Em memória dos meus queridos avós

Antônio e Alexandre.

Ao meu orientador Mauro, um grande amigo antes de tudo;
À professora Marina, por sua sabedoria e conselhos;
Às valiosas amizades conquistadas durante o período deste trabalho: Matheus, Val, Dani, Fernando, Thiago, Nassim e Calebe;
Aos meus amigos Arnaud e Denise, por todo o carinho dedicado a mim durante longos anos;
Aos meus amigos Cláudio e Dorta, amizades que resistiram à distância e ao tempo;
Aos meus amigos Moacyr e Gleyson, sempre presentes quando necessário;
Às minhas avós Olga e Lourdes, pelos ensinamentos, compreensão e amor;
Aos meus pais, pelo apoio contínuo e verdadeiro;
Aos meus irmãos Camila, Fernando e Lucas, amores incondicionais de minha vida;
A Marcia, pelo companheirismo, cumplicidade e amor;
E, principalmente, Àquele que tornou possível este sonho: muitíssimo obrigado Senhor Jesus.

# Resumo

Educação a Distância (EaD) é um tema de grande importância que vem sendo intensamente abordado devido à crescente demanda pela educação continuada e a modernização e disponibilização de novas tecnologias. Diversos ambientes de EaD surgiram com o aumento da popularidade da Internet, cada um oferecendo uma gama de recursos para, principalmente, maximizar o processo de ensino/aprendizado. Tal processo ocupa um lugar de destaque em várias pesquisas voltadas para o desenvolvimento desse tipo de ambiente devido à sua importância na teoria de EaD. Dessa forma, assuntos que não estão diretamente ligados a ele têm recebido pouca atenção por parte de projetistas e desenvolvedores. O objetivo deste trabalho é justamente atentar para um desses assuntos, apresentando uma arquitetura de segurança para ambientes de EaD que contempla os aspectos da autenticação e do controle de acesso de usuários. Em relação à autenticação, a arquitetura utiliza os mecanismos de senhas e questões randômicas codificadas através de chaves de criptografia. Já em relação ao controle de acesso, o esquema formulado para a arquitetura combina uma série de técnicas que permitem que o administrador de um curso de um ambiente de EaD tenha o máximo de flexibilidade e simplicidade no gerenciamento das autorizações atribuídas aos usuários pertencentes a esse mesmo curso. Tais técnicas englobam o uso de autorizações coletivas/individuais, autorizações positivas/negativas e autorizações explícitas/implícitas.

# Abstract

Distance Education (DE) is a very important theme that has been discussed intensively owing to the growing demand for continued education and to the modernization and availability of new technologies. Several DE environments have emerged as a result of the increasing popularity of the Internet, each offering a variety of resources, mainly to maximize the teaching/learning process. Due to its relevance to the DE theory, this process has been the focus of several researches aimed at developing this type of environment. As a result, designers and developers have devoted little attention to subjects not directly related to it. The purpose of this work, therefore, is precisely to broach on of these subjects, in the form of a safety architecture for DE environments that takes into account the aspects of authentication and control of user access. From the standpoint of authentication, the architecture employs the password and random question mechanisms coded using cryptographic keys. With regard to access control, the scheme formulated for the architecture combines a series of techniques that allow the manager of course in a DE environment the maximum flexibility and simplicity in managing the authorizations attributed to the users registered in the course. These techniques involve the use of collective/individual authorizations, positive/negative authorizations, and explicit/implicit authorizations.

# Sumário

# Lista de Figuras

# Lista de Tabelas

# Capítulo 1
# Introdução

## *1.1 Considerações Iniciais*

A educação e seus paradigmas vêm passando por um forte processo de modernização devido ao avanço tecnológico das últimas décadas. Alguns dos principais componentes relacionados ao ensino, como escolas, universidades, professores e alunos, vêm tendo sensíveis mudanças em relação ao desempenho de suas funções.

Considerando os diferentes processos educacionais existentes, a Educação a Distância (EaD) é, provavelmente, o que mais tem se modernizado. Inicialmente, a EaD consistia de cursos à distância que chegavam ao seu público através do uso dos correios. Nos dias de hoje, essa mesma distância é percorrida por conexões idealizadas através de modernos meios de comunicação, possibilitando uma interação muito mais eficiente. Além disso, com a tecnologia atual, muitas possibilidades de aplicação da EaD se fazem presentes com a utilização de recursos audiovisuais, entre outros.

Resumidamente, a EaD tem como proposta trazer inúmeros benefícios para o processo de ensino/aprendizagem, tais como atingir o maior número possível de pessoas, minimizar o problema da distância entre instituição educadora e aluno, proporcionar maior liberdade ao aluno na questão de horários, etc. Para viabilizar esses objetivos, vários recursos devem ser utilizados em harmonia, e que em conjunto formam um ambiente de EaD. Esses recursos compreendem os meios de comunicação para interconexão remota entre computadores, os softwares utilizados para dar suporte ao ensino/aprendizado e metodologias de ensino específicas.

Dentre esses softwares, há a necessidade de um que, para prover a segurança de um ambiente de EaD, seja responsável pelos aspectos da autenticação de usuários e do controle de acesso aos objetos protegidos através de autorizações. Segundo Linhalis [Linhalis, 2000], a autenticação é a propriedade de segurança que dita que as entidades envolvidas em um processo de comunicação devem ter meios de confirmarem suas identidades mutuamente, certificando-se de com quem estão se comunicando. Terminado o estágio de autenticação, durante todo o tempo restante da comunicação entre o usuário e o ambiente de EaD tem-se o

estágio de controle de acesso. Os mecanismos de controle de acesso garantem o alcance aos objetos protegidos apenas pelo conjunto de usuários autorizados.

Assim, os mecanismos de autenticação e controle de acesso devem estar inseridos em ambientes de EaD de forma a ditarem restrições para cada usuário em cada situação.

## *1.2 Motivação e Objetivo*

O grupo de Banco de Dados do Departamento de Computação da UFSCar vem desenvolvendo um projeto denominado AMMO (Authoring and Manipulation of Multimedia Objects). Esse projeto vem sendo utilizado para a pesquisa de vários aspectos computacionais ligados, principalmente, à manipulação de objetos multimídia. Nesse contexto, está sendo desenvolvido um ambiente denominado DE-MAW (Distance Education-Multimedia Application Web-builder) [Seno, 2000], no qual alguns resultados já foram alcançados e novos trabalhos de pesquisa encontram-se em fase de elaboração.

Entre os pontos que necessitam ser abordados e desenvolvidos referentes ao DE-MAW estão os que envolvem a autenticação de usuários e o controle de acesso sobre as ferramentas que compõem esse ambiente. Considerando esses pontos como motivação inicial, foram pesquisadas na literatura características de segurança estabelecidas sobre ambientes de EaD disponíveis no mercado.

A partir desse levantamento, o objetivo deste trabalho consistiu em criar uma arquitetura de segurança para ambientes de EAD que abrange as características de autenticação e de controle de acesso. Como resultado, foi desenvolvida uma arquitetura de segurança independente das particularidades desse tipo de ambiente. Essa característica permite que projetistas e desenvolvedores possam utilizá-la para a implementação da segurança em seus respectivos ambientes de EaD.

## *1.3 Organização da Monografia*

No capítulo 2 é mostrada uma revisão sobre os conceitos e mecanismos de autenticação e controle de acesso. O capítulo 3 realiza um levantamento relacionado aos conceitos básicos de EaD, incluindo uma avaliação dos aspectos de autenticação e controle de acesso de alguns dos ambientes de EAD disponíveis no mercado. No capítulo 4 é apresentada a arquitetura de segurança proposta neste trabalho de mestrado, tendo como enfoque as suas funcionalidades. No capítulo 5 são descritos os aspectos de implementação para que tais funcionalidades

possam ser efetivamente alcançadas. O capítulo 6 descreve um estudo de caso em que a arquitetura de segurança é aplicada no ambiente DE-MAW.

# Capítulo 2
# Segurança de Sistemas Computacionais

## *2.1 Considerações Iniciais*

São abordadas neste capítulo questões envolvendo segurança, desde os métodos de ataque usados contra sistemas computacionais até as técnicas de autenticação e de controle de acesso utilizadas para prevenir ações não autorizadas dentro desses sistemas.

A seção 2.2 descreve brevemente os métodos de ataque para o acesso às informações ou recursos protegidos de um sistema computacional. Na seção 2.3 são apresentadas de maneira sucinta as propriedades de segurança para o desenvolvimento de um sistema computacional consideravelmente seguro. As seções 2.4 e 2.5 mostram, respectivamente, a autenticação e o controle de acesso de maneira detalhada, abordando conceitos e métodos de implementação. Na seção 2.6 são citadas quais são os conceitos e mecanismos utilizados na implementação da arquitetura de segurança. Finalmente, a seção 2.7 aborda as considerações finais do capítulo.

## *2.2 Métodos de Ataque*

Um ataque é caracterizado através da intenção da destruição de informação ou de outros recursos, modificação ou deturpação da informação, roubo ou remoção de informação ou de outros recursos, revelação de informação sigilosa e interrupção de serviços.

De acordo com Coulouris [Coulouris et al., 1994], a utilização de um dos seguintes métodos é necessária para que um ataque seja concretizado:

- *Espreita*: é a obtenção de cópias de mensagens sem autorização, realizada obtendo-se as mensagens diretamente da rede ou examinando informações que estão armazenadas sem proteção adequada;
- *Personificação*: enviar ou receber mensagens utilizando a identidade de outra entidade;

- *Falsificação*: consiste em interceptar mensagens e alterar o seu conteúdo antes de enviá-las ao destino;

- *Vírus*: é um programa intruso com intenções maléficas que se instala no sistema a partir da sua execução, possuindo várias formas de ataque;

- *Infiltração*: ocorre quando um programa explora remotamente as facilidades de processos em execução;

- *Cavalo de Tróia*: consiste em um programa que aparentemente tem uma função útil, mas que na realidade tem por intenção lesar o sistema em que está instalado.

O cumprimento de certas propriedades de segurança evita em grande parte os ataques descritos acima. Essas propriedades são descritas na próxima seção.

## *2.3 Propriedades de Segurança*

Um sistema computacional é considerado seguro quando está de acordo com algumas propriedades de segurança. Tais propriedades definem normas que regulamentam o acesso às informações e recursos de forma segura.

De acordo com Chin [Chin, 1999], as propriedades que precisam ser consideradas na concepção de um sistema computacional seguro são:

- *Confidencialidade*: apenas as partes envolvidas na comunicação acessam o conteúdo dos dados que trafegam na rede, sendo que qualquer ação de monitoramento ilegal não deve ser capaz de usufruir desses dados.

- *Integridade*: os dados transmitidos na comunicação não devem ser alterados por acesso ilegal ou falha.

- *Autenticação*: é necessária a confirmação de identidade pelas partes integrantes de um processo de comunicação.

- *Controle de Acesso*: o acesso às informações e recursos é restrito às entidades autorizadas.

- *Não-Repudiação*: a autoria de uma ação não pode ser negada por seu executor.

Nas duas próximas seções é realizado um levantamento mais detalhado sobre duas das propriedades de segurança apresentadas acima: autenticação e controle de acesso.

## *2.4 Autenticação*

A necessidade de se identificar pessoas vem desde o início da história do homem e alcançou tais proporções na sociedade atual que ficou muito comum a preocupação com a verificação de identidade, em que possa ser confirmado se alguém é quem realmente diz ser.

Vários métodos criados no passado para a verificação de identidade ficaram mais aprimorados e confiáveis com o avanço tecnológico. Por exemplo, um dos primeiros métodos usados na identificação criminal baseado nas características físicas foi o sistema de medidas Bertillon, criado em 1870 [Everett, 1992]. Esse sistema usava o peso, tamanho da abertura dos braços, tamanho do tronco, tamanho do pé, tamanho dos dedos, tamanho do antebraço e largura da cabeça para o reconhecimento de um indivíduo. Atualmente, outras características consideradas mais confiáveis são utilizadas, entre elas voz, impressão digital, varredura de retina e aspectos faciais.

A autenticação pode ser definida como um processo no qual as entidades envolvidas em uma comunicação confirmam suas identidades [Linhalis, 2000], ou seja, é a utilização dos métodos de verificação de identidade em uma comunicação[1].

Os métodos de autenticação são baseados em quatro princípios [Everett, 1992]:

- Alguma coisa que o usuário conhece;
- Alguma coisa que o usuário tem;
- Alguma característica física do usuário;
- Algum padrão de comportamento do usuário;

Fiorese [Fiorese, 2000] classifica, respectivamente, esses quatro princípios em três categorias diferentes, sendo que a última engloba os dois últimos princípios. São elas: conhecimento, propriedade e biometria. Esta seção apresenta os principais e mais utilizados métodos de autenticação de cada uma.

[1] Como neste trabalho os métodos de verificação de identidade são analisados para serem aplicados na autenticação, os termos *método(s) de verificação de identidade* e *método(s) de autenticação* são utilizados como sinônimos.

### 2.4.1 Conhecimento

Na atualidade, essa é a categoria mais utilizada na implementação da autenticação e seus métodos baseiam-se principalmente na alocação de uma ou mais palavras secretas para um usuário. Tal fato pode ser verificado no cotidiano, no qual inúmeras senhas, *PIN's (Personal Identification Number)* e chaves de criptografia são utilizadas nos sistemas de software para o reconhecimento da identidade.

#### 2.4.1.1 Senhas (*Passwords*)

A base para o controle de acesso em ambientes computacionais reside no emprego de senhas, devido principalmente às facilidades e baixo custo de implementação que seu uso provê. Embora amplamente utilizadas, são as mais suscetíveis a fraudes através de meios como adivinhação da senha (*password guessing*), ataque de dicionário (*dictionary attack*), monitoramento do tráfego na rede (*sniffing*), cavalos-de-tróia e cópia de anotações [Fiorese and Tarouco, 1999].

Para se aumentar a dificuldade do sucesso de tais meios de ataque, a observação de alguns procedimentos simples descritos por Everett [Everett, 1992] na utilização de senhas tornam-se fundamentais. São eles:

- *Criação de uma senha*: senhas devem ser criadas pelo sistema de forma a dificultar a previsão de seus valores. Por exemplo, seis dígitos numéricos formam uma combinação relativamente segura. Se for permitida a escolha da senha pelo próprio usuário, deve ser estabelecido um tamanho mínimo de caracteres e o sistema deve checar a senha através de um dicionário de senhas previsíveis.

- *Armazenamento de uma senha*: um arquivo de senhas armazenado em um computador deve conter dados que foram transformados por uma função (criptografia). Os usuários devem ser instruídos a preservarem em segredo os valores de suas senhas.

- *Distribuição de uma senha*: as senhas devem ser distribuídas de forma segura para os usuários. Telefones nunca devem ser utilizados na distribuição de senhas.

- *Uso de uma senha*: usuários devem preservar a confidencialidade de suas senhas, não as compartilhando com outros usuários. O sistema deve restringir o número de

tentativas de acesso caso a senha digitada esteja errada. Por exemplo, após três tentativas o acesso é bloqueado e o log de autenticação deve ser investigado.

- *Mudança de uma senha*: um sistema deve forçar mudanças regulares de uma senha. A nova senha segue o mesmo padrão de criação citado acima.

- *Destruição de uma senha*: quando um indivíduo deixa de ser usuário de um sistema, sua senha deve ser destruída e seu valor não mais utilizado para novas senhas.

Deve-se mencionar que os procedimentos citados não estão associados a um método de autenticação específico e sim a todos os métodos de autenticação existentes. Sendo assim, ao se implementar um método de autenticação torna-se obrigatória a análise detalhada de cada um dos itens acima.

### *Senhas Descartáveis (One-time Passwords)*

Uma maneira de se superar a natureza estática das senhas é a utilização do método de senhas descartáveis. Nesse método uma senha é utilizada somente uma vez, sendo necessária em cada conexão uma nova senha. Isso evita que uma pessoa passe por um usuário de um sistema através de uma senha que tenha obtido ilegalmente. Várias implementações desse método estão disponíveis atualmente em hardware e software.

As senhas descartáveis são classificadas em duas categorias: sincronizadas no tempo e desafio/resposta [Fiorese, 2000]. As senhas descartáveis sincronizadas no tempo são calculadas baseando-se em um intervalo pré-determinado de tempo, por exemplo, trinta segundos. Nesse caso, uma nova senha é criada a cada trinta segundos a partir do momento em que o usuário entra em contato com o sistema. No método de senhas descartáveis baseadas em desafio/resposta o sistema envia um desafio para o usuário, geralmente um número ou uma string, e espera uma resposta compatível do usuário. Em ambos os métodos o usuário deve conhecer o algoritmo que gera as senhas para que os valores das respostas enviadas estejam corretos.

### *Perguntas Randômicas (Random Queries)*

Para se implementar esse método, primeiramente é necessário que o usuário preencha um cadastro que possui os mais variados tipos de questões. Por exemplo, qual é o número do seu CPF, qual é a sua data de nascimento, qual é a sua comida preferida, qual é o nome da sua mãe, etc. Após isso, toda vez que o usuário em questão tentar conectar-se com o sistema, uma dessas perguntas pessoais é feita e a autenticação só é realizada se a resposta for correta.

Esse método pode ser combinado com o uso de senhas normais ou senhas descartáveis. Muitas vezes ele é visto como um complemento na autenticação e não como o método principal. Empresas de cartão de crédito geralmente utilizam este método na autenticação de seus usuários em ligações telefônicas. A vantagem é que ele pode ser totalmente implementado em software, não necessitando de hardware adicional.

### 2.4.1.2 PIN's

Um PIN geralmente é formado de quatro a oito dígitos que são escolhidos pelo sistema e atribuídos ao usuário. De acordo com Jackson [Jackson and Hruska, 1992], um PIN é uma seqüência de dígitos usada para verificar a identidade de um usuário, comportando-se como um tipo de senha. Já para Martins [Martins, 2001], ele é definido por conjunto de caracteres numéricos utilizado como chave secreta para identificação do usuário em transações em automação bancária e comercial.

A principal diferença entre um PIN e uma senha é que muitas vezes o primeiro está associado a um dispositivo físico como, por exemplo, um cartão magnético. Isso pode ser observado em um sistema bancário, em que um cartão e uma senha são utilizados na autenticação. O PIN que está armazenado no cartão indica o usuário envolvido na operação para o sistema. Após isso, a senha é requerida para a confirmação de identidade, já que essa é, teoricamente, mantida em segredo pelo usuário legítimo.

### 2.4.1.3 Chaves de Criptografia

Antes de se tratar da autenticação por chaves de criptografia, é necessária uma visão geral dos conceitos envolvidos nesse mecanismo. Na realidade, a criptografia surgiu para manter informações confidenciais em sigilo através de métodos de criptografia, mas notou-se que ela, após a evolução desses métodos, também poderia ser utilizada de uma forma robusta na autenticação de usuários.

#### *Criptografia*

Essa técnica consiste em modificar na origem a mensagem a ser transmitida, gerando uma mensagem criptografada. O processo de codificação se dá através de um método de criptografia. Após o envio e a chegada da mensagem criptografada ao seu destino, ocorre o processo inverso, ou seja, através do mesmo método de criptografia a mensagem é decodificada.

O principal problema dessa técnica é que se o método de criptografia for descoberto por um indivíduo não autorizado, faz-se necessário criar outro método. Para a resolução desse problema agregou-se o uso de chaves. Dessa forma, através do uso de uma chave, não é mais necessário manter o método de criptografia em sigilo, pois a mensagem criptografada varia de acordo com a chave. Assim, chaves diferentes produzem mensagens criptografadas diferentes com o mesmo método de criptografia. As chaves podem ser utilizadas de várias maneiras, sendo que duas delas são discutidas a seguir.

### *Criptografia com Chave Secreta (Chave Simétrica)*

A mensagem a ser enviada é codificada através de um método de criptografia e uma chave secreta. Para que a mensagem seja decodificada por esse método é necessária a utilização da mesma chave [Silberschatz et al., 1999]. Ambas as partes envolvidas na comunicação, emissor e receptor, devem obter a chave secreta através de um canal seguro. É imprescindível a existência de confiança entre as partes comunicantes em relação ao sigilo da chave usada.

O DES (Data Encryption Standard) é um dos métodos baseados em chave secreta mais utilizados [Ganley and Piper, 1992].

### *Criptografia com Chave Pública (Chave Assimétrica)*

Nessa técnica, para que haja comunicação, são criadas duas chaves pelo receptor, uma pública e a outra secreta. O receptor envia a chave pública para o emissor e mantém a chave secreta em sigilo. O emissor codifica a mensagem a ser enviada através da utilização da chave pública e do método de criptografia pelo qual as chaves foram geradas anteriormente. Após receber a mensagem criptografada, o receptor utiliza a chave secreta para decodificá-la. O método utiliza uma função para definir uma relação entre as duas chaves, dificultando imensamente o conhecimento da chave secreta através da chave pública.

Uma das principais vantagens desse método é que não há necessidade de confiança entre as partes comunicantes, já que apenas o receptor é que detém o conhecimento sobre a chave secreta. Além disso, não é necessário um canal seguro para o envio da chave pública.

O método mais importante de criptografia com chave pública é o RSA, nome dado pelas iniciais dos seus autores Rivest, Shamir e Adleman [Bidzos, 1992].

### *Assinatura Digital*

Segundo Linhalis [Linhalis, 2000], embora a criptografia garanta a confidencialidade de uma mensagem, ela não pode garantir a autenticidade do emissor. Para que esse problema fosse solucionado foi criado o mecanismo de assinatura digital [Sherwood, 1992], que utiliza a criptografia com chave secreta e chave pública. O fator que é considerado como uma 'assinatura' é a própria chave secreta, já que apenas seu possuidor tem acesso a ela. Nota-se que nessa técnica ambas as chaves envolvidas no processo de comunicação têm a habilidade de codificar e decodificar uma mensagem, diferentemente do que foi apresentado na subseção anterior.

Um documento digital assinado consiste na tripla <M, P, {M}C>, no qual M é a mensagem a ser enviada, P é a entidade emissora e {M}C é uma cópia da mensagem M criptografada com a chave secreta, sendo essa última a assinatura digital. Assim, qualquer indivíduo que possuir a chave pública correspondente à chave secreta utilizada na assinatura poderá verificar a autenticidade do emissor. Para que M seja mantida em sigilo deve-se codificá-la através de alguma técnica de criptografia.

Para produzir a "impressão digital" de uma mensagem a ser enviada, ou seja, um identificador único e confiável, uma das técnicas utilizadas é a dos algoritmos *hash*. Esse tipo de algoritmo recebe uma entrada de tamanho variável e gera como saída um valor de tamanho fixo, chamado *hash* ou *digest*, que possui as seguintes propriedades:

- Deve ser computacionalmente impossível encontrar duas mensagens que originem o mesmo valor.
- O *hash* não revela nada sobre a entrada.

O uso de uma impressão digital fornecida por um algoritmo *hash*, em uma mensagem é ilustrado na Figura 2.1.

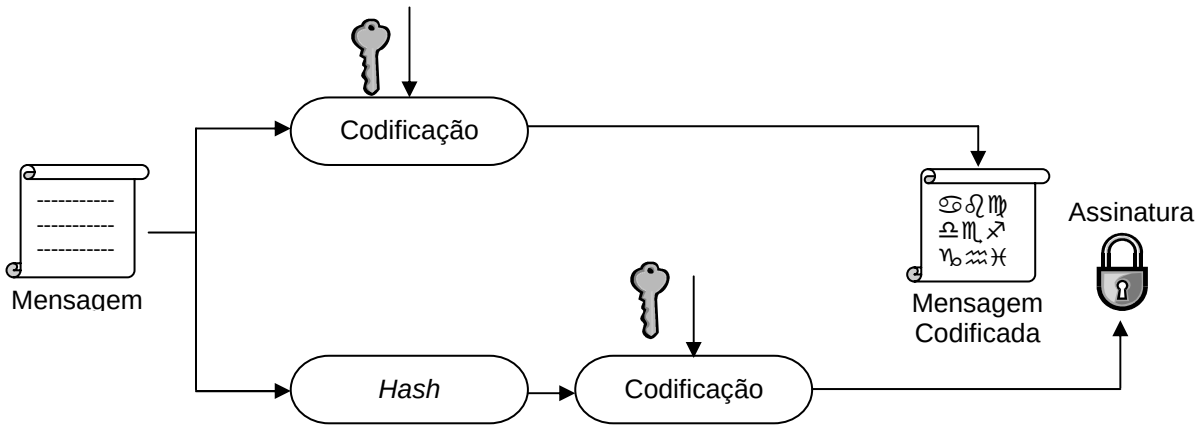



**Figura 2.1 - Criação de uma assinatura digital**

Para a criação da assinatura digital, a mensagem é submetida a uma função hash. O valor gerado pelo algoritmo é então criptografado com a chave secreta do emissor finalizando o processo. Deve-se lembrar que a assinatura em si não é a mensagem a ser enviada, sendo que essa última deve ser criptografada e transmitida junto com a assinatura digital.

O processo de verificação de uma assinatura digital é descrito pela Figura 2.2.

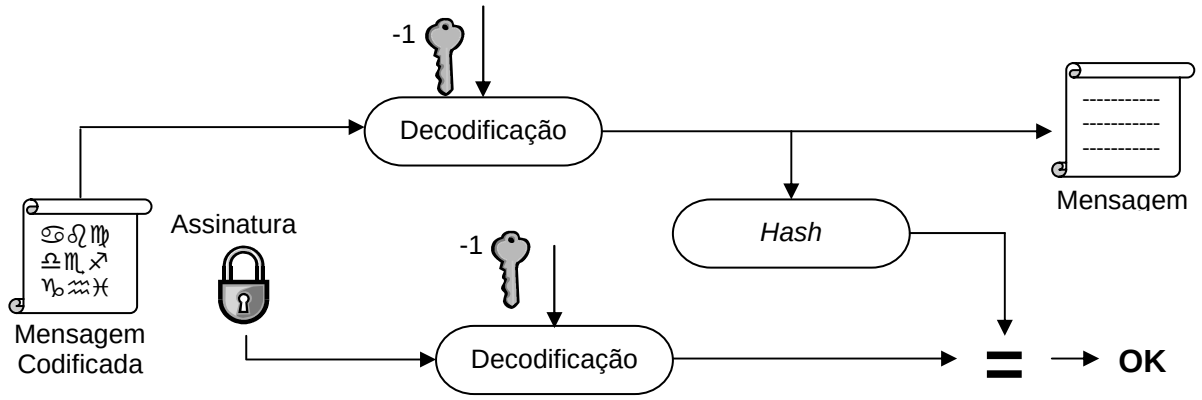



**Figura 2.2 - Verificação de uma assinatura digital**

Quando chega ao seu receptor, a mensagem é decodificada através do método de criptografia da origem e da chave pública do emissor, que pode ter sido enviada junto com a mensagem ou obtida por um serviço de distribuição de chaves públicas. Após o processo de decodificação, é aplicada à mensagem a mesma função *hash* da origem, gerando um valor que deverá ser o mesmo da assinatura enviada. A assinatura então é decodificada e o valor *hash* enviado pela origem é obtido. Se os valores hashes, um recuperado da assinatura digital e o outro gerado sobre a mensagem decodificada, forem iguais, significa que o conteúdo da mensagem não foi alterado.

### *Certificação*

Mesmo tendo certeza que a mensagem enviada não foi alterada devido ao uso dos mecanismos de chave privada e pública apresentados anteriormente, como ter certeza de que quem enviou a chave pública é realmente a entidade emissora? Nesse contexto surgem os certificados e as entidades certificadoras. Os certificados servem para documentar a associação de chaves públicas com entidades. São declarações que comprovam que uma determinada chave pública pertence a uma entidade em particular [Sherwood, 1992]. Nesses certificados são armazenadas informações específicas sobre o detentor da chave pública, tais como nome, endereço, telefone, e-mail, etc.

Para garantir que um certificado realmente é válido surgiram as entidades certificadoras. Uma entidade certificadora assina um certificado através de sua chave privada. Para obter as informações do certificado deve-se decodificá-lo com a chave pública da autoridade certificadora que o assinou.

## 2.4.2 Propriedade

A autenticação baseada na propriedade é realizada com o uso de um objeto, conhecido como *token*, pertencente ao usuário. Esse tipo de autenticação é conhecido como híbrida, pois geralmente utiliza uma senha ou *PIN* além do próprio *token*. Sua principal vantagem reside na dificuldade de se forjar um *token* [Everett, 1992]. Como desvantagens pode-se citar, além do custo adicional do hardware, o roubo e a perda desse objeto.

De acordo com Fiorese [Fiorese, 2000], um dos principais métodos utilizados com os *tokens* é o de desafio/respota.

### 2.4.2.1 *Tokens* de Desafio/Resposta

Nesse método, os *tokens* são como computadores de bolso com teclado e visor, possuindo uma pequena capacidade de processamento.

Para efetuar a sua identificação, o usuário informa o seu *PIN* para que o sistema possa verificar se trata-se de um usuário válido. Após a verificação, o sistema devolve para o usuário um número randômico como desafio. Por sua vez, o usuário informa o mesmo *PIN* anterior para o seu *token* com o objetivo de habilitar o dispositivo para uso. Logo após isso, o usuário entra com o número randômico do desafio no *token*. O dispositivo codifica esse número através de uma chave secreta que possui e mostra o resultado codificado em seu visor.

Na seqüência, o usuário informa ao sistema o resultado mostrado pelo *token*. Finalmente, o sistema decodifica a informação e compara com o número aleatório que ele tinha fornecido anteriormente. Se forem iguais, a autenticação é confirmada. Essa comparação só é possível porque o sistema conhece a chave secreta que está associada ao *token*, sendo ela recuperada através do *PIN* do usuário.

### 2.4.3 Biometria

Os métodos de autenticação baseados na biometria utilizam-se das características físicas e comportamentais dos usuários para o reconhecimento da identidade. As características físicas geralmente usadas para esse fim são face, íris, retina e impressão digital, enquanto que as características comportamentais são escrita, voz e padrão de digitação [Everett, 1992].

Para que uma característica física ou comportamental possa ser utilizada na autenticação, algumas propriedades devem ser observadas e avaliadas [Fiorese, 2000]:

- *Universalidade*: todos os seres humanos devem possuir tal característica;
- *Singularidade*: a característica não deve ser igual em diferentes seres humanos;
- *Permanência*: a característica não deve variar de acordo com o tempo;
- *Mensurabilidade*: a característica deve ser passível de ser mensurada.

De acordo com a característica usada, essas propriedades denotam diferentes valores. Assim, uma característica pode ser mais adequada para um determinado sistema, enquanto outra não. Por exemplo, a face possui uma alta universalidade, uma baixa singularidade, uma média permanência e uma alta mensurabilidade.

Os métodos que se baseiam na biometria não fornecem uma resposta absoluta como ‘sim/não’, diferenciando-se dos métodos baseados no conhecimento e propriedade. Ao contrário, eles fornecem a porcentagem de um determinado usuário ser ou não ser um usuário legítimo do sistema. Com o avanço tecnológico, as margens de erro dessa verificação diminuem constantemente.

Os principais componentes de um sistema biométrico geralmente são:

- *Dispositivo de medida*: possibilita a entrada da característica física ou comportamental a ser comparada. Deve oferecer simplicidade de uso;

- *Software de operação*: responsável pela comparação entre a característica extraída pelo dispositivo de medida e a característica tomada como padrão. Geralmente utiliza redes neurais e lógica *fuzzy* em sua implementação;

#### 2.4.3.1 Reconhecimento de Face

Para reconhecer o rosto de um ser humano, os softwares que implementam esse método mapeiam a geometria e as proporções da face. Para isso levam em conta vários pontos delimitadores que são capazes de definir distância, tamanho e forma de cada elemento de um rosto: orelhas, nariz, olhos, queixo, etc.

