogeneity and change, are inevitable concerns in the Technology and Information area, specially in the medical area. The first relate to the mechanisms of access control whereas the second relate to the low coupling architectures, like the service-oriented architectures.

To take care of confidentiality and privacy requirements of information management of the UFSM Hospital, a Contextual Information-Based Access Control model - CIBAC - was considered. The CIBAC was used to control the access to the hospital information management system, called System of Information for Education - SIE.

This work describes the implementation of model CIBAC on SIE using a Service-Oriented Architecture - SOA - and the Web Services technology, in order to facilitate to its use in heterogeneous and changeable environments. The SIE/CIBAC implementation was named SIE-Health.

Capítulo 1

Introdução

Executivos da área de Tecnologia da Informação (TI) têm enfrentado o desafio de cortar despesas e aumentar a utilização das tecnologias existentes; ao mesmo tempo procuram, continuamente, melhorar o serviço fornecido aos seus clientes, para que estes possam ser mais competitivos e se adequarem às estratégias de negócio. Existem dois aspectos importantes por trás desses desafios: heterogeneidade e mudança. A heterogeneidade é devida ao fato de muitas empresas possuirem diversos sistemas, aplicações e arquiteturas de diferentes épocas e tecnologias. Integrar produtos de diferentes fornecedores e plataformas tem sido uma tarefa difícil. As mudanças na área de TI acontecem por causa da alta competitividade no mercado e, conseqüentemente, observam a mudança das necessidades e requisitos dos clientes, Wiehler (2004).

Para enfrentar esses desafios é preciso uma arquitetura que forneça uma plataforma para o desenvolvimento de aplicações com baixo acoplamento, localização transparente e independência de protocolo, minimizando as dificuldades devido à heterogeneidade e mudanças nos sistemas utilizados, Erl (2005). Este trabalho propõe o uso de uma Arquitetura Orientada a Serviços (SOA) como forma de atender a esses requisitos.

SOA representa uma forma de construir sistemas distribuídos onde as funcionalidades da aplicação estão organizadas como serviços independentes que se comunicam com usuários finais ou outros serviços, com um baixo acoplamento entre as partes, Endrei (2004). Para a implementação dessa arquitetura é utilizado *Web*

Services.

Web Services é uma tecnologia relativamente nova e tem recebido grande aceitação como uma importante implementação de uma arquitetura orientada a serviços. Isto porque *Web Services* permite a integração de aplicações extremamente heterogêneas através da Internet. Sua especificação é completamente independente da linguagem de programação, sistema operacional e hardware utilizado, isso para obter um baixo acoplamento entre os serviços, Erl (2004).

Este trabalho utiliza *Web Services* para a implementação de um modelo de controle de acesso baseado em informações contextuais, chamado *Contextual Information-Based Access Control* (CIBAC), Soares (2006), junto ao Sistema de Informações para o Ensino (SIE) contendo o módulo Saúde, na forma de uma Arquitetura Orientada a Serviços (SOA). Essa arquitetura provê serviços para a utilização de políticas e recursos necessários à proteção de informações médicas.

A disponibilização de informações clínicas em redes de computadores levanta questionamentos sobre a privacidade dos pacientes e a integridade e confidencialidade dos dados. O controle de acesso é um ponto-chave para manter tais requisitos, e a utilização do mesmo como um serviço pode tornar esta funcionalidade facilmente acessível às diversas aplicações, em especial ao SIE. Com a utilização do CIBAC pretende-se tornar esse controle mais flexível, pois é possível mapear diversas informações em seus respectivos contextos, possibilitando a criação de políticas que atendam às necessidades da área de saúde.

1.1 Justificativa

O estudo sobre uma Arquitetura Orientada a Serviços para a implementação do CIBAC têm por objetivo conhecer e usar os benefícios desta para resolver os principais problemas enfrentados pelas empresas: heterogeneidade e mudança. Algumas justificativas do porquê da utilização da SOA são:

- a SOA fornece uma camada de abstração que permite uma organização continuar investindo em TI, através do encapsulamento dos recursos existentes na

forma de serviços que forneçem as funções de negócio, não havendo a necessidade de refazê-los integralmente;

- o ponto de integração em uma Arquitetura Orientada a Serviços é a especificação do serviço, e não sua implementação. Isto fornece uma transparência e reduz o impacto quando a infra-estrutura e implementação mudam. Fornecendo uma especificação de serviço para recursos de software existentes, construídos com o uso de diferentes plataformas, torna a integração destes mais fácil de gerenciar, pois a complexidade é isolada. Isto se torna cada vez mais importante visto que mais empresas trabalham juntas para fornecer funcionalidades agregadas, Endrei (2004);

- serviços de negócio expostos com baixo acoplamento podem mais facilmente serem usados e combinados para compor outros serviços. A habilidade de compor novos serviços através da interação de serviços existentes fornece uma grande vantagem às empresas que querem dar uma resposta rápida às demandas de mercado. O reuso de serviços reduz o tempo e recursos necessários para desenvolver as funcionalidades requeridas, o que, também, reduz o impacto das mudanças; e

- a utilização de *Web Services* é suportada por diversas plataformas de desenvolvimento, permitindo que sistemas legados possam acessar os serviços implementados no âmbito deste trabalho.

1.2 Objetivos

Este trabalho tem como objetivo a implementação de um modelo de controle de acesso baseado em informações contextuais junto ao Sistema Integrado de Ensino - módulo Saúde - usando uma arquitetura orientada a serviço, utilizando *Web Services* como implementação dessa arquitetura. E como objetivos específicos, têm-se:

- entender os conceitos que envolvem uma Arquitetura Orientada a Serviços;

- estudar os diversos padrões relacionados a *Web Services*, como SOAP, WSDL, UDDI etc;
- implementar o modelo CIBAC como estudo de caso para uma SOA;
- testar a implementação do modelo e adaptar o Sistema de Informações para o Ensino (SIE), desenvolvido pelo CPD/UFSM, para utilizá-la; e
- publicar os resultados do trabalho.

O restante deste texto está organizado em 3 capítulos:

- **Capítulo 2 - Revisão da Literatura:** apresenta as principais características dos conceitos e tecnologias usados neste trabalho e do CIBAC, o modelo de controle de acesso implementado;
- **Capítulo 3 - Implementação do CIBAC:** mostra os detalhes de implementação do modelo CIBAC; e
- **Capítulo 4 - Conclusão:** apresenta as considerações finais e trabalhos futuros.

Capítulo 2

Revisão da Literatura

Este capítulo contextualiza a pesquisa bibliográfica e expõe as principais características de uma Arquitetura Orientada a Serviços. Também é feita uma explicação sobre os padrões relacionados com a tecnologia *Web Services*, utilizados para a implementação do modelo CIBAC na forma de uma SOA. Em seguida, apresenta alguns detalhes sobre o SIE, com foco em seu módulo de Gerenciamento e Controle de Acesso (SGCA), e explica o modelo CIBAC, o qual pretende resolver os problemas do SIE relacionados com o controle de acesso a dados médicos. Por fim, é apresentada uma linguagem de descrição de políticas de acesso.

2.1 Arquitetura Orientada a Serviços

Uma Arquitetura Orientada a Serviços (SOA) é composta por um grupo de serviços que se comunicam uns com os outros para desempenhar um processo de negócio. Cada serviço, embora independente, é interoperável com os demais, podendo assumir diferentes papéis em diferentes processos, através de sua reutilização, Endrei (2004).

Dentro de uma SOA, serviços podem ser usados por outros serviços ou programas. Para a utilização de um determinado serviço é preciso ter conhecimento acerca deste. Dentro dessa arquitetura, este objetivo é alcançado através do uso de descritores de serviço. Um descritor, em seu formato mais básico, estabelece o nome do serviço, os dados esperados e retornados, e a sua localização.

Estabeler uma Arquitetura Orientada a Serviços dentro de uma empresa não necessariamente requer a substituição da estrutura pré-existente. Um dos aspectos mais atrativos dessa arquitetura é a habilidade de introduzir unidade em ambientes possivelmente diversos. Enquanto *Web Services* (ver seção 2.2) possibilita a união de tecnologias distintas, SOA a promove através do estabelecimento e padronização da habilidade para encapsular a lógica de aplicações legadas, ou não legadas, e expô-las via um *framework* de comunicação comum, aberto e padronizado, Erl (2005).

A figura 2.1 mostra as colaborações em uma Arquitetura Orientada a Serviços. As colaborações seguem o paradigma "busque, ligue e invoque" (*find, bind and invoke*), onde um "serviço consumidor" localiza dinamicamente um "serviço de registro" em busca de algum serviço que satisfaça determinados critérios. Se o serviço procurado existe, então o "serviço de registro" fornece ao consumidor a descrição daquele serviço, o qual contém informações sobre sua interface e localização, Erl (2004).

