ARMANDO LEMOS DA SILVA FILHO

ROTEIRO DE SEGURANÇA DA INFORMAÇÃO PARA ARQUITETURA ORIENTADA A SERVIÇO

Monografia apresentada à Escola Politécnica da Universidade de São Paulo para conclusão do curso de MBA em Tecnologia da Informação.

Área de Concentração: Tecnologia da Informação

Orientador: Prof. MSc. Renato Manzan

São Paulo
2008

FICHA CATALOGRÁFICA

Silva Filho, Armando Lemos da
Roteiro de segurança da informação para arquitetura orientada a serviço / A.L. da Silva Filho. -- São Paulo, 2007.
81 p.

Monografia (MBA em Tecnologia da Informação) - Escola Politécnica da Universidade de São Paulo. Programa de Educação Continuada em Engenharia.

1.Segurança da informação 2.Computação 3.Software de aplicação 4.SOA I.Universidade de São Paulo. Escola Politécnica. Programa de Educação Continuada em Engenharia II.t.

DEDICATÓRIA

Dedico este trabalho aos meus Pais,
Armando Lemos e Maria Lúcia

RESUMO

Roteiros práticos que demonstram como honrar aspectos básicos de segurança em sistemas tradicionais enumeram uma série de pontos chaves e críticos de sucesso para a implementação de novas aplicações. O objetivo deste trabalho é demonstrar os conceitos e características de segurança que devem fazer parte do planejamento de uma implementação de arquitetura orientada a serviço (SOA). O trabalho conceitua os principais aspectos de segurança, a mudança do paradigma de arquitetura e a implicação de segurança no novo contexto e novas situações que surgem por conta desta mudança.

Palavras-chave: Arquitetura Orientada à Serviço, Segurança da Informação, Sistemas de Informação, SOA, Controle de Acesso.

ABSTRACT

Practical processes that demonstrate how to migrate or evaluate a traditional system and comply with basic security principles is a key success factor when implementing new applications.

The objective of this paper is to demonstrate security concepts and features that must be part of the implementation planning for service oriented architecture (SOA). This paper defines the primary security aspects, the architecture paradigm change and the security implications in the new context and new situations that appear due to this model change.

Key-words: Service Oriented Architecture, Information Security, Information Systems, SOA, Access Control.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACL	Access Control List
CORBA	Common Object Request Broker Architecture
DAC	Discrete Access Control
DAC	Discretionary Access Control
EDA	Event-driven architecture
EJB	Enterprise Java Beans
ESB	Enterprise Service Bus
J2EE	Java 2 Platform, Enterprise Edition
JMS	Java Messaging Service
JSLEE	JAIN Service Logic Execution Environment
LUW	Logical unit of work
MAC	Mandatory Access Control
MDB	Message Driven Bean
MOM	Message-oriented middleware
MQ	Message Queue
MSMQ	Microsoft Message Queuing
ORB	Object request broker
REST	Representational State Transfer
ROI	Return On Investment
RPC	Remote procedure call
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SSO	Single Sign-On
TI	Tecnologia da Informação
TOCTTOU	Time-of-check to Time-of-use
WSDL	Web Service Description Language

SUMÁRIO

1 INTRODUÇÃO

CONSIDERAÇÕES INICIAIS

A tecnologia da Informação é uma necessidade não somente estratégica, mas sim um diferencial competitivo e que permite superar objetivos, respondendo aos desafios do mundo globalizado e oferecendo novas alternativas economicamente viáveis para garantir a sobrevivência das empresas e permitir novas formas de troca de conhecimento. Segundo a [ISO/IEC 27002:2005], define-se como segurança da informação como a preservação da confidencialidade, integridade e disponibilidade da informação, em complemento à outras propriedades tais como autenticação, responsabilidade de ações, não-repudiação e confiabilidade.

Alguns dos aspectos relevantes para segurança encontram respaldo na [ISO/IEC 9126-1:2001], nas categorias de funcionalidade e confiabilidade do modelo de qualidade interno e externo incluindo tolerância à falhas, usabilidade, eficiência e portabilidade e que expõe segurança como um dos recursos para viabilizar a qualidade de software.

Os novos mercados motivaram novas formas de pensar e de propor soluções de aplicação computacional, objetivando reaproveitamento dos recursos e evitando ciclos de desenvolvimento desnecessários.

Igualmente, o desafio de garantir que esta nova forma de pensar não implique em problemas maiores de segurança é a motivação desta monografia, em explorar os conceitos de segurança tradicionais e as situações práticas em que os mesmos são executados, ampliando para os novos modelos de pensamento e arquiteturas de aplicações.

OBJETIVO

O objetivo deste trabalho é evidenciar, através de um roteiro, a segurança para aplicações SOA (*Service Oriented Architecture*), ressaltando os aspectos necessários para separação da camada de aplicação e aproveitamento de serviços integrados de infra-estrutura para segurança, que devem estar disponíveis como serviços básicos para todas as aplicações.

Para tanto, este trabalho analisa os aspectos de segurança relevantes (identificação, autenticação, autorização e auditoria) e suas implicações em uma arquitetura orientada a serviços, considerando seus aspectos atômicos, mas também independentes nas aplicações e a mudança para uma demanda de integração que é o fundamento da mudança de paradigma que uma estrutura SOA precisa atuar.

O roteiro assume como entrada uma aplicação SOA em sua forma tradicional, e que deverá passar pelo processo de revisão descrito nesta monografia, para identificar o quê pode ser considerado como camada de segurança independente da lógica de negócio, aproveitando serviços de segurança que devem ser disponibilizados como infra-estrutura para aplicações SOA, e que também é defendido na argumentação desta monografia. No entanto, a definição dos serviços de segurança, soluções técnicas, fases de conversão de aplicações e como integrá-los em cada caso analisado real não fazem parte desta monografia, podendo ser objeto de estudos complementares.