Bons softwares de reconhecimento de face disponíveis no mercado podem reconhecer um indivíduo mesmo que ele tenha mudado o corte/cor dos cabelos, tenha deixado bigode/barba ou use, sem exageros, acessórios como chapéus, brincos e óculos. Isso é possível porque o software baseia-se em detalhes fundamentais para o reconhecimento de uma pessoa, da mesma forma que um ser humano reconhece um outro ser humano mesmo que esse último tenha envelhecido levemente ou esteja com maquiagem.

O dispositivo geralmente utilizado para capturar as imagens utilizadas é uma câmera digital acoplada a um computador. A captura de imagens é dividida em duas fases: registro e verificação. Na fase de registro, as imagens capturadas são utilizadas como as imagens padrão para as posteriores comparações. A verificação consiste na comparação das imagens padrão com uma outra capturada no momento que se deseja realizar o reconhecimento de face.

## *2.5 Controle de Acesso*

Neste ponto, é importante fazer uma clara distinção entre a autenticação e o controle de acesso para que não ocorram confusões quanto à função que cada um exerce. Como já foi visto, a autenticação confirma a identidade de um usuário legítimo em um sistema computacional. Já o propósito do controle de acesso é limitar as ações que esse usuário legítimo pode executar [Sandhu and Samarati, 1994], mantendo o nível de visibilidade de acordo com as regras definidas para esse sistema computacional.

Sendo assim, o primeiro processo pelo qual um usuário legítimo passa ao entrar em contato com o sistema é a autenticação. Após isso, é o processo de controle de acesso que fica ativo durante todo o tempo restante de interação entre esse usuário e o sistema.

Na teoria encontrada sobre controle de acesso, geralmente existe uma distinção entre os termos *política* e *mecanismo*. Uma *política* define uma norma de alto nível sobre como os acessos são controlados, enquanto que um *mecanismo* é um software ou hardware de baixo nível que implementa tais políticas.

Nesta seção são descritas algumas políticas de controle de acesso e políticas de administração de autorizações. Além disso, para a visualização de como duas das três políticas de controle de acesso são implementadas, é apresentado o modelo conceitual chamado *matriz de acesso*. Tal modelo é utilizado pela grande maioria dos softwares que possuem mecanismos de segurança.

### 2.5.1 Conceitos Básicos

Antes da descrição detalhada dos itens citados acima, faz-se necessário o conhecimento alguns conceitos básicos. São eles: objeto, sujeito, tipo de operação e a definição formal de uma autorização.

#### 2.5.1.1 Objeto

Muitas abstrações foram feitas no decorrer dos anos para se lidar com o controle de acesso, sendo a mais fundamental delas o conceito de objeto. Um objeto representa um recurso ou informação protegida dentro de um sistema computacional. Por exemplo, uma impressora ou uma tupla de uma relação de um banco de dados.

#### 2.5.1.2 Sujeito

Um sujeito representa um usuário legítimo de um sistema computacional ou um programa que está sendo executado em benefício desse usuário. Tal usuário requer acesso aos objetos protegidos do sistema através de requisições. Caso ele tenha a autorização correspondente à requisição, o direito de acesso é garantido. Caso contrário o direito de acesso é revogado.

Um usuário pode apresentar-se como diferentes sujeitos dentro de um mesmo sistema computacional em ocasiões distintas, dependendo dos direitos que ele necessita. Por exemplo, em um sistema de folha de pagamento um usuário pode possuir um sujeito que lhe dá mais direitos e um outro que lhe dá uma quantidade menor de privilégios. O primeiro pode ser considerado de caráter mais administrativo, enquanto que o segundo está ligado às atividades mais triviais de tal sistema. Em um determinado momento, o usuário citado pode preferir

assumir o segundo sujeito, evitando assim alterações acidentais que seriam permitidas se o primeiro sujeito estivesse sendo usado.

### 2.5.1.3 Tipo de Operação

Um sujeito que requer acesso a um objeto deve informar, além do objeto requerido, o tipo de operação a ser executada sobre esse objeto. Esse tipo de operação varia de acordo com o objeto em questão. Se o objeto for um *arquivo*, por exemplo, os tipos de operação usuais são *leitura*, *escrita*, *execução* e *propriedade*. O tipo de operação *leitura* permite um sujeito verificar o conteúdo de um arquivo. De maneira análoga, o tipo de operação *escrita* permite um sujeito modificar o conteúdo de um arquivo. Executar um arquivo de programa é a função do tipo de operação *execução*. E, finalmente, o tipo de operação *propriedade* permite um sujeito atribuir direitos a um outro sujeito sobre o arquivo. Em contrapartida, se o objeto representa uma *conta bancária*, os tipos de operações básicos são *consultar saldo*, *crédito* e *débito*. Essas operações, que possuem um nível de abstração maior, são implementadas pela aplicação que utiliza esse tipo de objeto.

### 2.5.1.4 Definição Formal de uma Autorização

As políticas de controle de acesso *discretionary* e baseada em papéis (subseções 5.2.1 e 5.2.3 deste capítulo) utilizam-se de uma unidade fundamental para determinar se um sujeito tem ou não tem direito de acessar um objeto protegido. Tal unidade chama-se *autorização*.

De acordo com Bertino [Bertino and Martino, 1993] e Fernandez [Fernandez et al., 1981] [Fernandez et al., 1993], uma autorização é definida formalmente através de uma tripla $<s, t, o>$ em que:

- $s \in S$, é a entidade ativa, denominada sujeito, que requer acesso aos objetos;
- $t \in T$, é a operação executada sobre o objeto, por exemplo, leitura ou escrita;
- $o \in O$, é o objeto acessado e/ou modificado.

As letras maiúsculas *S*, *T* e *O* representam os *domínios* aos quais, respectivamente, os elementos *s*, *t* e *o* pertencem.

Mediante uma requisição de acesso a um objeto por um sujeito, uma função *f* é definida para determinar se a autorização correspondente à requisição existe. Caso exista, é garantido o direito de acesso ao sujeito. Não existindo, o direito é revogado. Assim,

- *f: S x T x O → (Verdadeiro, Falso).*

Dada a requisição de um sujeito *s* para executar a operação *t* no objeto *o*, *f(s, t, o) = Verdadeiro* se <*s, t, o*> existir.

### 2.5.2 Políticas de Controle de Acesso

Os requerimentos de segurança são variáveis no mundo real, mudando de acordo com o problema que se deseja resolver. Devido a isso, diferentes políticas de controle de acesso foram criadas. Abaixo são descritas as três principais.

#### 2.5.2.1 Política *Discretionary*

Por ser muito flexível, a política *discretionary* é ideal para vários sistemas comerciais e industriais. Nessa política, para que um sujeito tenha acesso a um determinado objeto, é necessária a existência de uma autorização armazenada no sistema que o associe com o objeto desejado. Assim, a partir de uma requisição de acesso, uma busca é realizada para checar se há a autorização correspondente. Caso tal autorização seja encontrada, o direito de acesso é garantido ao sujeito.

Como exemplo, supõe-se que um DBA (Administrador de Banco de Dados) crie duas contas de sujeitos – A1 e A2 – e garanta o privilégio de criar tabelas para o sujeito da conta A1. Para isso, o DBA utiliza a seguinte declaração em SQL:

**GRANT** CREATETAB **TO** A1;

Supõe-se agora que o sujeito da conta A1 crie duas tabelas chamadas, respectivamente, de Empregado e Departamento. Sendo agora o proprietário dessas duas tabelas, o sujeito A1 tem todos os privilégios sobre cada uma delas. Ele está habilitado a usar, por exemplo, as seguintes declarações em SQL para garantir ou revogar direitos ao sujeito A2 sobre as tabelas Empregado e Departamento:

1) **GRANT** INSERT **ON** EMPREGADO, DEPARTAMENTO **TO** A2;

2) **GRANT** SELECT **ON** DEPARTAMENTO **TO** A2;

3) **REVOKE** SELECT **ON** DEPARTAMENTO **TO** A2;

Na primeira declaração é garantido ao sujeito da conta A2 estar habilitado a inserir registros nas tabelas Empregado e Departamento. Já na segunda, garante-se o direito de

recuperar os registros existentes na tabela Departamento. A última declaração revoga o direito dado anteriormente ao sujeito A2 de recuperar os registros da tabela Departamento.

A principal característica que deve ser notada na política *discretionary* é que para cada acesso deve sempre existir uma autorização ligando o sujeito *s* ao objeto *o* no tipo de operação *t*. Caso ela não exista, o direito de acesso é negado.

### 2.5.2.2 Política *Mandatory*

A política de controle de acesso *mandatory* baseia-se na classificação de sujeitos e objetos de um sistema em níveis de segurança. Os níveis de segurança dos objetos refletem o dano em potencial que pode ocorrer caso um sujeito acesse indevidamente um determinado objeto, enquanto que os níveis de segurança dos sujeitos indicam quais objetos um sujeito pode acessar.

A política de controle de acesso *discretionary* é conhecida como tudo ou nada, ou seja, o sujeito tem ou não tem o direito de acessar os objetos de acordo com a existência da tripla *(s, t, o)*, que corresponde a uma autorização. De forma diferente, a política de controle *mandatory*, também conhecida como multinível [Baraani-Dastjerdi et al., 1997], não exige o relacionamento de um sujeito e um objeto através de uma tripla para que o acesso seja concedido. Nessa política são os níveis de classificação do sujeito e do objeto que determinam os direitos de acesso através de alguma(s) regra(s) pré-estabelecida(s). Uma regra comum e geralmente utilizada é permitir que os sujeitos pertencentes a um certo nível de segurança apenas acessem os dados classificados no mesmo nível ou em um nível de segurança inferior.

Os níveis de segurança mais usuais são *Top Secret* (TS), *Secret* (S), *Confidential* (C) e *Unclassified* (U), em que TS é o nível mais alto e U o mais baixo. O modelo mais comumente usado para a segurança multinível é chamado Modelo de Bell-LaPadulla [Elmasri and Navathe, 2000], o qual classifica cada entidade (sujeito, conta, programa, etc.) e objeto (tabela, tupla, coluna, visão, operação, etc.) em uma das classes de segurança (TS, S, C e U).

Em um sistema gerenciador de banco de dados relacional, por exemplo, cada valor de atributo de uma tabela tem o seu nível de segurança. Cada tupla tem como seu nível de segurança (NT) o maior nível de segurança dos valores dos atributos que a compõe.

Os exemplos a seguir são formulados sobre a seguinte consulta:

**SELECT** * **FROM** EMPREGADO;

Exemplo 1 - Se um sujeito com nível de segurança S executa a consulta citada acima sobre a tabela Empregado, mostrada de acordo com os valores da Tabela 2.1, ele teria acesso a todos os dados da tabela. Isso porque os níveis de classificação de segurança de todos os valores dos atributos são menores ou iguais ao nível de segurança do sujeito.

| **Nome** | | **Salário** | | **Performance** | | **NT** |
|---|---|---|---|---|---|---|
| João | U | 3500 | C | Suficiente | S | S |
| Cléber | C | 5000 | S | Ótimo | C | S |

**Tabela 2.1 - Tabela de empregados**

Exemplo 2 - Se um sujeito com nível de segurança C executa a mesma consulta, ocorre o processo conhecido como filtragem, em que os dados com nível de segurança maior que C são omitidos, surgindo o valor *null*. Esse processo origina uma nova tabela com os níveis de segurança dos atributos omitidos e de toda a tupla (NT) alterados para C, de acordo com a Tabela 2.2.

| **Nome** | | **Salário** | | **Performance** | | **NT** |
|---|---|---|---|---|---|---|
| João | U | 3500 | C | null | C | C |
| Cléber | C | null | C | Ótimo | C | C |

**Tabela 2.2 - Tabela de empregados após filtragem**

### 2.5.2.3 Política Baseada em Papéis

Um papel é usado para simplificar o gerenciamento de segurança [Lunt, 1995], classificando os sujeitos em diferentes categorias. Ele pode ser visto como um conjunto de ações e responsabilidades que uma determinada categoria possui. Dessa forma, as autorizações são atribuídas para os papéis e não diretamente para os sujeitos. Um sujeito que assume um papel específico obtém indiretamente todos os direitos de acesso pertencentes a esse papel. Geralmente um sujeito pode assumir diferentes papéis em um sistema, dependendo da necessidade da ocasião. Em contrapartida, um papel pode ser assumido por vários sujeitos.

As principais vantagens do uso de papéis são mencionadas abaixo:

- *Gerenciamento de autorizações*: o uso de papéis torna independente o relacionamento entre um sujeito e um objeto. Dessa forma, a tarefa de atribuição de autorizações é dividida em duas partes. A primeira consiste em estabelecer o

relacionamento entre os papéis e os objetos, e a segunda entre os sujeitos e os papéis. Com isso, a realocação das autorizações se dá de uma maneira muito mais simples. Por exemplo, supõe-se que um gerente de um setor de uma empresa recebe uma promoção e tem seu cargo elevado para diretor. Se o sistema não utilizar a política baseada em papéis é necessária a remoção e criação de uma série de autorizações individuais. De modo contrário, a única ação a ser tomada se o sistema utilizar o recurso de papéis é a mudança do papel de gerente para diretor.

- *Hierarquia de papéis*: quando criados, os papéis podem ser organizados em uma hierarquia de acordo com a capacidade de acesso de cada papel. Ao aplicar-se tal hierarquia no exemplo citado no item anterior, o papel *Gerente* fica abaixo do papel *Diretor* devido o cargo de diretor possuir um poder maior que o cargo de gerente em uma empresa no mundo real. Se houvesse ainda um papel *Empregado*, ele estaria abaixo do papel *Gerente*. Tal hierarquia pode ser vista de acordo com a Figura 2.3.

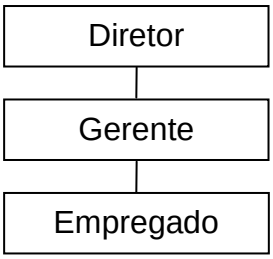



**Figura 2.3 - Exemplo de uma hierarquia de papéis**

A principal utilidade dessa hierarquia é a expansão das autorizações de um papel inferior para um papel superior. Na prática isso significa que o papel *Gerente*, por exemplo, pode acessar todos os objetos que o papel *Empregado* acessa, isto porque o primeiro está acima do segundo na hierarquia de papéis.

- *Privilégio mínimo*: a utilização de papéis permite a flexibilidade de um sujeito escolher um papel que seja suficiente para a realização de uma tarefa em particular, desde que esse sujeito possa assumir dois ou mais papéis e que o papel escolhido esteja abaixo do maior papel que ele possa assumir na hierarquia. Tal atitude diminui a chance de erros por negligência ou descuido.

- *Separação de responsabilidades*: algumas ações são, muitas vezes, conflitantes entre si e não podem ser executadas pelo mesmo sujeito. Por exemplo, um funcionário que é responsável pela distribuição dos salários mensais de uma fábrica não deve ser o

mesmo que faz os cheques para tal fim, pois ele pode fraudar o sistema emitindo um cheque com valor superior ao valor correto sem que ninguém saiba. Evitar tais quadros individualmente torna-se uma tarefa onerosa e cansativa. O uso de papéis simplifica esse procedimento, em que cada tarefa conflitante é atribuída a um papel e fica proibido pelo sistema um mesmo sujeito assumir dois papéis conflitantes.

### 2.5.3 Matriz de Acesso

A matriz de acesso é um modelo conceitual que especifica os direitos que cada sujeito possui sobre cada objeto [Sandhu and Samarati, 1994]. Considerando um conjunto de sujeitos *S*, um conjunto de objetos *O* e uma matriz de acesso *M* que tem uma linha para cada elemento do conjunto *S* e uma coluna para cada elemento do conjunto *O*, uma célula *M[s, o]* contém os tipos de operação que o sujeito *s* pode exercer sobre o objeto *o* [McLean, 1994]. A Tabela 2.3 exemplifica o conceito de matriz de acesso.

| | **Arquivo 1** | **Arquivo 2** | **Arquivo 3** | **Conta 1** | **Conta 2** |
|---|---|---|---|---|---|
| **João** | propriedade<br>escrita<br>leitura | | propriedade<br>escrita<br>leitura | c. saldo<br>crédito | |
| **Maria** | leitura | propriedade<br>escrita<br>leitura | Escrita | c. saldo<br>débito | c. saldo<br>crédito |
| **José** | escrita<br>leitura | leitura | | | c. saldo<br>débito |

**Tabela 2.3 - Exemplo de uma matriz de acesso**

De acordo com a tabela anterior, a primeira coluna indica que *João*, *Maria* e *José* são sujeitos do sistema. Da mesma forma, a primeira linha indique que *Arquivo 1*, *Arquivo 2*, *Arquivo 3*, *Conta1* e *Conta 2* são objetos do sistema. Essa matriz especifica que, por exemplo, o sujeito *João* possui os tipos de operação *propriedade*, *escrita* e *leitura* sobre o objeto *Arquivo 1*, enquanto que o sujeito *Maria* possui apenas o tipo de operação *leitura*.

Embora os tipos de operação *consultar saldo*, *débito* e *crédito* sejam mais abstratos, em suas execuções eles são desmembrados nos tipos de operação *leitura* e *escrita*. A questão envolvida é que um sujeito não pode executar os tipos de operação *leitura* e *escrita* diretamente sobre um objeto *conta bancária*. Por exemplo, quando o sujeito *José* executar o tipo de operação *débito* sobre o objeto *Conta 2*, na realidade o que ocorre é uma *leitura* do valor atual do saldo, a sua modificação e, finalmente, uma *escrita* para a atualização do valor modificado.

### 2.5.3.1 Abordagens de Implementação

Embora conceitualmente a matriz de acesso especifique e organize as informações de maneira não ambígua, sua implementação raramente é idealizada na forma de uma matriz. Isso porque em um sistema computacional de grande porte a tendência é que a matriz tenha um tamanho enorme e, o que é mais grave, inúmeras das células estariam vazias. Para evitar a ocorrência de tais problemas, são discutidas a seguir algumas abordagens de implementação do modelo conceitual da matriz de acesso.

#### *Lista de Controle de Acesso*

Através das *listas de controle de acesso* (ACLs) [Ryutov and Neuman, 2000] [Sandhu and Samarati, 1994] é possível recuperar eficientemente os sujeitos que acessam um determinado objeto. Isso porque cada objeto possui uma lista anexada a ele que informa quais são os sujeitos e os tipos de operações que esses sujeitos podem executar sobre tal objeto. Essa abordagem corresponde ao armazenamento da matriz pelas colunas. A ACL equivalente a uma parte da Tabela 2.3 é mostrada na Figura 2.4 a seguir.

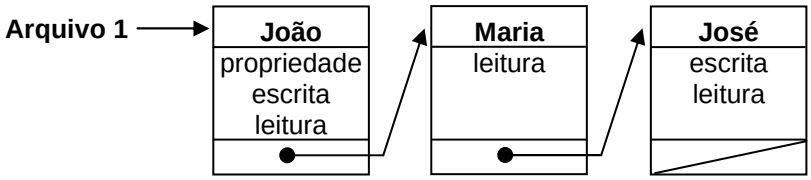



**Figura 2.4 - Um exemplo de ACL**

Uma desvantagem do uso das ACLs é recuperar todos os objetos que um sujeito pode acessar. Para tal tarefa seria necessário analisar a ACL de cada objeto do sistema e verificar se esse sujeito está incluso ou não. Esse problema se repete no caso da necessidade de se excluir todos os direitos de acesso de um sujeito aos objetos. Novamente seria necessário analisar a ACL de cada objeto e excluir os nós correspondentes a tal sujeito. Ambas as operações criam uma sobrecarga de processamento no sistema.

#### *Capacidade*

Uma *capacidade* [Gong, 1989] [Sandhu and Samarati, 1994] é um direito que um sujeito tem de executar um tipo de operação sobre um determinado objeto. Essa é uma abordagem de implementação da matriz de acesso muito semelhante às ACLs, tendo como diferença o armazenamento da matriz pelas linhas. Nela cada sujeito é associado a uma lista de capacidade que especifica quais objetos e tipos de operações sobre esses objetos ele tem

direito. A Figura 2.5 mostra uma parte da Tabela 2.3 através da utilização de uma lista de capacidade.

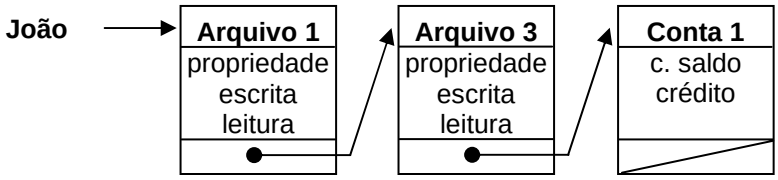



**Figura 2.5 - Um exemplo de lista de capacidade**

Os problemas gerados pelo uso das listas de capacidade são, de forma análoga as ACLs, a recuperação de todos os sujeitos que tem direito de acesso a um determinado objeto e a exclusão dos direitos de acesso para um objeto de todos os sujeitos do sistema computacional.

Para obter um controle de segurança mais refinado é possível combinar ACLs e Capacidades. Por exemplo, após a autenticação de um sujeito em um sistema distribuído, ele recebe suas capacidades e as apresenta aos servidores para obter os serviços dos objetos que eles disponibilizam. Da mesma forma, os servidores mantêm ACLs para confirmar os direitos de acesso desse sujeito.

### *Relações de Autorização*

A utilização de uma das duas abordagens anteriores ocasiona em ganho ou perda de desempenho, dependendo da informação que se deseja buscar. A implementação da matriz de acesso através de relações de autorização não ocasiona tais inconveniências. Cada linha da relação estabelece um tipo de operação que um sujeito pode executar sobre um objeto. A relação de autorização da Tabela 2.4 a seguir representa a matriz de acesso da Tabela 2.3.

| Sujeito | Objeto | Tipo de Operação |
|---|---|---|
| João | Arquivo 1 | propriedade |
| João | Arquivo 1 | escrita |
| João | Arquivo 1 | leitura |
| João | Arquivo 3 | propriedade |
| João | Arquivo 3 | escrita |
| João | Arquivo 3 | leitura |
| João | Conta 1 | c. saldo |
| João | Conta 1 | débito |
| Maria | Arquivo 1 | leitura |
| Maria | Arquivo 2 | propriedade |
| Maria | Arquivo 2 | escrita |
| Maria | Arquivo 2 | leitura |
| Maria | Arquivo3 | escrita |

| | | |
|---|---|---|
| Maria | Conta 1 | c. saldo |
| Maria | Conta 1 | débito |
| Maria | Conta 2 | c. saldo |
| Maria | Conta 2 | crédito |
| José | Arquivo 1 | escrita |
| José | Arquivo 1 | leitura |
| José | Arquivo 2 | leitura |
| José | Conta 1 | c. saldo |
| José | Conta 1 | débito |

**Tabela 2.4 - Exemplo de uma relação de autorização**

Nessa abordagem, se a tabela é ordenada pelo sujeito, consegue-se o efeito do uso da Capacidade. No caso da tabela ser ordenada pelo objeto obtém-se o efeito do uso da ACL.

### 2.5.4 Políticas Administrativas

Todo modelo de autorização deve incluir políticas administrativas que controlam a especificação das autorizações [Bertino et al, 1999], ou seja, elas determinam a quem é permitido agir sobre os direitos de acesso dos sujeitos e de que forma. Assim, um usuário ou um conjunto de usuários, geralmente conhecidos como administradores, são habilitados a garantir, alterar e revogar autorizações.

Na política de controle de acesso *mandatory*, os níveis de segurança dos sujeitos são designados pelo administrador, enquanto que os dos objetos são atribuídos pelos sujeitos que os criam. Na maioria das vezes, o único indivíduo autorizado a modificar os níveis de segurança tanto de sujeitos quanto de objetos nessa política é o administrador.

A política de controle de acesso *discretionary* permite algumas possibilidades de políticas administrativas em que, dependendo da situação, uma é mais apropriada que a outra. São elas:

- *Centralizada*: apenas um único administrador (ou grupo) é responsável pelo gerenciamento das autorizações.
- *Hierárquica*: um administrador central é encarregado de atribuir responsabilidades de gerenciamento a outros administradores. Por sua vez, esses administradores podem garantir ou revogar direitos aos sujeitos do sistema.

- *Cooperativa*: um determinado número de administradores é necessário para que se possa autorizar cooperativamente um sujeito a acessar um objeto de conteúdo muito valioso ou secreto.

- *Proprietária*: um determinado objeto pode ser acessado por diversos sujeitos desde que o seu sujeito criador assim o permita; nessa política cada sujeito é visto como administrador de seus próprios objetos.

- *Descentralizada*: um proprietário pode atribuir direitos administrativos a outros sujeitos sobre os objetos que lhe pertence.

A política de controle de acesso baseada em papéis possui políticas administrativas muito similares às apresentadas acima, sendo desnecessário apresentá-las.

### 2.5.5 Outros Conceitos Importantes

Nesta subseção são apresentados os conceitos de autorização implícita, autorização forte/fraca e autorização positiva/negativa. Além disso, é mostrado como as autorizações implícitas são deduzidas dos três domínios de uma autorização.

#### 2.5.5.1 Autorizações Explícitas/Implícitas

Armazenar todas as autorizações em que à função *f* retorna verdadeiro tornaria o mecanismo de segurança ineficiente em termos de espaço de armazenamento. As autorizações implícitas são utilizadas para resolver esse problema, diminuindo o número de autorizações explícitas. Uma autorização explícita existe fisicamente, sendo armazenada em um dispositivo de memória secundário. Já uma autorização implícita é deduzida de uma autorização explícita, existindo somente em tempo de execução. Segundo Bertino [Bertino and Martino, 1993] e Rabitti [Rabitti et al., 1991], os domínios de uma autorização permitem o surgimento das seguintes regras de derivação:

- Um gerente acessa todas as informações que seus empregados podem acessar; as autorizações podem derivar de acordo com o domínio de *S* (sujeitos).

- Um sujeito autorizado a modificar um objeto também pode ler esse mesmo objeto; as autorizações podem derivar de acordo com o domínio de *T* (tipos de operação).

- Um sujeito que tem direitos sobre uma classe implicitamente os tem sobre as suas instâncias; as autorizações podem derivar de acordo com o domínio de *O* (objetos).

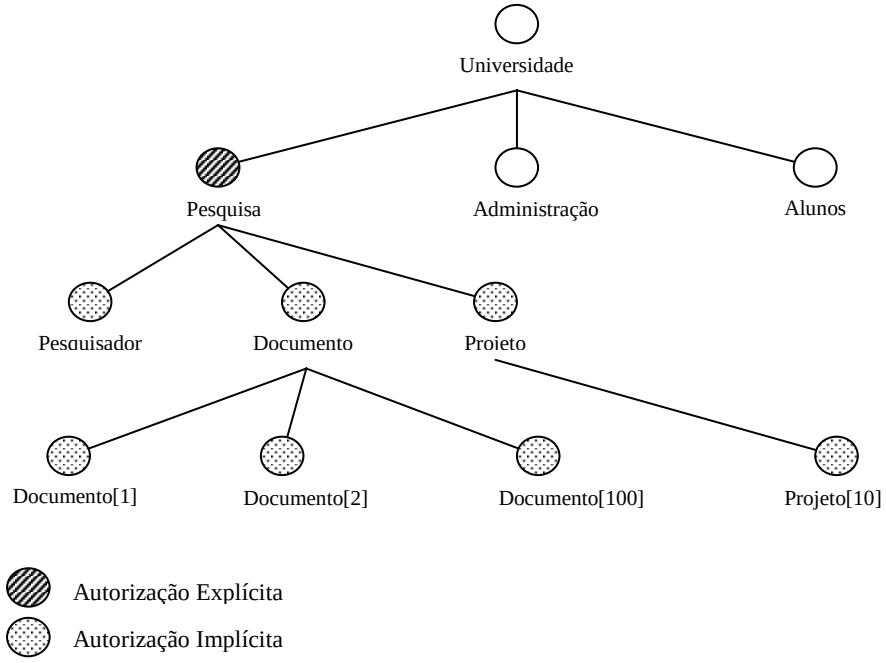



**Figura 2.6 - Autorização implícita**

De acordo com a Figura 2.6, a existência da autorização <*Professor, Escrita, Pesquisa* > permite que o sujeito *Professor* tenha direito de alterar todos os objetos da classe *Pesquisa*. Implicitamente ele tem o mesmo direito sobre as classes agregadas à classe *Pesquisa* e seus objetos, ou seja, as classes *Pesquisador*, *Documento* e *Projeto* com suas respectivas instâncias.

### 2.5.5.2 Autorizações Positivas/Negativas e Fracas/Fortes

Ao se dar uma autorização explícita para um sujeito sobre um objeto, às vezes, deseja-se também que ele não acesse alguns dos objetos componentes do objeto autorizado, ou seja, pretende-se suprimir algumas das autorizações implícitas. Para resolver esse problema foi criado o mecanismo de autorização positiva/negativa e fraca/forte.

Uma autorização forte não admite exceções para suas autorizações implícitas, enquanto que uma autorização fraca admite, sendo essas exceções expressas na forma de autorizações positivas ou negativas.

Uma autorização é positiva por padrão. Caso haja necessidade, pode-se declarar autorizações negativas. As autorizações negativas são úteis para neutralizar as autorizações implícitas vindas de uma autorização fraca positiva explícita atribuída a uma classe ou objeto.

Com o uso combinado da autorização positiva e negativa/forte e fraca, o mecanismo de segurança alcança um nível muito maior de flexibilidade.

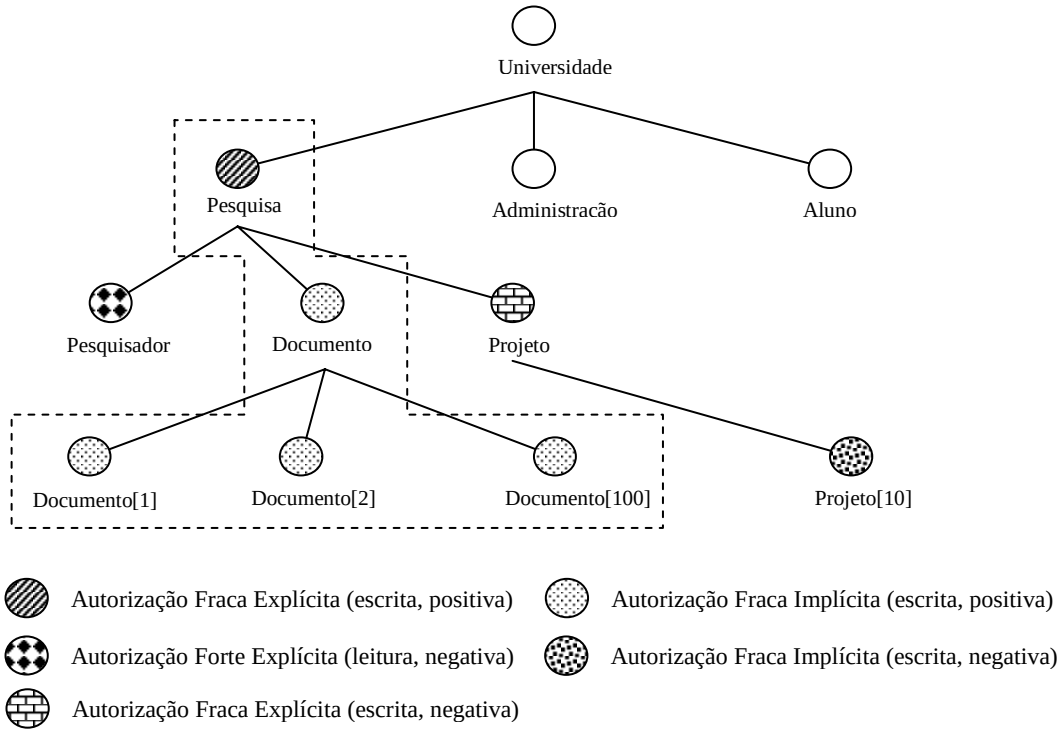



**Figura 2.7 - Autorização positiva/negativa e forte/fraca**

A Figura 2.7 ilustra o mesmo exemplo dado anteriormente, mas com algumas restrições. Considere que os símbolos para se representar autorizações fortes e fracas são, respectivamente, *()* e *[]*, e para se representar autorizações positivas e negativas são os sinais + e – no tipo de operação da autorização. A autorização fraca positiva *[Professor, +Escrita, Pesquisa]*, atribui o direito ao sujeito *Professor* de poder alterar os objetos da classe *Pesquisa* e, implicitamente, todos os objetos de suas classes agregadas. Com a inserção da autorização forte negativa *(Professor, -Leitura, Pesquisador)* e da autorização fraca negativa *[Professor, -Escrita, Projeto]*, o sujeito *Professor* é impedido de ler qualquer objeto da classe *Pesquisador* e de escrever em qualquer objeto da classe *Projeto*. A diferença entre ambas é que na primeira não é possível qualquer tipo de exceção nas classes de nível inferior diretamente ligadas à classe que aparece na autorização, ou seja, não é permitida a inserção de autorizações positivas abaixo da classe *Pesquisador* para o sujeito *Professor*. Na segunda são permitidos tais tipos de exceções.