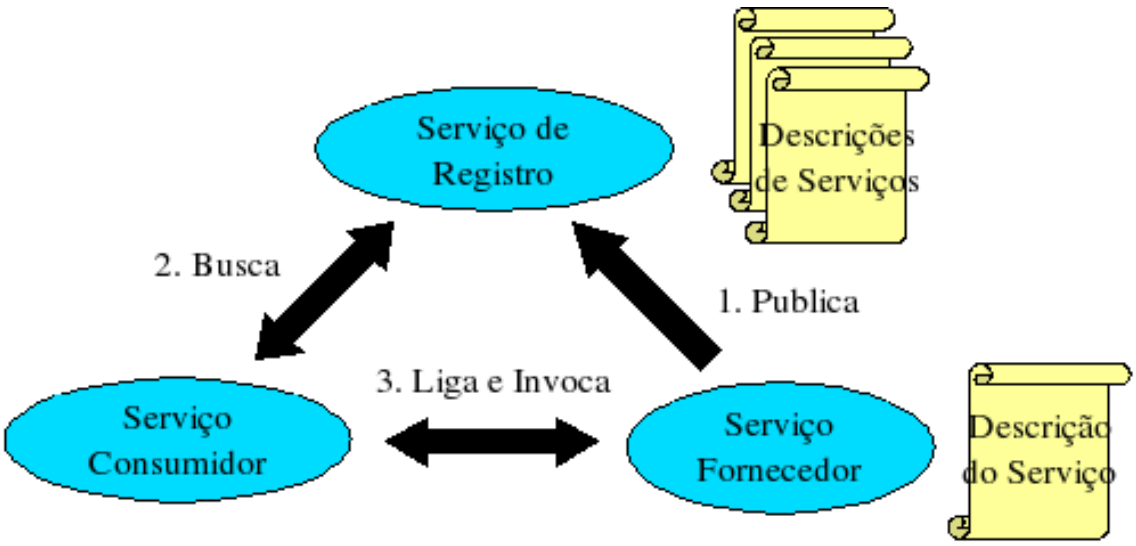

Figura 2.1: Colaborações entre os serviços numa SOA

Os papéis em uma Arquitetura Orientada a Serviços são:

- **Serviço Consumidor:** o serviço consumidor pode ser uma aplicação, um módulo de software ou outro serviço. Ele inicia pela busca em um serviço de registro, logo após a localização, ele se liga ao serviço fornecedor encontrado

através de um protocolo de transporte e executa operações desejadas. O serviço consumidor executa operações de acordo com a interface estabelecida pela descrição do serviço fornecedor.

- **Serviço Fornecedor:** o serviço fornecedor é uma entidade que possui um endereço acessível pela rede, aceitando e executando requisições feitas por consumidores. Publica seu serviço e o contrato de sua interface para o serviço de registro, permitindo que consumidores possam facilmente descobri-lo e acessá-lo.

- **Serviço de Registro:** o serviço de registro é quem possibilita a descoberta de serviços. Contém um repositório dos serviços disponíveis e permite a busca destes, feita por serviços consumidores.

2.1.1 Princípios de uma Arquitetura Orientada a Serviços

A seguir mostra-se os princípios básicos de uma Arquitetura Orientada a Serviços.

2.1.1.1 Serviços são reusáveis

A orientação a serviços encoraja o reuso de todos os serviços, não importando se os requisitos para o reuso inicialmente existem. Através da aplicação de padrões de projeto que tornam cada serviço potencialmente reusável, aumentam as chances dele ajudar em futuros requisitos sem grandes esforços de desenvolvimento.

Este princípio facilita todas as formas de reuso, incluindo operabilidade entre aplicações, composição, e serviços de utilidade geral. Um serviço é simplesmente uma coleção de operações relacionadas. É, portanto, a lógica encapsulada pelas operações individuais que devem ser consideradas reusáveis para garantir a existência de um serviço reusável.

2.1.1.2 Serviços possuem baixo acoplamento

Um serviço possui baixo acoplamento quando adquire conhecimento de outro serviço e mesmo assim continua independente. Essa característica é alcançada através do contrato de serviço estabelecido, permitindo que eles interajam através da troca de mensagens pré-definidas. O uso de um modelo de troca de mensagens aberto e padronizado, o qual faz uso da *eXtensible Markup Language* (XML) para descrever seus tipos de dados e operações, promove o baixo acoplamento entre os serviços.

Uma das características que tendem a evoluir naturalmente com a aplicação dos princípios de uma Arquitetura Orientada a Serviços é a abstração. Camadas de abstração são obtidas posicionando os serviços como pontos de acesso a uma variedade de recursos e lógica de processamento. O baixo acoplamento dos serviços permite que diferentes consumidores interajam com eles. Esses consumidores podem estar localizados em qualquer lugar e usarem tecnologias de desenvolvimento diversas.

2.1.1.3 Serviços são autônomos

Para que um serviço seja autônomo, é necessário que a lógica encapsulada por ele possua limites bem definidos. Isso elimina a dependência com outros serviços, o que poderia inibir o seu desenvolvimento e evolução. A autonomia do serviço é a consideração primária quando se está decidindo como a lógica de uma aplicação deve ser dividida entre os serviços e quais operações devem ser agrupadas dentro de um serviço.

A autonomia de um serviço não necessariamente lhe garante a exclusividade sobre a lógica encapsulada. Garante somente que em um determinado tempo de execução, o serviço tem controle total sobre a lógica que ele representa, Krafzig (2004). Devido a isso, faz-se distinção entre dois tipos de autonomia:

- **Autonomia a nível de serviço:** os limites entre dois serviços são bem definidos, mas ambos compartilham os mesmos recursos. Por exemplo, um

serviço que encapsula a lógica de um sistema legado, a qual também é usada, independentemente do serviço, por algum cliente antigo.

- **Autonomia pura:** a lógica básica está sob controle somente do serviço. Isso tipicamente ocorre quando se está construindo uma SOA desde o início.

2.1.1.4 Serviços são compostos

Um serviço pode representar uma variedade de lógica fornecida por diferentes fontes, incluindo outros serviços. A principal razão para implementar este princípio é garantir que os serviços possam participar como membros ativos em outras composições de serviços se for preciso. A composição de serviços é uma forma de reutilização. Portanto, as operações desses serviços devem ser bem projetadas para maximizar a ocorrência deste princípio.

A composição de serviços é uma das fortes características de uma SOA, podendo ser feita em diferentes níveis. Estimulando o desenvolvimento e a evolução de serviços compostos, a SOA suporta a automatização de processos de negócios flexíveis e altamente adaptativos. Como mencionado anteriormente, serviços existem como unidades lógicas independentes. Um processo de negócio pode então ser dividido em uma série de serviços, cada um responsável pela execução de parte do processo, Krafzig (2004).

2.1.1.5 Serviços não possuem estados

Serviços deveriam minimizar a quantidade de informações de estado que eles gerenciam. Informação de estado é um dado específico a uma atividade corrente. Enquanto um serviço está processando uma mensagem, por exemplo, ele possui um estado temporário. Se um serviço é responsável por reter o estado por longos períodos de tempo, sua habilidade de manter-se disponível para outros requisitantes será impedida.

Serviços sem estado são condições importantes para promover reusabilidade e escabilidade dentro de uma Arquitetura Orientada a Serviços.

2.1.1.6 Serviços podem ser encontrados facilmente

Para prevenir a criação acidental de um serviço redundante ou serviços que implementam uma lógica já existente, a existência de um registro central de serviços é encorajada.

2.2 Web Services

De acordo com o Grupo de Trabalho do *World Wide Web Consortium* (W3C) responsável pela arquitetura dos *Web Services*, surgiu a seguinte definição sobre essa tecnologia:

> Um *Web Service* é uma aplicação de software identificada através de uma Identificação Uniforme de Recursos (URI) e contém interfaces e ligações capazes de serem definidas, descritas e descobertas através do uso de uma Linguagem de Marcação eXtendida (XML). Um *Web Service* suporta interações diretas com outros agentes de softwares através da troca de mensagens baseadas em XML via protocolos de comunicações da Internet, Web (2004)

Web Services são baseados em padrões abertos, sendo que sua estrutura básica combina o poder de duas tecnologias largamente empregadas em todo o mundo: XML, a linguagem universal de descrição de dados, e HTTP, o protocolo da camada de aplicação usado para a transferência de dados na *World Wide Web*.

A vasta aceitação do *Web Service* resultou na criação de novas tecnologias que se tornaram padrões de fato, dentre elas as mais importantes são a Linguaguem de Descrição de *Web Services* (WSDL), que descreve a interface e o protocolo de comunicação, e o *Simple Object Access Protocol* (SOAP), um protocolo baseado em XML para Chamada de Procedimento Remoto (RPC). Também, definiu-se um sistema de registro chamado *Universal Description, Discovery, and Integration* (UDDI), responsável por armazenar documentos WSDL, permitindo que serviços sejam publicados e encontrados, Erl (2005).

2.2.1 A linguagem WSDL

Uma Linguaguem de Descrição de *Web Services* (WSDL) descreve o ponto de contato de um serviço, fornecendo uma definição formal de sua interface, a qual permite que requisitantes querendo se comunicar com o serviço conheçam exatamente como estruturar as mensagens de requisição, e estabelecendo sua localização (endereço). Um documento no formato WSDL pode ser separado em duas partes, uma contendo a descrição abstrata e a outra a descrição concreta, Erl (2005).