JUSTIFICATIVA

Segundo (NATIS *et al.*, 2007), a arquitetura orientada a serviços está prioritariamente surgindo na maioria dos principais projetos de sistemas. Sua contribuição e impacto nas aplicações, em especial para segurança da informação, ainda merecem maior detalhamento e padronização, o que cria novos desafios e oportunidades no ciclo de desenvolvimento de sistemas.

Por conta desta prerrogativa, faz-se necessário avaliar aspectos de segurança da informação inerentes à identificação, autenticação e autorização, e seus requisitos de disponibilidade, confidencialidade e privacidade dentro de uma arquitetura orientada a serviços, mensagens, transações e que integram múltiplos aplicativos de forma colaborativa, mas ainda buscando preservar sua segurança atômica e principalmente entre aplicativos distintos e fora do domínio de segurança individual.

A segurança é um dos fatores críticos de sucesso em aplicações de alta confiabilidade, e as novas arquiteturas de aplicativos obrigatoriamente deverão estar capacitadas a lidar com aspectos fora do seu controle absoluto, muitas delas ligadas a conceitos e habilidades relativas à segurança.

Este estudo visa apresentar aspectos relevantes para uma aplicação ser considerada segura e capaz de trabalhar nos novos modelos de negócio

considerando-se a arquitetura orientada a serviços, propondo um roteiro e uma forma de estruturação para análise de uma aplicação e sua aderência aos aspectos de segurança relevantes, obedecendo à metodologia exposta.

METODOLOGIA

Esta monografia utilizou-se dos seguintes recursos para sua elaboração:

1.1.1 Fase levantamento de dados

A pesquisa de material relativo ao tema levou em conta três tipos de fontes principais: artigos publicados por instituições de pesquisa, recomendações de empresas e normas técnicas de domínio público, que pode ser consultada na relação de referências no final desta monografia. Além disso, foi explorado o estado da arte desses conceitos em um modelo de referência comercial conforme (BUECKER et al., 2007) para apoiar a proposta deste trabalho e dar respaldo a sua aplicação imediata e justificada no contexto atual. Por fim, o conhecimento sobre segurança da informação, arquitetura SOA e oportunidades de aproveitamento de segregação e reuso de processos de segurança até o nível mais flexível de serviço foi usado para direcionar o trabalho e criar a argumentação de integração destes dois temas.

1.1.2 Fase elaboração e montagem da proposta de trabalho

Nesta fase são conceituados os aspectos de segurança e de arquitetura SOA, levando em conta as diferenças que SOA propicia em relação à serviços distribuídos e a preocupação em manter aspectos de segurança como identificação, autenticação, autorização e auditoria suficientemente confiáveis para uso em um ambiente não mais controlado por uma aplicação exclusiva, mas dependente de um eco-sistema de aplicativos colaborativos, sendo a segurança um dos pilares necessários para esta infra-estrutura.

1.1.3 Fase estudo de caso

Nesta fase, usam-se os conceitos resumidos do roteiro de segurança para selecionar e avaliar os aspectos de segurança que devem ser; (a) adicionados ao contexto da aplicação (b) modificados em relação ao seu uso e (c) removidos por se tratar de algo inerente a infra-estrutura, e portanto, não de responsabilidade individual da aplicação.

1.1.4 Fase análise dos resultados

Esta fase finaliza as experiências obtidas e sinaliza a importância e o aproveitamento que pode ser obtido ao usar a camada de segurança como um serviço, liberando as aplicações de preocupar-se com esta lógica de programação e abrindo potencial de flexibilidade para estas aplicações executarem em ambientes computacionais complexos.

2 SEGURANÇA

Segurança de Informação está relacionada com métodos de proteção aplicados sobre um conjunto de dados no sentido de preservar o valor que possui para um indivíduo ou uma organização. São características básicas da segurança da informação os aspectos de confidencialidade, integridade e disponibilidade, não estando restritos somente a sistemas computacionais, informações eletrônicas ou sistemas de armazenamento. O conceito se aplica a todos os aspectos de proteção de informações e dados.

Atualmente o conceito de segurança está padronizado pela norma ISO/IEC influenciado pelo padrão inglês (*British Standard*) BS 7799. A série de normas ISO/IEC 27000 foi reservada para tratar de padrões de Segurança da Informação, incluindo a complementação ao trabalho original do padrão inglês. A ISO/IEC 27002:2005 continua sendo considerada formalmente como 17799:2005 para fins históricos.

CONCEITOS DE SEGURANÇA

A história da computação mostra que no passado, os recursos de hardware e software eram disponíveis apenas para pessoas com acesso especial, como por exemplo, estudantes que necessitavam de recursos de grandes computadores centralizados, onde normalmente não havia tempo e recursos disponíveis para executar os trabalhos completos, o que implicava na execução de parte dos trabalhos de forma manual, fora do ambiente de computação. Embora estas condições restringiam o uso livre dos computadores, elas também dificultavam o uso indevido dos recursos computacionais de forma maliciosa. Com o avanço do hardware e sua redução em termos de tamanho e preço, e considerando os recursos de conectividade e acesso remoto que permitem acesso rápido e fácil aos computadores, o processo administrativo de controle de segurança que era usado para proteção dos grandes centros tornou-se obsoleto. Em outras palavras, os avanços na usabilidade do sistema contabilizam uma perda em segurança. Pessoas remanescentes da antiga filosofia de segurança ainda confiam em métodos como

portas trancadas para evitar acesso aos dados de forma indevida. Tais medidas são obviamente inadequadas quando consideramos o recurso de acesso remoto nos sistemas de hoje. No entanto, e menos óbvio de que mesmo sem o acesso remoto, a simples obtenção de software por fontes externas maliciosas provê os meios para um ataque de segurança.