### 2.5.5.3 Regras de Derivação

Na subseção 2.5.5.1 deste capítulo, as regras de derivação definidas sobre os domínios *S*, *T* e *O* de uma autorização explícita foram vistas sucintamente. As subseções seguintes descrevem com maiores detalhes cada um desses domínios em relação à derivação.

#### *Domínio de Sujeitos*

Em relação ao domínio *S*, usualmente os sujeitos são agrupados de forma a pertencerem a um ou mais papéis. Através de uma hierarquia formada por esses papéis é possível encontrar as autorizações implícitas desse domínio. Para a notação gráfica dessa hierarquia é usado um grafo de papéis, de acordo com a Figura 2.8.

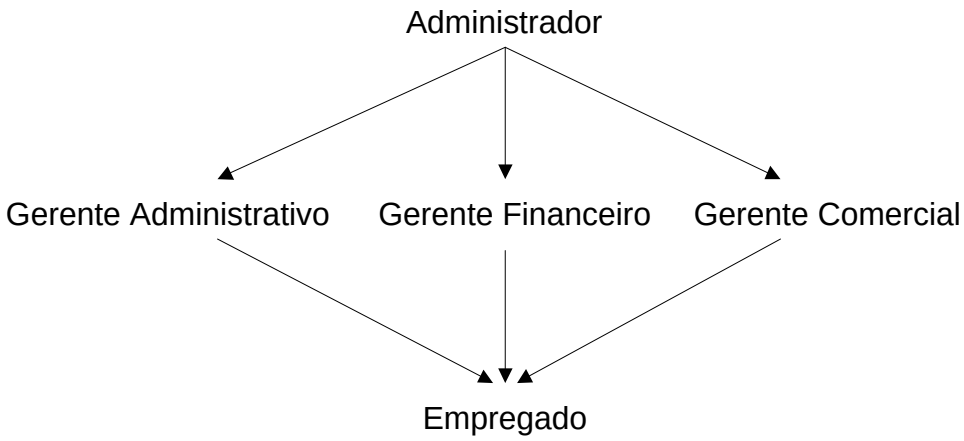



**Figura 2.8 - Grafo de papéis**

Se o papel *Gerente Financeiro* tem o direito *t* sobre o objeto *o*, então o papel *Administrador* tem o mesmo direito. Isso porque *Administrador* é um ancestral de *Gerente Financeiro* dentro do grafo de papéis. Na forma de autorização tem-se que $\langle Gerente\ Financeiro, +t, o\rangle \rightarrow \langle Administrador, +t, o\rangle$. Conseqüentemente, se houver uma autorização negativa para o papel *Administrador* sobre o objeto *o* com o direito *t*, tem-se que nenhum papel abaixo dele na hierarquia de papéis tem o direito *t* sobre o objeto *o*. Como autorizações tem-se que $\langle Administrador, -t, o\rangle \rightarrow \langle Gerente\ Administrativo, -t, o\rangle$, $\langle Gerente\ Financeiro, -t, o\rangle$, $\langle Gerente\ Comercial, -t, o\rangle$, $\langle Empregado, -t, o\rangle$. Observe que o lado direito das implicações não existe fisicamente, ou seja, são autorizações implícitas.

### *Domínio de Tipos de Operação*

Em relação ao domínio *T*, têm-se as mesmas considerações e características do domínio *S*. Um grafo de tipos de operação também é definido, de acordo com a Figura 2.9.


**Figura 2.9 - Grafo de tipos de operação**

Com o uso das operações de leitura (*L*), escrita (*E*) e criação (*C*) pode-se definir que se um sujeito possuir um desses tipos de operação sobre um objeto, ele também tem direito a todos os tipos de operação de níveis inferiores do grafo apresentado. Para as autorizações com o tipo de operação negativa, o sujeito não tem direito a nenhuma operação acima do nível da operação em que o direito lhe foi negado. Por exemplo, a implicação *[Professor, -L, Projeto]* $\rightarrow$ *[Professor, -E, Projeto]* é óbvia, pois se um sujeito não tem direito de ler um objeto, também não tem o direito de escrever nesse objeto.

### *Domínio de Objetos*

Para se explicar a derivação no domínio de objetos, utiliza-se nesta subseção um dos esquemas propostos para sistemas gerenciadores de banco de dados orientados a objetos.

Devido ao fato de um banco de dados conter um grande número de objetos, a derivação em relação ao domínio *O* torna-se particularmente atraente. Os conceitos *AOS* (Esquema de Autorização de Objetos) e *AOL* (Grafo de Autorização de Objetos) são utilizados para a definição das regras de derivação. Enquanto o *AOS* é o esquema da autorização sobre objetos, o *AOL* é uma instanciação do *AOS* em um banco de dados específico.

O conceito *AOS* define a estrutura de como as autorizações serão propagadas em relação ao domínio de objetos. A Figura 2.10 apresenta o *AOS* definido para o gerenciador *Orion* [Banerjee et al., 1987] [Kim et al., 1988].

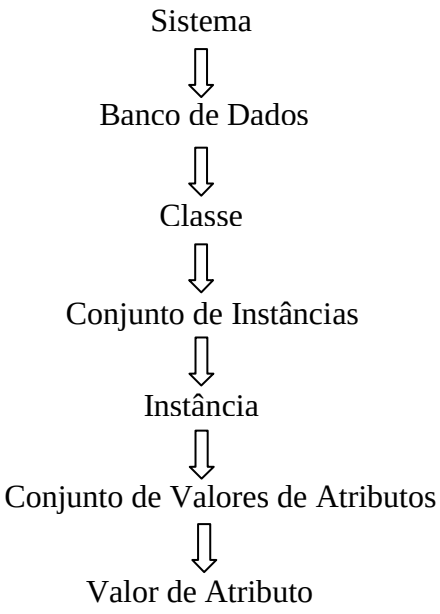



**Figura 2.10 - Esquema de autorização de objetos**

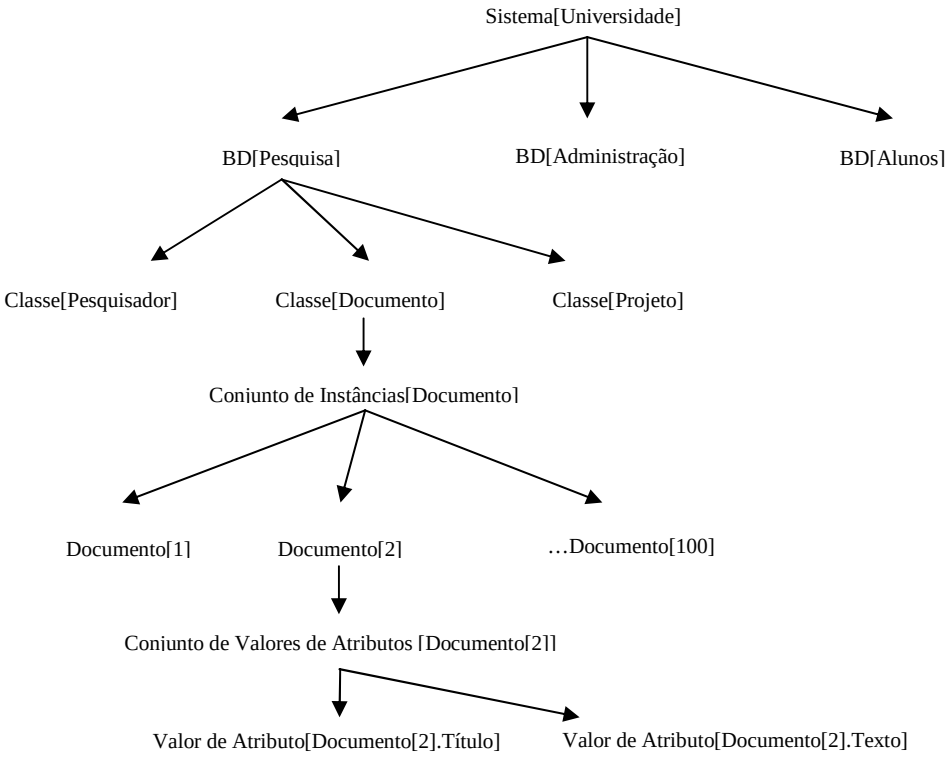



**Figura 2.11 - AOL (instanciação do AOS)**

No esquema da Figura 2.10, se um sujeito tem direito de acessar o item *sistema*, ele tem o mesmo direito sobre todos os níveis inferiores ao nível do item *sistema*.

Uma instanciação do esquema citado pode ser vista na Figura 2.11. Nota-se que essa figura é uma variação do esquema da Figura 2.6 e Figura 2.7, isso porque ela define realmente em que item da hierarquia o sujeito tem uma autorização. Por exemplo, uma autorização na forma *<Professor, +L, Classe[Documento]>* permite ao sujeito *Professor* a leitura da definição da classe *Documento* e de suas instâncias, enquanto que uma autorização do tipo *<Professor, +L, Conjunto de Instâncias[Documento]>* permite apenas a leitura de suas instâncias.

## *2.6 Técnicas Utilizadas na Arquitetura de Segurança*

Todo o material sobre as técnicas de segurança que originaram este capítulo foi cuidadosamente analisado durante o desenvolvimento desse trabalho de mestrado para que fosse possível elaborar uma arquitetura de segurança para ambientes de EaD com um bom nível de confiabilidade, com um alto nível de flexibilidade e de baixo custo de implementação.

Dessa forma, para a autenticação de usuários são usados os conceitos de autenticação por conhecimento, mais especificamente senhas e perguntas randômicas codificadas através de chaves de criptografia. A escolha dessas técnicas para a autenticação deve-se principalmente ao bom nível de segurança que elas oferecem juntas e do baixo custo de implementação, já que nenhum hardware adicional é necessário.

Já para a elaboração do mecanismo de controle de acesso são usados os conceitos de autorização individual/coletiva, autorização positiva/negativa e autorização explícita/implícita. A combinação dessas técnicas na arquitetura de segurança fornece um alto grau de flexibilidade na administração das autorizações.

## *2.7 Considerações Finais*

Neste capítulo foram apresentados os métodos de ataque a sistemas computacionais, as propriedades de segurança e uma descrição detalhada sobre autenticação e controle de acesso. Na seção de autenticação, além da base conceitual sobre o assunto, tópicos como criptografia, assinatura digital e certificação foram descritos. Já na seção de controle de acesso foi realizada uma revisão sobre as políticas de segurança *discretionary* e *mandatory*, além da apresentação de algumas políticas de administração de autorizações. Uma discussão adicional sobre tópicos como autorizações implícitas, positivas/negativas, fortes/fracas e regras de derivação também foi apresentada.

# Capítulo 3
# Educação a Distância

## *3.1 Considerações Iniciais*

A educação a distância (EaD) vem sendo intensivamente estudada nas últimas décadas devido principalmente à necessidade da educação continuada imposta pelo mercado de trabalho e aos avanços tecnológicos que possibilitaram uma revolução nos recursos utilizados para o processo de ensino/aprendizagem. Alguns dos principais componentes relacionados ao ensino, como escolas, universidades, professores e alunos, vêm tendo sensíveis mudanças em relação ao desempenho de suas funções diante desse paradigma.

A seção 3.2 mostra a definição de EaD comumente encontrada na literatura. Na seção 3.3 um breve histórico sobre o surgimento e evolução da EaD é apresentado. A seção 3.4 aborda os principais objetivos da utilização da EaD. A seção 3.5 demonstra alguns dos principais ambientes de EaD disponíveis no mercado e suas características em relação aos seus mecanismos de autenticação e de autorização. Na seção 3.6 é mostrada a avaliação desses mesmos mecanismos. E, finalmente, na seção 3.7 são realizadas as considerações finais deste capítulo.

## *3.2 Definição*

Em seu sentido mais fundamental, EaD significa a separação física e/ou temporal entre o professor e o aluno [Gonçalves, 1996]. Para que ela seja caracterizada é necessária a presença de alguns elementos [Sá, 2000], sendo eles:

- Separação do professor e do aluno no espaço e/ou tempo;
- O controle da aprendizagem é realizado mais intensamente pelo aluno do que pelo professor;
- A interação entre professores e alunos é mediada por alguma forma de tecnologia.

“A educação a distância é uma aprendizagem planejada que normalmente ocorre em um lugar diferente do tradicional e como resultado requer projeto de curso e técnicas instrucionais

especiais, métodos especiais de comunicação eletrônica ou outra tecnologia, bem como sistemas organizacionais e administrativos especiais" [Moore and Kearsley, 2001].

## *3.3 Histórico*

EaD não é um tema que surgiu recentemente e suas origens primitivas remontam a séculos e séculos atrás. Com o advento da escrita, o homem pôde passar para o papel o que antes só podia ser dito, nascendo tempos depois a primeira forma de interação entre o educador e o educando: a correspondência. Chaves [Chaves, 2001] cita que as epístolas do Novo Testamento são claros exemplos de EaD, pois possuem um caráter didático nítido e foram destinadas a comunidades inteiras.

O marco da EaD moderna é apontado como sendo o ano de 1840 quando Issac Pitman lançou no Reino Unido um curso a distância por correspondência. Em 1881, Willian Rainey Harper, respeitado por ser o fundador e reitor da Universidade de Chicago, ofereceu um curso de Hebreu por correspondência que foi um grande sucesso. Após isso, algumas instituições de ensino começaram a se conscientizar para a importância que a EaD teria nas próximas décadas. O *Queen's College* do Canadá foi uma delas e em 1889 iniciou a oferta de diversos cursos a distância. A procura por eles foi massiva devido, principalmente, ao baixo custo e a distância existente entre os centros urbanos desse país [Loyolla and Prates, 2001].

O processo de desenvolvimento da EaD está diretamente ligado ao avanço das tecnologias que ela utiliza como meio para disseminar informação. Essas tecnologias evoluíram gradativamente e algumas delas tornaram-se obsoletas devido a problemas como falta de recursos audiovisuais e lentidão na comunicação entre professor e aluno. De acordo com Moraes [Moraes, 2001], são estas as quatro gerações de tecnologias utilizadas na EaD:

I. Correspondência;

II. Rádio e televisão;

III. Sistemas multimídia integrados;

IV. Informática e telemática.

Antes da expansão da eletrônica e computação, a correspondência através dos correios era a única forma de comunicação para os indivíduos que faziam parte de um curso a distância. Com o tempo, foi percebido que era necessária a existência de recursos que

excedessem a utilização de textos e figuras, pois certos elementos eram extremamente difíceis de se explicar. O rádio e a televisão deram suporte para o fim desse problema, envolvendo recursos de áudio e vídeo na elaboração do conteúdo das aulas. Com a utilização conjunta dos elementos impressos, do rádio e da televisão chega-se aos sistemas multimídia integrados. Após isso, o crescimento do uso do computador e o surgimento da internet permitiram que todas as tecnologias antes utilizadas fossem concentradas em um só local, criando um meio de comunicação muito mais interativo e atraente tanto para professores quanto para alunos.

## *3.4 Objetivos*

De acordo com Seno [Seno, 2001] o principal objetivo da EaD é aplicar a prática educativa e o processo de ensino/aprendizagem de uma forma diferente, de modo que o estudante tenha a possibilidade de aprender a aprender, saber pensar, criar, inovar, construir conhecimentos e participar ativamente de seu próprio aprendizado.

Para Aretio [Aretio, 1994] os objetivos da EaD são:

- *Democratizar o acesso à educação*: igualdade de oportunidades de educação para todos os indivíduos; quebra das barreiras geográficas entre instituição de ensino e aluno; novas possibilidades para as pessoas que não puderam iniciar ou concluir seus estudos; permanência dos alunos em seu meio cultural e natural, evitando êxodos que incidem negativamente no desenvolvimento regional;

- *Propiciar uma aprendizagem autônoma e ligada à experiência*: afastamento do contexto da sala de aula na formação; conscientização por parte do aluno da importância da aprendizagem continuada; o aluno é considerado o sujeito ativo no aprendizado e o professor atua apenas como facilitador do processo; proposta de independência do aluno através de seus próprios critérios, aprimorando sua capacidade para pensar, trabalhar e decidir por si mesmo;

- *Promover um ensino inovador e de qualidade*: aumento da oferta e da diversificação de cursos regulares e não regulares; nova sistemática e recursos didáticos instrucionais; possibilidade de comunicação bidirecional freqüente; elaboração de recursos didáticos e planejamento da instrução por especialistas de comprovada competência; diagnóstico de problemas através da constante avaliação do sistema, verificando se os objetivos da instituição e do curso foram alcançados;

- *Incentivar a educação continuada*: promoção de atividades de extensão educacional e cultural para o atendimento da crescente demanda; reciclagem e aperfeiçoamento profissional através de instrumentos e estratégias de formação continuada;

- *Reduzir os custos*: custos iniciais compensados com a economia em escala; 50% de economia em relação aos sistemas de ensino tradicionais;

## *3.5 Ambientes para EaD e seus Mecanismos de Segurança*

Como a Ead é uma realidade incontestável e há uma grande tendência do seu uso aumentar muito nos próximos anos, vários ambientes já estão implementados e disponíveis na Web. Cada um deles procurou trazer inovações em relação aos seus concorrentes anteriores, iniciando uma busca saudável pela melhoria da tecnologia e do processo remoto de ensino/aprendizagem. Tal fato resultou em ambientes com uma grande quantidade de recursos úteis para professores e alunos.

As próximas subseções apresentam alguns ambientes de EaD que possuem mecanismos de segurança [Morgan, 1999], descrevendo rapidamente suas características e suas ferramentas referentes à autenticação e ao controle da acesso.

### 3.5.1 Blackboard (www.blackboard.com)

Nesse ambiente, professores podem criar e gerenciar cursos via Web sem terem conhecimento de HTML. O material de um curso é facilmente disponibilizado através das ferramentas que compõem o ambiente Blackboard. Como os cursos são protegidos por senhas, apenas os estudantes registrados têm acesso aos materiais. Em adição, o ambiente oferece opções de gerenciamento de curso, como, por exemplo, estatísticas para a avaliação do conteúdo acessado pelos alunos.

Para se utilizar as funcionalidades do ambiente, antes é necessário registrar-se como um usuário, fornecendo informações pessoais (nome, e-mail, etc.), informações sobre a conta a ser criada (nome de usuário e senha) e informações genéricas (sexo, nível de formação, endereço, etc.). Após isso é possível matricular-se em cursos disponíveis ou criar novos cursos via Web.

Informações como título, descrição, área de interesse e disciplina são obrigatórias na criação de um curso, sendo que o usuário registrado que o criou torna-se seu instrutor. São oferecidas também algumas opções para a personalização da interface, como, por exemplo,

formato dos botões e cores. Outra facilidade é o *cartucho de curso*, que representa um conjunto de materiais pré-formatados que o criador de um curso pode utilizar como matéria-prima no desenvolvimento de seus próprios materiais.

Em relação à autenticação, o ambiente apresenta um mecanismo simples que se utiliza de um nome de usuário e uma senha. A partir do momento que um usuário deseja entrar em uma área restrita, ele fornece essas informações e seu *login*[2] é realizado. A sessão[3] de um *login* dura enquanto não for efetuado um *logout*[4] explícito ou enquanto o navegador Web não for fechado. Uma falha relacionada a esse fato ocorre no Blackboard quando um usuário legítimo muda o domínio da *URL* (*Uniform Resource Locator*) do ambiente para um outro domínio qualquer, sendo que a sessão referente a seu curso não é encerrada. Um indivíduo não autorizado pode executar todas as ações que o perfil do usuário legítimo permite caso o primeiro acesse um computador em que tal falha tenha ocorrido.

Várias opções de controle de acesso são disponibilizadas no Blackboard. Através delas o instrutor pode disponibilizar ou não as áreas de comunicação e as ferramentas de um curso. Além disso, é possível deixar certas áreas visíveis somente aos professores e alunos que estão atualmente matriculados, não permitindo que um convidado que esteja visitando o curso possa interagir com elas. É observada na Figura 3.1 uma parte dos itens disponíveis para a definição dos direitos de acesso de um curso.

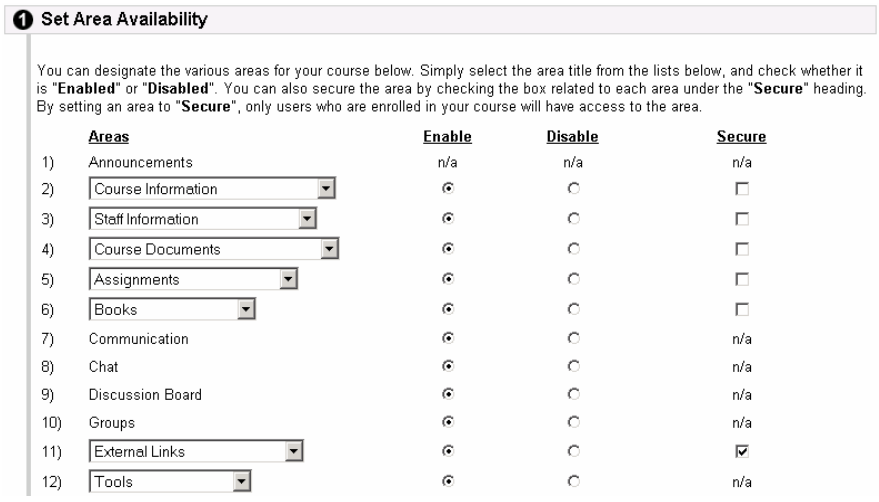



**Figura 3.1 - Algumas opções de controle de acesso do ambiente Blackboard**

[2] Procedimento em que um usuário é reconhecido em um serviço através de um nome de usuário e uma senha.
[3] Série de interações relacionadas entre um cliente e um servidor web [Ayers et al., 1999].
[4] Procedimento em que um usuário encerra explicitamente uma sessão iniciada por um *login*.

### 3.5.2 Intralearn (www.intralearn.com)

Possuindo várias funcionalidades, o ambiente Intralearn oferece meios para criar, distribuir e avaliar o aprendizado interativo através da Internet e intranets. Foi inicialmente projetado para conceder certificados de treinamento em nome de organizações e instituições educacionais que exigiam testes e provas para a avaliação de conhecimento. Algumas características desse ambiente são: recursos multimídia, interface para o envio de e-mail, chat, classes de discussão, quadro de avisos, *FAQ* (*Frequently Asked Question*), glossário, referências bibliográficas, etc.

Tornar-se um usuário nesse ambiente não é tão simples quanto no Blackboard. Primeiramente, é necessário preencher um formulário on-line e requerer a autorização para se testar o ambiente. Nesse formulário, todos os dados obrigatórios para a criação de um usuário são passados: nome, empresa, cargo, endereço de correio eletrônico, interesse no ambiente e interesse em educação a distância. Após algum tempo, um e-mail contendo instruções de como criar um curso é enviado por uma pessoa do suporte do ambiente. Nesse ponto o usuário do interessado já está criado, sendo o seu papel designado como *instrutor*.

Uma ferramenta muito útil para a construção de cursos é fornecida pelo ambiente e é possível um curso ficar pronto para a disponibilização com apenas quatro passos muito simples. No primeiro passo, é requerido um código para o curso que o distinguirá de todos os outros existentes. A categoria, as informações de descrição (título, objetivo e descrição) e a escolha dos componentes de comunicação de um curso são solicitadas no segundo passo. No terceiro, algumas opções de controle do curso são apresentadas. É escolhido nesse passo, por exemplo, se um aluno poderá refazer um exame, se o progresso do aluno será acompanhado, se o curso será disponibilizado logo após a criação, etc. E, finalmente, no quarto passo é definida a primeira aula do curso, juntamente com os seus tópicos.

A autenticação desse ambiente é realizada da mesma forma que no Blackboard, ou seja, através do par nome de usuário/senha. A duração de uma sessão é equivalente ao tempo em que o usuário efetuou o *login* e uma das três seguintes ações: execução de um *logout*, fechamento do navegador Web e mudança do domínio da *URL* do ambiente para um outro domínio qualquer.

Em relação ao controle de acesso, o instrutor pode habilitar/desabilitar ferramentas que um estudante pode utilizar. Na Figura 3.2, as opções disponíveis para isso são demonstradas.

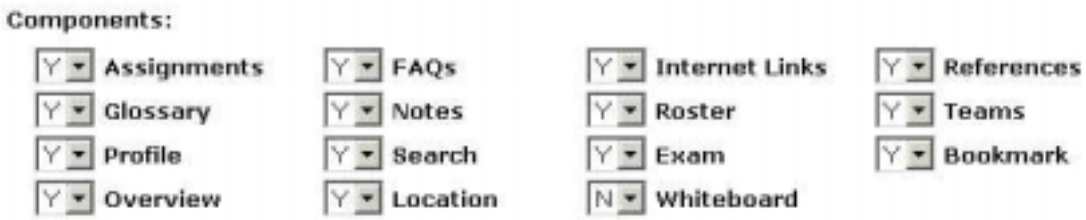



**Figura 3.2 - Algumas opções de controle de acesso do ambiente Intralearn**

Uma característica de segurança interessante nesse ambiente é o *remote proctoring*, em que um fiscal tem controle sobre as ações que um estudante pode tomar. Assim, é possível a prevenção de ações maliciosas ou errôneas. Tal característica torna-se altamente relevante em casos de testes on-line.

### 3.5.3 eCollege (www.ecollege.com)

O ambiente eCollege foi criado para prover universidades e centros de treinamento on-line. É constituído de uma variedade de recursos que tornam o ensino via Web mais efetivo. Permite a personalização de acordo com a necessidade do cliente.

Uma pessoa interessada em tornar-se um usuário desse ambiente realiza o preenchimento de um cadastro on-line. As informações requeridas são: nome completo, nome de usuário, senha, e-mail.

Após a criação do usuário, é possível criar um curso para a avaliação do ambiente. As informações necessárias para se criar o curso são: título, identificador, descrição, nível do curso, categoria e data de início/fim que o curso estará visível para alunos.

O mecanismo de autenticação não se difere dos ambientes anteriores, sendo necessário um nome de usuário e uma senha para o reconhecimento de identidade. Existe o problema do não-encerramento de sessão quando um usuário muda o domínio da *URL* do ambiente para outro domínio qualquer. Em relação ao controle de acesso, não existe nenhuma forma de se controlar quais ferramentas ou qualquer outro item os alunos de um curso acessam.

## *3.6 Avaliação dos Aspectos e dos Mecanismos de Segurança dos Ambientes de EaD Pesquisados*

Os mecanismos de autenticação e de controle de acesso dos ambientes de EaD pesquisados mostraram-se muito semelhantes, inclusive com as mesmas deficiências.

Em relação à autenticação, os ambientes de EaD utilizam o par nome de usuário/senha. Embora esse mecanismo traga um bom nível de segurança, utilizá-lo sem nenhum outro mecanismo adicional pode facilitar a invasão do ambiente por um usuário não autorizado. Por exemplo, um *cracker*[5] pode obter, através de algum método de espreita, o par de informações necessário para efetuar o *login* e, conseqüentemente, acessar o ambiente como um usuário legítimo se esse não utilizar um mecanismo de autenticação adicional.

No que se refere ao controle de acesso, os ambientes de EaD pesquisados comportam-se de uma maneira bastante inflexível. Isso porque as autorizações são atribuídas de maneira coletiva, não permitindo que um administrador de um curso possa configurar os direitos de acesso individualmente, se assim ele o desejar. Além disso, não existe nesses ambientes nenhum esquema de reaproveitamento de autorizações. Tal fato ocasiona no aumento do número necessário de autorizações para se controlar os direitos de acesso de um curso e, conseqüentemente, em um respectivo acréscimo do tempo necessário para gerenciá-las.

Portanto, para ambientes de EaD, notou-se a necessidade do uso de um mecanismo de autenticação adicional e, além disso, a criação de um modelo de controle de acesso que tenha a flexibilidade e a simplicidade no gerenciamento das autorizações de um curso como as suas principais características.

A observação de tais detalhes foi fundamental para a construção da arquitetura de segurança para ambientes de EaD descrita neste trabalho.

## *3.7 Considerações Finais*

Este capítulo abordou tópicos referentes à EaD, como sua definição, uma breve noção sobre como ela surgiu e quais são os seus objetivos. Além disso, foram apresentados alguns ambientes de EaD, juntamente com seus mecanismos de autenticação e autorização. Em seqüência, a avaliação desses mecanismos foi abordada.

No capítulo a seguir é apresentada a proposta de trabalho deste projeto de mestrado, envolvendo a definição de uma arquitetura de segurança que trata, principalmente, dos aspectos da autenticação e do controle de acesso.

[5] Criminoso virtual que usa seus conhecimentos para invadir sistemas, quebrar senhas, roubar dados, etc.

# Capítulo 4
# Uma Arquitetura de Segurança para Ambientes de EaD

## *4.1 Considerações Iniciais*

Com a disponibilização de ambientes de EaD na Web, existe a necessidade de se criar mecanismos de segurança que controlem o acesso dos usuários legítimos a tais ambientes. Atentos a isso, os projetistas de alguns ambientes têm criado cada qual os seus próprios meios de segurança, não levando em consideração a reutilização dos conceitos teóricos e das ferramentas geradas.

Dessa forma, o objetivo deste trabalho é definir uma arquitetura de segurança [Castro, 2002] que pode ser utilizada na concepção e implementação dos aspectos de autenticação e do controle de acesso de ambientes de EaD.

Em relação à autenticação, é apresentado um mecanismo que é utilizado toda vez em que um ator tenta se conectar ao ambiente. Esse mecanismo utiliza as técnicas de nome de usuário/senha e questões randômicas. Tratando-se do controle de acesso, é mostrado um mecanismo que permite atribuição de autorizações coletivas/individuais e atribuição de autorizações positivas/negativas, além de utilizar o conceito de autorizações explícitas/implícitas para o reaproveitamento das autorizações.

Embora a arquitetura de segurança suporte a interação com todos os tipos de papéis que um ator pode assumir (professor, monitor, aluno, etc), este capítulo focaliza com maior ênfase o papel *administrador de curso*, que representa um ator que gerencia os direitos de acesso de todos os outros atores de um determinado curso. Em alguns ambientes de EaD existentes, esse papel também é chamado de instrutor ou administrador.

A seção 4.2 traz uma descrição geral da arquitetura proposta neste trabalho e, em seguida, o funcionamento de cada módulo é explicado detalhadamente. Na seção 4.3 é abordada a integração de um ambiente de EaD com a arquitetura de segurança definida. A seção 4.4 apresenta suas vantagens e desvantagens. E, finalmente, na seção 4.5 são feitas as considerações finais deste capítulo.