2.2.1.1 Descrição abstrata

Uma descrição abstrata estabelece as características da interface de um *Web Service* sem qualquer referência à tecnologia usada para hospedá-lo ou para possibilitar a transferência de suas mensagens. Através da separação dessas informações, a integridade da descrição do serviço pode ser preservada, não importando as mudanças que podem ocorrer com a tecnologia utilizada. A seguir, resume-se as três partes principais que formam a descrição abstrata: *portType*, *operation* e *message*.

A seção chamada *portType* (interface) fornece uma visão geral da interface do serviço através da classificação das mensagens que ele pode processar em grupos de funções chamadas *operations* (operações). Cada *operation* representa uma ação específica realizada pelo serviço. Muito parecida com métodos de objetos, as operações também possuem parâmetros de entrada e saída. Como *Web Services* usam exclusivamente comunicação baseada na troca de mensagens, parâmetros são representados pelo campo *message*. Portanto, uma operação consiste em um grupo de mensagens de entrada e saída.

2.2.1.2 Descrição concreta

Para que um *Web Service* possa executar qualquer lógica, é necessário que sua definição de interface abstrata seja conectada a algo real, tecnologicamente implementado. Como a execução dessa lógica sempre envolve comunicação, a interface abstrata do *Web Service* precisa ser conectada a um protocolo de transporte. Essa conexão é definida na descrição concreta do arquivo WSDL, a qual consiste das

seguintes partes relacionadas: *binding*, *port* e *service*.

Uma descrição *bindind* (ligação) do WSDL descreve os requisitos para que uma conexão física seja estabelecida com o serviço. Em outras palavras, a ligação representa uma possível tecnologia de transporte que o serviço pode usar para se comunicar. SOAP é a forma mais comum de ligação, embora outros protocolos também sejam suportados. Uma ligação pode ser aplicada a toda interface (*portType*) ou apenas para operações específicas.

Relacionado com a ligação está a marcação *port* (porta), a qual representa o endereço físico pelo qual um serviço pode ser acessado usando um protocolo específico. Esta parte dos dados da implementação física existe separadamente a fim de permitir que informações sobre a localização sejam mantidas independentemente de outros aspectos da descrição concreta. Dentro do WSDL, o campo *service* é usado para agrupar diversos pontos de contato do serviço, representados pelas portas, que associam um endereço a uma ligação.

2.2.2 O padrão UDDI

Como descrito anteriormente, para um serviço utilizar outro serviço, ele deve ter acesso à descrição de tal serviço procurado. Com a quantidade de serviços aumentando dentro e fora de uma organização, mecanismos para publicar e descobrir descrições de serviços tornam-se necessários. Por exemplo, um sistema de registro torna-se uma opção atrativa para manter o controle de diversos serviços. Esses repositórios permitem que pessoas (ou mesmo serviços requisitantes) localizem a última versão de uma descrição de serviço conhecida ou mesmo novos *Web Services* que atendam determinados requisitos.

Universal Description, Discovery, and Integration (UDDI) especifica um padrão relativamente aceito para estruturar registros que tomam conta de descritores de serviços. O registro é baseado em XML e é independente de plataforma, podendo ser usado para buscas manuais ou via uma API padronizada, UDDI (2006).

Registros públicos aceitam o cadastramento de qualquer organização, não importanto se ela tem serviços a oferecer. Uma vez autenticadas, organizações atuando

como entidades provedoras de serviços podem registrar seus serviços.

Registros privados podem ser implementados dentro das fronteiras de uma organização para prover um repositório central para a descrição de todos os serviços que uma organização desenvolve, compra ou vende.

2.2.3 O protocolo SOAP

Como toda a comunicação entre serviços é baseada na troca de mensagens, a especificação do Protocolo Simples de Acesso à Objetos (SOAP) tem como principal objetivo definir um formato de mensagem padronizado, o qual consiste em um documento XML capaz de armazenar tanto dados de um documento como os de uma Chamada de Procedimento Remoto (RPC). A estrutura desse formato é muito simples, mas a habilidade de estendê-lo e personalizá-lo tem possibilitado a criação de novas especificações, relacionadas a *Web Services*, Erl 2004.

Uma característica importante do SOAP é que ele usa o protocolo HTTP para o transporte do seu conteúdo, permitindo que os documentos XML passem através de *firewalls* sem problemas, pois, por motivos de segurança, os administradores de sistema geralmente bloqueiam todas as portas de comunicação dos seus servidores, exceto a porta 80, usada pelo HTTP.

Toda mensagem SOAP é empacotada dentro de um *container* conhecido como envelope. O envelope é responsável por conter todas as partes da mensagem. Cada mensagem pode conter um cabeçalho, área dedicada a armazenar meta informações. Embora seja opcional, o cabeçalho é usado na implementação de várias extensões. O atual conteúdo da mensagem é armazenado no corpo da mensagem, o qual consiste tipicamente de dados formatados em XML, Erl (2005). A listagem 2.1 mostra um exemplo de requisição SOAP, realizada por um serviço implementado no caso de estudo.

Listagem 2.1: Requisição SOAP no estilo RPC

```
1 <?xml version='1.0' encoding='UTF-8'?>
2   <soap:Envelope xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/'
        xmlns:n='x' xmlns:xsd='http://www.w3.org/2001/XMLSchema'
       xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'>
```

```
3     <soap:Body soap:encodingStyle='http://schemas.xmlsoap.org/soap/
          encoding/'>
4       <n:buscaAcoesDelegadas>
5         <arg0 xsi:type='xsd:string'>Cesar</arg0>
6         <arg1 xsi:type='xsd:string'>Listar Prontuarios</arg1>
7         <arg2 xsi:type='xsd:string'>Servicos</arg2>
8       </n:buscaAcoesDelegadas>
9     </soap:Body>
10  </soap:Envelope>
```

O protocolo SOAP também define meios para enviar arquivos anexos, facilitando a entrega de dados que não são facilmente representadas através de um documento XML. Anexos SOAP são comumente usados para transportar arquivos binários, como imagens.

Finalmente, as mensagens SOAP oferecem a habilidade de adicionar uma lógica para tratamento de exceções, através do uso de uma seção opcional chamada *fault*, que reside dentro do corpo da mensagem. O uso típico dessa seção é para armazenar uma simples mensagem para informar o serviço requisitante que alguma exceção ocorreu.

2.3 Modelo Utilizado pelo SIE

O Sistema de Informações para o Ensino (SIE) implementado pelo CPD/UFSM utiliza um Sistema de Gerenciamento e Controle de Acesso (SGCA) para suas aplicações. O SGCA é uma ferramenta utilizada para controlar o acesso dos usuários às aplicações do sistema, armazenando senhas e garantindo que cada usuário tenha acesso somente às aplicações e registros que lhe forem repassados. Embora existam poucas aplicações já desenvolvidas para o Hospital Universitário de Santa Maria (HUSM), existe um interesse do CPD/UFSM nessa área. A principal motivação é que o SIE não está preparado para tratar dados clínicos, sobre os quais a legislação exige confidencialidade e privacidade, pois o SGCA não considera informações importantes para o controle de acesso a esses dados, como o médico assistente, local e hora de acesso e delegações de acesso. Como um dos objetivos deste trabalho é agregar funcionalidades aos módulos do SIE que atendem ao HUSM, utilizar-se-á

os conceitos básicos do SIE para demonstrar como ele está estruturado e, posteriormente, estabelecer-se-á a solução proposta pelo modelo que se chama SIE-Saúde, Soares (2006), com o SGCA atuando em conjunto com a implementação do modelo CIBAC .

No SIE/SGCA, o acesso se dá através do conteito de Usuário, o qual possui uma senha que, por definição, é pessoal e intransferível. Um Usuário do sistema está vinculado a um registro do Cadastro Único de Pessoas, sendo que uma mesma pessoa pode possuir vários Usuários (perfis) cadastrados no SGCA.

As Aplicações são os programas aos quais os Usuários terão acesso. Cada módulo (inclusive o SGCA) é composto por um conjunto de Aplicações, programas executáveis, que são cadastradas no SGCA. São exemplos de aplicações: "Serviço de Informações Cadastrais"; "Serviço de Prescrição Médica"; "Serviço de Agendamento e Internação"; "Serviço de Pesquisa Clínica", entre outros, Soares (2006).

O SGCA também utiliza o conceito de Grupo, que são conjuntos de Aplicações, pertinentes a um mesmo módulo, gerenciadas por um ou mais Usuários. Uma vez definidas as Aplicações que compõem um Grupo, um Usuário, vinculado ao Grupo em questão, tem acesso a todas essas Aplicações.

As restrições no SGCA são modeladas como Restrições de Funcionalidade (ou Ações) e Restrições de Dados. As Restrições de Funcionalidade definem o quê os Usuários podem fazer quando estão executando uma Aplicação. São exemplos de Restrições de Funcionalidade: "Inserir", "Alterar", "Excluir", "Delegar", etc. O universo de Restrições de Funcionalidade válidas pode ser diferente para diferentes Aplicações e está restrito ao conjunto de Restrições previamente estabelecido. As Restrições de Dados definem quais registros de quais tabelas os Usuários podem acessar quando estão executando uma determinada Aplicação. São exemplos de Dados no SIE-Saúde: "Dados de Identificação do Paciente", "Dados Demográficos", "Prescrição", "Agendamento", "Internação", etc. Por utilizar um controle discricionário, o SGCA não suporta delegação de atribuições, que são vislumbradas no âmbito do SIE-Saúde, e nem regras dinâmicas, que podem ser manipuladas contextualmente.