Pode-se verificar (SUMMERS, 1984) que um dos motivos para medidas de segurança é proteger a privacidade dos indivíduos, assim como dar algum controle sobre a informação que é mantida em sistemas de computadores. Um resumo dos princípios de privacidade adotados pelos membros da Organização para cooperação econômica e desenvolvimento (OECD – *Organization for Economic Cooperation and Development*) inclui:

- Deve haver limites para a coleta de dados pessoais, e esses dados devem ser obtidos de forma legal e honesta, com a ciência e consentimento do sujeito, quando aplicado. O sujeito representa a pessoa sobre a qual os dados estão sendo coletados;
- Os dados devem ser relevantes para os propósitos para os quais foram coletados, assim como serem precisos, completos e atuais;
- Os dados não devem, em geral, ser usados para outros propósitos;
- Os dados devem ser protegidos por medidas razoáveis contra perda, acesso indevido, destruição, modificação ou disseminação. Em outras palavras, deve-se prover segurança de dados;
- Deve ser possível descobrir quais sistemas de registro existem, seu propósito principal, bem como as pessoas que são responsáveis pelo controle dos registros;
- Um indivíduo deve ser capaz de descobrir se uma empresa detém informações sobre ele, e neste caso, obter acesso a esta informação, devendo ser capaz de questionar tais dados, e se for o caso, solicitar que sejam apagados ou corrigidos.
- Um administrador de dados deve ser responsável por adequar as medidas de implementação destes princípios.

Considere os aplicativos que são utilizados nos computadores pessoais, onde alguns deles não possuem nenhum tipo de segurança inerente. Por exemplo, uma calculadora eletrônica não depende de nenhum dado para usá-la. Por outro lado, existem aplicativos que gerenciam dados pessoais, onde nestes casos são normalmente controlados por um processo de senha (Um correio eletrônico pede um usuário e senha antes de abrir as mensagens). Existem ainda outros tipos de aplicativos que utilizam esquemas de segurança ainda maiores. O sistema operacional não permite troca de arquivos a menos que o usuário tenha privilégios de administrador. O navegador web (*web browser*) normalmente criptografa todos os dados que são trocados com o banco.

Gerenciamento de segurança em ambientes de TI é comumente classificado em segurança de rede (*network security*); segurança de plataforma (*platform security*) e segurança de aplicação (*application security*). Dentre estas, as duas primeiras tem pouco relacionamento com segurança SOA conceitualmente, o que faz com que segurança de aplicação seja o foco maior deste trabalho.

Uma das motivações fundamentais de segurança é garantir o uso dos sistemas computacionais para os usuários. A qualidade desta oferta passa pela garantia da estabilidade do ambiente envolvendo os componentes de hardware e software dos mais variadas complexidades e criticidades (AMOROSO, 1994). O primeiro dilema em segurança passa pelo entendimento do que pode comprometer este uso continuado, que representa a taxonomia de eventos e situações negativas para um sistema de computadores.

2.1.1 Ameaças, vulnerabilidades e ataques

Existe um conjunto de situações em que os computadores podem ser submetidos e que fazem parte dos conceitos básicos da computação. Uma ameaça a um sistema de computação é definida como qualquer ocorrência potencial, seja ela maliciosa ou não, a qual tenha um efeito indesejável nos bens e recursos associados com o sistema de computadores. De uma forma simples, uma **ameaça** trata de algo ruim que pode ocorrer no sistema. (AMOROSO,1994)

Uma **vulnerabilidade** em um sistema de computador representa alguma característica inadequada que torna possível uma ameaça ocorrer. Uma

vulnerabilidade permite que coisas ruins ocorram em um sistema de computador. Desta forma, as ameaças aos sistemas de computador podem ser reduzidas ao se identificar e remover vulnerabilidades. (AMOROSO,1994)

Um **ataque** em um sistema de computador representa alguma ação executada por um intruso que envolve a exploração de uma vulnerabilidade com a finalidade relativa à uma ameaça que possa ocorrer. Os ataques envolvem normalmente conhecimento da vulnerabilidade por parte do atacante (*hacker*). Embora esta definição desconsidere os ataques provenientes de erros inocentes, para efeitos de modelagem de sistemas de segurança, deve-se ter contramedidas para identificar e lidar com ambos os casos. (AMOROSO,1994)

Para entender as possíveis ameaças e a maneira como elas são endereçadas pelos sistemas de segurança, (AMOROSO,1994) categoriza as ameaças em três tipos distintos, considerando que a maioria das ameaças em um sistema de computadores pode ser vista como relacionada a um tema de disseminação, integridade ou negação de serviço, conforme explicado a seguir:

- **Disseminação**: Esta ameaça envolve a disseminação de informações para um indivíduo para o qual não deveria ser vista. No contexto da segurança de computadores, a ameaça de disseminação ocorre quando uma informação secreta que está armazenada no sistema ou está em trânsito entre dois sistemas é comprometida para alguém que não deveria conhecer aquele segredo. Às vezes o termo "vazamento" é usado em conjunto com uma ameaça de disseminação. Exemplos incluem informações pessoais divulgadas para colegas, dados sigilosos como construção de produtos e estratégias militares, bem como divulgação de informações de saúde. Esta categoria também é referenciada na literatura de segurança como confidencialidade.
- **Integridade**: Esta ameaça contempla a mudança não autorizada de dados armazenados em sistema de computadores ou em trânsito entre dois sistemas. Quando um intruso maliciosamente altera um dado, diz-se que a informação foi comprometida. Esta situação vale também para um erro inocente que resulta em mudanças não autorizadas. Mudanças autorizadas são aquelas realizadas por indivíduos com propósitos justificáveis. Assim como na ameaça de disseminação, a ameaça de integridade pode ter conseqüências leves quando dados não críticos são modificados. No entanto, esta situação pode também ser extrema, se

dados médicos forem alterados e não forem percebidos, com eventual perda de vidas humanas como resultado deste tipo de ataque.