## *4.2 Descrição da Arquitetura*

O lado servidor da arquitetura de segurança é idealizado através de módulos para uma divisão mais clara de responsabilidades, sendo que cada um tem uma função específica que é ativada caso seja necessário. De acordo com a figura abaixo, os módulos são: módulo de autenticação, módulo de controle de sessão, módulo de autorização, módulo monitor e módulo auditor.

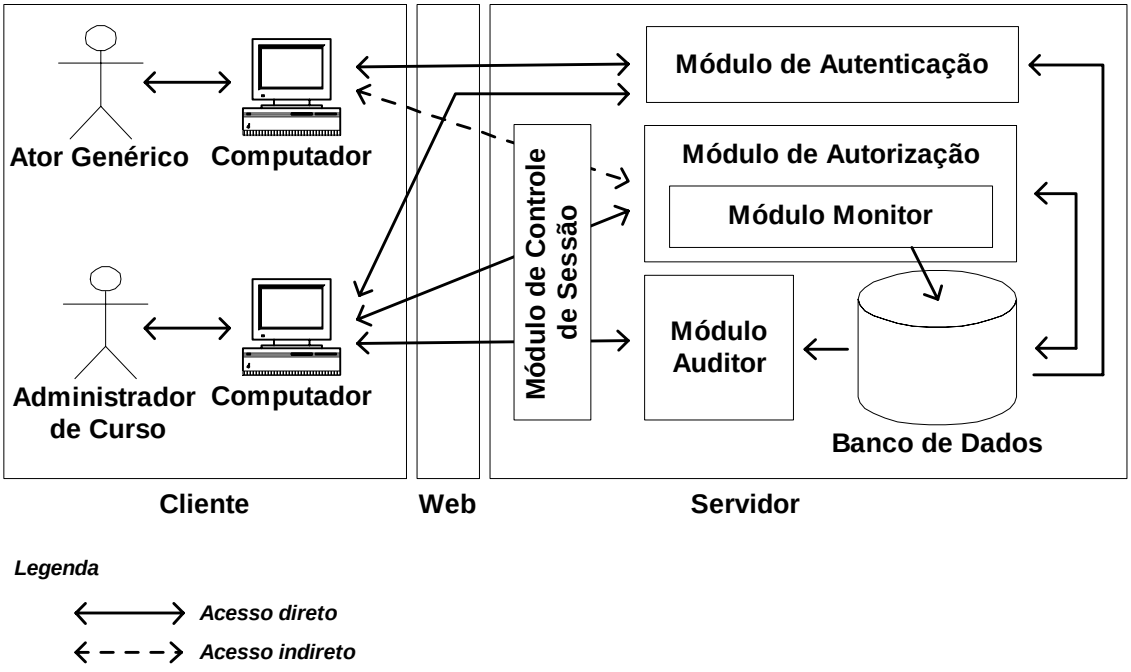



**Figura 4.1 - Arquitetura de segurança**

Como pode ser visto na Figura 4.1, os atores situam-se no lado cliente da arquitetura e interagem através da Web com os módulos que compõem o lado servidor. Todos os atores que assumem os diferentes papéis de um ambiente de EaD têm direito de interagir com os módulos de uma maneira semelhante, mas o ator que assumem o papel administrador de curso possui a característica de gerenciar os direitos de acesso de um curso. Por essa razão, todos os demais atores são representados na ilustração pelo elemento *ator genérico*, enquanto que o administrador de curso é utilizado diretamente.

Um administrador de curso entra em contato com a arquitetura através de um computador conectado à Internet. Utilizando um navegador web, ele passa pelo módulo de autenticação para que seja feito o seu reconhecimento dentro do ambiente. Após esse processo, o administrador de curso pode interagir diretamente (seta cheia) com os módulos de autorização e auditor. Isso significa que ele tem a sua disposição um software respectivo para

cada um desses módulos. Tais softwares permitem o uso de todas as funcionalidades que esses módulos oferecem.

Da mesma forma que o administrador de curso, um ator genérico passa pelo módulo de autenticação. A partir do momento da confirmação da sua identidade, ele acessa o módulo de autorização indiretamente (seta tracejada). Isso significa que ele não tem direito de utilizar o software que permite o acesso a todas as funcionalidades do módulo. Somente através das verificações que as ferramentas, páginas e outros componentes do ambiente realizam é que esse ator interage com o módulo de autorização. Deve-se notar que essas verificações são feitas em segundo plano, ou seja, o ator não sabe que elas estão acontecendo. Além disso, o ator genérico não tem direito de acesso, seja direto ou indireto, ao módulo auditor.

Após a autenticação, todas as interações entre os atores e os módulos de autorização e auditor passam obrigatoriamente pelo módulo de controle de sessão.

Com exceção do módulo de controle de sessão, todos os outros módulos restantes interagem com o banco de dados que mantém as informações de segurança do ambiente. Os módulos de autenticação e auditor apenas recuperam informações, enquanto que o módulo monitor apenas realiza inserções. O único módulo que tem a capacidade de recuperar e inserir informações no banco de dados é o de autorização.

As subseções seguintes explicam detalhadamente as funcionalidades de cada módulo.

### 4.2.1 Módulo de Autenticação

Usualmente, a autenticação é o primeiro processo que um ator é submetido quando ele passa a interagir com uma área restrita do ambiente. Na arquitetura de segurança apresentada, ela é dividida em dois estágios. Primeiramente, um identificador e uma senha do ator são requeridos pelo módulo de autenticação através de um navegador Web. A senha é codificada no cliente por meio de um método de criptografia[6] e ambas informações são enviadas para o servidor. No servidor ambas informações são comparadas com os dados armazenados no BD de segurança para validação. Se os valores não forem compatíveis é requerida do ator a reentrada dos dados. O acesso ao ambiente é proibido por uma faixa de tempo caso a incompatibilidade se repita por um número pré-determinado de vezes. Tal número é configurado pelo administrador de curso.

[6] Neste ponto não é importante a definição de um método de criptografia específico e sim notar que é necessária a codificação das informações a serem enviadas. A escolha do método em si é de responsabilidade do implementador do ambiente de EaD.

Se o ator obtiver sucesso no primeiro estágio, uma questão randômica é apresentada a ele para a confirmação da autenticação. Geralmente essa questão é de cunho pessoal, sendo conhecida apenas pelo próprio ator. Por exemplo, 'Quais são os primeiros três dígitos do seu CPF?'. Cada vez que o ator conecta-se ao ambiente, uma questão diferente é apresentada.

Novamente a informação é codificada no cliente, enviada para o servidor, decodificada e validada através da comparação com os dados armazenados no BD de segurança. Um redirecionamento para o primeiro estágio da autenticação é efetuado se a resposta estiver errada. Em contrapartida, o ator tem acesso garantido às áreas restritas do ambiente caso forneça a resposta correta.

### 4.2.2 Módulo de Controle de Sessão

Após o processo de autenticação, a arquitetura de segurança inicia o controle de sessão da conexão do ator com o ambiente de EaD para evitar brechas que permitam o acesso não-autorizado. Esse controle é realizado pelo módulo de controle de sessão e não permite, por exemplo, o ator voltar para uma página Web pertencente ao ambiente se ele mudou o domínio da *URL* do ambiente para um outro domínio qualquer, sendo necessária uma nova autenticação caso isso ocorra.

As informações são armazenadas no cliente na forma de pares chave=valor pelo módulo de controle de sessão. Essas informações constituem a base para que esse módulo possa decidir se uma sessão deve ou não deve ser encerrada. Por exemplo, a partir do momento que um ator é autenticado em um ambiente de EaD, o par *hora_conexao=16h27min* é armazenado para futuras consultas do ambiente.

### 4.2.3 Módulo de Autorização

Geralmente, os ambientes de EaD disponíveis na Web trabalham com um esquema de autorização que atribui os direitos de acesso aos papéis que os atores assumem. Tal fato tem como conseqüência uma maior simplicidade no gerenciamento das autorizações, pois cada autorização possui um caráter coletivo que evita um grande número de autorizações individuais. Assim, por exemplo, se um ambiente controla o acesso às ferramentas que possui, deve existir um conjunto de autorizações que exprimam os seguintes fatos: o administrador de curso pode acessar a ferramenta de gerenciamento de cursos, os professores podem acessar a ferramenta de construção de aulas, os alunos podem acessar a ferramenta de chat, etc.

Embora possua a vantagem descrita acima, esse tipo de esquema traz consigo a inconveniência da inflexibilidade. Essa dificuldade fica notada quando é necessário que apenas um subconjunto dos atores que assumem um determinado papel acesse uma ferramenta qualquer do ambiente. Por exemplo, ‘x’ e ‘y’ são os únicos alunos autorizados a utilizarem a ferramenta ‘z’.

O módulo de autorização apresentado nesta subseção tem como principal objetivo gerar uma maior flexibilidade para o gerenciamento de autorizações sem perder a facilidade alcançada pelos esquemas tradicionais, permitindo a atribuição de autorizações coletivas/individuais e autorizações positivas/negativas, além de utilizar o conceito de autorizações explícitas/implícitas.

#### 4.2.3.1 Operações de Procura, Inserção e Remoção de uma Autorização

As operações que esse módulo permite podem ser uma procura por uma autorização, uma inserção de uma autorização e uma remoção de uma autorização.

A operação de procura é realizada passando-se, a partir do cliente, um ator ou um papel, um tipo de operação e um objeto protegido como parâmetros para o servidor. É retornado o valor verdadeiro se a autorização é encontrada, caso contrário é retornado o valor falso. Essa é a operação mais utilizada desse módulo devido às constantes requisições de acesso dos atores aos objetos protegidos de um ambiente de EaD.

A inserção de uma autorização necessita de um ator e/ou um papel, um tipo de operação e um objeto como parâmetros, sendo realizada em duas fases. A primeira, após o envio dos dados para o servidor, checa-se a consistência da inserção, ou seja, se realmente ela é possível. As verificações executadas nessa fase são: se não existe uma autorização explícita atualmente armazenada que seja igual àquela que será inserida ou que gere uma autorização implícita equivalente àquela que será armazenada. Não havendo essas duas inconsistências, a segunda fase é realizada, consistindo no armazenamento da nova autorização no banco de dados e o envio de uma mensagem para o cliente que a operação foi executada com sucesso.

Para executar a operação de remoção é necessário passar para o servidor como parâmetros um ator ou um papel, o tipo de operação e o objeto da autorização que se deseja remover. Se for encontrada, ela é eliminada do banco de dados e é enviada uma mensagem para o cliente que a operação foi executada com sucesso. Deve ficar claro que a operação de remoção só é possível em autorizações explícitas.

#### 4.2.3.2 Utilização de Autorizações Coletivas/Individuais

Como já foi dito no capítulo anterior, os ambientes de EaD pesquisados permitem apenas a atribuição de autorizações coletivas, ou seja, sobre os papéis que os atores podem assumir. Isso faz com que o administrador de curso tenha pouca flexibilidade no gerenciamento das autorizações. Por exemplo, em uma determinada situação pode ser necessário atribuir autorizações somente para alguns atores que assumem um papel específico. Tal situação seria impossível de se contemplar nesses ambientes.

O módulo de autorização da arquitetura de segurança elimina esse problema através da possibilidade de se atribuir autorizações coletivas ou individuais. Em uma autorização coletiva, todos os atores que assumem o papel que faz parte dessa autorização têm o direito especificado por ela. Em contrapartida, apenas um ator específico tem o direito que uma autorização individual determina.

#### 4.2.3.3 Utilização de Autorizações Explícitas/Implícitas

Com o uso do conceito de autorizações coletivas/individuais, o número de autorizações existentes aumenta. Caso esse número cresça em proporções geométricas, o gerenciamento das autorizações torna-se uma tarefa onerosa.

Para permitir os benefícios do uso das autorizações coletivas/individuais sem ter o inconveniente citado acima, o módulo de autorização utiliza o conceito de autorizações explícitas/implícitas, que permite uma redução no número de autorizações e, conseqüentemente, uma maior simplicidade de gerenciamento.

Para deixar esse conceito mais claro, é realizada uma comparação entre o esquema tradicional de autorizações utilizado pelos ambientes existentes e o esquema de autorizações explícitas/implícitas usado por esse módulo. Assim, supõe-se que em um curso deseja-se atribuir direitos que relacionam vinte atores que assumem o papel *professor* com as seguintes ferramentas: *ferramenta de construção de aula* e *ferramenta de visualização de aula*. Essas atribuições estão de acordo com os seguintes casos:

- Todos os professores têm direito de acessar as duas ferramentas;
- Apenas três professores têm direito de acessar as duas ferramentas.

No primeiro caso, para os mecanismos de autorização dos ambientes existentes, seria necessária a criação de duas novas autorizações que associam o papel *professor* com as

ferramentas citadas. Já no módulo de autorização, apenas uma autorização que associa o papel professor à ferramenta de construção de aula precisa ser criada. A segunda autorização que dá aos professores o direito de utilizarem a ferramenta de visualização de aula é deduzida implicitamente. Isso porque existe uma hierarquia entre as ferramentas, de acordo com a Figura 4.2 abaixo.

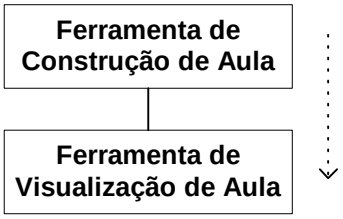



**Figura 4.2 - Hierarquia entre as ferramentas**

Tal hierarquia expressa a seguinte relação: quem constrói uma aula tem direito de assisti-la. Assim, um ator que tem direito de acesso sobre a primeira ferramenta, também o tem sobre a segunda. A seta tracejada demonstra esse fato.

No segundo caso, primeiramente, não é possível atribuir autorizações individuais nos ambientes de EaD pesquisados. O módulo de autorização permite tais atribuições, utilizando o mesmo método descrito para o primeiro caso. Assim, apenas três autorizações seriam criadas, uma para cada professor. As autorizações que permitem o acesso à ferramenta de visualização de aula são deduzidas implicitamente.

Deve-se notar que em ambos os casos houve uma redução de 50% no número das autorizações armazenadas.

### 4.2.3.4 Utilização de Autorizações Positivas/Negativas

Até esse momento, apenas os métodos de atribuição de direitos de acesso foram tratados, ou seja, questões referentes às autorizações positivas, que permitem que um ator ou um papel acesse um determinado objeto. Para negar o direito de acesso a um objeto para um ator ou um papel, o módulo de autorização possui duas formas: não-existência de autorizações e autorizações negativas.

Os ambientes de EaD pesquisados utilizam somente a primeira forma de negação, em que assume-se que um ator não tem o direito de acesso a um objeto temporariamente devido ao fato de uma autorização que permita tal acesso poder ser inserida a qualquer momento. Na não-existência de uma autorização, há a possibilidade de atribuição errônea de direitos através

da inserção negligente de autorizações. Por exemplo, um administrador de curso pode errar ao escolher o parâmetro ator no momento da inserção de uma autorização positiva, mesmo sabendo previamente que esse ator não pode acessar o objeto da autorização. Deve-se notar que, mesmo sabendo de antemão que o ator escolhido erroneamente não pode receber a autorização ao objeto, não existe nenhum mecanismo que garanta que essa autorização equivocada seja inserida.

A proibição do direito de acesso na segunda forma se dá através da inserção de uma autorização negativa, deixando explícito que o ator não pode acessar o objeto. Nesse caso, não há a atribuição errônea de direitos. Retomando o exemplo anterior, a inserção de uma autorização negativa a partir do conhecimento prévio de que um ator não pode acessar um determinado objeto evita todo o problema. Isso porque o módulo de autorização impede a inserção e avisa ao administrador de curso que já existe uma autorização negativa correspondente à positiva que se deseja armazenar.

Uma autorização negativa auxilia diretamente em alguns casos em que uma autorização implícita pode não ser desejada. Por exemplo, modificando-se o segundo caso da subseção anterior, agora se deseja que apenas um professor não tenha o direito implícito sobre a ferramenta de visualização de aula. Para realizar tal tarefa, basta a criação de uma autorização negativa que associe esse professor a essa ferramenta.

As autorizações negativas também são muito úteis para simplificar o gerenciamento das autorizações. Por exemplo, se for desejado que dezenove dos vinte professores acessem a ferramenta de construção de aula, em vez de se criar dezenove autorizações individuais, cria-se apenas duas autorizações: uma que dá o direito ao papel professor de acessar a ferramenta e outra negativa para o único professor que não tem tal direito.

Deve ficar claro que apenas as autorizações positivas são propagadas implicitamente, enquanto que as autorizações negativas são absolutamente estáticas quanto a essa questão.

### 4.2.4 Módulo Monitor

Enquanto utilizam um ambiente de EaD, os atores fazem uma grande quantidade de requisições para o módulo de autorização. Tais requisições podem ser aproveitadas para gerar informações úteis de caráter gerencial para um professor ou para o administrador de curso. Para tanto, é necessário que algumas informações sejam armazenadas cada vez que um ator faça uma requisição de acesso a um objeto para o módulo de autorização. Essas informações

podem englobar, por exemplo, a data e a hora da requisição, se a requisição foi atendida ou não, o tempo que o ator utilizou ou permaneceu no objeto requerido, o objeto previamente acessado e o objeto subseqüente acessado.

Assim, de acordo com a Figura 4.1, o módulo monitor é acoplado ao módulo de autorização para interceptar cada requisição que chega para esse último, mantendo um 'log' das ações de cada ator do ambiente.

### 4.2.5 Módulo Auditor

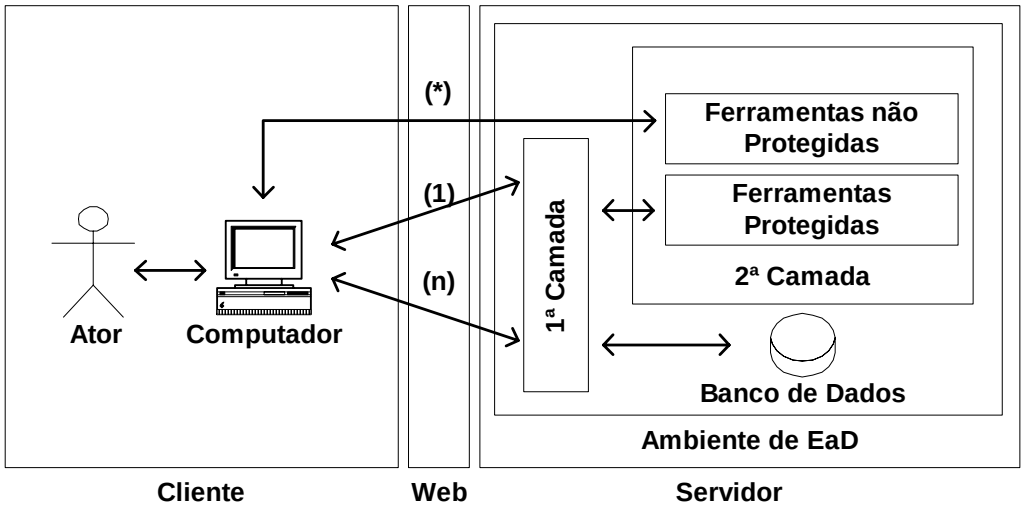
Em algum momento, um professor ou o administrador de curso pode desejar saber, por exemplo, sobre as ações que um ator tomou no decorrer de um curso ou sobre o número de vezes que um objeto em particular foi acessado. O módulo auditor é o responsável por executar esse trabalho.

Desde que as informações necessárias já foram armazenadas pelo módulo monitor, o módulo auditor as recupera e as analisa. O resultado retornado depende de que maneira as informações foram processadas.

Os dois principais objetivos deste módulo são ajudar na avaliação de alunos e criar perfis estatísticos dos atores do ambiente. No primeiro caso, informações do tipo "uma das possibilidades do estudante X obter a nota Y foi devido ele acessar os materiais necessários para o estudo da disciplina Z vezes" podem ser obtidas. Já no segundo caso, informações do modo "o estudante X tem como rotina acessar o curso Y às 14h com um tempo de permanência de 36min".

## *4.3 Integração da Arquitetura de Segurança com Ambientes de EaD*

Na Figura 4.1, a arquitetura de segurança foi mostrada isoladamente para simplificar a compreensão da interação dos atores com os módulos e dos módulos com o banco de dados, mas na realidade ela é uma parte do ambiente que a utiliza. Assim, ela pode ser vista como a primeira camada do ambiente de EaD que protege uma segunda camada dos acessos não permitidos. Essa segunda camada é constituída por elementos como, utilizando o exemplo anterior, as ferramentas do ambiente que os atores usam durante o tempo de conexão. Ambas camadas são ilustradas abaixo pela Figura 4.3.




***Legenda***

***(1) Autenticação***

***(n) Autorizações***

***(*) Após a autenticação***

**Figura 4.3 - Integração da arquitetura de segurança com um ambiente de EaD**

Assim, de acordo com a figura, para poder interagir com a segunda camada pela primeira vez (1), o ator deve passar pelo módulo de autenticação. Nas sucessivas interações (n) isso não é necessário. Na segunda camada, existe um conjunto de elementos protegidos e um outro conjunto de elementos não protegidos. O módulo de autorização controla o acesso apenas ao primeiro conjunto, ou seja, para cada requisição de acesso a uma ferramenta protegida por um ator, a primeira camada verifica no banco de dados se existe uma autorização correspondente, seja ela explícita ou implícita. Excluindo-se a autenticação (*), nenhuma precaução é tomada pela arquitetura caso o ator deseje acessar alguma ferramenta do segundo conjunto.

## *4.4 Vantagens e Desvantagens da Arquitetura de Segurança*

Os diversos módulos que compõem a arquitetura possuem algumas vantagens e desvantagens. Abaixo elas são citadas para cada módulo.

Em relação ao módulo de autenticação, a vantagem é ele define um mecanismo adicional além do uso do par nome de usuário/senha. Deve-se levar em conta também a utilização de chaves de criptografia. Tais fatos aumentam a segurança contra invasões de usuários não-autorizados em um ambiente de EaD. A desvantagem é que nunca um processo de autenticação será 100% seguro, independente da tecnologia que ele utilize.

O módulo de controle de sessão possui a vantagem de se controlar mais rigorosamente a conexão do ator com o ambiente de EaD. Dessa forma, é possível evitar brechas de segurança que colocariam em risco a área particular de um determinado ator ou o próprio ambiente.

Em relação ao módulo de autorização, diversas vantagens são obtidas com o seu uso. São elas: alto nível de flexibilidade, simplicidade no gerenciamento das autorizações, redução no número de autorizações armazenadas, etc. Como desvantagens devem ser levados em consideração o impacto no desempenho computacional causado pela utilização das autorizações explícitas/implícitas e a dificuldade no entendimento desse conceito pelos administradores de curso. No primeiro caso, para que uma autorização implícita possa ser encontrada, várias outras autorizações precisam ser analisadas. Já no segundo, um administrador de cursos pode levar um certo tempo para se familiarizar com o conceito das autorizações explícitas/implícitas.

Os módulos monitor e auditor permitem a criação de informações de caráter gerencial, que são de suma importância para os cursos em andamento num ambiente de EaD. A principal desvantagem é o tempo gasto em analisar os dados primários e gerar essas informações gerenciais.

## *4.5 Considerações Finais*

Neste capítulo foi apresentada uma arquitetura de segurança para ambientes de EaD. Suas duas principais características são: maior segurança na realização da autenticação, maior controle da conexão entre um ator e o ambiente, uma maior flexibilidade e simplicidade na administração das autorizações, que torna o ambiente mais flexível para um administrador de curso, e a geração de informações de caráter gerencial para tomada de decisões. Cada módulo pertencente à arquitetura teve sua funcionalidade detalhadamente descrita e, além disso, foi mostrado a integração dela com um ambiente de EaD. E, para finalizar, foram abordadas as vantagens e desvantagens do uso da arquitetura de segurança.

No capítulo seguinte são apresentados os aspectos de implementação da arquitetura de segurança, ou seja, os tópicos mais relevantes para colocar-se em funcionamento os conceitos apresentados neste capítulo. Tais aspectos são enfatizados nos módulos de autenticação, de controle de sessão e de autorização. Embora os módulos monitor e auditor façam parte do esquema conceitual da arquitetura, maiores estudos sobre eles e suas respectivas implementações são deixados como trabalhos futuros.

# Capítulo 5
# Aspectos de Implementação

## *5.1 Considerações Iniciais*

Enquanto o capítulo anterior preocupou-se em apresentar a arquitetura de segurança em uma visão mais conceitual, este capítulo prende-se a uma visão mais prática, abordando os aspectos que devem ser observados para obter-se uma implementação coerente com as funcionalidades de cada módulo.

Não foi escolhida nenhuma plataforma de hardware ou software específica para apresentar esses aspectos de implementação, buscando-se direcionar o desenvolvedor do ambiente para uma solução de 'alto nível' através da discussão de pontos fundamentais e das estruturas de armazenamento das informações necessárias.

As seções deste capítulo seguem a mesma seqüência em que os módulos foram apresentados no capítulo anterior. Assim, a seção 5.2 apresenta os itens mais relevantes sobre o módulo de autenticação. Na seção 5.3 são abordados os tópicos referentes ao módulo de autorização. A seção 5.4 aborda o banco de dados de segurança. Finalmente, a seção 5.5 finaliza este capítulo, apresentando suas conclusões.

## *5.2 Módulo de Autenticação*

Várias tecnologias necessárias para a implementação do módulo de autenticação já foram criadas e testadas, sendo preciso apenas a sua incorporação na arquitetura de segurança. Assim, esta seção aponta quais dessas tecnologias são as mais adequadas para atender as funcionalidades desse módulo.

A autenticação é utilizada partindo do princípio que apenas o ator que possui o par de informações nome de usuário/senha as conhece. Para dificultar que outros indivíduos tenham acesso a essas informações durante a sua transmissão, é recomendada a utilização de chaves de criptografia. Dessa forma, o princípio da confidencialidade também é garantido (subseção 2.3 do capítulo 2).

A arquitetura de segurança adota o conceito de chaves devido a dois motivos principais:

- Independência do sigilo do algoritmo de criptografia;
- Existência de algoritmos já consagrados no mercado.

Com a vantagem da independência do sigilo do algoritmo de criptografia, caso fique detectado que um acesso não autorizado ocorreu, basta apenas trocar-se a chave utilizada na codificação, sem maiores problemas em relação ao algoritmo. A vantagem da existência de algoritmos consagrados, como o DES (Data Encryption Standard) [Ganley and Piper, 1992] e o RSA [Bidzos, 1992], que utilizam chaves traz consigo uma grande confiabilidade no mecanismo utilizado pelo ambiente, pois tais algoritmos já foram exaustivamente testados.

### 5.2.1 Modelagem de Classes

Para que as informações referentes à autenticação sejam armazenadas no BD de segurança é necessária a existência de uma classe *Ator*, de acordo com a Figura 5.1.

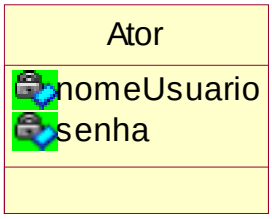



**Figura 5.1 - Informações utilizadas pelo módulo de autenticação**

A classe *Ator* representa os atores do ambiente de EaD. Os atributos *nomeUsuario* e *senha* denotam, respectivamente, um identificador único e a sua senha correspondente. Ambos são utilizados para a autenticação simples, sendo que outros atributos adicionais (CPF, nome, endereço, telefone, etc.) que são utilizados para complementar esse tipo de autenticação através de questões randômicas não foram colocados na figura por serem diferentes de um ambiente para outro.

## *5.3 Módulo de Controle de Sessão*

Para dar suporte ao controle de sessão da conexão entre o ator e o ambiente de EaD, o módulo de controle de sessão utiliza a tecnologia de *cookies*. *Cookies* são pequenos arquivos texto que armazenam conjuntos de pares chave=valor [Ayers et al., 1999]. Inicialmente, eles originam-se no servidor e são enviados para o cliente como instruções no cabeçalho da resposta HTTP (Hiper Text Transfer Protocol). Essas instruções informam ao navegador para

criar um novo *cookie* com um determinado nome (chave), armazenando um determinado valor. A única informação exigida pela arquitetura a ser armazenada é o horário do início da conexão. A partir dela é possível invalidar uma seção se não houve nenhuma interação entre o cliente e o servidor em um determinado intervalo de tempo. Mudar do domínio da *URL* ou fechar o navegador web fará com que o próprio navegador invalide automaticamente o *cookie* e, conseqüentemente, a sessão.

## *5.4 Módulo de Autorização*

A implementação do módulo de autorização exige o esclarecimento dos seguintes pontos: a definição de uma autorização que atenda os requisitos impostos pelas funcionalidades do módulo, as hierarquias necessárias para se descobrir autorizações implícitas, a formulação de uma estrutura de classes para se armazenar autorizações explícitas e os algoritmos de busca, inserção e remoção de uma autorização e suas implicações.

### 5.4.1 Definição de uma Autorização para Controle de Acesso em Ambientes de EaD

Para que o módulo de autorização possa cumprir suas funcionalidades, é necessário definir uma autorização que atenda os seguintes requerimentos:

- Atribuição de autorizações coletivas/individuais;
- Atribuição de autorizações positivas/negativas;

Assim, tal autorização é formada pela extensão da tripla $<s, t, o>$ para uma quádrupla $<s, p, {}^{\{+/-\}}t, o>$ em que os elementos *s*, *t* e *o* têm os mesmos significados da tripla mostrada anteriormente (subseção 3.5.1.4 do capítulo 3). O elemento *p* demonstra um papel que o ator *s* assume dentro de um curso, sendo que $p \in P$. O domínio *P* representa os papéis definidos para o ambiente de EaD. O conjunto sobrescrito $^{\{+/-\}}$ indica que a autorização pode ser positiva ou negativa.

Utilizando a definição acima, basta a criação da autorização individual positiva *<camila, aluno, +acesso, agenda>* para que o sujeito *camila* possa executar o tipo de operação *acesso* sobre o objeto *agenda* assumindo o papel aluno.

Para se criar uma autorização coletiva, não é necessário indicar um ator em particular. Assim, a autorização coletiva negativa *<null, professor, -acesso, elaboração de provas>* indica que o papel *professor* não pode executar o tipo de operação *acesso* sobre o objeto

*elaboração de provas*. Deve-se notar que no lugar do sujeito aparece o indicador *null*, mostrando que se trata de uma autorização coletiva.

### 5.4.2 Construção das Hierarquias dos Domínios de uma Autorização

Para se encontrar as autorizações implícitas através do algoritmo de busca de autorizações, faz-se necessária a construção das hierarquias dos domínios de papéis (*P*), tipos de operação (*T*) e objetos (*O*). Esses domínios variam de ambiente para ambiente, por isso são apresentadas nesta subseção apenas algumas considerações de como construí-las através de alguns exemplos práticos.

#### 5.4.2.1 Hierarquia de Papéis

A hierarquia mais intuitiva de se construir é a hierarquia de papéis. Isso porque os atores de um ambiente de EaD exercem diferentes funções que podem ser agrupadas em papéis. Cada um desses papéis demanda uma carga de responsabilidades menor ou maior no gerenciamento de um curso. Assim, a hierarquia pode ser construída justamente através das cargas de responsabilidades que cada papel possui. Por exemplo, o papel *administrador de curso* possui a maior carga de responsabilidades, tornando-se o papel de nível mais alto na hierarquia, enquanto que o papel *visitante*, que representa um ator que está conhecendo o ambiente, fica no nível mais baixo da hierarquia. A Figura 5.2 mostra um exemplo da hierarquia de papéis.