A utilização de *Web Services* visa a implementação de um modelo de controle

de acesso baseado em informações contextuais, denominado CIBAC (*Contextual Informations-Based Access Control*), Soares(2006), capaz de ser conectado e usado pelo SIE atual através de alterações de baixo custo efetuadas em seu código-fonte, em especial no SGCA.

Veja na figura 2.2 como os módulos do SIE atual estão integrados. Ao centro, visualiza-se o Sistema de Gerenciamento e Controle de Acesso (SGCA) circundado pelo Sistema de Tramitações, que fará a integração de todos os módulos. Para o SIE-Saúde, a principal modificação é a adição de um módulo responsável pela parte da Saúde e a inclusão do CIBAC junto ao SGCA (figura 2.3).

Figura 2.2: Módulos SIE

2.4 Modelo CIBAC

O modelo CIBAC, ou *Contextual Information-Based Access Control*, foi proposto por Soares (2006) e tem como característica principal o uso de informações contextuais para prover o controle de acesso a dados médicos, em especial aqueles gerenciados pelo SIE-Saúde.

Figura 2.3: Módulos SIE-Saúde

Por contexto, entende-se qualquer informação relevante que possa ser utilizada para caracterizar a situação de uma entidade. Uma entidade pode ser uma pessoa, um lugar, ou um objeto, relevantes para a interação entre o usuário e a aplicação. No CIBAC, estas informações são modeladas de acordo com a política adotada, e por esse motivo são definidas como propriedade, pois fornece os dados necessários à montagem da informação contextual. Cada propriedade distinta pode, por sua vez, ser agrupada diretamente nos contextos a que se referem; isto é, um contexto pode ser definido como um conjunto de informações sobre determinadas entidades, como um usuário ou mesmo um objeto, Soares 2006.

A combinação de determinadas informações sobre uma dada entidade, por exemplo, um dado usuário, quando agrupadas numa classe de informações caracteriza-se como um tipo de contexto. Na área da saúde, podem-se determinar tipos de contextos de acordo com a política de acesso utilizada. Por exemplo, um médico plantonista na emergência deve ser autorizado a acessar dados sobre pacientes em atendimento na emergência, mas não sobre pacientes internados em outras unidades do hospital, a menos que ele seja também o médico de um determinado paciente.

Outro ponto interessante abordado pelo CIBAC é que, em um ambiente hospitalar, pode ser comum a delegação de atribuições a médicos ou especialistas de outras áreas da saúde, a fim de prover assistência a um determinado paciente. Quando trata-se a questão da delegação como sendo uma informação contextual e gera-se regras para que esta delegação seja coerente, facilita-se a modelagem de dados temporais, locais e relacionados a uma dada assistência, permitindo a adequação do controle de acesso à legislação pertinente.

De acordo com Soares (2006), o CIBAC define informação como sendo uma Propriedade de Contexto na forma de um par (P, V), onde P é o nome da propriedade e V é o valor da propriedade P. Ainda estabelece o conceito de condição de contexto, que é uma fórmula booleana em forma de tupla, tal como (CT, P, $\oplus$, V), onde CT é o tipo de contexto, P é o nome da propriedade, $\oplus$ é um operador (por exemplo, $>$, $<$, $=$) e V é o valor que a propriedade assume conforme o operador $\oplus$. As condições de contexto são usadas na criação de políticas de controle de acesso. Essas políticas podem ser representadas de diversas formas. Pode-se usar, por exemplo, a linguagem XML ou, uma linguagem derivada dessa, o XACML.

2.4.1 Descrição de políticas de acesso

XACML (*eXtensible Access Control Markup Language*) é uma nova linguagem de marcação, baseada no XML, usada na descrição de políticas de controle de acesso. Esta linguagem define o formato de uma requisição que contém informações sobre o usuário, recurso, ação e ambiente, afim de achar políticas que se aplicam a ela e, então, seja tomado uma decisão, gerando uma resposta XACML, Griffin (2004).

Ao se criar um arquivo contento uma política define-se, através do elemento *Target* [1] , quais são os atributos da requisição que são analisados para verificar se esta política é aplicável. Se a política é aplicável, então, avalia-se as regras existentes nela. O XACML possui diversas funções para comparações de atributos e construções de expressões, que podem ser usadas tanto nas regras como no *Target*.

Como muitas políticas podem ser aplicáveis a uma determinada requisição,

[1] marcação, de uma política, que contêm informações referentes a uma requisição XACML

além de que cada política pode conter várias regras, existem algoritmos que devem ser usados para se gerar uma decisão final, que pode ser: permitido, negado, não aplicável ou indeterminado. Como exemplo, existe o algoritmo que permite o acesso desde que pelo menos uma regra, de alguma política, seja atendida.

Quando se usa o XACML, cria-se um Ponto de Decisão de Políticas, que é o responsável por receber a requisição XACML, procurar todas as políticas aplicáveis através do campo *Target*, avaliar as regras e usar um algoritmo para tomar a decisão final, retornando uma resposta XACML. Esta resposta pode conter informações adicionais, como o tempo de duração dessa permissão.

Capítulo 3

Implementação do CIBAC

Neste capítulo é apresentado a implementação do modelo de Controle de Acesso Baseado em Informações Contextuais (CIBAC) na forma de uma Arquitetura Orientada a Serviços. Implementação esta que propõe integrar-se no SIE-Saúde, em conjunto com o SGCA, para atender as necessidades de controle de acesso aos dados hospitalares.

A arquitetura do SIE-Saúde, com seu Sistema de Gerenciamento e Controle de Acesso (SGCA) comunicando-se com a implementação do CIBAC, é ilustrada através da figura 3.1. Nela é mostrada a interação do SGCA com o Serviço de Autorização, além da troca de mensagens entre este e os demais serviços. O CIBAC utiliza um banco de dados para armazenar todas suas informações, exceto as políticas de acesso. Para um melhor entendimento sobre as interações mostradas nesta figura, ver a seção 3.2.3.

3.1 Modelo do Banco de Dados

Utilizou-se um banco de dados relacional para armazenar as informações necessárias ao funcionamento da implementação do modelo CIBAC. Desta forma, o gerenciamento e o relacionamento entre os dados foi simplificado. Somente as políticas não se encontram armazenadas nesse local, pois elas estão em arquivos XACML independentes, localizados em um diretório configurado na inicialização do Serviço de Decisão. A figura 3.2 mostra o modelo do banco de dados criado. É importante

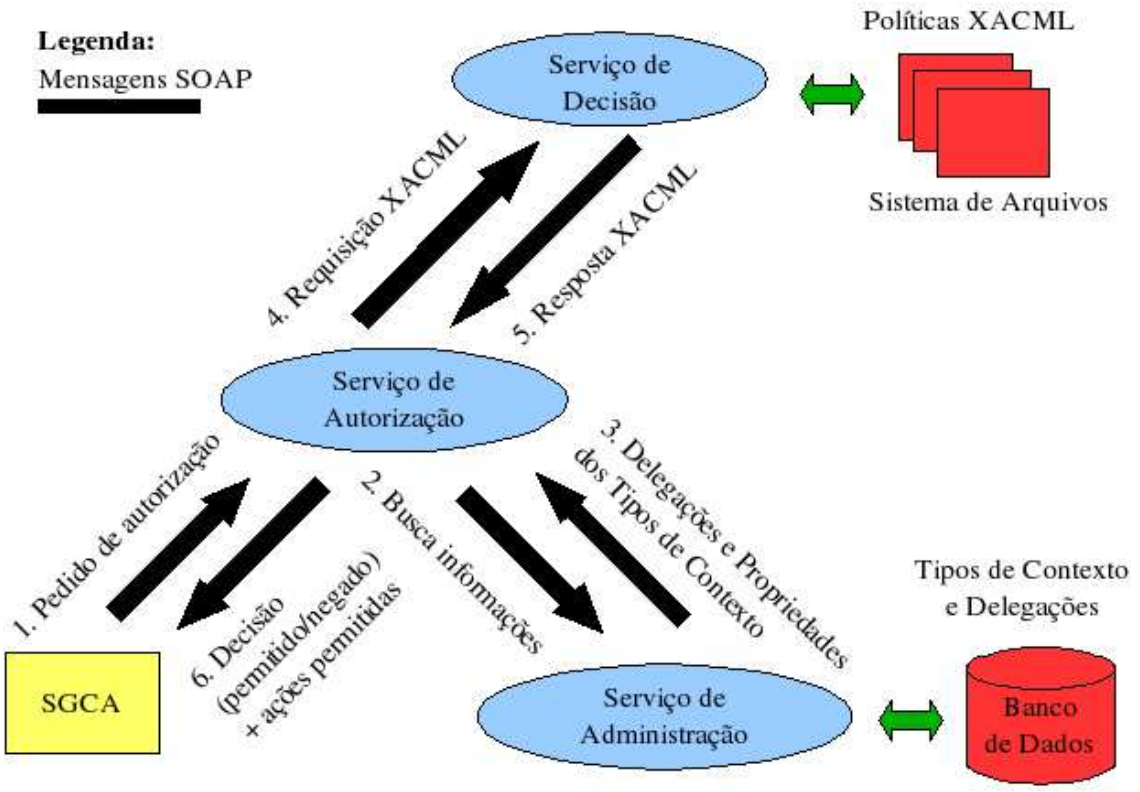

Figura 3.1: Comunicação do SGCA e os serviços para um pedido de autorização

notar que tal modelo, embora usado primeiramente para a área médica, é de uso geral e tem como objetivo armazenar uma varidade de informações, definidas após a instalação do sistema CIBAC.