- **Negação de serviço**: Este tipo de evento ocorre quando o acesso a um determinado recurso do sistema é intencionalmente bloqueado, como resultado da ação maliciosa de um usuário. Ou seja, se um usuário solicita um determinado recurso e este recurso não pode ser atendido por conta de uma ação que previne o sistema de atendê-lo, então houve uma negação de serviço. Esta categoria é também referenciada na literatura de segurança como disponibilidade.

Uma aplicação disponível para múltiplos usuários e que compartilha dados tem que cobrir os aspectos a seguir.

2.1.2 Identificação

A identificação pode ser definida como o processo ou mecanismo que permite que um agente externo ao sistema de computadores notifique o sistema sobre sua identidade. A necessidade de identidade surge quando se deseja associar cada ação com um causador desta ação. Os sistemas de computadores podem determinar quem invocou uma determinada operação examinando a identidade reportada do agente que iniciou a sessão na qual aquela operação estava envolvida. Esta identidade é tipicamente estabelecida por meio de uma seqüência de *login*. Por exemplo, na maioria dos sistemas os usuários possuem uma identificação de *login* que é a primeira coisa mostrada quando o sistema é ligado. Este processo depende do fornecimento da identificação de usuário (*userid*), normalmente expressa como uma sigla ou abreviação (ex: admin, alemos, T001, etc.). Este processo demonstra a forma mais simples possível que compõe a identificação. Importante observar que existe uma distinção entre a suposta declaração de identidade do usuário e a sua efetiva veracidade. Como conseqüência disso, a identificação é combinada com outros aspectos para assegurar que a seqüência de identificação está correta e pode ser acreditada.

2.1.3 Verificação da identidade dos usuários

A autenticação pode ser definida com sendo o conjunto de procedimentos e mecanismos que permitem ao sistema de computação assegurar que uma determinada identidade apresentada ao sistema está correta. A autenticação envolve normalmente alguma forma de validação para produzir evidência ou confiança que a identidade reportada deva ser válida. Em muitos sistemas, este processo é implementado com a identificação seguida de uma senha que é usada como prova da identidade.

Existem várias abordagens para autenticação. Existem três grupos distintos que podem ser combinados entre si:

- Algo que você sabe: O tipo mais disseminado de autenticação envolve o conhecimento de algo. As senhas representam a forma mais comum onde, este segredo é a base da confiança. Um processo alternativo envolve o recurso associativo ou de perguntas e respostas, que devem ser respondidas pelo usuário. A sofisticação destes recursos pode variar na quantidade e qualidade das perguntas para evitar o problema de quebra de confidencialidade, muito mais sensível no caso de uma senha. De qualquer forma, estes métodos baseiam-se em dados previsíveis no tempo, mesmo que usados de forma dinâmica ou combinada.
- Algo que você possui: Esta abordagem normalmente prevê o uso de um componente físico que tem a propriedade de assegurar um elo entre o sistema computacional e um usuário previamente cadastrado para uso deste recurso. Um exemplo bastante convencional é uma chave ou um *smart-card*, onde pode-se armazenar um segredo que torna difícil para um usuário malicioso burlar. É importante notar que um nível de segurança mais sofisticado deve compor um segredo (algo que você sabe) com um dispositivo (algo que você possui) para dificultar ainda mais a possibilidade de falhas ou ataques.
- Algo que você é: Como uma terceira forma de autenticação, esta envolve o uso de alguma característica inerente ao agente que participa da autenticação. Esta forma é normalmente usada com seres humanos, utilizando recursos como reconhecimento de voz, impressões digitais ou padrões de retinas, cada uma com um nível de confiabilidade e aplicação com outras formas de autenticação para garantir sua confiabilidade (também conhecido como problema do falso-positivo). Esta forma é a mais cara de todas e a mais complexa por necessitar de equipamentos específicos e que são recentes quando comparados com as outras

técnicas em uso, o que dificulta também a sua utilização massiva e apresenta outras dificuldades de implementação além do escopo técnico deste trabalho.

Uma premissa importante é que a autenticação de um usuário seja feita durante uma sessão de trabalho, onde os recursos computacionais são alocados e disponibilizados para o usuário, de forma que o sistema controle o que está sendo feito. Caso a identificação do usuário mude durante uma sessão, o sistema deve ter meios de detectar e validar esta mudança (como por exemplo, o comando su - *super user* do ambiente UNIX). Esta condição é necessária para garantir o registro adequado e o controle das operações executadas pelo usuário durante toda a sua sessão de trabalho.

Uma condição vital para autenticação é a confiança no canal de comunicação entre o terminal que o usuário utiliza e o sistema que pode estar remoto. O caminho seguro (*trusted path*) é a comunicação direta entre o usuário e a rotina no sistema responsável pelo processo de autenticação, de forma que não possa ser burlada. Dessa forma, o caminho seguro deve prover confiança para o usuário, ao contrário dos esquemas anteriores de autenticação.

2.1.4 Política de segurança

Virtualmente todos os sistemas de computação podem ser modelados em termos de sujeitos e objetos para acesso. Por exemplo, quando um usuário inicia uma sessão de trabalho o sistema opera um conjunto de processos em nome deste usuário para acessar recursos do sistema. Esta condição sugere que existe um processo de decisão para mediar quais acessos podem prosseguir e quais não podem. Uma outra forma de pensar na mediação seria um filtro nos quais os acessos precisam passar.