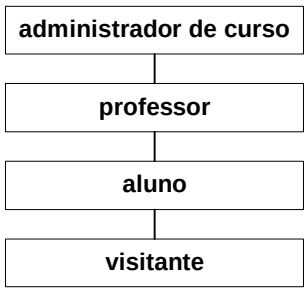



**Figura 5.2 - Hierarquia de papéis em um ambiente de EaD**

#### 5.4.2.2 Hierarquia de Objetos

A criação da hierarquia de objetos dentro de um ambiente de EaD é divida em duas fases principais. Primeiramente, os objetos que necessitam de proteção contra acessos não-autorizados devem ser identificados. Por exemplo, um ambiente pode controlar o acesso sobre as ferramentas que os atores utilizam, enquanto outro ambiente pode controlar o acesso às páginas Web que o compõe. Responder a seguinte questão pode auxiliar nessa fase: ‘Quais

são os itens do ambiente que nem todos os atores podem acessar?'. A segunda fase é constituída da criação da hierarquia propriamente dita. O principal problema dessa fase é que nem sempre o domínio de objetos forma uma hierarquia de modo tão natural quanto o domínio de papéis, requerendo uma análise mais cuidadosa de como esses objetos se relacionam uns com os outros. Para ilustrar a realização dessa análise, são construídas as duas hierarquias de objetos citadas acima: ferramentas e páginas Web.

- *Ferramentas*: uma política para se construir uma hierarquia de ferramentas é agrupá-las de acordo com os papéis que as utilizam. Assim, se o ambiente contém o papel *administrador de curso*, provavelmente existe um conjunto de ferramentas administrativas e gerenciais. Da mesma forma, se existe um papel *professor*, é provável que exista um conjunto de ferramentas para a construção de aulas e avaliação de alunos. Essa correspondência direta facilita a construção da hierarquia, sendo que o nível de cada ferramenta corresponde ao nível do papel que a utiliza, de acordo com a Figura 5.3.

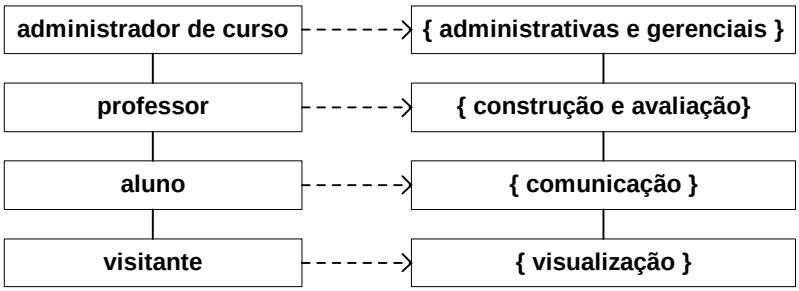



**Figura 5.3 - Hierarquia de ferramentas em correspondência com a hierarquia de papéis**

Na figura acima, cada nível da hierarquia de ferramentas é delimitado pelos símbolos *{}*, significando que em cada nível existe uma ou mais ferramentas. É comum ocorrer situações em que dois papéis acessam a mesma ferramenta. Nesses casos, a ferramenta deve ser colocada na hierarquia em correspondência com o papel de nível mais inferior. Assim, o papel de nível mais superior herda implicitamente o direito de acessar tal ferramenta.

- *Páginas Web*: uma abordagem que pode ser tomada nesse caso é analisar como um ator pode navegar entre as diversas páginas que compõem um curso de um ambiente de EaD. Através dessa análise a hierarquia de páginas pode ser construída, colocando-se a página principal do curso como o nível mais alto. O próximo nível é

formado pelas páginas que a página principal dá acesso e assim sucessivamente. A Figura 5.4 mostra um exemplo simples dessa hierarquia de páginas.

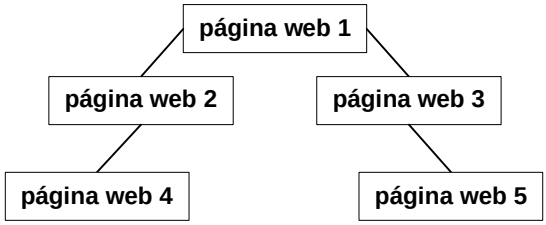



**Figura 5.4 - Hierarquia de páginas web**

### 5.4.2.3 Hierarquia de Tipos de Operação

Se existir mais de uma operação aplicável nos objetos protegidos definidos pelo ambiente de EaD, é possível se encontrar uma hierarquia entre elas. Essa hierarquia é construída a partir da verificação de qual operação é pré-requisito de uma outra. A operação considerada como pré-requisito é colocada em um nível inferior da hierarquia de operações, enquanto que a outra operação é posta em um nível superior.

Para a hierarquia de ferramentas citada anteriormente, apenas a operação *acesso* é necessária. Nesse caso não há como se construir uma hierarquia, devido o domínio de tipos de operação possuir somente uma operação. Já para a hierarquia de páginas web, as operações de *interação* e *visualização* podem ser definidas. Nesse caso, a operação de *interação* ocupa um lugar superior na hierarquia, pois para que um ator possa interagir com uma página, antes é necessária a sua visualização. A Figura 5.5 demonstra a hierarquia entre as duas operações.

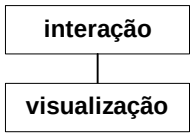



**Figura 5.5 - Hierarquia de operações em um ambiente de EaD**

### 5.4.2.4 Interação entre as Hierarquias

Mediante uma requisição é necessário analisar como as hierarquias de papéis, tipos de operação e objetos interagem entre si para verificar se um determinado ator tem direito de acesso sobre um objeto. Para ilustrar essas interações, são consideradas as hierarquias apresentadas anteriormente, sendo que as hierarquias de papéis e objetos são ligeiramente simplificadas, de acordo com o seguinte: *{administrador de curso, professor, aluno}* e

*{página web 1, página web 2, página web 3}*. Supõe-se um exemplo em que um ator que assume o papel *professor* faz uma requisição com o tipo de operação *visualização* sobre o objeto *página web 3*. Tal ator tem o direito implicitamente garantido devido ao papel *estudante* ter uma autorização explícita que lhe dá o direito de executar o tipo de operação *interação* sobre o objeto *página web 2*. Todos os passos necessários para se encontrar essa autorização implícita são mostrados na Tabela 5.1.

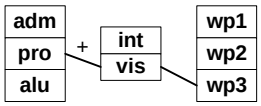
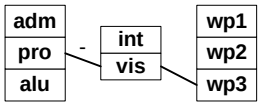
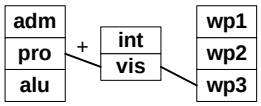
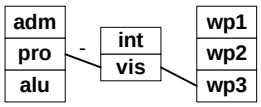
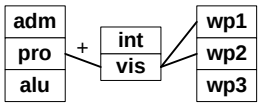
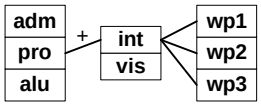
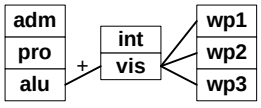
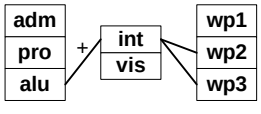
| Passo | Ilustração | Descrição |
|---|---|---|
| 1º |  | Procura-se pela autorização explícita *<ator, professor, +visualização, página web 3>*. Retorna-se verdadeiro se for encontrada, encerrando-se a execução da busca. Caso contrário, passa-se para o próximo passo. |
| 2º |  | Procura-se pela autorização explícita *<ator, professor, -visualização, página web 3>*. Retorna-se falso se for encontrada, encerrando-se a execução da busca. Caso contrário, passa-se para o próximo passo. |
| 3º |  | Procura-se pela autorização explícita *<null, professor, +visualização, página web 3>*. Retorna-se verdadeiro se for encontrada, encerrando-se a execução da busca. Caso contrário, passa-se para o próximo passo. |
| 4º |  | Procura-se pela autorização explícita *<null, professor, -visualização, página web 3>*. Retorna-se falso se for encontrada, encerrando-se a execução da busca. Caso contrário, passa-se para o próximo passo. |
| 5º |  | Procura-se seqüencialmente pelas autorizações explícitas *<{ator/null}, professor, +visualização, página web 2>* e *<{ator/null}, professor, +visualização, página web 1>*. Retorna-se verdadeiro se uma delas for encontrada, encerrando-se a execução da busca. Caso contrário, passa-se para o próximo passo. |
| 6º |  | Como não há mais objetos na hierarquia de objetos, muda-se para a próxima operação na hierarquia de operações e volta-se para o objeto da requisição. Em seguida, procura-se seqüencialmente pelas autorizações explícitas *<{ator/null}, professor, +interação, página web 3>*, *<{ator/null}, professor, +interação, página web 2>* e *<{ator/null}, professor, +interação, página web 1>*. Retorna-se verdadeiro se uma delas for encontrada, encerrando-se a execução da busca. Caso contrário, passa-se para o próximo passo. |
| 7º |  | Como não há mais operações na hierarquia de operações e objetos na hierarquia de objetos, muda-se para o próximo papel da hierarquia de papéis e volta-se para a operação e objeto da requisição. Em seguida, procura-se seqüencialmente pelas autorizações explícitas *<{atores/null}, aluno, +visualização, página web 3>*, *<{atores/null}, aluno, +visualização, página web 2>* e *<{atores/null}, aluno, +visualização, página web 1>*. Retorna-se verdadeiro se uma delas for encontrada, encerrando-se a execução da busca. Caso contrário, passa-se para o próximo passo. |
| 8º |  | Como não há mais objetos na hierarquia de objetos, muda-se para a próxima operação na hierarquia de operações e volta-se para o objeto da requisição. Em seguida, procura-se seqüencialmente pelas autorizações explícitas *<{atores/null}, aluno, +interação, página web 3> e <{atores/null}, aluno, +interação, página web 2>*. Como no último caso a autorização é encontrada, retorna-se verdadeiro e encerra-se a execução da busca. |

**Tabela 5.1 - Interações entre as hierarquias para se encontrar uma autorização**

Como pode ser visto acima, as procuras são feitas de forma similar, mudando-se apenas os parâmetros *papel, tipo de operação* e *objeto*. Cada vez que todos os elementos de uma hierarquia são verificados, a procura recomeça do elemento correspondente ao elemento da primeira busca, ou seja, da requisição propriamente dita.

Do primeiro ao quarto passo, são procuradas as autorizações positivas e negativas referentes ao ator e ao papel que ele assume. Se alguma delas for encontrada, não há necessidade da busca por autorizações implícitas.

A partir do quinto passo, o direito de acesso pode ser garantido implicitamente, ou seja, os elementos das hierarquias começam a variar para a realização de novas procuras. A notação *{ator/null}* significa que duas autorizações são procuradas, uma especificando o ator e outra levando em conta somente o seu papel. Por exemplo, a representação *<{ator/null}, professor, +visualização, página web 2>* mostra que as autorizações *<ator, professor, +visualização, página web 2>* e *<null, professor, +visualização, página web 2>* são procuradas. O quinto passo denota a dedução na hierarquia de objetos, pois apenas o parâmetro referente ao objeto é trocado. Já no sexto passo, fica caracterizada a dedução nas hierarquias de tipos de operação e objetos, já que ambos os elementos mudam.

Nos dois últimos passos, a notação *{atores/null}* denota que é procurada, além da autorização do papel (*null*), a autorização individual referente a cada ator que assume o papel *aluno*. Isso porque é necessário que somente um aluno tenha direito de acesso de acordo com o tipo de operação e objeto atuais para que se possa permitir implicitamente o direito de acesso requisitado pelo professor. O sétimo passo mostra a dedução em todas as hierarquias, já que todos os parâmetros mudam. No oitavo passo, a autorização encontrada *<null, aluno, +interação, pw2>* é que garante o direito de acesso. Nesse último ocorre a dedução na hierarquia de objetos.

### 5.4.2.5 Observações Importantes

Os seguintes pontos são de grande relevância e devem ser notados na construção das hierarquias dos domínios de uma autorização:

- *Ordem de definição dos elementos dos domínios de uma autorização*: os objetos a serem protegidos devem ser previamente definidos em relação às operações aplicadas sobre eles. Isso principalmente devido às operações estarem semanticamente ligadas aos objetos, ou seja, antes de se determinar uma operação é

necessário saber se os objetos a suportam ou a requerem. Não existe qualquer restrição de ordem em relação à definição dos papéis.

- *Não-obrigatoriedade da construção das três hierarquias*: não é obrigatória a construção de todas as hierarquias para se encontrar as autorizações implícitas. O algoritmo de busca pode ser adaptado para se encontrar essas autorizações a partir de apenas uma ou duas hierarquias. Por exemplo, se não for possível que objetos de um ambiente de EaD formem a hierarquia de objetos, as autorizações implícitas são deduzidas através das hierarquias de papéis e tipos de operação.

### 5.4.3 Modelagem de Classes

As autorizações coletivas/individuais e as autorizações positivas/negativas necessitam de uma estrutura de classes para que elas possam ser armazenadas. Tal estrutura é mostrada de acordo com a modelagem da Figura 5.6.

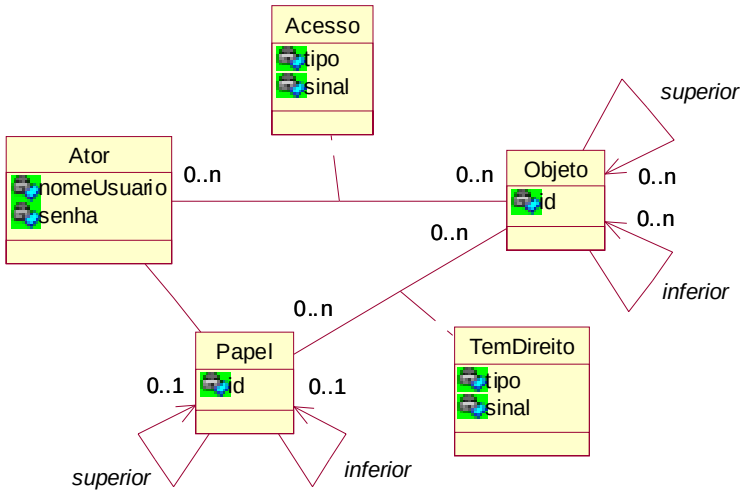



**Figura 5.6 - Estrutura de classes do módulo de autorização**

Existe uma correspondência entre a modelagem acima e a autorização definida na subseção 5.4.1 deste capítulo. A classe *Ator* representa os atores que acessam um ambiente de EaD. A classe *Papel* corresponde aos papéis que o ator assume. A classe *Objeto* armazena os objetos protegidos do ambiente. A classe de relacionamento *Acesso* abstrai as autorizações individuais que um ator tem sobre os objetos, sendo considerada como o primeiro nível de autorizações. O atributo *tipo* representa a operação pertencente a hierarquia de operações a ser executada, enquanto que o atributo *sinal* indica se trata-se de uma autorização positiva ou negativa. A classe de relacionamento *TemDireito* representa as autorizações coletivas, ou seja,

as autorizações que um papel possui, sendo considerada o segundo nível de autorizações. Seus atributos têm os mesmos significados dos atributos anteriores.

De acordo com os relacionamentos, um ator e um papel podem ter direitos de acesso a vários objetos e um objeto pode ser acessado por vários atores e papéis. Os auto-relacionamentos *superior* e *inferior* da classe *Objeto* formam a hierarquia de objetos apresentada anteriormente. Assim, um objeto pode ter zero ou mais objetos superiores e zero ou mais objetos inferiores, sendo que ter zero objetos em um dos dois relacionamentos indica que esse objeto está, respectivamente, no topo ou no nível mais baixo da hierarquia. As mesmas considerações valem para os auto-relacionamentos da classe *Papel*, com a diferença que um papel pode ter zero ou um papel superior e inferior. As cardinalidades do relacionamento entre a classe *Ator* e a classe *Papel* não foram colocadas devido às diferentes modelagens que podem surgir de ambiente de EaD para outro. Tais diferenças são abordadas na seção 5.5.

#### 5.4.3.1 Políticas de Gerenciamento

Algumas políticas são necessárias para evitar situações em que duas autorizações possam ser contraditórias ou redundantes, garantindo a consistência das instâncias da estrutura de classes. As políticas de gerenciamento são:

- *Não-redundância*: não existem duas autorizações com o mesmo ator, papel, tipo de operação e sinal.
- *Não-conflito*: não existem duas autorizações no mesmo nível com o mesmo ator, papel, tipo de operação e diferentes sinais.
- *Sobreposição*: se existirem duas autorizações em níveis distintos que possuam o mesmo ator, papel, tipo de operação e diferentes sinais, aquela que pertencer ao primeiro nível prevalece sobre a do segundo nível.

Cada uma dessas políticas exerce um efeito prático sobre as classes de relacionamento *Acesso* e *TemDireito.*

A política de não-redundância evita a inserção de autorizações iguais, independente se no mesmo nível ou em níveis diferentes. Isso porque se um papel tem direito de acesso sobre um determinado objeto, não há razão para que um ator que assume esse papel tenha uma autorização à parte sobre esse mesmo objeto. Caso a situação contrária ocorra, ou seja, se um

ator tem direito de acesso sobre um objeto e deseja-se inserir o mesmo direito ao papel que esse ator assume, a autorização relacionada ao ator é removida antes da inserção da autorização pertencente ao papel.

A política de não-conflito impede que sejam inseridas no mesmo nível duas autorizações que tenham apenas seus sinais como diferença. Devido a ela, é possível definir se um ator tem ou não tem direito sobre um determinado objeto. Isso porque se fosse possível inserir no mesmo nível uma autorização positiva e uma outra negativa referentes aos mesmos ator/papel, tipo de operação e objeto, não haveria como o módulo de autorização decidir se o direito de acesso ao ator que fez a requisição é garantido.

Finalmente, a política de sobreposição determina que se existir um conflito entre duas autorizações pertencentes a níveis diferentes, a autorização pertencente ao primeiro nível têm precedência sobre a autorização do segundo nível. Essa política define uma das características de facilidade de gerenciamento de autorizações do módulo de autorização citadas no capítulo anterior, em que o número de autorizações inseridas em algumas situações diminui em grande escala (exemplo da subseção 4.2.3.3 do capítulo 4).

### 5.4.4 Algoritmos de Procura, Inserção e Remoção de Autorizações

O módulo de autorização oferece três operações que são executadas sobre as autorizações: Procura, inserção e remoção, sendo que cada uma delas segue rigorosamente as políticas de gerenciamento apresentadas anteriormente. Os respectivos algoritmos dessas operações são mostrados a seguir.

#### 5.4.4.1 Algoritmo de Procura de uma Autorização

A partir das interações das hierarquias de papéis, tipos de operação e objetos (subseção 5.4.2.4 deste capítulo), o algoritmo de procura é constituído para verificar se um determinado ator tem ou não tem o direito de acesso requisitado sobre um objeto. Tal algoritmo pode ser visto abaixo:

```
01: procurar por uma autorização individual igual à requisição;
02: se encontrar
03: .  retornar verdadeiro
04: senão
05: .  procurar por uma autorização individual com o sinal contrário ao da requisição;
06: .  se encontrar
07: .  .  retornar falso
08: .  senão
09: .  .  procurar por uma autorização coletiva igual à requisição;
10: .  .  se encontrar
```

```
11: . . . retornar verdadeiro
12: . . senão
13: . . . procurar por uma autorização coletiva com sinal contrário ao da requisição;
14: . . . se encontrar
15: . . . . retornar falso
16: . . . senão
17: . . . . mudar para o objeto superior da hierarquia de objetos;
18: . . . . enquanto houver papéis na hierarquia de papéis faça
19: . . . . . enquanto houver operações na hierarquia de tipos de operação faça
20: . . . . . . enquanto houver objetos na hierarquia de objetos faça
21: . . . . . . . procurar por uma autorização coletiva com os parâmetros atuais;
22: . . . . . . . se encontrar
23: . . . . . . . . retornar verdadeiro
24: . . . . . . . fim se;
25: . . . . . . . se o papel atual for igual ao papel da requisição
26: . . . . . . . . procurar por uma autorização individual com os parâmetros atuais
27: . . . . . . . . se encontrar
28: . . . . . . . . . retornar verdadeiro
29: . . . . . . . . fim se;
30: . . . . . . . senão
31: . . . . . . . . enquanto houver atores que assumem o papel atual faça
32: . . . . . . . . . procurar por uma autorização individual com os parâmetros atuais;
33: . . . . . . . . . se encontrar
34: . . . . . . . . . . retornar verdadeiro
35: . . . . . . . . . fim se;
36: . . . . . . . . fim enquanto;
37: . . . . . . . fim se;
38: . . . . . . . mudar para o objeto superior da hierarquia de objetos;
39: . . . . . fim enquanto;
40: . . . . . mudar para a operação superior da hierarquia de tipo de operação;
41: . . . . . reiniciar a hierarquia de objetos no objeto da requisição;
42: . . . . fim enquanto;
43: . . . . mudar para o papel inferior na hierarquia de papéis;
44: . . . . reiniciar a hierarquia de tipos de operação na operação da requisição;
45: . . . fim enquanto;
46: . . . retornar falso;
47: . . fim se;
48: . fim se;
49: fim se;
50: fim se;
```

**Algoritmo 5.1 - Busca de uma autorização**

Em todo o algoritmo, as procuras pelas autorizações individuais são feitas na classe *Acesso*, enquanto que as procuras pelas autorizações coletivas são feitas na classe *TemDireito*.

Inicialmente, as procuras por autorizações individuais (linha 1 até linha 7) são realizadas primeiro devido à política de sobreposição. As duas procuras seguintes (linha 9 até linha 15) se referem às autorizações coletivas, ou seja, o papel que o ator da requisição assume. Caso uma autorização positiva seja encontrada, é retornado verdadeiro (linhas 3 e 11). Se uma autorização negativa for encontrada, é retornado falso (linhas 7 e 15). Essas procuras iniciais representam, respectivamente, os quatro primeiros passos da Tabela 5.1.

Após elas, é necessário mudar-se o objeto da requisição para o próximo objeto da hierarquia de objetos (linha 17) para que o direito de acesso possa ser dado implicitamente caso uma autorização seja encontrada.

Os laços de iteração (linha 18 até linha 20) juntamente com as mudanças dos parâmetros (linhas 38, 40, 41, 43 e 44) representam as interações entre as hierarquias de papéis, tipos de operação e objetos como visto do quinto até o oitavo passo da Tabela 5.1.

Primeiramente, uma procura por uma autorização coletiva referente aos parâmetros papel, tipo de operação e objeto atuais é realizada (linha 21). Retorna-se verdadeiro caso ela seja encontrada (linha 23). Nesse ponto, o papel atual ainda é o papel da requisição (linha 25) e, devido a isso, faz-se necessária a procura de uma autorização individual referente ao ator da requisição e o tipo de operação e objeto atuais (linha 26).

Após a primeira mudança na hierarquia de papéis, deve-se verificar a existência de uma autorização individual (linha 32) para cada ator que assume o papel atual (linha 31). Como esses atores assumem um papel inferior ao papel que o ator da requisição assume, o direito de acesso é implicitamente garantido ao ator (linha 34) se qualquer um desses atores possuir uma autorização de acordo com o tipo de operação e objeto atuais.

Terminadas todas interações entre as hierarquias, retorna-se falso (linha 46) devido a nenhuma autorização ter sido encontrada.

### 5.4.4.2 Algoritmo de Inserção de uma Autorização

O algoritmo de inserção de uma autorização é o que mais está relacionado com as políticas de gerenciamento e, devido a esse fato, é o algoritmo que, após sua implementação, requer mais tempo de processamento quando executado. Ele foi elaborado para suportar as possíveis combinações entre os conceitos de autorizações coletivas/individuais, autorizações positivas/negativas e autorizações explícitas/implícitas. Tal algoritmo pode ser visto abaixo:

```
01: se a autorização a ser inserida for individual
02: .  procurar por uma autorização individual igual à autorização a ser inserida;
03: .  se encontrar
04: .  .  retornar falso
05: .  senão
06: .  .  procurar por uma autorização individual com sinal contrário à autorização a ser inserida;
07: .  .  se encontrar
08: .  .  .  retornar falso
09: .  .  fim se;
10: .  fim se;
11: fim se;
```

```
12: procurar por uma autorização coletiva igual à autorização a ser inserida;
13: se encontrar
14: .  retornar falso
15: fim se;
16: se a autorização a ser inserida for coletiva
17: .  procurar por uma autorização coletiva com sinal contrário à autorização a ser inserida;
18: .  se encontrar
19: .  .  retornar falso
20: .  fim se;
21: fim se;
22: se a autorização a ser inserida for positiva
23: .  mudar para a operação superior da hierarquia de tipos de operação;
24: .  enquanto houver papéis na hierarquia de papéis faça
25: .  .  enquanto houver operações na hierarquia de tipos de operação faça
26: .  .  .  enquanto houver objetos na hierarquia de objetos faça
27: .  .  .  .  procurar por uma autorização coletiva com os parâmetros atuais;
28: .  .  .  .  se encontrar
29: .  .  .  .  .  retornar falso
30: .  .  .  .  fim se;
31: .  .  .  .  se o papel atual for igual ao papel da autorização que se deseja inserir
32: .  .  .  .  .  se a autorização a ser inserida for individual
33: .  .  .  .  .  .  procurar por uma autorização individual com os parâmetros atuais;
34: .  .  .  .  .  .  se encontrar
35: .  .  .  .  .  .  .  retornar falso
36: .  .  .  .  .  .  fim se;
37: .  .  .  .  .  fim se;
38: .  .  .  .  senão
39: .  .  .  .  .  enquanto houver atores que assumam o papel atual faça
40: .  .  .  .  .  .  procurar por uma autorização individual com os parâmetros atuais;
41: .  .  .  .  .  .  se encontrar
42: .  .  .  .  .  .  .  retornar falso
43: .  .  .  .  .  .  fim se;
44: .  .  .  .  .  fim enquanto;
45: .  .  .  .  fim se;
46: .  .  .  .  mudar para o objeto superior da hierarquia de objetos;
47: .  .  .  fim enquanto;
48: .  .  .  mudar para a operação superior da hierarquia de tipo de operação;
49: .  .  .  reiniciar a hierarquia de objetos no objeto da autorização a ser inserida;
50: .  .  fim enquanto;
51: .  .  mudar para o papel inferior na hierarquia de papéis;
52: .  .  reiniciar a hierarquia de tipos de operação na operação da autorização a ser inserida;
53: .  fim enquanto;
54: .  se a autorização a ser inserida for individual
55: .  .  reiniciar a hierarquia de papéis no papel superior no papel da autorização a ser inserida
56: .  senão
57: .  .  reiniciar a hierarquia de papéis no papel da autorização a ser inserida
58: .  fim se;
59: .  enquanto houver papéis na hierarquia de papéis faça
60: .  .  enquanto houver operações na hierarquia de tipos de operação faça
61: .  .  .  enquanto houver objetos na hierarquia de objetos faça
62: .  .  .  .  se o papel atual for diferente do papel da autorização a ser inserida
63: .  .  .  .  .  procurar por uma autorização coletiva com os parâmetros atuais;
64: .  .  .  .  .  se encontrar
65: .  .  .  .  .  .  desativar a autorização
66: .  .  .  .  .  senão
67: .  .  .  .  .  .  enquanto houver atores que assumam o papel atual faça
68: .  .  .  .  .  .  .  procurar por uma autorização individual com os parâmetros atuais;
69: .  .  .  .  .  .  .  se encontrar
70: .  .  .  .  .  .  .  .  desativar a autorização
71: .  .  .  .  .  .  .  fim se;
```

72: . . . . . . **fim enquanto;**
73: . . . . . **fim se;**
74: . . . . **senão**
75: . . . . . **enquanto** houver atores que assumam o papel atual **faça**
76: . . . . . procurar por uma autorização individual com os parâmetros atuais;
77: . . . . . **se** encontrar
78: . . . . . . desativar a autorização
79: . . . . . **fim se**;
80: . . . . . **fim enquanto;**
81: . . . . **fim se**;
82: . . . . mudar para o objeto inferior da hierarquia de objetos;
83: . . . **fim enquanto**;
84: . . . mudar para a operação inferior da hierarquia de tipo de operação;
85: . . . reiniciar a hierarquia de objetos ao objeto da autorização a ser inserida;
86: . . **fim enquanto**;
87: . . mudar para o papel superior na hierarquia de papéis;
88: . . reiniciar a hierarquia de tipos de operação à operação da autorização a ser inserida;
89: . **fim enquanto**;
90: **fim se;**
91: inserir a autorização;
92: **retornar** verdadeiro;

**Algoritmo 5.2 - Inserção de uma autorização**

De acordo com o algoritmo, algumas procuras são previamente necessárias antes de se inserir uma autorização para que as políticas de gerenciamento sejam mantidas. Caso a autorização a ser inserida seja individual (linha 1), procura-se por uma autorização igual a ela (linha 2) e por uma outra com sinal contrário (linha 6) ao sinal dela. Essas duas procuras são realizadas para se conservar no primeiro nível de autorizações, respectivamente, a política de não-redundância e a política de não-conflito. A procura seguinte (linha 12) mantém a política de não-redundância tanto para o primeiro quanto para o segundo nível de autorizações. Se a autorização a ser inserida é coletiva (linha 16), é necessário procurar pela autorização com sinal contrário ao seu sinal (linha 17). Essa checagem conserva a política de não-conflito para o segundo nível de autorizações. Caso qualquer uma dessas autorizações seja encontrada, retorna-se falso (linhas 4, 8, 14 e 19).

Nesse ponto, se a autorização a ser inserida for negativa, basta realizar a inserção (linha 91) e o algoritmo é finalizado retornando-se verdadeiro (linha 92). Caso contrário, uma nova série de averiguações são necessárias para se manter a consistência entre as autorizações implícitas e a autorização que se deseja inserir.

Após as primeiras procuras é necessário mudar da operação da requisição para a próxima operação da hierarquia de operações (linha 23). Da mesma forma que no algoritmo de busca, os laços de iteração (linha 24 até linha 26) juntamente com as mudanças dos

parâmetros (linhas 46, 48, 49, 51 e 52) representam as interações entre as hierarquias de papéis, tipos de operação e objetos.

Durante cada iteração, uma procura por uma autorização coletiva é realizada (linha 27) independentemente se a autorização a ser inserida é individual ou coletiva. A inserção não é permita caso essa autorização seja encontrada (linha 29).

Caso o papel atual seja igual ao papel original[7] (linha 31) e se essa autorização for individual, uma procura referente ao ator que consta na autorização a ser inserida e o tipo de operação e objeto atuais é feita (linha 33). Se a autorização for encontrada, a inserção é proibida (linha 35). A última condição (linha 32) é necessária porque a inserção da nova autorização não deve ser realizada se existir uma autorização individual que dê direito a esse ator sobre um objeto pertencente a um nível superior da hierarquia de objetos, pois a primeira pode ser deduzida da segunda.

Após a primeira mudança na hierarquia de papéis (linha 38), deve-se verificar a existência de uma autorização individual (linha 40) para cada ator que assume o papel atual (linha 39). Como esses atores assumem um papel inferior ao papel original, a inserção não é permitida (linha 42) se qualquer um desses atores possuir uma autorização de acordo com o tipo de operação e objeto atuais.

Ao término dos laços de iteração, todas as verificações de redundância e contrariedade foram feitas, mas antes da inserção é necessário checar se não há nenhum conflito entre a autorização a ser inserida e alguma autorização coletiva ou individual pertencente a um papel igual ou superior ao papel original. Isso porque a autorização a ser inserida permite os mesmos direitos implicitamente aos papéis superiores e, caso um desses já possua uma autorização equivalente, torna-se necessário desativá-la, ou seja, armazená-la em um local onde os algoritmos de busca e de inserção não detectem sua presença. O único algoritmo que trabalha diretamente sobre essas autorizações desativadas é o de remoção de uma autorização. Maiores detalhes sobre esse assunto são dados na próxima subseção.