Para implementar o modelo CIBAC foi definido dois tipos de contexto, sujeitos e objetos, cujas propriedades precisavam ser armazenadas em um meio persistente. Para cada um destes tipos foi criada uma tabela com o mesmo nome, sendo que ambas tem ligações com suas respectivas tabelas de propriedades. Como a criação de propriedades é dinâmica, foi criada a tabela `Tipos Propriedades` para armazenar informações sobre elas e seus tipos de contexto. As informações sobre o ambiente também são utilizadas, como hora do Serviço de Autenticação e endereço IP do computador do usuário. Por serem dinâmicas e obtidas facilmente através ds computadores utilizados, essas informações não precisam ser armazenadas em um banco de dados.

Como os sujeitos possuem funções (ou papéis) desempenhados dentro de um

hospital, foi criada uma tabela `Funções`, além de outra tabela para ligar a tabela `Sujeitos` com `Funções`, chamada `Sujeitos Funções`. Já para os objetos, estes possuem tipos que os classificam em diferentes grupos. Para esta tabela, que se relaciona com a tabela `Objetos`, deu-se o nome de `Tipos Objetos`.

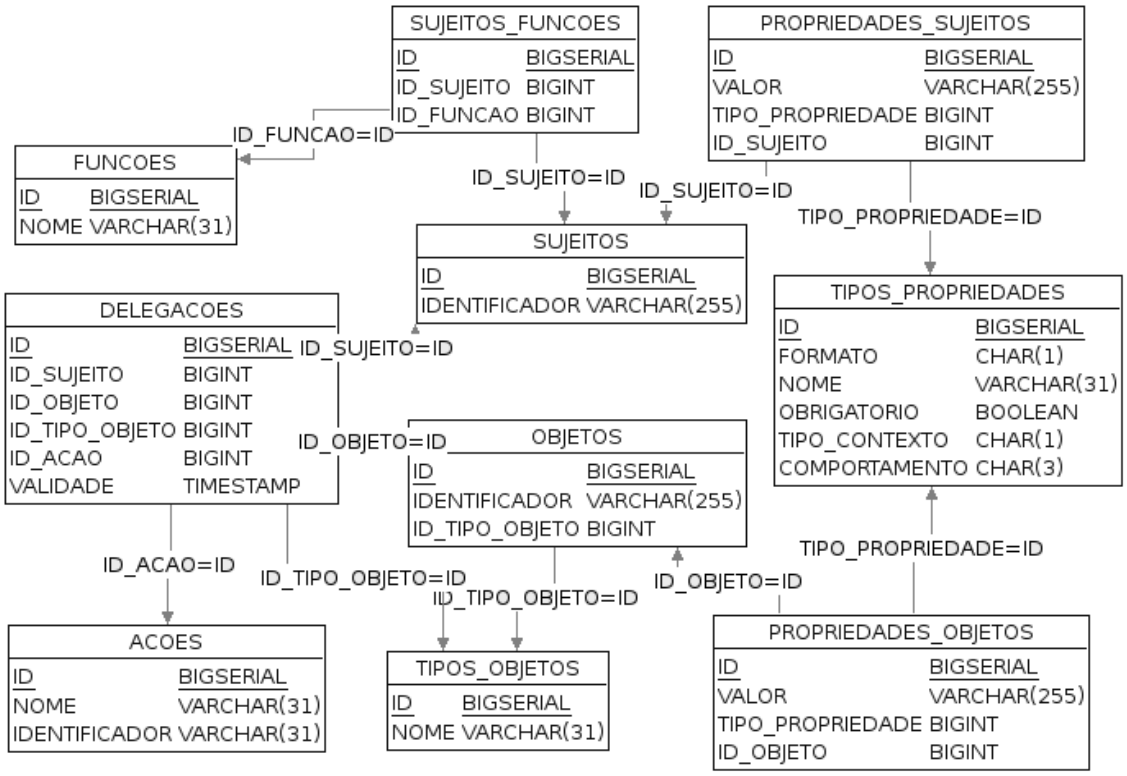

Figura 3.2: Modelo do Banco de Dados do CIBAC

Cada tipo de contexto possui um `ID` único, usado apenas pelo banco de dados e pela implementação do CIBAC. Como é necessário haver uma ligação com as informações mantidas pelo SIE atual, optou-se por agregar o campo `identificador`, capaz de diferenciar cada tipo de contexto. No caso do Sujeito, este `identificador` é único, podendo ter valores como o CPF ou RG de uma pessoa. Para facilitar a integração com o SIE, utiliza-se o valor da chave primária encontrado na tabela que armazena informações sobre os usuários desse sistema. Já para o Objeto, o `identificador` é único somente em conjunto com o nome existente na tabela `Tipos Objetos`. Optou-se por essa abordagem para permitir que, por exemplo, um prontuário tenha o mesmo `identificador` que uma aplicação, o que os diferenciaria

seria o seu tipo: prontuário, aplicação, etc.

De acordo com o modelo e suas definições, cada informação é agregada ao seu tipo de contexto na forma de propriedades. Estas propriedades, por sua vez, são armazenadas em tabelas contendo seus valores. Os metadados relativos a cada propriedade são armazenados na tabela `Tipos Propriedades`, a qual contém:

- **formato:** Contém a forma de representação do valor da propriedade em uma linguagem de programação, como `string, int, float`;
- **nome:** Possui o nome propriamente dito da propriedade, como: contador de acesso, médico assistente, entre outros;
- **obrigatório:** Determina se o preenchimento do valor da propriedade é obrigatório ou não;
- **tipo de contexto:** Relaciona a propriedade a um objeto ou a um sujeito; e
- **comportamento:** Determina o comportamento que o valor da propriedade deve assumir, de acordo com características de cada informação, como o incremento do valor na propriedade contador de acesso.

Como a utilização do modelo no SIE prevê o uso de delegações de atribuições para o controle de acesso a prontuários eletrônicos, esta característica é tratada com o auxílio da tabela `Delegações`. Esta tabela armazena uma referência sobre um sujeito, a ação delegada a ele, e sobre qual objeto (ou tipo de objeto) tal ação pode ser executada, além da validade da delegação criada. Veja a seção 3.3 para mais informações sobre o processo de delegação.

A tabela `Ações` foi criada para armazenar todas as ações que o CIBAC dá autorização para serem executadas. O campo `nome` é usado na criação de políticas, por exemplo, inserir, alterar, excluir, etc. Já o campo `identificador` serve para criar uma relação com as ações existentes no modelo SIE. Tal campo armazena a chave primária de uma tabela contendo as restrições de funcionalidade (ações existentes).

3.2 Serviços

Para prover as principais funcionalidades do CIBAC criou-se uma arquitetura orientada a serviços. Essa arquitetura é formada pela implementação de três *Web Services* com responsabilidades bem distintas, são eles: Serviço de Administração, Serviço de Decisão e Serviço de Autorização.

3.2.1 Serviço de Administração

O Serviço de Administração contempla o acesso ao banco de dados e provê diversas operações para manipulação destes dados, listadas a seguir. É importante dizer que para manipular as propriedades dos tipos de contexto, usa-se as operações específicas para cada tipo de contexto.

- gravação (inserção ou atualização) de uma delegação, objeto, sujeito, função, tipo de propriedade e tipo de objeto;
- busca de todas as delegações, objetos, sujeitos, funções, tipos de propriedades e tipos de objetos;
- busca de uma delegação, objeto, sujeito, função, tipo de propriedade e tipo de objeto usando o `id`, `identificador` ou `nome`, dependendo de qual for a entidade;
- verificação da existência de um objeto, sujeito, função, tipo de propriedade e tipo de objeto baseado em algum parâmetro, como o `identificar` ou `nome` de uma entidade; e
- exclusão de uma delegação, objeto, sujeito, função, tipo de propriedade e tipo de objeto usando o `id` de uma entidade;

No documento WSDL que descreve este serviço, existe o campo *types*, o qual define os tipos de dados usados nas mensagens de entrada e saída das operações do serviço. No trecho do documento mostrado na listagem 3.1, pode-se ver dois

tipos complexos, `Entidade` (linha 4) e `Funções` (linha 10). Onde `Funções` extende `Entidade` (linha 12), bem como ocorre numa linguagem orientada a objetos.