Este filtro é conhecido como monitor de referência (*reference monitor*) ou modelo de referência. Esta terminologia surgiu na época dos antigos sistemas operacionais quando os sistemas eram também conhecidos como monitores. A funcionalidade básica pode ser vista na figura 1. O fluxo de entrada e saída indicado demonstra o sujeito e recurso no processo normal, abrangendo as regras e políticas de acesso que modelam e alteram o resultado final, gerando um evento de controle (trilha de auditoria) conforme o caso. Uma entrada (sujeito) é submetida ao modelo de

referência, que por sua vez utiliza entrada de conhecimento de políticas e regras, produzindo uma decisão de acesso como saída e um registro de auditoria pertinente. Note que um sujeito deve produzir uma decisão de acesso, assim como o conjunto de políticas e regras definir um evento de auditoria.

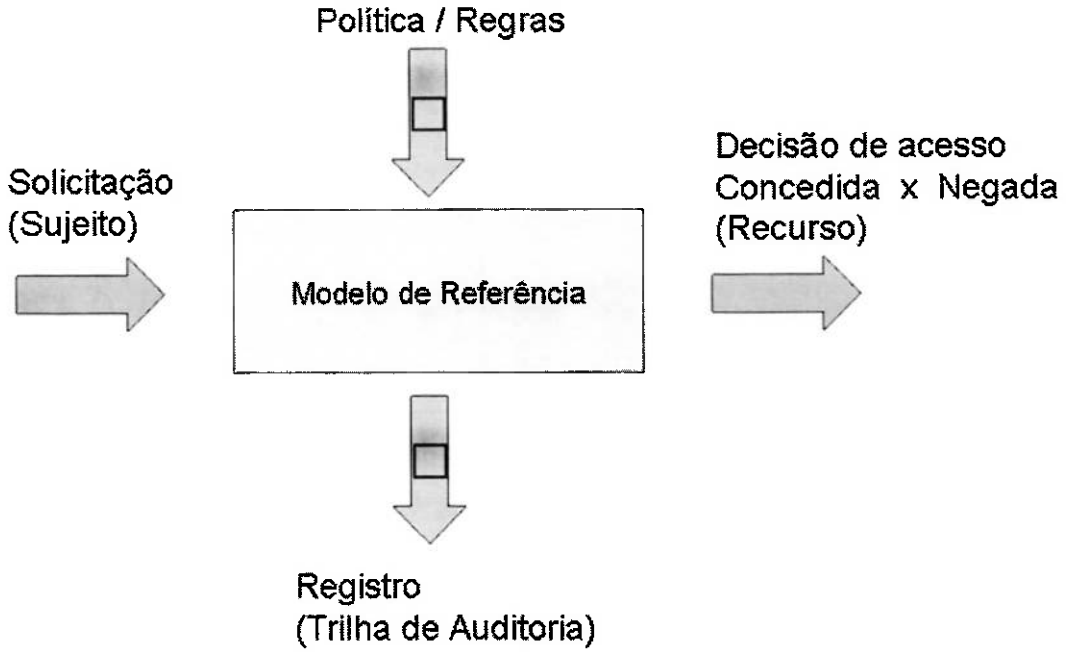

Figura 1 – Modelo de referência

A política de segurança define as condições nas quais um acesso de um sujeito é mediado pela funcionalidade do monitor de referência. Estas condições são representadas normalmente por um conjunto de regras de segurança para o referido sistema. Uma política de segurança define as regras pelas quais os sujeitos poderão operar. Como é de se esperar, deve haver mecanismos adequados no sistema para assegurar as regras desejadas da política de segurança. As políticas de segurança podem ser expressas em uma notação informal como formal. Para o caso informal, adota-se uma tabela simples que representa determinadas premissas de operação do sistema, incluindo uma coluna para os vários tipos de acesso definidos e uma outra coluna para as relações correspondentes que devem ocorrer entre os sujeitos e objetos naquele tipo de acesso. A tabela 2.1 demonstra esta relação.

Tipo de Acesso	Relação (Sujeito;Objeto)
Leitura	Domínio
Execução	Domínio
Escrita	Igual

Tabela 2.1 – Tipo de acesso x relação (adaptado de AMOROSO,1994)

A tabela 2.1 define a existência de uma função entre o sujeito e um objeto, de tal forma que para ter acesso de leitura, o modelo de referência deve validar que o sujeito tem um acesso maior ou igual ao declarado para o recurso, caracterizando assim a relação de domínio. A partir desta tabela, e considerando uma primitiva de segurança de domínio, pode-se concluir que o nível de segurança dos sujeitos deve dominar ou exceder o nível do objeto para que as operações de leitura e execução sejam concedidas. Da mesma forma, pode-se concluir que o nível do sujeito para uma operação pode ser equivalente ao do recurso para permitir uma operação de escrita.

Considerando uma notação formal, as expressões de segurança serão demonstradas usando uma coleção de funções com valores de retorno booleanos que se referem a conjuntos dos diferentes acessos, sujeitos e objetos que existem em um sistema de computação. Estas funções oferecem uma notação mais formal para expressar as políticas de segurança. Embora este seja um conjunto de notações possíveis, vale ressaltar que este não é o único meio de expressar uma política de segurança, e foi escolhido devido a sua natureza de funções de retorno booleanas para criar expressões matemáticas rapidamente e sem precisar de muitos detalhes técnicos.