O ponto em que a hierarquia de papéis deve ser reiniciada deve ser o primeiro item a ser reconsiderado. Caso a autorização a ser inserida seja individual (linha 54), a hierarquia de papéis deve ser reiniciada ao papel superior do papel original devido a todas as checagens de consistência já terem sido realizadas (linha 1 até linha 15). Se a autorização a ser inserida for

[7] Entende-se por ‘original’ o papel ou o tipo de operação ou o objeto da autorização a ser inserida.

coletiva (linha 56), a hierarquia de papéis deve ser reiniciada ao papel original devido à possibilidade de existência de autorizações individuais que entrem em conflito com essa autorização coletiva. Dessa forma, pode haver autorizações individuais que se tornam redundantes após a inserção. Para ilustrar essa situação, considera-se um exemplo em que existe a autorização *<felipe, professor, +visualização, página web 2>* e se deseja inserir a autorização *<null, professor, + visualização, página web 2>*. Essa operação é possível, pois é requerida uma mudança da situação em que o ator *felipe* tem o direito de *visualização* sobre o objeto *página web 2* para a situação em que todos os professores possuem esse mesmo direito. Como esse caso pode ocorrer apenas quando o papel da autorização coletiva a ser inserida é igual ao papel das autorizações individuais, é necessário que a hierarquia de papéis seja reiniciada no papel original para que se façam tais verificações de redundância.

Novamente são necessárias procuras de autorizações utilizando-se as hierarquias de papéis, tipos de operação e objetos (linha 59 até linha 61). Se forem encontradas autorizações coletivas ou individuais (linhas 64, 69 e 76), elas são desativadas (linhas 65, 70 e 77). O teste para verificar a igualdade entre o papel atual e papel da autorização a ser inserida (linha 62) é necessário somente para evitar uma procura, já realizada anteriormente, de uma autorização coletiva. Deve-se notar que todas as mudanças de parâmetros (linhas 82, 84 e 87) ocorrem de maneira inversa às anteriores (linhas 46, 48 e 51) devido à necessidade de se verificar autorizações com papéis superiores ao papel original, tipos de operação inferiores ao tipo de operação original e objetos inferiores ao objeto original.

Após todas as desativações, a autorização é inserida (linha 91), retorna-se verdadeiro (linha 92) e o algoritmo é finalizado.

### 5.4.4.3 Algoritmo de Remoção de uma Autorização

Como dito anteriormente (subseção 4.2.3.3 do capítulo 4), caso não se deseje uma autorização implícita, deve-se inserir uma autorização negativa correspondente. Já no caso das autorizações explícitas, o algoritmo de remoção é utilizado. Tal algoritmo pode ser visualizado abaixo:

```
001: procurar pela autorização a ser removida;
002: se for encontrada
003: .  se a autorização a ser removida for positiva
004: .  .  enquanto houver autorizações a serem reativadas faça
005: .  .  .  iniciar a hierarquia de papéis no papel superior ao papel da autorização original;
006: .  .  .  iniciar a hierarquia de tipos de operação na operação superior à operação da
     .  .  .  autorização original;
007: .  .  .  enquanto houver papéis inferiores ao papel da autorização a ser reativada faça
```

```
008: . . . . enquanto houver operações na hierarquia de tipos de operação faça
009: . . . . . enquanto houver objetos na hierarquia de objetos faça
010: . . . . . . procurar por uma autorização coletiva com os parâmetros atuais;
011: . . . . . . se encontrar
012: . . . . . . . associá-la com a autorização que seria reativada;
013: . . . . . . . interromper a iteração atual;
014: . . . . . . fim se;
015: . . . . . . enquanto houver atores que assumam o papel atual faça
016: . . . . . . . procurar por uma autorização individual com os parâmetros atuais;
017: . . . . . . . se encontrar
018: . . . . . . . . associá-la com a autorização que seria reativada;
019: . . . . . . . . interromper a iteração atual;
020: . . . . . . . fim se;
021: . . . . . . fim enquanto;
022: . . . . . . se a autorização da procura anterior foi encontrada
023: . . . . . . . interromper a iteração atual;
024: . . . . . . fim se;
025: . . . . . . mudar para o objeto superior da hierarquia de objetos;
026: . . . . . fim enquanto;
027: . . . . . se a autorização da procura anterior foi encontrada
028: . . . . . . interromper a iteração atual;
029: . . . . . fim se;
030: . . . . . mudar para a operação superior da hierarquia de tipos de operação;
031: . . . . . reiniciar a hierarquia de objetos no objeto da autorização original;
032: . . . . fim enquanto;
033: . . . . se a autorização da procura anterior foi encontrada
034: . . . . . interromper a iteração atual;
035: . . . . fim se;
036: . . . . mudar para o papel superior da hierarquia de papéis;
037: . . . . reiniciar a hierarquia de tipos de operação na operação superior à operação da
     . . . . autorização original;
038: . . . fim enquanto;
039: . . . se não houve nenhuma associação
040: . . . . se a autorização a ser reativada é individual
041: . . . . . procurar por uma autorização individual com sinal contrário à autorização a ser
     . . . . . reativada;
042: . . . . . se não encontrar
043: . . . . . . iniciar a hierarquia de tipos de operação na operação superior à operação da
     . . . . . autorização a ser reativada;
044: . . . . . . iniciar a hierarquia de objetos no objeto da autorização a ser reativada;
045: . . . . . . enquanto houver operações na hierarquia de tipos de operações faça
046: . . . . . . . enquanto houver objetos na hierarquia de objetos faça
047: . . . . . . . . procurar por uma autorização individual com o papel da autorização a ser
     . . . . . . . . reativada e com a operação e o objeto atuais;
048: . . . . . . . . se encontrar
049: . . . . . . . . . associar a autorização encontrada com a autorização que seria
     . . . . . . . . . reativada;
050: . . . . . . . . . interromper a interação atual;
051: . . . . . . . . fim se;
052: . . . . . . . . mudar para o objeto superior da hierarquia de objetos;
053: . . . . . . . fim enquanto;
054: . . . . . . . se a autorização da procura anterior foi encontrada
055: . . . . . . . . interromper a iteração atual;
056: . . . . . . . fim se;
057: . . . . . . . mudar para a operação superior da hierarquia de tipos de operação;
058: . . . . . . . reiniciar a hierarquia de objetos no objeto da autorização a ser reativada;
059: . . . . . . fim enquanto;
060: . . . . . senão
061: . . . . . . apagar a autorização que seria reativada
062: . . . . . fim se;
```

```
063: . . . . senão
064: . . . . . procurar por uma autorização coletiva com sinal contrário à autorização a ser
     . . . . . reativada;
065: . . . . . se encontrar
066: . . . . . . apagar a autorização que seria reativada
067: . . . . . fim se;
068: . . . . fim se;
069: . . . . se  a autorização da procura anterior não foi encontrada ***
070: . . . . . iniciar a hierarquia de tipos de operação na operação superior à operação da
     . . . . . autorização a ser reativada;
071: . . . . . enquanto houver operações na hierarquia de tipos de operações faça
072: . . . . . . enquanto houver objetos na hierarquia de objetos faça
073: . . . . . . . procurar por uma autorização coletiva com o papel da autorização a ser
     . . . . . . . reativada e com a operação e o objeto atuais;
074: . . . . . . . se encontrar
075: . . . . . . . . associar a autorização encontrada com a autorização que seria
     . . . . . . . . reativada;
076: . . . . . . . . interromper a interação atual;
077: . . . . . . . fim se;
078: . . . . . . . mudar para o objeto superior da hierarquia de objetos;
079: . . . . . . fim enquanto;
080: . . . . . . se a autorização da procura anterior foi encontrada
081: . . . . . . . interromper a iteração atual;
082: . . . . . . fim se;
083: . . . . . . mudar para a operação superior da hierarquia de tipos de operação;
084: . . . . . . reiniciar a hierarquia de objetos no objeto da autorização a ser reativada;
085: . . . . . fim enquanto;
086: . . . . . se a autorização da procura anterior não foi encontrada
087: . . . . . . reativar autorização
088: . . . . . fim se;
089: . . . . fim se;
090: . . . fim se;
091: . . fim enquanto;
092: . senão
093: . . se a autorização a ser removida for coletiva
094: . . . enquanto houver autorizações individuais a serem reativadas faça
095: . . . . reativar autorização;
096: . . . fim enquanto;
097: . . fim se;
098: . fim se;
099: . remover autorização;
100: . retornar verdadeiro;
101: senão
102: . retornar falso;
103: fim se;
```

**Algoritmo 5.3 - Remoção de uma autorização**

A primeira verificação realizada é se a autorização a ser removida existe (linha 1 e 2). Caso ela não seja encontrada, retorna-se falso (linha 102) e o algoritmo é encerrado. Se ela existir e for positiva (linha 3), antes de removê-la é necessário verificar se alguma outra autorização não precisa ser reativada. Esse fato se deve à operação de inserção, em que uma nova autorização pode se tornar redundante em relação a uma existente. Assim, essa última autorização deve ser armazenada em uma estrutura auxiliar (desativação) e recuperada (reativação) caso a autorização que foi inserida seja removida depois de algum tempo.

Para cada autorização (linha 4) que tem possibilidade de ser reativada é preciso executar alguns testes para se checar a consistência da sua restauração. Pode não ser admissível reativar uma autorização que foi desativada, dependendo das ações tomadas no espaço de tempo entre a inserção da autorização que desencadeou tais desativações e a sua posterior remoção. Por exemplo, adotando-se as hierarquias apresentadas anteriormente (subseção 5.4.2.4 deste capítulo), considera-se uma situação inicial em que o administrador de curso possui a autorização *<null, administrador de curso, +visualização, página web 2>*. Em seguida deseja-se inserir a autorização *<null, aluno, +visualização, página web 2>*, que dá implicitamente o mesmo direito de acesso que o administrador de curso já possui. Dessa forma, a autorização *<null, administrador de curso, +visualização, página web 2>* precisa ser movida (desativada) para uma estrutura auxiliar que a relaciona à autorização inserida, sendo possível sua reativação caso a última venha a ser removida. Supõe-se agora que a autorização *<null, professor, +interação, página web 1>* também é inserida sem problemas, já que não ocorre nenhum conflito ou redundância entre ela e a autorização inserida anteriormente. A partir da remoção da autorização *<null, aluno, +visualização, página web 2>* é esperado que a autorização referente ao administrador de curso seja reativada, mas isso não ocorre devido à existência da autorização *<null, professor, +interação, página web 1>*, que implicitamente permite o mesmo direito de acesso. Assim, é necessário que a autorização *<null, administrador de curso, +visualização, página web 2>* permaneça na estrutura auxiliar e esteja relacionada com a autorização que não permitiu sua reativação após a operação de remoção. Caso a autorização *<null, professor, +interação, página web 1>* seja removida, a autorização original é restaurada, voltando-se para o estado inicial do exemplo.

Primeiramente, as hierarquias devem ser iniciadas (linha 5 e 6) com os elementos pertinentes, evitando-se buscas desnecessárias. Dessa forma, os papéis utilizados nas procuras estão localizados na hierarquia de papéis entre o papel da autorização a ser removida e o papel da autorização atual que tem possibilidade de ser reativada.

Os laços de iteração (linha 7 até linha 9) juntamente com as mudanças dos parâmetros (linhas 2, 30, 31, 36 e 37) representam as interações entre as hierarquias de papéis, tipos de operação e objetos.

Nas iterações procura-se por autorizações coletivas (linha 10) e por autorizações individuais (linha 16). Caso qualquer uma delas seja encontrada, associa-se a ela a autorização que seria restaurada (linhas 12 e 18), os laços de iteração são interrompidos (linhas 23, 28 e

34) e a busca é novamente iniciada (linha 4) para a próxima autorização que tem possibilidade de ser reativada.

Caso não tenha existido nenhuma associação durante as iterações (linha 39), ainda é necessário verificar a consistência referente ao ator/papel que possui a autorização desativada anteriormente pela operação de inserção.

Se a autorização a ser recuperada for individual (linha 40), é realizada uma busca por uma autorização negativa (linha 41) correspondente a ela. Caso seja encontrada (linha 42), a autorização que seria restaurada é eliminada definitivamente (linha 61) devido à mudança ocorrida referente ao direito de acesso do ator/papel, ou seja, ele tinha direito ao objeto protegido antes da autorização ser desativada, mas atualmente esse direito foi bloqueado explicitamente.

Se a autorização negativa não for localizada, procura-se por uma autorização (linha 47) que pode dar implicitamente o mesmo direito de acesso. Se tal autorização for encontrada, ela é associada com a autorização que seria reativada (linha 49).

A procura por uma autorização coletiva negativa (linha 64) correspondente à autorização a ser reativada é necessária caso essa última não seja individual (linha 40). Se ela for encontrada (linha 65), a autorização que seria restaurada é eliminada (linha 66).

Se nenhuma autorização foi encontrada em qualquer uma das três buscas anteriores (linhas 41, 47 e 54), faz-se necessário procurar por uma autorização coletiva (linha 73) que pode dar implicitamente o mesmo direito de acesso da autorização a ser reativada. Caso seja localizada (linha 74), ela é associada com a autorização que seria reativada (linha 75), caso contrário, a reativação é efetivada (linha 87).

Se a autorização a ser removida for coletiva (linha 93) e negativa (linha 92), basta apenas restaurar as autorizações (linha 95) enquanto elas existirem (linha 94). Isso porque as autorizações reativadas tratam-se apenas de autorizações individuais negativas que foram movidas devido à inserção da autorização coletiva negativa.

### 5.4.5 Relação entre os Conceitos Utilizados pelo Módulo de Autorização

Os conceitos de autorização coletiva/individual, autorização positiva/negativa e autorização explícita/implícita, utilizados pelo módulo de autorização, não são dependentes entre si em relação à implementação. Isso significa que a utilização de um desses conceitos

não implica na utilização de outro, trazendo mais flexibilidade caso o desenvolvedor não se interesse por algum deles.

O impacto da mudança pode recair sobre a estrutura de classes do módulo de autorização e/ou sobre os algoritmos de procura, inserção e remoção de uma autorização.

Caso não se utilize o conceito de autorizações explícitas/implícitas, os algoritmos têm que ser modificados de forma a executarem somente buscas diretas. Na prática, isso significa que os laços de iteração de todos os algoritmos são eliminados, simplificando muito suas implementações.

Se o conceito de autorização coletiva/individual não for usado, somente uma das classes que representa os usuários do ambiente de EaD deve restar. A outra é eliminada. Por exemplo, se o ambiente tratar os usuários individualmente, a classe *Ator* permanece na estrutura de classes, caso contrário, a classe *Papel* é mantida. Além disso, os algoritmos devem ser modificados a fim de realizar buscas somente individuais ou coletivas.

Finalmente, caso não se empregue o conceito de autorização positiva/negativa, o atributo *sinal* pertencente às classes de relacionamento *Acesso* e *TemDireito* são removidos. Nos algoritmos devem ser eliminadas as referências às autorizações negativas, já que são consideradas apenas as autorizações positivas.

## *5.5 Estrutura do Banco de Dados de Segurança*

Um banco de dados é utilizado pelos módulos para a inserção e recuperação de informações referentes à autenticação e à autorização. Sua estrutura é formada pela junção das diversas classes e relacionamentos apresentados até esse ponto, de acordo com a Figura 5.6 mostrada anteriormente. Deve-se lembrar que faltam as classes necessárias para os módulos monitor e auditor, mas, como dito antes, esses módulos foram deixados como trabalhos futuros e, conseqüentemente, a modelagem de suas classes também.

A integração do BD de segurança com o BD de um ambiente de EaD depende da análise de algumas particularidades que variam de um ambiente para outro. São elas:

- De que maneira um ator assume os diferentes papéis frente aos cursos que participa;
- Verificar se os objetos pertencem a um curso em particular ou ao ambiente de EaD.

Essas particularidades são verificadas nas próximas seções.

### 5.5.1 Relacionamento entre Atores e Papéis de um Ambiente de EaD

A forma que um ator pode associar-se a diferentes papéis depende da maneira que o ambiente de EaD disponibiliza seus cursos para o público:

- Um ator pode freqüentar vários cursos, assumindo um papel diferente em cada um.
- Um ator pode freqüentar apenas um curso, assumindo um único papel.

Do ponto de vista prático, a primeira forma de disponibilização de cursos é mais eficiente, pois somente com um *login*, o ator recupera todos os cursos que ele administra, que ele está matriculado como aluno, etc. Esse caso é modelado abaixo na Figura 5.7.

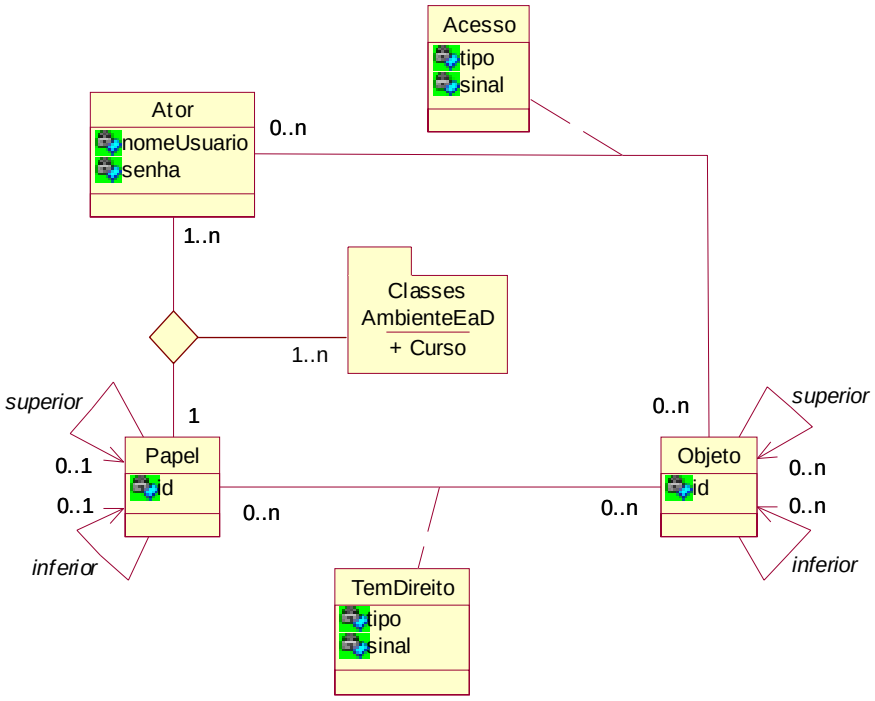



**Figura 5.7 - Relacionamento Ator / Papel - 1º Caso**

De maneira contrária, na segunda forma, cada vez que o usuário desejar mudar de curso é necessário um novo *login*. Assim, ele é considerado como um ator diferente para cada curso. Esse caso é modelado na Figura 5.8.

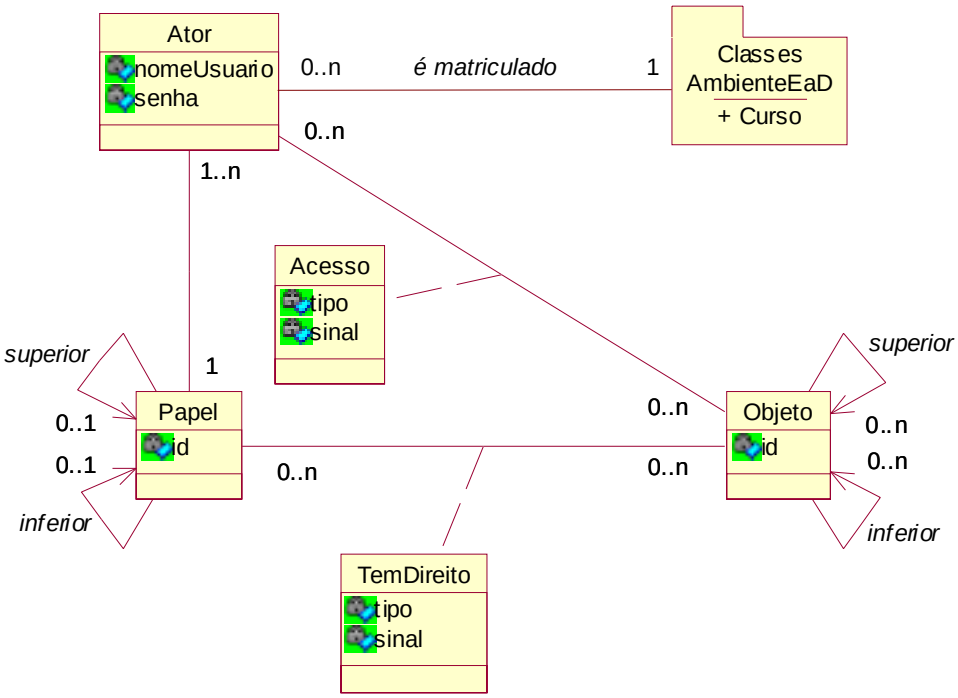



**Figura 5.8 - Relacionamento Ator / Papel - 2º Caso**

Em ambas as figuras, a classe + *Curso* representa os cursos disponibilizados por um ambiente EaD. Ela é envolvida pelo pacote[8] *ClassesAmbienteEaD* para deixar claro que sua especificação não é determinada diretamente pela arquitetura como o restante das classes, ou seja, ela faz parte do BD geral do ambiente e é utilizada pelo BD de segurança.

As diferenças entre as duas figuras anteriores é que na Figura 5.7 existe um relacionamento ternário entre as classes *Ator*, *Papel* e *Curso* para que a primeira forma de disponibilização de cursos seja obtida. Isso implica na existência de apenas uma instância da classe *Ator* relacionada com diversas instâncias da classe *Curso*, sendo que cada par de instâncias ator/curso se relaciona com uma instância da classe *Papel*. Já na Figura 5.8 esse relacionamento ternário é transformado em um relacionamento binário entre as instâncias de *Ator* e *Curso*, permitindo a segunda forma de disponibilização. Nesse caso uma instância da classe *Ator* relaciona-se com somente uma instância das classes *Curso* e *Papel*.

### 5.5.2 Propriedade dos Objetos

Os objetos protegidos pela arquitetura de segurança podem pertencer aos cursos em particular ou pertencer ao ambiente de EaD. Exemplos respectivos a essas duas situações são: as páginas de um curso em que navegam os atores nele inscritos e as ferramentas que um ambiente disponibiliza. A modelagem da primeira abordagem é visualizada na Figura 5.9.

[8] Um pacote representa um conjunto de classes que se relacionam por compartilharem objetivos em comum.

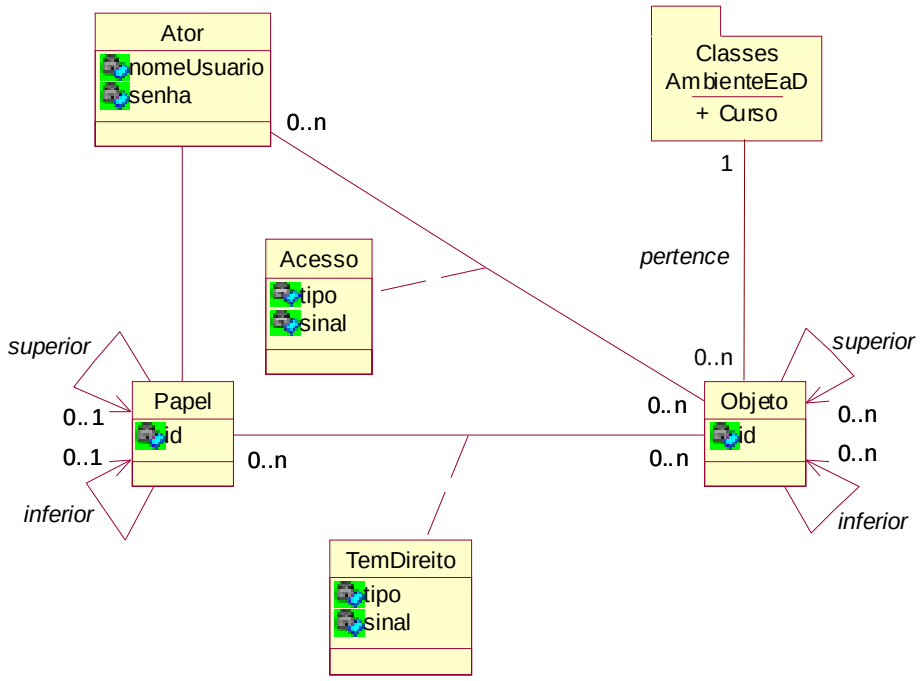



**Figura 5.9 - Objetos protegidos pertencentes aos cursos**

Como visto na figura, é necessário apenas um relacionamento entre as classes *+Curso* e *Objeto* para que um curso seja proprietário dos objetos protegidos. De acordo com esse relacionamento, um curso pode ter zero ou mais objetos protegidos, enquanto que um objeto pode pertencer somente a um curso.

A segunda abordagem é conseguida simplesmente eliminando-se o relacionamento anterior, restando apenas as classes *Ator*, *Papel* e *Objeto* juntamente com suas classes de relacionamento *Acesso* e *TemDireito*.

### 5.5.3 Exemplo de Integração entre os Diferentes Casos

A partir desse ponto é mostrado um exemplo da integração dos conceitos apresentados pelas duas subseções anteriores. Assim, deseja-se aplicar a arquitetura de segurança em um ambiente de EaD que permite um ator assumir diferentes papéis em diferentes cursos, sendo um papel por curso. Nota-se também que os objetos protegidos pertencem ao ambiente devido se tratar das ferramentas que são disponibilizadas aos diversos atores que freqüentam os diversos cursos existentes. Dessa forma, a modelagem para esses requerimentos fica de acordo com a figura abaixo.

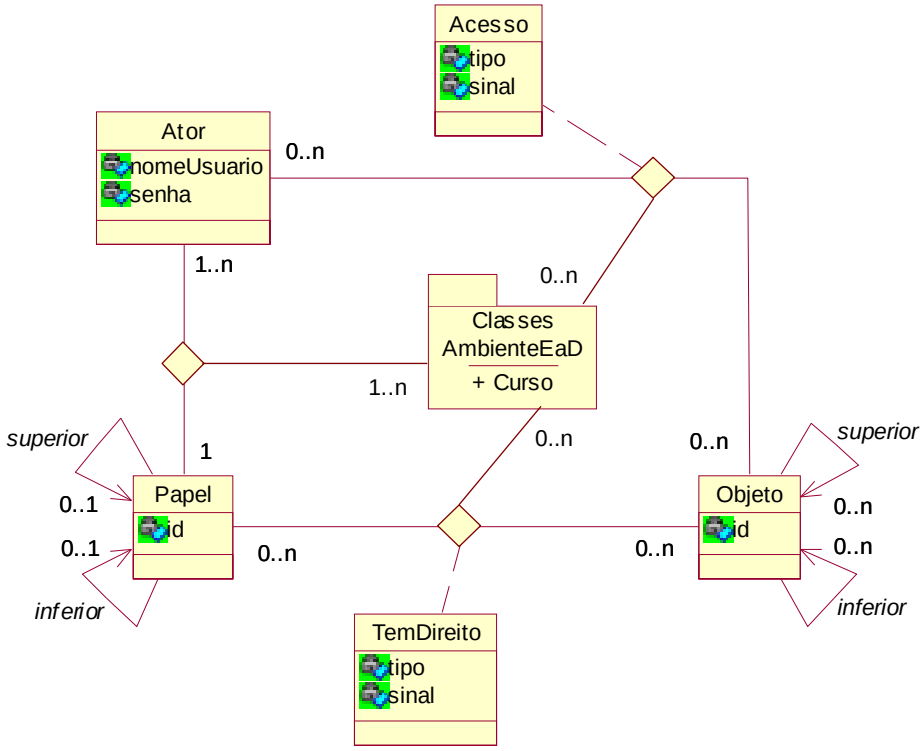



**Figura 5.10 - Integração dos conceitos anteriores**

A Figura 5.10 mostra o mesmo relacionamento ternário entre as classes *Ator*¸ *Papel* e *+Curso*. Dessa forma, como visto na Figura 5.7, é tratado o requerimento de um ator poder assumir diferentes papéis, sendo um papel para cada curso.

Como os objetos não pertencem a cursos específicos, nenhum relacionamento é criado entre as classes *+Curso* e *Objeto*. Devido a esse fato, é necessário controlar a qual curso uma autorização coletiva/individual pertence. Os relacionamentos ternários entre as classes *Ator/Papel*, *+Curso* e *Objeto* são responsáveis por essa funcionalidade. Dessa forma, um ator/papel nunca terá direito sobre um objeto duas vezes no mesmo curso.

## *5.6 Considerações Finais*

Neste capítulo foram abordados os aspectos de implementação mais importantes para a aplicação da arquitetura de segurança. As soluções foram apresentadas de forma independente de plataformas de hardware/software específicas, sendo essa atitude tomada para se direcionar a implementação de uma maneira mais flexível. Algumas considerações foram feitas sobre o módulo de autenticação, mas os esforços foram principalmente direcionados ao módulo de autorização devido à dificuldade de sua implementação.

No próximo capítulo é visto um estudo de caso em que a arquitetura de segurança é utilizada em um ambiente de EaD.

# Capítulo 6
# Estudo de Caso

## *6.1 Considerações Iniciais*

Neste capítulo é apresentado um estudo de caso em que a arquitetura de segurança é aplicada no ambiente DE-MAW[9] (Distance Education-Multimedia Application Web-builder). Para isso são considerados os aspectos de implementação apresentados anteriormente.

A seção 6.2 apresenta o projeto do ambiente DE-MAW, trazendo sua arquitetura e as funcionalidades de cada módulo que a compõe. Na seção 6.3 é abordada a plataforma de software utilizada no desenvolvimento do estudo de caso. A seção 6.4 mostra as ferramentas criadas a partir dos módulos de autenticação e autorização, fazendo uma descrição de suas interfaces. Na seção 6.5 são apresentadas as conclusões deste capítulo.

## *6.2 Ambiente DE-MAW*

O ambiente DE-MAW tem suas origens no projeto AMMO (Authoring and Manipulation of Multimedia Objects), utilizando toda a potencialidade de recursos de armazenamento, recuperação e autoria de aplicações multimídia que esse último oferece. Sendo assim, uma breve descrição do projeto AMMO é realizada para uma melhor compreensão do ambiente DE-MAW.

### 6.2.1 Projeto AMMO

O principal objetivo do AMMO é fornecer suporte para a criação e manipulação local ou remota de objetos multimídia. A arquitetura desse projeto é ilustrada na figura a seguir.

[9] O ambiente DE-MAW não está totalmente implementado, sendo que vários trabalhos de pesquisa em andamento visam esse objetivo.