Listagem 3.1: Campo *types* de um documento WSDL

```
1  <wsdl:types>
2          <xsd:schema targetNamespace="http://www.webmethods.com/package/
               projeto.modelo/">
3
4                  <xsd:complexType name="Entidade" abstract="true">
5                          <xsd:sequence>
6                                  <xsd:element name="id" nillable="true"
                                       type="soapenc:long"/>
7                          </xsd:sequence>
8                  </xsd:complexType>
9
10                 <xsd:complexType name="Funcao">
11                         <xsd:complexContent>
12                                 <xsd:extension base="Entidade">
13                                         <xsd:all>
14                                                 <xsd:element name="nome
                                                       " nillable="true"
                                                       type="xsd:string"/>
15                                         </xsd:all>
16                                 </xsd:extension>
17                         </xsd:complexContent>
18                 </xsd:complexType>
19
20         </xsd:schema>
21 </wsdl:types>
```

A listagem 3.2 mostra um trecho da interface deste serviço, onde aparece a operação `buscaFuncao` (linha 10) com suas duas mensagens esperadas. A mensagem de entrada `buscaFuncaoIn` (linha 1) possui um `long` como argumento, e a mensagem de saída `buscaFuncaoOut` (linha 5) retorna uma `Função`.

Listagem 3.2: Trecho da interface do Serviço de Administração

```
1  <wsdl:message name="buscaFuncaoIn">
2          <wsdl:part name="arg0" type="soapenc:long"/>
3  </wsdl:message>
4
5  <wsdl:message name="buscaFuncaoOut">
6          <wsdl:part name="Result" type="Funcao"/>
7  </wsdl:message>
8
9  <wsdl:portType name="AdministrationService">
10         <wsdl:operation name="buscaFuncao" parameterOrder="arg0">
11                 <wsdl:input name="buscaFuncaoIn" message="buscaFuncaoIn
                       "/>
12                 <wsdl:output name="buscaFuncaoOut" message="
                       buscaFuncaoOut"/>
```

```
13          </wsdl:operation>
14  </wsdl:portType>
```

A ligação da interface do serviço, descrita anteriormente, com um protocolo que define um formato para as mensagens e um meio de transporte é mostrado na listagem 3.3, mais especificamente na linha 2. Neste caso, será usado o protocolo SOAP, cujo corpo das mensagens estão no estilo RPC. Estas mensagens serão transportadas usando o protocolo HTTP.

Listagem 3.3: Definição de Protocolos

```
1  <wsdl:binding name="AdministrationService" type="AdministrationService"
       >
2          <soap:binding style="rpc" transport="http://schemas.xmlsoap.org
               /soap/http"/>
3          <wsdl:operation name="buscaFuncao">
4                  <soap:operation soapAction="buscaFuncao" style="rpc"/>
5                  ...
6          </wsdl:operation>
7  </wsdl:binding>
```

Finalizando o documento WSDL, encotramos o endereço físico para acessar este serviço, como mostra a linha 3 da listagem 3.4. Este serviço poderia estar replicado em diversos computadores, acarretando em mais ocorrências do campo *port*, o qual especifica seu endereço físico.

Listagem 3.4: Endereço do Serviço de Administração

```
1  <wsdl:service name="$Proxy1">
2          <wsdl:port name="AdministrationService" binding="
               AdministrationService">
3                  <soap:address location="http://cesar:8002/glue/
                       administrationService"/>
4          </wsdl:port>
5  </wsdl:service>
```

Para interagir com este serviço foi criada uma aplicação web, chamada Administração do CIBAC, para que um usuário administrador pudesse popular o banco de dados com informações relevantes ao sistema. Esta manipulação dos dados também poderia ser feita através de uma interface de usuário do sistema SIE, bastando este se comunicar com o Serviço de Administração.

A página inicial da Administração do CIBAC é mostrada na figura 3.3, a qual contem *links* para formulários de cadastro e tabelas mostrando o conteúdo

armazenado no banco de dados.

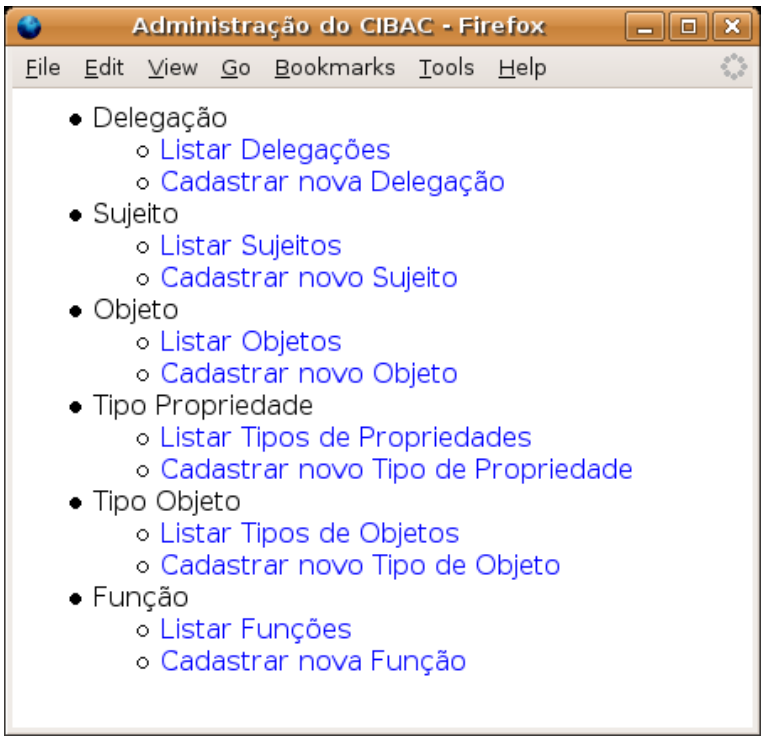

Figura 3.3: Administração do CIBAC - página inicial

3.2.2 Serviço de Decisão

O Serviço de Decisão é responsável por gerenciar os arquivos XACML contendo políticas de acesso, as quais são excênciais ao sistema CIBAC. Estas políticas são armazenadas em um diretório do servidor no qual este serviço executa. A localização de tal diretório é informado ao serviço via parâmetro de inicialização.

Outra funcionalidade desse serviço é receber mensagens SOAP contendo uma requisição XACML e confrontá-la com as diversas políticas previamente cadastradas, seguindo a especificação XACML. Após isso, é gerado uma resposta XACML que é retornada ao requisitante. Para essa funcionalidade, existe uma operação, descrita em seu WSDL, chamada `decisão`.

Para tratar políticas XACML, assim como criar requisições e usá-las, em conjunto com as políticas, para tomar decisões e gerar respostas XACML, foi utilizada a implementação *Sun's XACML*, Sun's (2004). *Sun's XACML* é um projeto que for-

nece um suporte completo para todas as principais características da especificação do XACML, é código aberto e escrito em Java.

A listagem 3.5 mostra o campo *Target*, responsável por determinar se uma política se aplica a uma requisição XACML. Este é um trecho de uma política XACML usada nos testes do CIBAC. Pode-se ver através do conteúdo encontrado entre o campo *Target* que esta política se aplica aos sujeitos que tem função (linha 4) de médico ou enfermeiro (linha 5) e querem acessar uma aplicação (linha 11). Se esta política for aplicável, então o Serviço de Decisão passa a avaliar as regras dessa política.

Listagem 3.5: Campo *Target* de uma Política XACML

```
1  <Target>
2    <Subjects><Subject>
3      <SubjectMatch MatchId="...function:regexp-string-match">
4        <SubjectAttributeDesignator DataType=".../XMLSchema#string"
             AttributeId="_funcoes"/>
5        <AttributeValue DataType=".../XMLSchema#string">.*# Medico
             #.*|.*# Enfermeiro #.*</AttributeValue>
6      </SubjectMatch>
7    </Subject></Subjects>
8    <Resources><Resource>
9      <ResourceMatch MatchId="...function:string-equal">
10       <ResourceAttributeDesignator DataType=".../XMLSchema#string"
             AttributeId="...resource:resource-id"/>
11       <AttributeValue DataType=".../XMLSchema#string">Aplicacao</
             AttributeValue>
12     </ResourceMatch>
13   </Resource></Resources>
14 </Target>
```

As regras da política XACML usada como exemplo são mostradas na listagem 3.6. A primeira regra (linha 1), se satisfeita, permite a execução de ações definidas posteriormente. Esta regra usa uma função (linha 2) que lida com um intervalo de tempo, tal intervalo situa-se das 8h às 20h (linha 6 e 7). Caso a regra não seja verdadeira, a segunda regra (linha 11) nega a autorização.

Listagem 3.6: Campos *Rule* de uma Política XACML

```
1  <Rule RuleId="Access" Effect="Permit">
2    <Condition FunctionId="...function#time-in-range">
3      <Apply FunctionId="...function:time-one-and-only">
4        <EnvironmentAttributeDesignator DataType=".../XMLSchema#time"
             AttributeId="...environment:current-time"/>
5      </Apply>
```

```
6         <AttributeValue DataType=".../XMLSchema#time">08:00:00</
              AttributeValue>
7         <AttributeValue DataType=".../XMLSchema#time">20:00:00</
              AttributeValue>
8     </Condition>
9   </Rule>
10
11  <Rule RuleId="Access" Effect="Deny"/>
```

Por fim, a listagem 3.7 mostra as ações (linha 3) que devem ser informadas ao SGCA caso a autorização seja permitida. Além da hora máxima (linha 4) que essa autorização é válida, a qual está de acordo com a primeira regra, mostrada anteriormente.