As funções com retorno booleano descrevem expressões matemáticas simples na forma (1) abaixo:

allow(s,o,a) iff P **(1)**

Onde esta regra especifica que em todos os estados do sistema, um sujeito *s* pode solicitar um acesso *a* no objeto *o* se e somente se a propriedade *P* for verdadeira. Considerando que a função pode ser para um conjunto de *s* ítens do universo S, de *a* acessos do universo A e de *o* objetos do universo O, a sintaxe expandida (2) deve ser:

qq s v S, o v O, a v A: allow(s,o,a) iff P **(2)**

Alguns exemplos de regras formais incluem a regra de acesso totalmente aberta:

Sem restrição: qq s v S, o v O, a v A: allow(s,o,a) iff true

Ao estipular que todas as ocorrências de *s*, *o*, *a* retornam o valor verdadeiro, esta regra define um acesso universal sem restrição para o recurso.

Uma segunda mediação bastante desejada nos sistemas de segurança trata da propriedade de um dado, expressa pela seguinte política:

propriedade: qq s v S, o v O, a v A: allow(s,o,a) iff own(s,o)

Neste caso, os acessos são mediados de forma a permitir apenas aqueles sujeitos que são donos de objetos nos quais desejam operar.
Uma terceira política refere-se aos direitos administrativos, expressa por:

admin: qq s v S, o v O, a v A: allow(s,o,a) iff admin(s)

Esta condição permite que administradores tenham acesso a qualquer objeto, de forma a superar a política de propriedade.
Por fim, um exemplo de política para controle de classificação de informação seria:

domínio: qq s v S, o v O, a v A: allow(s,o,a) iff dominates(label(s), label(o))

Esta política demonstra como os rótulos de classificação e segurança de dados seriam usados como base da política de segurança e mediação do acesso.
A diferença primária entre todas estas políticas e as políticas implementadas em sistemas reais é que a parte P da regra normalmente é mais complexa e requer páginas de documentação e explicação.
Para efeitos deste estudo, é suficiente assumir que deve haver um conjunto de regras explícitas para o modelo de referência de segurança avaliar os acessos de sujeitos em objetos em tempo de execução, e que dependendo da dinâmica das entidades (sujeitos, acessos e objetos) estas regras podem ser modificadas para satisfazer necessidades de controle diferentes, e que tanto os modelos informais como os formais podem ser usados para esta representação.

2.1.5 Controle de acesso

Os conceitos descritos até o presente momento sugerem a participação de certos controles de acesso para prover proteção contra ataques maliciosos. Em todos os casos, o controle de acesso é entendido como uma implementação do modelo de refêrencia com base em políticas de segurança. Os mecanismos de controle de acesso representam a forma que a maioria dos projetistas de segurança asseguram

a funcionalidade do monitor de referencia, onde é definido em termos de sujeitos e objetos. Especificamente, a definição de controle de acesso são os mecanismos que garantem a mediação das solicitações de sujeitos para acesso a recursos dentro de uma política de segurança. O termo autorização é usado neste documento de forma intercambiável para expressar o resultado que o modelo de referência produz, normalmente expresso em uma condição de permissão ou de negação.

Existem duas categorias para sistemas de acesso, sendo o controle de acesso discreto e o controle de acesso mandatório.

Em um mecanismo de acesso discreto, os processos e mecanismos para assegurar a mediação de segurança são verificados no nível do usuário. Os sistemas DAC (*Discrete Access Control*) são vistos como tendo um conjunto de parâmetros definidos pelo usuário que podem afetar o processo de mediação do sistema. Obviamente, isso requer que a política de segurança expresse de forma suficiente estas operações e que esta capacidade de alteração não implique numa violação de política. De uma forma geral, os mecanismos DAC provêem flexibilidade para proteger arquivos e recursos computacionais configurando parâmetros DAC conforme a necessidade. A vantagem óbvia é que este esquema provê grande flexibilidade para os usuários, mas também apresenta o risco de proteção inadequada quando estes mesmos usuários esquecem de proteger os dados ou então não atentam para a proteção efetiva necessária. Além disso, como os parâmetros DAC são facilmente modificáveis, os usuários protegidos por recursos DAC são suscetíveis a ataques de cavalo de tróia.

Em contrapartida, o controle de acesso mandatório é definido abrangendo os procedimentos e mecanismos que reforçam a mediação especificada, não no nível do usuário, mas de acordo com um sistema centralizado de administração. Os parâmetros MAC (*Mandatory Access Control*) somente podem ser definidos por um administrador ou oficial de segurança. Os mecanismos MAC não permitem que o usuário altere os parâmetros de controle de acesso, centralizando esta tarefa com uma pessoa de sistema ou de segurança. A vantagem do esquema MAC é que ele permite a proteção de arquivos importantes e é menos suscetível a ataques de cavalo de tróia. No entanto, os mecanismos MAC são significativamente menos flexíveis que DAC e normalmente impõem limites para troca de dados de uma forma inaceitável para os usuários.

Um modelo para representar as abordagens DAC como MAC é a matriz de acesso. A matriz de acesso é uma tabela *N* x *M*, sendo que cada linha corresponde a um dos *N* sujeitos e cada coluna corresponde a um dos *M* objetos. Cada célula da matriz define os direitos de acesso que aquele determinado sujeito pode ter no objeto, conforme a figura 2.