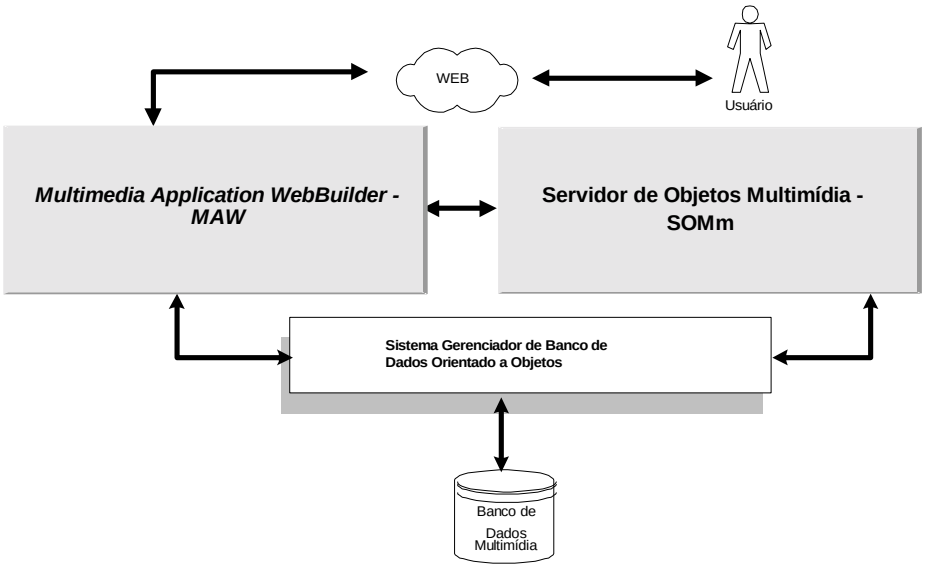



**Figura 6.1 - Arquitetura do projeto AMMO**

De acordo com a Figura 6.1, o AMMO é constituído pelos seguintes elementos:

- *Servidor de Objetos Multimídia (SOMm)*: responsável pelo gerenciamento, criação local e recuperação de aplicações multimídia [Santos and Vieira, 1998] [Santos et al., 1997] [Santos and Vieira, 1998a] [Vieira and Santos, 1997] [Felipe and Vieira, 2000] [Felipe and Vieira, 2000a] [Felipe, 2000].

- *Construtor de Aplicações Multimídia através da Web (MAW)*: fornece o suporte à autoria de aplicações multimídia através da Web [Figueiredo, 2000] [Macedo, 1999] utilizando o padrão SMIL[10].

- *Sistema Gerenciador de Banco de Dados Orientado a Objetos*: mantém as estruturas de classes utilizadas pelo AMMO, gerencia chamadas aos métodos e o armazenamento dos objetos multimídia.

- *Banco de Dados Multimídia*: armazena os objetos multimídia de acordo com as estruturas de classes dos padrões MHEG-5 e SMIL.

### 6.2.2 Descrição do Ambiente

Com o objetivo de englobar todo o processo de ensino/aprendizagem, o ambiente DE-MAW [Seno, 2000] [Seno and Vieira, 2001] [Seno and Vieira, 2001a] visa utilizar toda a potencialidade de recursos de armazenamento, recuperação e autoria de aplicações multimídia

[10] O padrão SMIL (*Synchronized Multimedia Integration Language*) foi desenvolvido pela W3C (*World Wide Web Consortium*) de forma a fornecer uma linguagem que permita a utilização e sincronização de informações multimídia na Web. Para maiores informações consultar W3C [W3C, 1998].

do AMMO. Nesse ambiente, as aulas podem ser construídas com um módulo de autoria próprio e armazenadas em um banco de dados multimídia. Suas metas principais são fazer com que o educador consiga preparar e aplicar suas aulas via Web, acompanhando todo o processo de ensino/aprendizagem dos alunos, e fornecer meios flexíveis para um aluno controlar o seu aprendizado em um curso.

Os seis módulos que compõem o projeto do ambiente DE-MAW são: módulo de segurança, módulo de criação, módulo de visualização, módulo de avaliação e acompanhamento, módulo de manutenção e módulo de apoio. A figura abaixo mostra a arquitetura do ambiente.

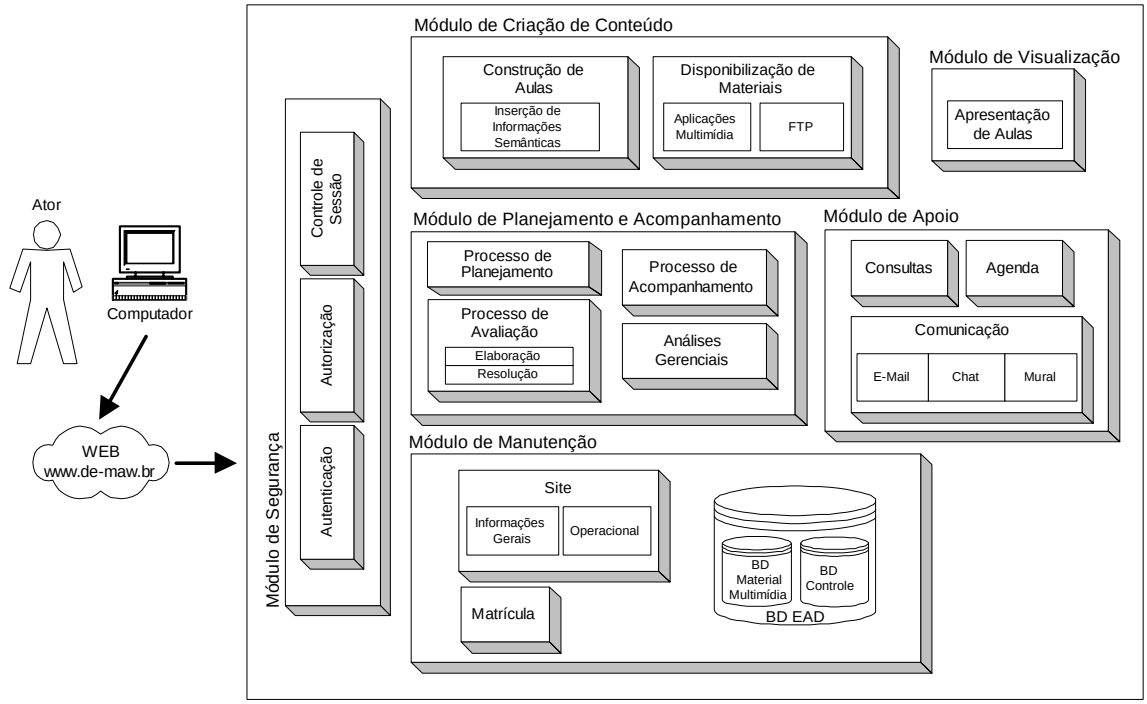



**Figura 6.2 - Arquitetura do ambiente DE-MAW**

De acordo com a Figura 6.2, um ator pode acessar o DE-MAW através de um computador conectado à internet. Primeiramente, ele passa pelo módulo de autenticação. Após o processo de reconhecimento de identidade, o ator tem acesso, dadas as devidas restrições pelo módulo de autorização, aos módulos de criação, visualização, avaliação e acompanhamento, manutenção e apoio. A seguir tem-se a descrição de cada módulo, sendo que maiores detalhes são encontrados em Seno [Seno and Vieira, 2001].

- *Módulo de Segurança*: implementa as funcionalidades da autenticação simples pelo submódulo de autenticação, controla a conexão dos atores ao ambiente através do

submódulo de controle de sessão e determina os direitos de acesso às ferramentas do DE-MAW pelo submódulo de autorização. Este módulo representa a arquitetura de segurança aplicada nesse ambiente de EaD.

- *Módulo de Criação de Conteúdo*: tem como finalidade a construção de aulas e a disponibilização de materiais através, respectivamente, do submódulo de construção de aulas e do submódulo de disponibilização de materiais. Aulas são aplicações multimídia que são exibidas no ambiente de EaD. A disponibilização de materiais é direcionada para os atores, disciplinas e cursos que possuem os direitos para acessá-los, sendo realizada através das aulas multimídia ou pela área de FTP.

- *Módulo de Visualização*: permite que as aulas construídas sejam visualizadas pelos atores, além de fornecer o acesso aos materiais da aula exibida no momento.

- *Módulo de Planejamento e Acompanhamento*: a utilização do submódulo de planejamento permite a criação de planejamentos flexíveis, quebrando a linearidade na execução de cursos a distância. Dessa forma, o aluno tem uma participação mais ativa na construção de seu conhecimento. O submódulo de avaliação fornece meios de analisar o desempenho de um aluno através da Web. Tem como primeira fase a elaboração de um teste, e, como segunda fase, a sua resolução. A elaboração compreende o processo pelo qual o professor ou responsável define de que é constituído e como é realizado um teste. O processo de resolução é definido como sendo a própria resolução do teste pelo aluno, podendo ser realizado on-line ou submetido ao professor após um prazo determinado. Com o uso do submódulo de acompanhamento é possível analisar o rendimento de um aluno no processo de aprendizagem, auxiliando-o no que for necessário através do módulo de apoio.

- *Módulo de Apoio*: divide-se nos submódulos de consulta, comunicação e agenda. O primeiro é responsável por oferecer modos de consulta às bases de dados de material multimídia e de controle. O submódulo de comunicação permite a comunicação unidirecional (e-mail) ou bidirecional (chats, sala de discussões, etc.) dos atores do ambiente de EaD. Através do submódulo de agenda, um aluno pode organizar-se pelo planejamento de suas atividades, anotações, datas importantes, etc.

- *Módulo de Manutenção*: responsável pela manutenção de todo o ambiente DE-MAW. Toda a base de dados e todo o site são mantidos por esse módulo.

## *6.3 Plataforma de Software*

O ambiente DE-MAW está sendo implementado principalmente em *Java* [Lemay and Cadenhead, 2001] [Ayers et al., 1999]. Devido a esse fato, na implementação dos módulos de autenticação, autorização e controle de sessão utilizou-se a mesma linguagem para manter-se o padrão adotado. Nas interfaces das ferramentas de autenticação e autorização a tecnologia de *applets* foi utilizada. Como tecnologia de comunicação entre o cliente e o servidor usou-se *servlets*. O software *Jakarta Tomcat* foi adotado como servidor web para o ambiente. Finalmente, o banco de dados relacional *Interbase* foi usado para armazenar as informações necessárias. As seguintes subseções abordam cada um desses softwares.

### 6.3.1 Applets

Um *applet* [Sun, 2002] é um pequeno programa escrito em *Java* incorporado em uma página web, sendo executado quando é realizada uma requisição a essa página. Os *applets* foram o primeiro conteúdo interativo que podia ser distribuído como parte de uma página da web [Lemay and Cadenhead, 2001].

Uma das principais vantagens de se usar um *applet* é que, como qualquer outro programa escrito na linguagem *Java*, ele pode beneficiar-se dos recursos fornecidos pela *API Java*. Tal fato permite a construção de uma interface mais elaborada e amigável com o usuário através da utilização do pacote[11] *javax.swing*.

A extensão do *HTML* (*Hipertext Markup Language*) através da tag <*APPLET*> possibilita que um *applet* seja inserido em uma página web. A partir do momento que o navegador web encontra essa tag, ele localiza o *applet* no servidor, descarrega-o no cliente e o executa. Um exemplo de código *HTML* utilizando a tag <*APPLET*> é mostrado abaixo.

```
01: <html>
02:     <head>
03:         <title> Exemplo de Applet </title>
04:     </head>
05:     <body>
06:         <applet code = “Exemplo.class” height = “50” width = “345”>
07:             Este programa requer um navegador compatível com Java
08:         </applet>
09:     </body>
10: </html>
```

[11] Um pacote representa um conjunto de classes que se relacionam por compartilharem objetivos em comum.

No exemplo acima, a tag <*APPLET*> está destacada em negrito. Os atributos *code*, *heigth* e *width* representam, respectivamente, o nome do arquivo de classe principal do *applet*, largura e altura da janela do *applet* na página web.

Devido a um *applet* ser executado na máquina cliente e não na máquina servidora, algumas restrições foram criadas para se evitar que ações prejudiciais a um usuário fossem executadas através dos *applets* conhecidos como *hostile applets*. Sendo assim, não é permitido a um *applet*: ler ou escrever arquivos no cliente; comunicar-se com um outro servidor além daquele que forneceu a página web que o inclui; executar um programa no cliente; carregar programas pertencentes ao cliente. Vários navegadores web incluem opções para que as restrições sejam mais rígidas ou maleáveis em referência aos *applets*.

### 6.3.2 Servlets

Um *servlet* pode ser entendido como um *applet* sem interface visual que é executado no servidor [Servlet, 2002], estendendo as funcionalidades de um servidor web de uma maneira simples e consistente. Os *sevlets* oferecem meios para a escrita de programas baseados em componentes e independentes de plataforma.

A utilização da tecnologia de *servlets* atualmente é mais voltada para o protocolo *HTTP*, embora ela não esteja ligada a nenhum protocolo específico. As *API*'s utilizadas na construção de um *servlet* é disponibilizada nos pacotes *javax.serlet* e *javax.servlet.http*. O primeiro é considerado o *framework* básico para a construção de um *servlet*, enquanto o segundo é uma extensão do *framework* básico para se atender a requisições *HTTP*.

Os *servlets* não possuem as mesmas restrições encontradas nos *applets*¸ pois sua execução ocorre na máquina servidora. Eles também oferecem um desempenho muito maior em relação aos programas *CGI* (*Commom Gateway Interface*).

### 6.3.3 Jakarta Tomcat e Interbase

O servidor web gratuito *Tomcat* é considerado como um *container* para os *servlets* devido ele implementar as especificações da *servlet API* [Tomcat, 2002]. Dessa forma, o *Tomcat* manipula as requisições provindas do cliente, transforma os parâmetros da requisição em objetos compreensíveis pelos métodos de um *servlet* e retorna a resposta para esse cliente.

O sistema gerenciador de banco de dados gratuito *Interbase* [Interbase, 2002] fornece um poderoso meio para armazenamento de grandes quantidades de dados com estabilidade e

segurança. Através de uma interface gráfica amigável, o seu ambiente permite a criação e manutenção de banco de dados que possuem todos os mecanismos desejáveis em banco de dados relacionais como procedimentos armazenados, gatilhos, papéis, etc.

### 6.3.4 Integração das Tecnologias

Até o momento as tecnologias utilizadas no desenvolvimento foram apresentadas separadamente. A Figura 6.3 mostra como essas tecnologias interagem entre si.

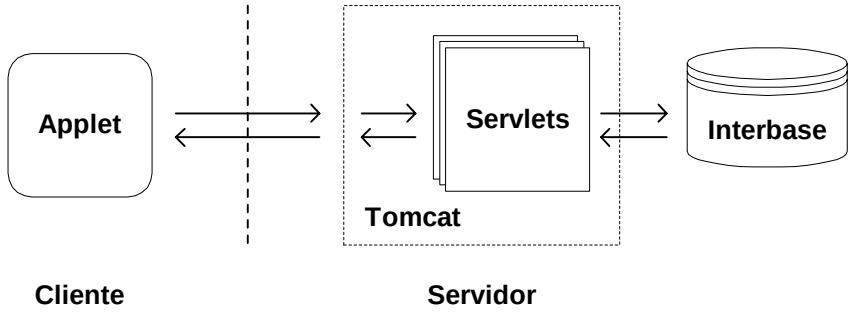



**Figura 6.3 - Integração das tecnologias**

Inicialmente, o *applet* é trazido para a máquina cliente através do acesso de uma página web do ambiente de EaD. Através das interações do usuário com a interface do *applet*, requisições *http* são enviadas para o servidor. Essas requisições geralmente representam manipulações de dados contidos no banco de dados. O servidor web *Tomcat* verifica a chegada de uma requisição e transforma os parâmetros pertencentes a ela em objetos inteligíveis para o *servlet* que realizará o processamento das informações. Esse *servlet* realiza requisições de seleção e/ou inserção e/ou atualização e/ou remoção de dados para o banco de dados *Interbase*. A operação é realizada pelo BD e é retornada uma resposta para o *servlet*, que por sua vez a retorna para o *Tomcat*. Após isso, ela é enviada para o *applet* que formata os dados e os apresenta via interface para o usuário que iniciou todo o processo.

O fato do *servlet* funcionar como um elemento intermediário entre o banco de dados e o *applet* traz fim a uma importante limitação desse último. Como o *applet* pode acessar apenas a máquina servidora da qual ele foi descarregado, teoricamente esse servidor web teria que ser também o servidor de banco de dados para atender às requisições de dados do *applet*. O *servlet* acaba com essa limitação, pois é ele que faz acesso ao banco, que pode estar em uma máquina servidora distinta.

## *6.4 Ferramentas Geradas*

A implementação da arquitetura de segurança aplicada no ambiente DE-MAW envolveu os módulos de autenticação, autorização e controle de sessão. Para os módulos de autenticação e autorização foram geradas ferramentas com as quais um ator desse ambiente pode interagir. O módulo de controle de sessão não possui interface, pois se trata de um módulo para controle interno utilizado pelo DE-MAW. A seguir são feitas considerações sobre o desenvolvimento de cada módulo.

### 6.4.1 Ferramenta de Autenticação

Seguindo a especificação da arquitetura para o módulo de autenticação simples, foi utilizado o conceito de chaves na implementação da ferramenta de autenticação utilizada no ambiente DE-MAW.

Para implementar o esquema de chaves, algumas bibliotecas de segurança pertencentes à linguagem *Java* foram usadas. Essas bibliotecas estão localizadas nos pacotes *java.security* e *javax.cripto*, sendo que maiores detalhes sobre eles e sobre a arquitetura de segurança *Java* são encontrados no apêndice A.

#### 6.4.1.1 Descrição de Interfaces

Quando um ator deseja entrar em alguma área restrita do ambiente DE-MAW, ele é direcionado para a página de *login*. A interface dessa página é visualizada de acordo com a Figura 6.4. Para que seja autenticado, o ator fornece o seu nome de usuário e a sua senha. A partir do momento em que ele clica no botão *Enviar*, é criada uma chave de criptografia baseada no seu nome de usuário. Tal chave é utilizada pelo algoritmo de criptografia na codificação da senha. Uma vantagem desse algoritmo a ser ressaltada é que a chave não é armazenada fisicamente para posterior recuperação, existindo somente na memória volátil da máquina cliente enquanto a senha é codificada. Assim, ela é descartada após sua utilização e criada novamente caso haja necessidade. Isso evita a obtenção da chave por um indivíduo não autorizado caso haja invasão dessa máquina cliente.

Com a chegada dessas informações na máquina servidora do ambiente DE-MAW, primeiramente é verificado se o nome de usuário existe no banco de dados. Caso não exista é solicitado que o ator informe os dados novamente. Existindo, a senha armazenada no banco de dados é recuperada. Logo após isso, o algoritmo de criptografia é aplicado para decodificação tanto na senha enviada quanto na recuperada. Se os valores forem

incompatíveis, é solicitada a reentrada dos dados pelo ator. Caso eles sejam iguais, a página que contém a pergunta randômica é apresentada para o ator, de acordo com a Figura 6.5.

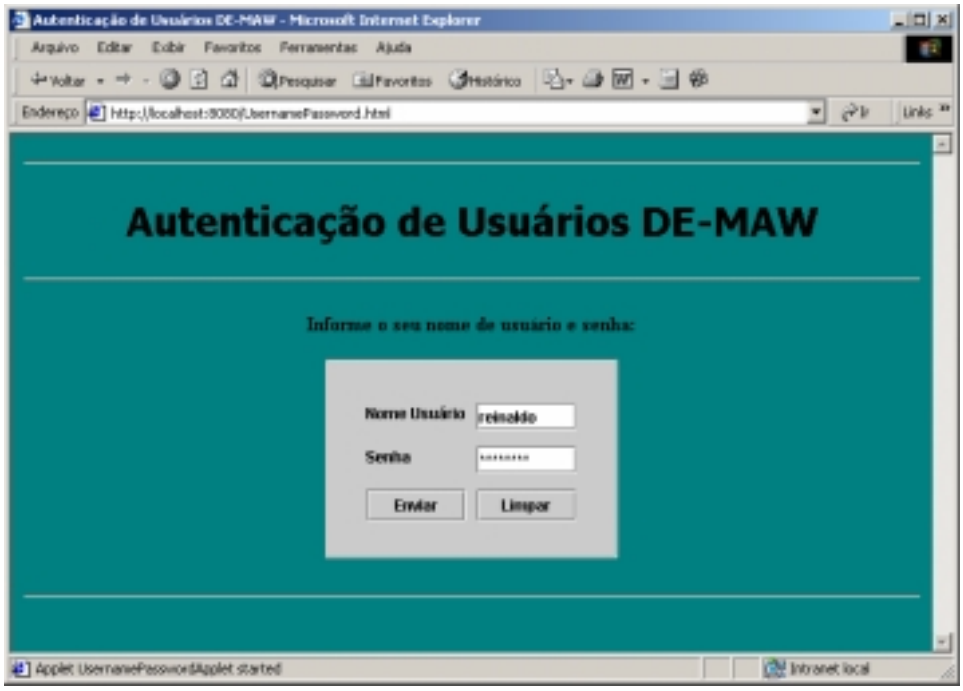



**Figura 6.4 - Interface da ferramenta de autenticação do ambiente DE-MAW / Nome de usuário e senha**

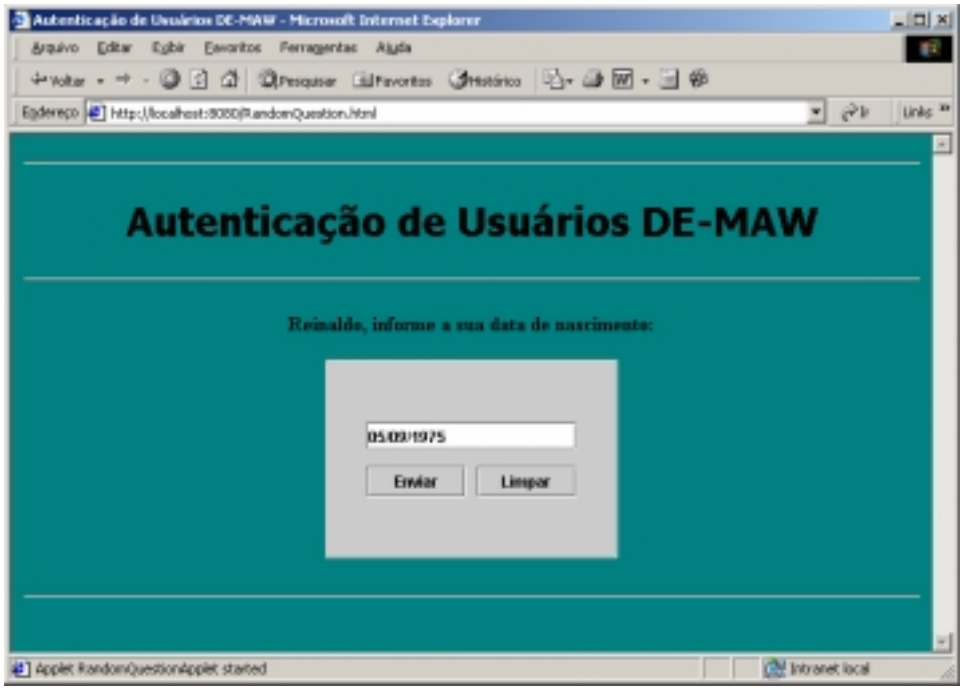



**Figura 6.5 - Interface da ferramenta de autenticação do ambiente DE-MAW / Questão randômica**

O ator tem direito de acessar o ambiente DE-MAW caso responda corretamente a questão randômica. A resposta da questão também é codificada via chave de criptografia para que possa ser enviada em segurança para o servidor.

A página apresentada para o ator, caso haja incompatibilidade na conferência das informações em qualquer um dos passos da autenticação, é mostrada na Figura 6.6. O *link Tentar novamente* remete o ator para a página inicial de autenticação (Figura 6.4).

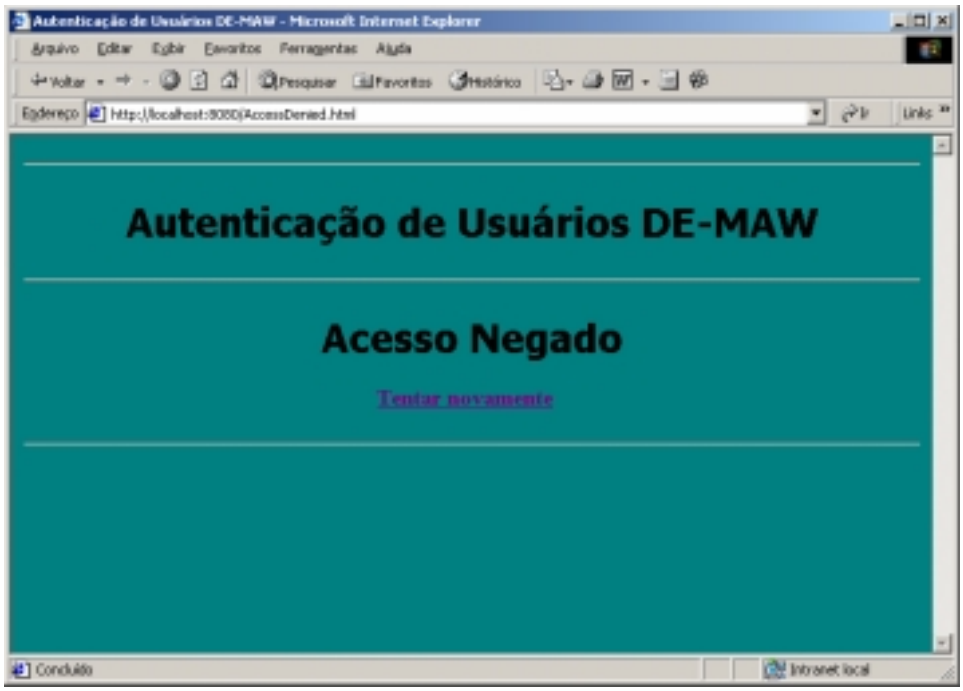



**Figura 6.6 - Interface da ferramenta de autenticação do ambiente DE-MAW / Acesso negado**

### 6.4.2 Ferramenta de Autorização

O ambiente DE-MAW controla o acesso de seus atores sobre as ferramentas que ele disponibiliza para utilização nos cursos. A ferramenta de autorização criada para esse fim contempla os três recursos especificados pelo módulo de autorização no capítulo 5. São eles: autorizações coletivas/individuais, autorizações positivas/negativas e autorizações explícitas/implícitas.

#### 6.4.2.1 Extensão do BD do Ambiente DE-MAW

Para que tais recursos fossem possíveis no ambiente, a extensão do seu banco de dados foi necessária. As modificações foram feitas de acordo com as considerações da seção 5.5 do capítulo 5 e podem ser visualizadas na Figura 6.7.

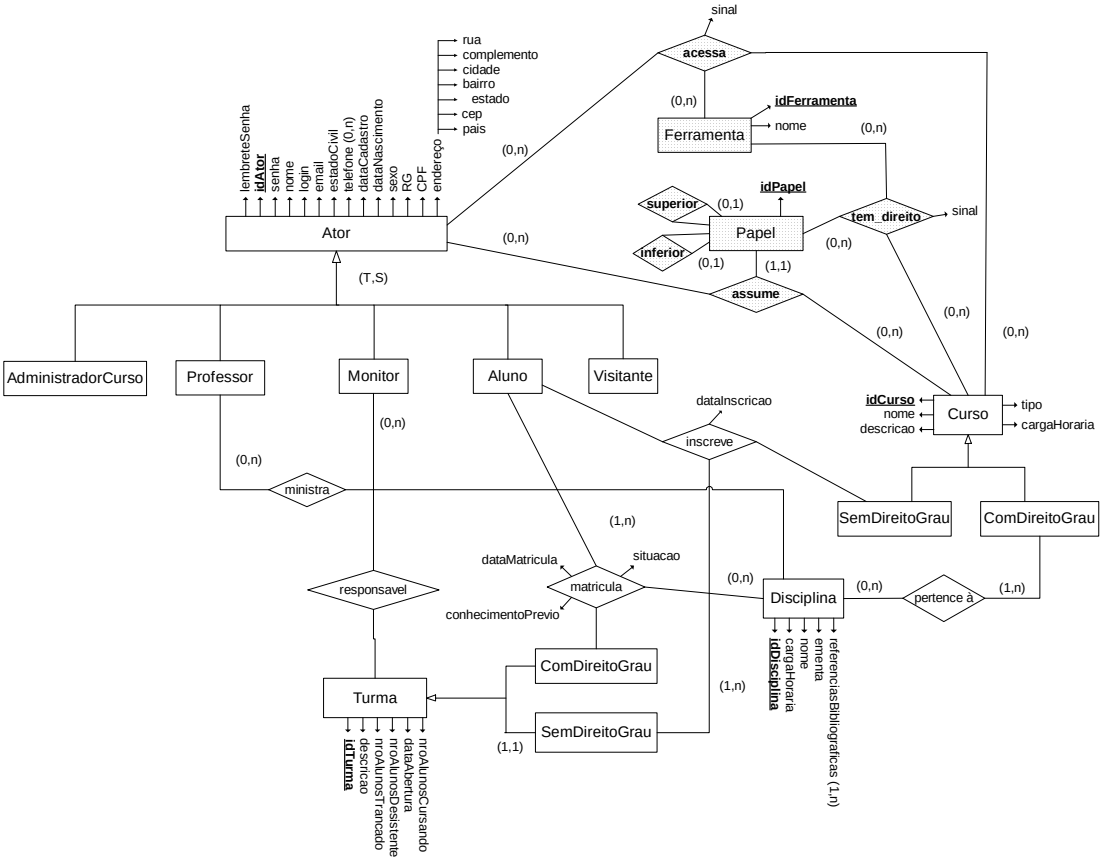



**Figura 6.7 - BD do ambiente DE-MAW estendido para dar suporte à arquitetura de segurança**

A partir do banco de dados original [Silva, 2002] foram inseridos os relacionamentos e as entidades destacadas na figura acima. Essas entidades correspondem à modelagem de classes apresentada na subseção 5.4.3 do capítulo 5, enquanto que os relacionamentos ternários correspondem à situação mostrada na subseção 5.5.3 desse mesmo capítulo.

Passar a hierarquia de generalização/especialização da entidade ator de total/exclusiva (T/E) para total/sobreposição (T/S) é uma outra proposta de modificação. Esse fato permite que fique caracterizada a situação tratada anteriormente ($1^{o.}$ caso da subseção 5.5.1 do capítulo 5) em que um ator pode assumir diversos papéis em diferentes cursos.

Como é necessária a criação da entidade *Papel* para que as autorizações coletivas possam existir, é necessário que as aplicações que acessam esse banco mantenham o controle de consistência entre essa entidade e as entidades que representam os papéis. Isso para evitar, por exemplo, que um aluno matriculado em uma disciplina de um determinado curso possa assumir um papel diferente nesse mesmo curso.

### 6.4.2.2 Construção da Hierarquia de Papéis

Embora o módulo de autorização possa executar a derivação de autorizações implícitas através dos domínios de papéis, de tipos de operação e de objetos, decidiu-se que, para o DE-MAW, somente o primeiro domínio seria utilizado. A razão de tal decisão foi a facilidade de compreensão do conceito de autorizações explícitas/implícitas pelos usuários do ambiente. Isso porque esses usuários vêm das mais diversas áreas de conhecimento e, muitas vezes, têm somente noções de computação. Tal fato retoma o conceito da não-obrigatoriedade das hierarquias apresentado na subseção 5.4.2.5 do capítulo anterior.

A construção da hierarquia de papéis foi baseada nos papéis que o ambiente DE-MAW define, ou seja, administrador de curso, professor, monitor, aluno e visitante. Dessa forma, tal hierarquia é visualizada na Figura 6.8.

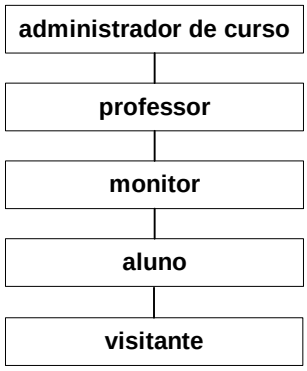



**Figura 6.8 - Hierarquia de papéis no ambiente DE-MAW**

Mudanças estruturais na hierarquia de papéis, tais como inserir ou remover um papel, não afetam a ferramenta de autorização, pois ela foi implementada para prever essas situações que podem ocorrer durante a evolução do ambiente.