Listagem 3.7: Campos *Obligations* de uma Política XACML

```
1  <Obligations>
2    <Obligation ObligationId="Resposta" FulfillOn="Permit">
3      <AttributeAssignment AttributeId="acoes" DataType=".../XMLSchema#
           string">inserir, aterar, excluir</AttributeAssignment>
4      <AttributeAssignment AttributeId="hora" DataType=".../XMLSchema#
           time">20:00:00</AttributeAssignment>
5    </Obligation>
6  </Obligations>
```

3.2.3 Serviço de Autorização

O Serviço de Autorização desempenha a principal função dentro do CIBAC, ou seja, autorizar ou não um determinado sujeito a executar uma ação que envolve algum objeto (recurso). Para que esse serviço retorne uma resposta apropriada, ele se comunica com os dois serviços descritos anteriormente.

Para mostrar com mais detalhes todo o processo de autorização usou-se como exemplo o caso de uso onde uma aplicação do SIE-Saúde (figura 3.4) solicita ao SGCA permissão para habilitar os botões de ações de seu menu. O SGCA agora comunica-se com o CIBAC, através do Serviço de Autorização, para tomar uma decisão, ao invés de consultar sua base de dados. A integração entre o SGCA e o CIBAC para prover essa funcionalidade foi de fácil implementação, em virtude da grande aceitação e suporte à tecnologia *Web Services* pela maioria das plataformas de desenvolvimento. A seguir é explicado os detalhes das interações mostradas na figura 3.1.

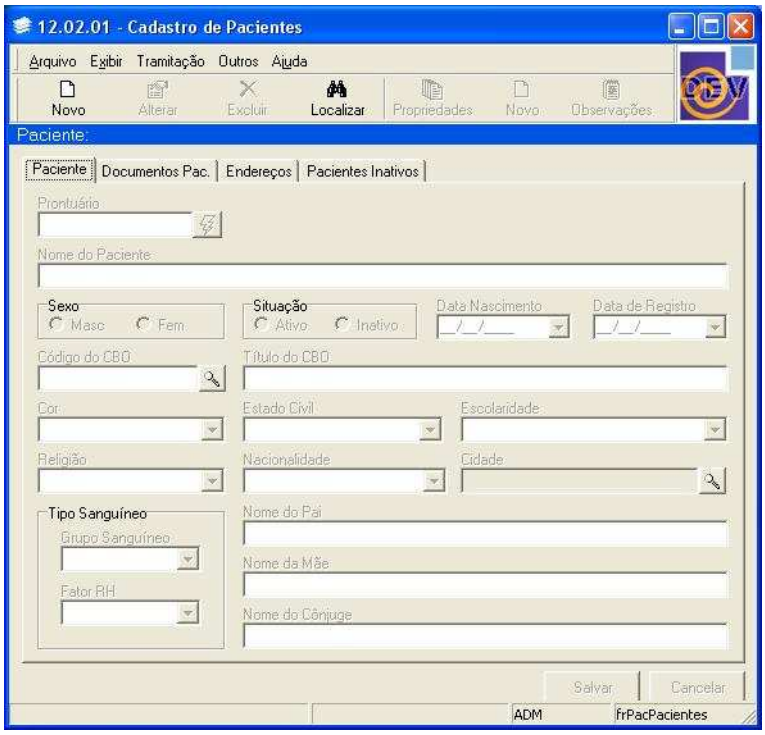

Figura 3.4: Aplicação do SIE-Saúde usada como teste de integração com o CIBAC

3.2.3.1 Interação entre o SGCA e o Serviço de Autorização

Primeiramente a aplicação do SIE-Saúde se comunica com o SGCA para obter os identificadores dos botões que devem ser habilitados para o usuário corrente. O SGCA então, de posse do identificador do usuário e da aplicação que fez o pedido, se comunica com o Serviço de Autorização através de uma mensagem SOAP. Nessa mensagem o SGCA coloca os tais identificadores, informa que o objeto em questão é uma aplicação e envia também o endereço IP do computador que abriu a aplicação.

Após o Serviço de Autorização efetuar seu trabalho, ele envia uma mensagem SOAP como resposta para o SGCA. Essa mensagem contém uma `string` informando a decisão resultante, que pode ser permitida, negada, não aplicável ou indeterminada. Um decisão não aplicável significa que não foi possível encontrar uma política que se aplica aos dados enviados pelo SGCA, talvez porque esses dados não estejam presentes no banco de dados do CIBAC. Uma decisão indeterminada é retornada quando ocorre alguma exceção na execução do CIBAC.

Caso a decisão, na resposta do Serviço de Autorização, seja permitida, é enviado também os identificadores de todas as ações que o usuário autenticado tem permissão de efetuar na aplicação aberta. Junto com os identificadores é enviado o tempo, em milissegundos, da duração dessa autorização. O SGCA poderá então utilizar esse tempo para garantir as regras temporais de controle de acesso.

3.2.3.2 Interação entre os Serviços

Logo após receber a mensagem contendo o pedido de autorização, feita pelo SGCA, o Serviço de Autorização se comunica com o Serviço de Administração para obter as propriedades referentes ao sujeito e objeto, além das delegações válidas. Para obter essas informações, o Serviço de Administração executa consultas no banco de dados. Se houver alguma delegação é feita uma união entre as ações delegadas e as ações permitidas pelo resultado da avaliação das políticas. Se as políticas negarem a autorização, então somente as ações delegadas serão informadas ao SGCA.

De posse das informações sobre os tipos de contexto, o Serviço de Autorização cria uma requisição XACML com esses dados e envia uma mensagem SOAP para o Serviço de Decisão. Este, por sua vez, verifica todas as políticas que foram carregadas do sistema de arquivos para sua memória, em busca de alguma que seja aplicável à requisição. A resposta do Serviço de Decisão também está no formato XACML.

Ao receber a resposta XACML, o Serviço de Autorização verifica a decisão recebida. Caso houver uma política aplicável, cujo resultado seja permitido, o serviço em questão faz uma união das ações permitidas pela política com as ações delegadas e envia, como resposta, uma mensagem SOAP para o SGCA.

3.3 Delegação

Um dos grandes benefícios do sistema CIBAC, implementado neste trabalho, é a possibilidade de um usuário delegar uma ação sobre um objeto para outro usuário. Dentro da área médica esta funcionalidade é de grande valia, pois permite que, por exemplo, um médico responsável por determinado prontuário possa delegar a ação

de consultá-lo a outro médico. A seguir é explicado como funciona o processo de delegação e sua influência no resultado de um pedido de autorização, fornecido pelo Serviço de Autorização.

Para que um usuário possa delegar uma ação, foi criado uma aplicação *Web* chamada de Administração do CIBAC, onde existe um formulário para preenchimento e cadastro de delegações (veja figura 3.5). A aplicação é ainda um protótipo, possuindo uma interface pouco amigável. Outra funcionalidade é informar ao usuário quais delegações já foram cadastradas, desde que possuam seu prazo de validade não vencido. Para consultar e popular o banco de dados, a Administração do CIBAC se comunica com o Serviço de Administração. Veja na figura 3.6 a comunicação entre a Administração do CIBAC e os Serviços de Administração e Autorização, para realizar o gerenciamento de delegações.

Figura 3.5: Administração do CIBAC - cadastro de delegações

Uma delegação é concedida a um usuário (sujeito) específico, permitindo-o efetuar determinada ação sobre um objeto ou um grupo de objetos, definido pelo tipo de objeto. Como exemplo, pode-se ter uma delegação onde o usuário 1 possa alterar o prontuário 120 e outra onde o mesmo usuário possa visualizar todos os prontuários. Nesse caso, prontuário é um tipo de objeto. Além dessas informações, para o cadastro de uma delegação é informado a data do prazo de validade. Quando o prazo expira, a delegação não é mais levada em conta pelo sistema CIBAC.

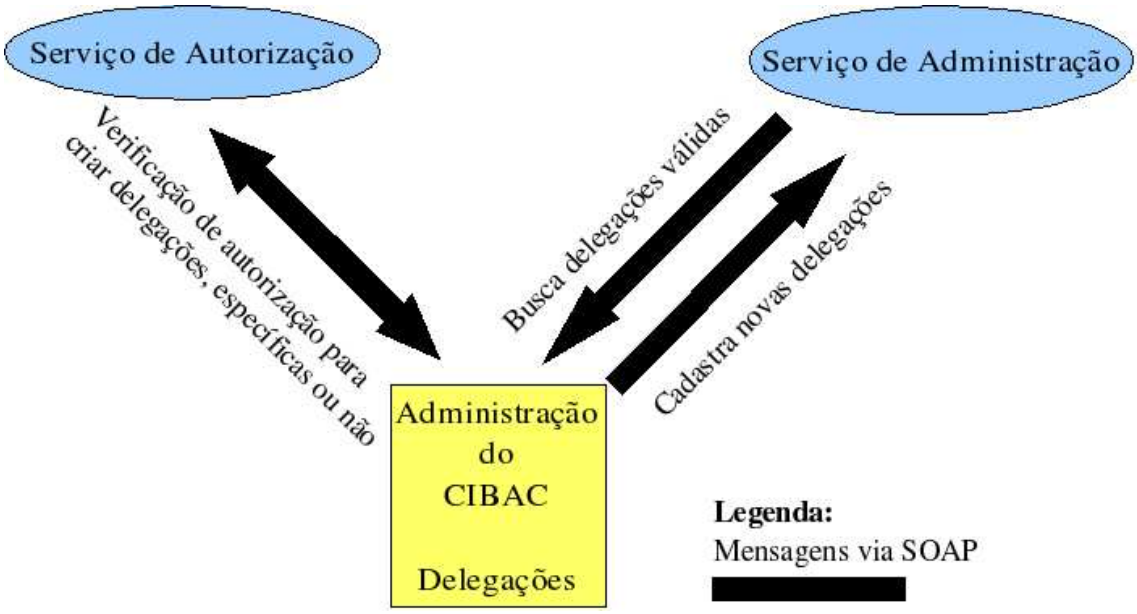

Figura 3.6: Comunicação entre a Administração do CIBAC e alguns serviços, para o gerenciamento de delegações