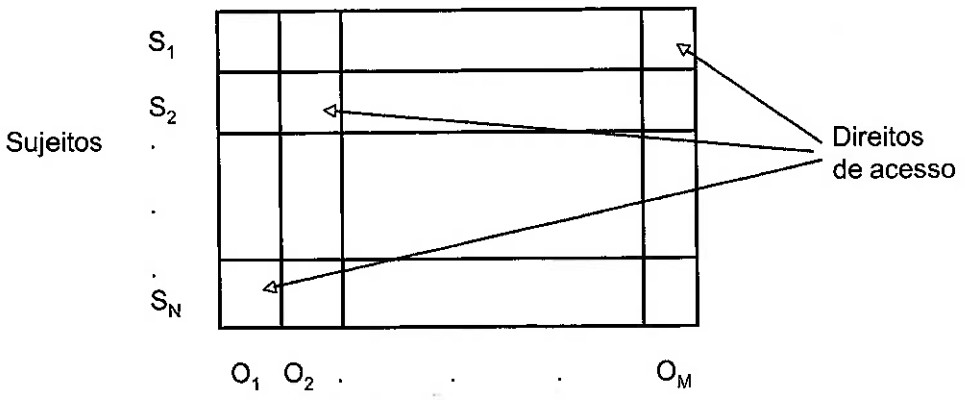

Figura 2 – Matriz de acesso (adaptado de AMOROSO, 1994)

Um tipo de mecanismo de controle são as listas de controle de acesso ACL (*Access Control List*), que são estruturas onde para cada objeto, uma lista de sujeitos válidos é atribuída. Uma ACL é normalmente um esquema DAC pois a administração é mais adequada ao nível de usuários individuais. A tabela 2.2 mostra uma ACL para os objetos X, Y e Z e os sujeitos A, B e C:

Objeto	Sujeito
X	A, B
Y	A
Z	A, B, C

Tabela 2.2 – Lista de controle de acesso (adaptado de AMOROSO,1994)

Esta representação abstrai a noção de direitos de acesso por questões de simplicidade. Em um sistema real, os sujeitos podem ter direitos específicos para os recursos conforme a tabela 2.3.

Objeto	Sujeito (acesso)
X	A(leitura), B(execução)
Y	A(escrita)

Z	A(escrita, execução), B(leitura), C(leitura, execução)

Tabela 2.3 – Lista com tipos de acesso (adaptado de AMOROSO,1994)

Neste caso, o recurso *X* pode ser lido pelo sujeito *A* e pode ser lido ou executado pelo sujeito *B*. Assume-se que existe uma hierarquia de direitos, onde ler < executar < gravar.

Um conceito que se parece muito com a ACL é o mecanismo de controle de capacidades. Ao invés de associar uma lista de sujeitos com objetos, capacidades associa listas de objetos com sujeitos. Desta forma, em termos de modelo de acesso, uma capacidade para um determinado sujeito será os valores de todos os objetos que estão especificados em uma linha da matriz. Considerando a representação ACL para este caso, a estrutura para os sujeitos A, B e C e recursos X, Y e Z é mostrada na tabela 2.4.

Sujeito	Recurso
A	X, Y, Z
B	X, Z
C	Z

Tabela 2.4 – Lista de capacidades (adaptado de AMOROSO,1994)

As capacidades são representadas como ACL reversas com o sujeito *A* podendo acessar os objetos *X*, *Y* e *Z*. Assim como as ACL, um modelo de capacidades pode representar dados mais refinados de tipos de acesso.

Considerando os sistemas comerciais e sua implementação de segurança centrado em mecanismos de ACL, não faz parte deste estudo a análise de modelos MAC de aplicações, devendo ter um estudo aprofundado e detalhado para este tema.

2.1.6 Auditoria

O mecanismo de auditoria trata de reduzir os efeitos de diferentes tipos de ataques, usando mecanismos automáticos e procedimentos de associação que causam um registro de uma atividade do sistema ser gerada e armazenada em um local protegido como um registro computadorizado. Esta coleção de registros (*log*) provê os meios para identificar a hora, origem e outras informações importantes sobre um determinado comportamento dos usuários no sistema. Os requisitos de um sistema de auditoria em um sistema são os seguintes:

- Mecanismos e processos: Um mecanismo de auditoria deve estar presente para automaticamente obter dados sobre a atividade do sistema e colocá-los em uma *log*. Deve haver procedimentos para interpretar e recuperar os dados de auditoria no *log* protegido.
- Registro de atividade relevante: Um registro de todas as atividades suspeitas deveria ser colocado no *log* de auditoria. O termo atividade crítica é normalmente usado para cobrir este tipo de atividade. Para abranger todas atividades críticas, um sistema de auditoria pode requerer a inclusão de todos os tipos de atividades possíveis no sistema (por exemplo, incluindo leitura e gravação de arquivos). No entanto, esta granularidade permite a reconstrução de um ataque.
- Efeito mínimo em funcionalidade: O esquema de auditoria não deve penalizar o desempenho normal do sistema, incentivando, por exemplo, que os administradores desabilitem a sua atividade.
- Formato do registro de auditoria: Os registros de auditoria devem ser construídos obedecendo a blocos bem-definidos para assegurar a interpretação fácil.
- Registro em *log* protegida: Uma vez que o registro de auditoria tem a finalidade de reconstruir os passos de um ataque, e necessário que os *logs* sejam protegidos contra ataques, pois um dos passos que um usuário malicioso incluiria no seu processo seria cobrir os rastros na trilha de auditoria.

2.1.7 Confidencialidade

Os dados trocados via rede ou armazenados pelos aplicativos precisam de medidas que os preservem da visão indiscriminada. Uma técnica empregada para salvaguardar os dados de forma confidencial é usar criptografia, a qual é largamente aplicada em operações via *web* e permite que apenas as entidades origem e destino tenham condições de entender a mensagem enviada.