Como efeitos práticos da não utilização das hierarquias de tipos de operação e de objetos pelo DE-MAW, a existência do atributo *tipo*, que define o tipo de operação que os atores executam sobre os objetos (Figura 5.6 do capítulo 5), e dos auto-relacionamentos da entidade *Ferramenta*, que denotam a hierarquia de objetos, já não são mais necessárias (Figura 6.7). Além disso, os laços de iteração referentes às hierarquias de tipos de operação e de objetos dos algoritmos anteriores (subseções 5.4.4.1, 5.4.4.2 e 5.4.4.3 do capítulo 5) são eliminados.

### 6.4.2.3 Descrição de Interfaces

Após o processo de autenticação, um ator que assume o papel *administrador de curso* tem direito de acessar a ferramenta de autorizações. Essa ferramenta permite que tal ator atribua direitos para outros atores que pertencem ao curso que ele administra. A interface da ferramenta de autorização é visualizada de acordo com a Figura 6.9.

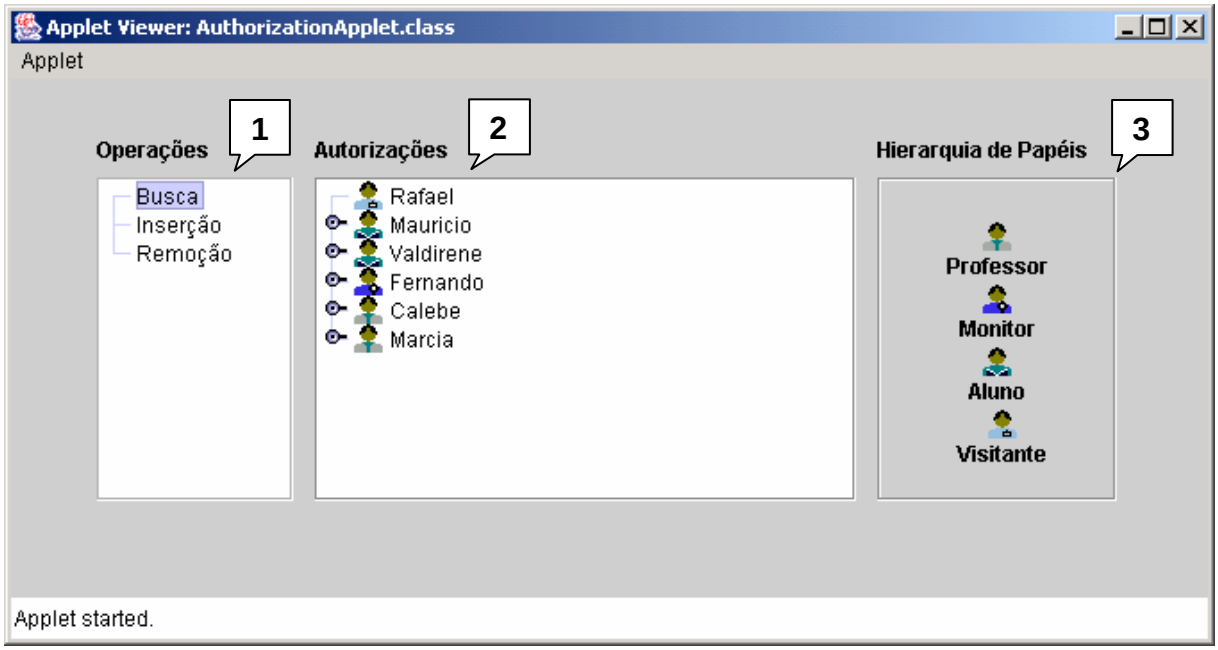



**Figura 6.9 - Interface da ferramenta de autorização do ambiente DE-MAW / Busca**

Na figura acima, a caixa *Operações* (1) mostra as possíveis operações a serem executadas através dessa interface, sendo que a busca é a operação atualmente selecionada.

Todos os atores do curso são listados na caixa *Autorizações* (2). Dessa forma, para verificar se um ator tem ou não tem uma autorização, basta selecioná-lo para visualizar todos os direitos de acesso que ele possui.

A *Hierarquia de Papéis* (3) mostra de que forma os papéis são arranjados para que se possa encontrar as autorizações implícitas. Essa hierarquia corresponde àquela mostrada na Figura 6.8. Os ícones , , e representam, respectivamente, os papéis visitante, aluno, monitor e professor do ambiente DE-MAW. O papel administrador não consta na visualização porque, na prática, o ator que o assume interage diretamente com a ferramenta.

Quando o administrador de curso seleciona um ator na caixa *Autorização* (2) o respectivo papel desse ator é destacado na *Hierarquia de Papéis* (3) juntamente com os seus papéis superiores. O papel que fica com o texto em vermelho é aquele atualmente selecionado

na caixa *Autorização* (2), enquanto que os papéis com texto em azul são os superiores ao atual. Tal funcionalidade facilita a visualização dos papéis que herdam as autorizações implícitas do ator atualmente selecionado. Figura 6.10 demonstra esse recurso.

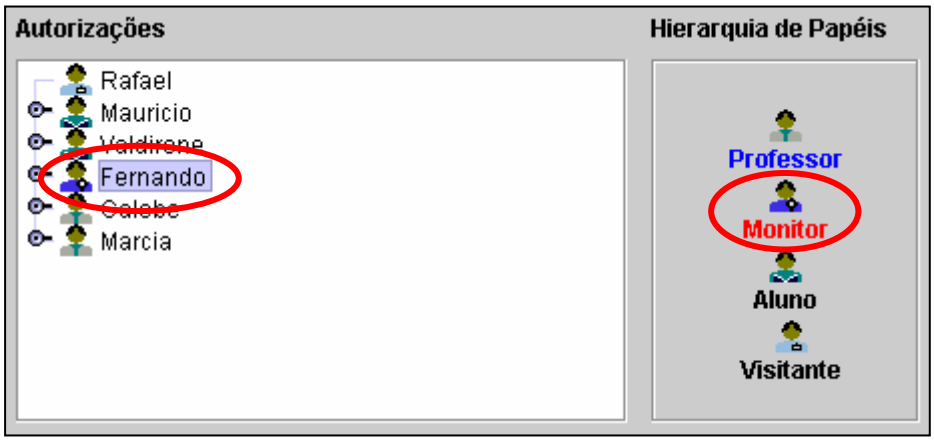



**Figura 6.10 - Atualização da hierarquia de papéis através da seleção de um ator**

A caixa *Autorização* (2) possui diferentes ícones para representar as diferentes formas das autorizações sobre as ferramentas do ambiente. Os significados desses ícones são:

- Autorização individual positiva explícita ();
- Autorização individual negativa explícita ();
- Autorização coletiva positiva explícita ();
- Autorização coletiva negativa explícita ();
- Autorização implícita ().


**Figura 6.11 – Autorizações pertencentes aos atores**

De acordo com a figura acima, o aluno *Mauricio* tem as autorizações coletivas positivas explícitas sobre as ferramentas *Comunicação* e *Visualização de Aulas* e a autorização individual positiva explícita sobre a ferramenta *Consulta*. Sendo assim, o monitor *Fernando* herda todas essas autorizações, pois está um nível acima na hierarquia de papéis.

A interface da operação de inserção de uma autorização na ferramenta de autorização do DE-MAW apresenta-se de acordo com a Figura 6.12. Os passos para a inserção são seqüenciais, sendo que a ferramenta só permite passar para o próximo passo quando o anterior foi executado corretamente.

Primeiramente, o administrador de curso deve selecionar o tipo da autorização a ser inserida: coletiva ou individual (1). Após isso, a opção para a escolha do sinal da autorização é liberada (2). Terminado esse passo, uma lista dos papéis ou dos atores do ambiente é mostrada na caixa de listagem mais a esquerda (3). Nesse passo, mostrar os papéis ou os atores depende da escolha do tipo da autorização (1). Com a escolha do papel ou do ator no passo anterior, apenas as ferramentas que não entram em conflito com nenhuma autorização existente são apresentadas na caixa de listagem (4).

Para confirmar a inserção, o administrador de curso deve clicar no botão "*Gravar*". Caso contrário ele clica no botão "*Cancelar*".

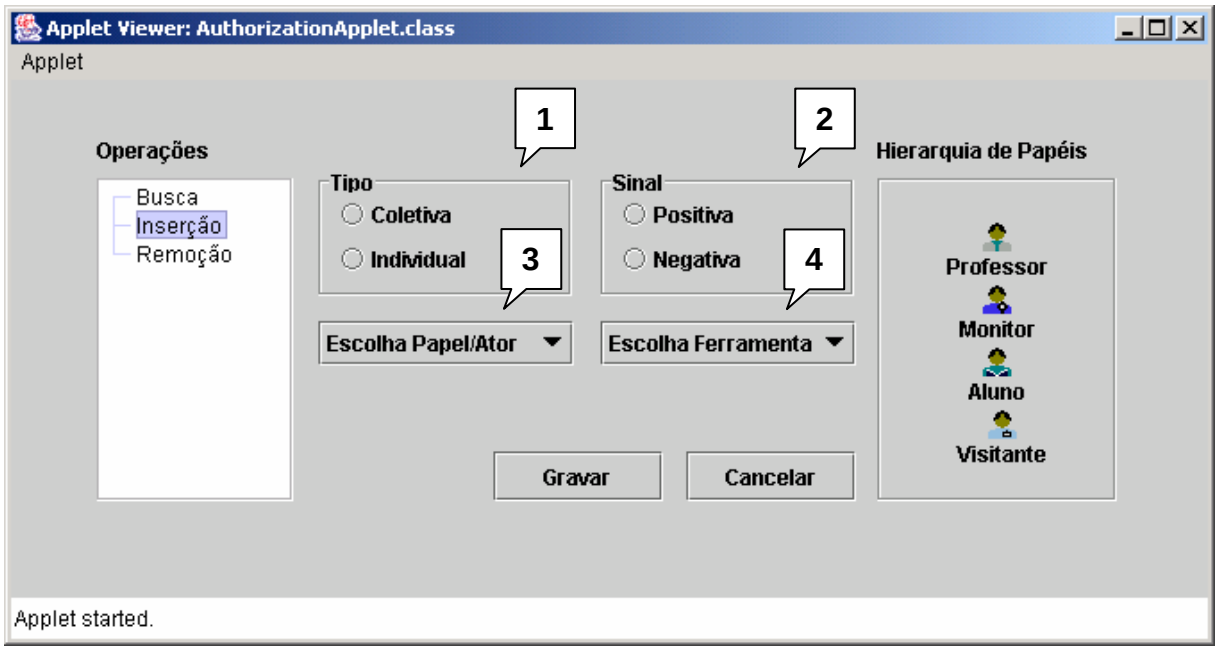



**Figura 6.12 - Interface da ferramenta de autorização do ambiente DE-MAW / Inserção**

A interface da operação de remoção de uma autorização pode ser visualizada em conformidade com a Figura 6.13.

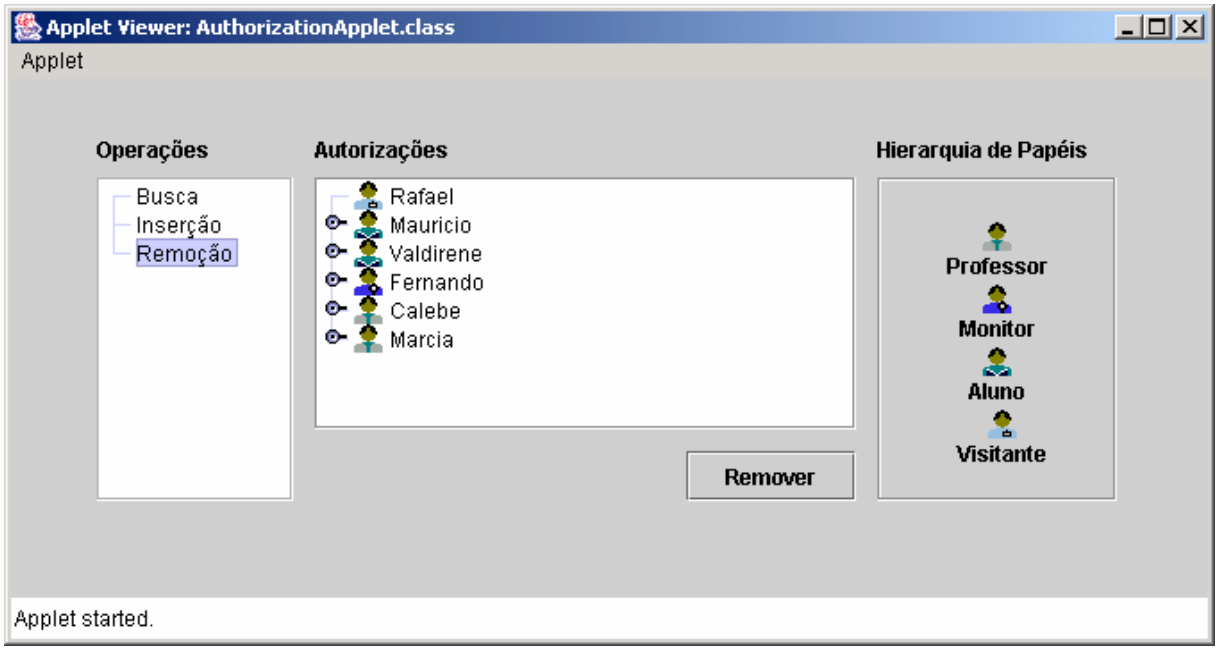



**Figura 6.13 - Interface da ferramenta de autorização do ambiente DE-MAW / Remoção**

Assim, para remover uma autorização, basta que o administrador de curso a selecione e, em seguida, clique no botão '*Remover*'. No exemplo, a autorização individual positiva explícita sobre a ferramenta *Agenda*, pertencente ao aluno *Valdirene*, está selecionada para remoção.

## *6.5 Considerações Finais*

Neste capítulo foi apresentado um estudo de caso em que a arquitetura de segurança foi aplicada ao ambiente de EaD DE-MAW. Primeiramente, uma descrição do ambiente DE-MAW foi realizada, explicando-se suas origens e as funcionalidades de cada módulo que o compõe. Logo após, a plataforma de software utilizada na implementação do estudo de caso foi abordada através de suas principais características. Finalmente, as ferramentas de autenticação e autorização, geradas para validar este projeto, foram descritas.

No próximo capítulo são apresentadas as conclusões deste trabalho de mestrado.

# Capítulo 7
# Conclusões

Um forte processo de modernização, impulsionado pelo avanço tecnológico, vem ocorrendo na educação nos últimos anos. Devido a esse fato, alguns dos principais paradigmas e componentes educacionais vêm sofrendo mudanças em relação ao desempenho de suas funções.

Nesse contexto, a EaD exerce um papel fundamental, tendo como proposta trazer inúmeros benefícios para o processo de ensino/aprendizagem, entre eles, proporcionar ao aluno maior liberdade na questão de horários, atingir o maior número de pessoas, minimizar o problema da distância entre instituição educadora e aluno, etc. Para que esses objetivos possam ser atingidos, um dos principais meios de comunicação utilizados vem sendo a web. Através dela, softwares conhecidos como ambientes de EaD são disponibilizados para uso.

Esses ambientes vêm sendo desenvolvidos com a devida atenção para as questões metodológicas de ensino e de aprendizagem, oferecendo uma gama de recursos para, por exemplo, a construção de aulas através de ferramentas que possibilitam a inclusão de elementos audiovisuais.

Embora de grande importância, poucos trabalhos citam os recursos de segurança para ambientes de EaD. Tais recursos permitem um maior controle do funcionamento do ambiente, possibilitando restringir o acesso às suas partes componentes.

Esta monografia teve como objetivo atentar para tal questão, criando uma arquitetura de segurança que abrange os aspectos da autenticação e controle de acesso. Tal arquitetura pode ser utilizada por projetistas e desenvolvedores na implementação de seus ambientes de EaD.

## *7.1 Contribuições*

As contribuições obtidas no desenvolvimento deste trabalho de mestrado foram:

- Criação de uma arquitetura de segurança independente em relação às características particulares de ambientes de EaD;

  - Definição e formalização de uma autorização para controle de acesso em ambientes de EaD;
  - Modelagem de uma estrutura de classes para suportar os conceitos de autorização coletiva/individual, autorização positiva/negativa e autorização explícita/implícita;
  - Criação de políticas de gerenciamento para manter-se a consistência das instâncias da estrutura de classes;
  - Concepção dos algoritmos de busca, inserção e remoção de autorizações de acordo com as políticas de gerenciamento;
- Aplicação dessa arquitetura no ambiente DE-MAW;
- Construção das ferramentas de autenticação e autorização para o ambiente DE-MAW.

## *7.2 Trabalhos Futuros*

A partir da arquitetura de segurança, os seguintes trabalhos futuros podem ser considerados para a criação de outros projetos de pesquisa:

- Adaptação da arquitetura proposta para a criação de um *framework* de segurança para ambientes de EaD;
- Análise e implementação dos módulos monitor e auditor;
- Incorporação das ferramentas geradas por essas pesquisas no ambiente DE-MAW.

# Referências Bibliográficas


[Aretio, 1994] Aretio, G. L., *Educación a Distancia Hoy*, Madrid: UNED, 1994, http://www.cciencia.ufrj.br/educnet/EDUOBJ.HTM, extraído em março de 2001.

[Ayers et al., 1999] Ayers, D., Bergsten, H., Bogovich, M., Diamond, J., Ferris, M., Fleury, M., Halberstadt, A., Houle, P., Mohseni, P., Patzer, A., Phillips, R., Li, S., Vedati, K., Wilcox, M., and Zeiger, S., **Java Professional Server Programming**, Professional Series, Wrox Press Ltda, 1999.

[Banerjee et al., 1987] Banerjee, J., Chou, H. T., Garza, J., Kim, W., Woelk, D., Ballou, N., and Kim, H. J., *Data Model Issues for Object-oriented Applications*, **ACM Trans. Off. Inf. Sys.**, Vol. 5, No 1, pp. 3-26, 1987.

[Baraani-Dastjerdi et al., 1997] Baraani-Dastjerdi, A., Pieprzyk, J., and Safavi-Naini, R., *A Multi-level View Model for Secure Object-oriented Databases*, ***Data & Knowledge Engineering***, Vol. 23, pp. 97-117, 1997.

[Bertino and Martino, 1993] Bertino, E., and Martino, L., **Object-oriented Database Systems**, Addison-Wesley Publishing Company Inc, 1993.

[Bertino et al, 1999] Bertino, E., Jajodia, S., and Samarati, P., *A Flexible Authorization Mechanism for Relational Data Management Systems*, ***ACM Transactions on Information Systems***, Vol. 17, No. 2, pp. 101-140, 1999.

[Bidzos, 1992] Bidzos, D. J., *Public Key Cryptography*, **Computer Security Reference Book**, CRC Press, pp. 121-133, 1992.

[Booch et al., 1999] Booch, G., Rumbaugh, J., and Jacobson, I., **Unified Modeling Language User Guide**, Addison-Wesley Publishing Company Inc, 1999.

[Castro, 2002] Castro, R. O., and Biajiz, M., *A Proposal for Web-based Learning Environment Security*, In Proceedings of IEEE International Conference on Advanced Learning Technologies, To be published.

[Chaves, 2001] Chaves, E. *Ensino a Distância: Conceitos Básicos*, http://www.edutecnet.com.br, extraído em fevereiro 2001.

[Chin, 1999] Chin, S. K., *High-Confidence Design for Security*, ***Communications of the ACM***, Vol. 42, No. 37, pp. 33-37, 1999.

[Coulouris et al., 1994] Coulouris, G., Dollimore, J., and Kindberg, T., **Distributed Systems: Concepts and Design**, Addison-Wesley, 1994.

[Elmasri and Navathe, 2000] Elmasri, R., and Navathe, S. B., **Fundamentals of Database Systems**, Addison-Wesley Publishing Company Inc, 2000.

[Everett, 1992] Everett, D., *Identity Verification and Biometrics*, **Computer Security Reference Book**, CRC Press, pp. 37-73, 1992.

[Felipe and Vieira, 2000] Felipe, J. C., and Vieira, M. T. P., *An Environment for Knowledge Discovery from Multimedia Aplication Metadata in an MHEG-5 Server*, In Proceedings of the VI Brazilian Symposium on Multimedia and Hypermedia Systems, pp. 213-230, 2000.

[Felipe and Vieira, 2000a] Felipe, J. C., and Vieira, M. T. P., *Modeling of an Object-oriented Data Mart for Knowledge Discovery in Multimedia Metadata*, In Proceedings of the XV Brazilian Symposium on Database, 2000.

[Felipe, 2000] Felipe, J. C., Extração de Conhecimento sobre Metadados de Aplicações Multimídia em um Ambiente Multidimensional Orientado a Objetos, Dissertação de Mestrado, Departamento de Computação, UFSCar, 2000.

[Fernandez et al., 1981] Fernandez, E. B., Summers, R. C., and Wood, C., **Database Security and Integrity**, Addison-Wesley Publishing Company Inc, 1981.

[Fernandez et al., 1993] Fernandez, E. B., Larrondo-Petrie, M. M., and Gudes, E., *A Method-Based Authorization Model for Object-Oriented Databases*, In Proceedings of the OOPSLA-93, pp. 135-150, 1993.

[Figueiredo, 2000] Figueiredo, J. M., Um Ambiente de Autoria Multimídia para a World Wide Web, Dissertação de Mestrado, Departamento de Computação, UFSCar, 2000.

[Fiorese and Tarouco, 1999] Fiorese, M., and Tarouco, L. M. R., *Uma Solução na Autenticação de Usuários para Ensino a Distância*, disponível em http://www.inf.ufrgs.br/pos/SemanaAcademica/Semana99/fiorese/fiorese.html, 1999.

[Fiorese, 2000] Fiorese, M., *Uma Proposta para Autenticação de Usuários para Ensino a Distância*, Dissertação de mestrado, Instituto de Informática, UFRGS, 2000.

[Gal-Oz and Fernandez, 1993] Gal-Oz, N., Gudes, E., and Fernandez, E. B., *A Model of Methods Access Authorization in Object-oriented Databases*, In Proceedings of the 19ª VLDB Conference, pp. 52-61, 1993.

[Ganley and Piper, 1992] Ganley, M., and Piper, F., *Encryption Algorithms*, **Computer Security Reference Book**, CRC Press, pp. 91-119, 1992.

[Gonçalves, 1996] Gonçalves, C. T. F., *Quem Tem Medo de Ensino a Distância*, Revista Educação a Distância, No. 7-8, INED/IBASE, 1996.

[Gong, 1989] Gong, L., *A Secure Identity-based Capability System*, in Proceedings of 1989 IEEE Symposium on Research in Security and Privacy, Oakland, CA USA, pp. 56-63, 1989.

[Interbase, 2002] *Interbase Document Set*, Borland/Imprise, 2002.

[ISO/IEC, 1996] [ISO/IEC, 1996] ISO/IEC, **Information Technology Coding of Multimedia and Hypermedia Information, Part 5: Support for Base-Level Interactive Applications**, MHEG-5 IS Document, 1996.

[Jackson and Hruska, 1992] Jackson, K. M., and Hruska, J., **Computer Security Reference Book**, CRC Press, 1992.

[Kim et al., 1988] Kim, W., Ballou, N., Chou, H. T., Garza, J., Woelk, D., and Banerjee, J., *Integrating an Object-oriented Programming System with a Database System*, In Proceedings of the 3rd International Conference on Object-oriented Programming Systems, Languages, and Applications, pp. 142-152, 1988.

[Lemay and Cadenhead, 2001] Lemay, L., and Cadenhead, R., **Aprenda em 21 Dias Java 2 Professional Reference**, Editora Campus, 2001.

[Linhalis, 2000] Linhalis, F., Interface Básica para um Servidor Universal, Dissertação de Mestrado, Instituto de Ciências Matemáticas e Computação – USP São Carlos, 2000.

[Loyolla and Prates, 2001] Loyolla, W. P. D. C., Prates, M., *Educação a Distância Mediada por Computador (EAD): Uma Proposta Pedagógica*, http://www.puccamp.br/~prates/edmc.html, extraído em fevereiro de 2001.

[Lunt, 1995] Lunt, T. F., *Authorization in Object-oriented Databases*, **Modern Database System: The Object Model, Interoperability, and Beyond**, ACM Press and Addison-Wesley Publishing Company Inc, pp. 130-145, 1995.

[Macedo, 1999] Macedo, M. R., *MHEG-5/SMIL: Um Ambiente de Integração entre os Padrões*, Dissertação de Mestrado, Departamento de Computação, UFSCar, 2000.

[Martins, 2001] Martins, G. A., *Vocábulos sobre Métodos e Técnicas de Pesquisa*, disponível em http://www.eac.fea.usp.br/metodologia/elucidario.asp, extraído em julho de 2001.

[McLean, 1994] McLean, J., *Security Models*, **Encyclopedia of Software Engineering**, Wiley Press, 1994.

[Moore and Kearsley, 2001] Moore, M., and Kearseley, G., *Distance Education: A System View*, http://www.cde.psu.edu/de/what_is_de.html#definition, extraído em fevereiro de 2001.

[Moraes, 2001] Moraes, P., *O que é EaD*, http://www.terravista.pt/Enseada/2023/a1.htm, extraído em março de 2001.

[Morgan, 1999] Morgan, M. B., *Comparison of Online Course Delivery Software Products*, http://www.marshall.edu/it/cit/webct/compare/comparison.html, extraído em março de 2001.

[Rabitti et al., 1991] Rabitti, F., Bertino, E., Kim, W., and Woek, D., *A Model of Authorization for Next-generation Database Systems*, **ACM Transactions on Database Systems**, Vol. 16, pp. 288-299, 1991.

[Ryutov and Neuman, 2000] Ryutov, T., and Neuman, C., *Representation and Evaluation of Security Policies for Distributed System Services*, in Proceedings of DARPA Information Survivability Conference and Exposition, Hilton Head, South Carolina, pp. 172-183, 2000.

[Sá, 2000] Sá, P. S. S., Gerador Automático de Arquivos HTML de Ajuda para Aplicação em Educação a Distância, Dissertação de Mestrado, Instituto de Ciências Matemáticas e de Computação de São Carlos, USP, 2000.

[Sandhu and Samarati, 1994] Sandhu, R. S., and Samarati, P., *Access Control: Principles and Practice*, IEEE Communications Magazine, Vol. 32, No. 9, pp. 40-60, 1994.

[Santos and Vieira, 1998] Santos, F. C., Vieira, M. T. P., *SMART – Ambiente para Autoria de Aplicações MHEG-5*, Anais do IV Simpósio Brasileiro de Sistemas Multimídia e Hipermídia, pp. 39-50, 1998.

[Santos and Vieira, 1998a] Santos, M. T. P., and Vieira, M. T. P., *Sistema de Recuperação de Informações em um Servidor de Objetos Multimídia*, Anais do IV Simpósio Brasileiro de Sistemas Multimídia e Hipermídia, pp. 249-260, 1998.

[Santos et al., 1997] Santos, M. T. P., Vieira, M. T. P., and Figueiredo, J. M., *Extensão de um Banco de Dados de Objetos MHEG-5 para Suportar Busca por Conteúdo*, Anais do XII Simpósio Brasileiro de Banco de Dados, pp. 107-121, 1997.

[Seno and Vieira, 2001] Seno, W. P., and Vieira, M. T. P., Uma Arquitetura de um Ambiente de Educação a Distância, Relatório Técnico RT001/2001, Departamento de Computação, UFSCar, 2001.

[Seno and Vieira, 2001a] Seno, W. P., and Vieira, M. T. P., *DE-MAW – Um Ambiente de Educação a Distância*, Revista Brasileira de Informática na Educação – RBIE, Vol. 10, No. 1, pp 21-29, 2002.

[Seno, 2001] Seno, W. P., Modelo e Base de Dados para Automatização do Planejamento e Execução de Cursos em Ambientes de Educação a Distância, Dissertação de Mestrado, Departamento de Computação, UFSCar, 2001.

[Servlet, 2002] *Java Servlet Technology*, http://java.sun.com/products/servlet, extraído em março de 2002.

[Sherwood, 1992] Sherwood, J., *Distributed Systems*, **Computer Security Reference Book**, CRC Press, pp. 797-819, 1992.

[Silberschatz et al., 1999] Silberschatz, A., Korth, H. F., and Sudarshan, S., **Sistema de Banco de Dados**, Makron Books, 1999.

[Silva, 2002] Silva, D. R., Uma Ferramenta para Descoberta de Conhecimento com Suporte de Data Warehousing e sua Aplicação para Acompanhamento do Aluno em Educação a Distância, Dissertação de Mestrado, Departamento de Computação, UFSCar, 2002.

[Sun, 2002] **Java 2 SDK Standard Edition Documentation**, Version 1.4, Sun Microsystems, 2002.

[Tomcat, 2002] *Tomcat 4.0*, http://jakarta.apache.org/tomcat/tomcat-4.0-doc, extraído em

março de 2002.

[Vieira and Santos, 1997] Vieira, M. T. P., and Santos, M. T. P., *Content-based Search on an MHEG-5 Standard-based Multimedia Database*, In Proceedings of the QPMIDS DEXA 97, pp. 154-159, 1997.

[W3C, 1998] W3C Working Draft, **Synchronized Multimedia Integration Language (SMIL) 1.0 Specification**, http://www.w3.org/TR/1998/REC-smil19980615, 1998.

# Apêndice A – Segurança em Java

A linguagem Java oferece uma poderosa *API* para tornar aplicações mais confiáveis e seguras através dos conceitos que envolvem criptografia (chave simétrica, chave assimétrica, assinatura digital, etc.), sendo que as classes que encapsulam esses conceitos estão localizadas no pacote *java.security* e *javax.cripto* [Ayers et al., 1999]. Tal *API* é dividida em duas partes:

- *Java Cryptography Architecture* (*JCA*): considerada como o núcleo para o desenvolvimento de criptografia na linguagem Java;
- *Java Cryptography Extension*[12] (*JCE*): utilizada para a construção de criptografia mais poderosa.

Dois princípios direcionaram a implementação da *JCA* e *JCE*. Primeiramente, elas devem suportar independência de algoritmos e extensibilidade. Esse primeiro princípio está ligado ao fato de um desenvolvedor poder escrever aplicações sem amarrá-las de maneira muito próxima a um algoritmo de criptografia em particular e, além disso, novos algoritmos devem ser facilmente integrados com aqueles já existentes.

O segundo princípio dita que a *JCA* e a *JCE* devem dar suporte à independência de implementação e interoperabilidade. Isso implica que um desenvolvedor pode escrever aplicações sem amarrá-las a implementações particulares de algoritmos de criptografia providas por terceiros. Além disso, implementações de algoritmos fornecidas por terceiros devem operar em conjunto com a arquitetura de segurança Java.

Devido a esses dois princípios, a *JCA* e a *JCE* foram construídas sobre os conceitos de *providers* e *engines*.

Um *provider* é um pacote ou biblioteca de terceiros que provê implementações de algum subconjunto da *API* de segurança Java, sendo que essa *API* vem com um *provider* padrão fornecido pela *SUN*. Desenvolvedores podem implementar suas próprias versões de algoritmos de criptografia populares ou implementar novos algoritmos de criptografia e empacotá-los como um *provider*.

[12] A JCE foi incorporada para uso em outros países além dos Estados Unidos da América a partir do JDK 1.4 [Sun, 2002].

Uma *engine* fornece uma interface uniforme para os *providers*. Por exemplo, a classe *Signature* é uma *engine* que pode gerar instâncias de assinaturas digitais pertencentes às classes providas pelos *providers* disponíveis.

Assim, essa abordagem deixa os desenvolvedores de aplicações livres de diversos detalhes sobre os algoritmos de criptografia que eles possam vir a utilizar e permite que os usuários dessas aplicações mudem de um *provider* para outro quando necessário[13].

[13] Várias bibliotecas relacionadas à criptografia são proibidas de serem exportadas (diversos países restringem a exportação de criptografia) e importadas (alguns países restringem o uso de criptografia).