É interessante notar que uma delegação não possui informações sobre o usuário que a criou. Isto porque, para o CIBAC, não importa quem a cadastrou. Como é inconveniente que qualquer usuário possa cadastrá-las, deve-se criar políticas para definir quem pode delegar, quais ações possíveis, e para quais objetos ou tipos de objetos possam ser destinadas. Para verificar se um usuário tem autorização para cadastrar determinada delegação, a aplicação de Administração do CIBAC usa o Serviço de Autorização.

Após criada, uma delegação influi na resposta de um pedido de autorização da seguinte forma: logo após o SGCA fazer um pedido de autorização para o Serviço de Autorização, este serviço verifica os parâmetros recebidos e busca informações contextuais sobre o sujeito e objeto especificado, através da comunicação com o Serviço de Administração. Também são buscadas todas as ações que foram delegadas para o sujeito atual, desde que tais delegações sejam ainda válidas. As ações delegadas serão informadas ao SGCA independentemente do resultado obtido pela avaliação das políticas cabíveis.

3.4 Testes de Desempenho

Testes de desempenho são de grande importância para um sistema, principalmente se esse for usado continuamente e exija um tempo de resposta pequeno para requisições feitas por diversos usuários. O sistema desenvolvido neste trabalho será usado pelo SIE-Saúde dentro de um módulo muito utilizado, o SGCA. Em virtude disso, é necessário realizar testes para obter dados que possibilitem uma análise do desempenho do sistema CIBAC.

Para realizar os testes foram utilizados dois computadores. Um deles executou todos os serviços desenvolvidos neste trabalho, além do banco de dados PostgreSQL 8, usado pelo Serviço de Administração. Já o segundo, efetuou as requisições para o Serviço de Autorização, obtendo os resultados e os tempos de execução. O primeiro computador, que desempenhou o papel de servidor nos testes, foi o mais exigido, tendo este um processador Intel(R) Pentium(R) 4, com uma frequência de 1.80GHz. Sua memória era de aproximadamente 996 MBytes.

Antes de iniciar os testes, foram criadas dez políticas XACML, cada qual era aplicável a uma determinada função do sujeito e a um tipo de objeto. Como exemplo, existia uma política destinada aos Médicos acessando uma Aplicação e aos Médicos acessando um Prontuário. O banco de dados possuia o mínimo necessário para efetuar os testes. Foram cadastrados cinco funções e dois tipos de objetos, combinação usada para gerar as dez políticas.

Depois de estabelecidas as informações para os testes, foi criado um código para realizar várias requisições simultâneas, executadas por um número variado de *threads*. Para cada grupo de *threads* requisitantes, foi calculado o tempo médio para atender a todas elas. O gráfico da figura 3.7 mostra os tempos médios para um número de requisições que varia de 1 até 50.

Como pode-se ver através do gráfico, o tempo médio para atender até 10 requisições simultâneas ficou abaixo de 1 segundo. Embora o tempo médio seja acrescido pela adição de novas políticas e pelo tempo necessário para o SIE-Saúde realizar o seu trabalho, pode-se melhorar o desempenho utilizando um servidor com

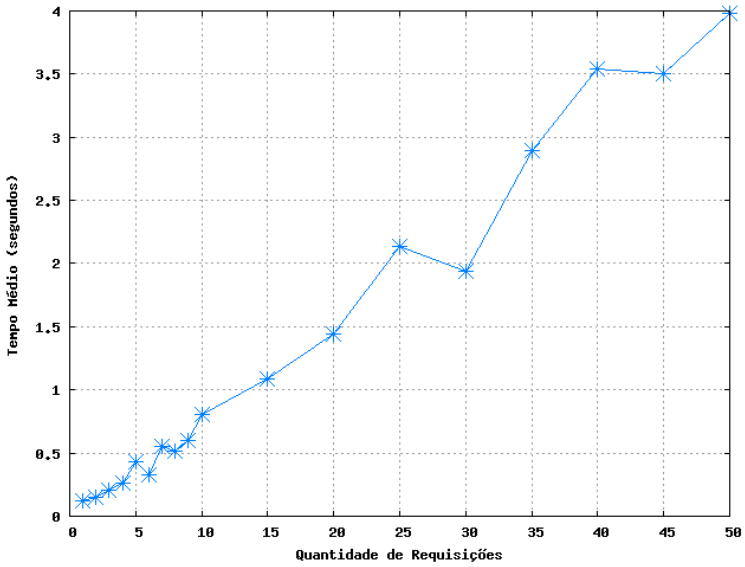

Figura 3.7: Gráfico do Tempo Médio de Resposta

maior poder de processamento, ou mesmo colocando alguns serviços executando em diferentes computadores.

Capítulo 4

Conclusão

Este trabalho apresentou uma implementação do modelo de controle de acesso baseado em informações contextuais, denominado CIBAC, o qual fornece autorizações de acesso relacionando informações de diferentes contextos. Seguindo os princípios de uma Arquitetura Orientada a Serviços, esta implementação foi feita com o uso da tecnologia *Web Services* para possibilitar que sistemas, construídos com diferentes tecnologias, pudessem facilmente fazer uso do CIBAC.

Os objetivos, citados na seção 1.2, foram todos alcançados. Foi possível a implementação do modelo proposto, assim como a realização de testes e a interligação com o SIE atual, formando o chamado SIE-Saúde. Apesar de que a utilização do CIBAC pelo SGCA tenha sido restrita à verificação da habilitação de botões em aplicações, um uso completo do CIBAC parece não apresentar grandes obstáculos. Isso graças ao modelo flexível e a utilização de *Web Services* para disponibilizá-lo.

A disponibilização de uma aplicação *Web* para administrar as informações contidas na base de dados facilita o processo de criação de entidades. Mas esse processo pode ser automatizado, desde que o SIE-Saúde interaja com o Serviço de Administração.

Em trabalhos futuros, a criação de políticas merece atenção especial. É necessário contruir uma interface que facilite o processo, bem como é preciso fazer um levantamento sobre que dados devem ser inseridos no CIBAC e como devem ser as políticas de acesso, de forma que melhor atendam os requisitos dos profissionais da área médica.

Referências Bibliográficas

ENDREI, M. et al. **Patterns:** service-oriented architecture and web services. IBM, 2004.

ERL, T. **Service-oriented architecture:** a field guide to integrating xml and web services. Pretience Hall, 2004.

ERL, T. **Service-oriented architecture:** concepts, technology, and design. Pretience Hall, 2005.

FERRAIOLO, D. F. et al. **Proposed NIST standard for role-based access control**. Information and System Security, 2001.

GRIFFIN, P. **Introduction to XACML**, 2004. Disponível em: <http://dev2dev.bea.com/pub/a/2004/02/xacml.html>, Acesso em: jul. 2006.

KRAFZIG, D.; BANKE, K.; SLAMA, D. **Enterprise SOA**: service-oriented architecture best practices. Pretience Hall, 2004.

MOTTA, G. H. M. B.; FURUIE, S. S. **Um modelo de autorização contextual para o controle de acesso baseado em papéis**. II Workshop em Segurança de Sistemas Computacionais (WSeg2002), Porto Alegre-RS, Brasil, SBC, 2002.

SOARES, G. A.; NUNES, R. C. **Controle de acesso baseado em credenciais hierárquicas dinâmicas - DHCBAC**. II Latin-American Symposium on Dependable Computing -Workshop on Theses and Dissertations. Proceedings of the LADC Wokshops, Salvador-BA, Brasil, 2005.

SOARES, G. A.; NUNES, R. C.; AMARAL, E. M. H. do. **Um modelo de controle de acesso baseado em contexto para autorizações a informações médicas**. XXXII Conferência Latinoamericana de Informática CLEI, Santiago, Chile, 2006.

SUN'S xacml, sun's xacml implementation, 2004. Disponível em: <http://sunxacml.sourceforge.net>. Acesso em: jun. 2006.

UDDI specifications. Disponível em: <http://www.oasisopen.org/committees/uddi-spec/doc/tcspecs.htm>. Acesso em: jul. 2006.

WEB services, web services architecture requirements, 2004. Disponível em: <http://www.w3.org/TR/wsa-reqs>. Acesso em: jun. 2006.

WIEHLER, G. **Web services and service oriented architectures**. Siemens, 2004.