2.1.8 Não-repudiação

Uma mensagem autêntica não pode ser negada pela sua origem nem pelo seu destinatário. Por exemplo, uma entidade origem, ao enviar uma ordem de venda de um produto, não pode posteriormente alegar que a ordem não foi gerada ou

transmitida. Ao mesmo tempo, uma entidade receptora precisa ter certeza da origem e do conteúdo que a mensagem pode ser executada na sua íntegra. A técnica utilizada para atingir este objetivo é a assinatura digital, que provê capacidades de reconhecimento de integridade para mensagens.

2.1.9 Checagem de sistema

Existem várias situações não cobertas na literatura de segurança que passam pela exploração de condições internas de aplicativos, sistemas operacionais e fatores humanos na operação que apresentam riscos para a segurança dos aplicativos em execução. Classes de ataques tais como negação de serviço (*denial of service*) ou exploração de erros fatais no código do sistema (*buffer overflows*) são exemplos destas condições. Normalmente, estas situações tiram proveito da ausência de checagem de parâmetros, limites de variáveis internas do programa (vetores, cadeias de caracteres), ou de concorrência entre processos (*race conditions*), sendo esta última, conhecida na literatura de segurança como TOCTTOU (Time-of-check to Time-of-use). Trata-se de uma anomalia de software onde se executa mudanças entre o tempo da checagem de uma condição de segurança (tal como a verificação de uma credencial de usuário) e o uso do resultado desta checagem. Esta condição também pode ser explorada em processos de autorização, onde o sistema constrói a lista de recursos autorizados para um usuário, mas sem validar a veracidade futura desta informação durante a execução das operações.

2.1.10 Privacidade

Com relação a direito, a privacidade estipula restrições na divulgação de dados pessoais para outros. Isso apresenta duas visões possíveis para uma aplicação. Uma diz respeito a como os dados são armazenados e liberados. A outra diz respeito aos serviços propriamente ditos e como os dados são expostos para uso.
Todas estas questões de segurança são bem entendidas e mapeadas conforme AMOROSO(1994) e em SUMMERS(1984).
Existem diversas literaturas que as descrevem e auxiliam na sua resolução. Podem-se encontrar sugestões de reaproveitamento de código, bibliotecas de programas e

frameworks que auxiliam na implementação das soluções. No entanto, todas elas lidam tipicamente com a proteção de uma única aplicação (que pode ser um aplicativo individual ou um conjunto de aplicativos distribuídos em várias camadas) ou em aplicações cliente-servidor.

Considerando os 10 temas abordados nesta seção, de uma forma geral o roteiro de segurança desta monografia aborda estes temas na forma direta para 4 disciplinas (identificação, autenticação, autorização e auditoria), e utiliza os conceitos relativos a política se segurança, confidencialidade, privacidade e não-repudiação para explicar a necessidade dos serviços de segurança, cuja motivação recai sobre o risco de ameaças, vulnerabilidades e ataques e que é o objetivo e motivação de se pensar em segurança.

3 SERVICE ORIENTED ARCHITECTURE

Service Orientented Architecture (SOA), ou em português, arquitetura orientada a serviços (AOS), é um estilo de arquitetura de software cujo princípio fundamental preconiza que as funcionalidades implementadas pelas aplicações devem ser disponibilizadas na forma de serviços. Freqüentemente estes serviços são organizados através de um "barramento de serviços" (*enterprise service bus*) que disponibiliza interfaces, ou contratos, acessíveis através de *web services* ou outra forma de comunicação entre aplicações. A arquitetura SOA é baseada nos princípios da computação distribuída e utiliza o paradigma *request/reply* para estabelecer a comunicação entre os sistemas clientes e os sistemas que implementam os serviços.

Uma solicitação (*request*) é a formulação e envio de um pedido para um sistema que deve receber e processar os dados recebidos, normalmente produzindo uma ação e uma resposta. Uma resposta (*reply*) é um retorno que é entregue ao solicitante, em contrapartida a uma informação iniciada por ele. Este paradigma trata essencialmente de uma forma de obter uma interação síncrona sobre um mecanismo assíncrono. De uma forma geral, assume-se que uma solicitação deve produzir uma resposta, e que esta deve produzir uma ação de finalização com sucesso ou falha. Um caso bastante utilizado são as transações, caracterizadas pela rápida finalização uma vez iniciadas.

Além da perspectiva estritamente técnica, a arquitetura orientada a serviços também se relaciona com determinadas políticas e conjuntos de "boas práticas" que pretendem criar um processo para facilitar a tarefa de encontrar, definir e gerenciar os serviços disponibilizados.

A arquitetura orientada a serviços também se insere em um processo de reorganização dos departamentos de tecnologia da informação das organizações, permitindo um melhor relacionamento entre as áreas que dão suporte tecnológico à empresa e as áreas responsáveis pelo negócio propriamente dito, graças a maior agilidade na implementação de novos serviços e reutilização dos ativos existentes.

SERVIÇO

O serviço, no ponto de vista de SOA, é uma função de um sistema computacional que é disponibilizado para outro sistema na forma de um serviço. Um serviço deve funcionar de forma independente do estado de outros serviços e deve possuir interface bem definida. Normalmente, a comunicação entre o sistema cliente e aquele que disponibiliza o serviço é realizada através de *web services*.

PROCESSOS

Mais do que uma tecnologia, SOA também influencia regras e processos de negócios, além de muitas vezes implicar reengenharia de software simultaneamente. - Gartner Group

A Arquitetura Orientada a Serviços pode ser bem representada a partir do seguinte processo, chamado de "*find-bind-execute paradigm*" o que significa aproximadamente paradigma de "procura-consolida-executa". Tal conceito é análogo a "Ciclo de Deming" aplicado aos serviços, que define o ciclo que envolve o planejamento, a execução, o monitoramento e a tomada de ação pró ativa para a melhoria da qualidade. A figura 3 exemplifica o ciclo do processo.

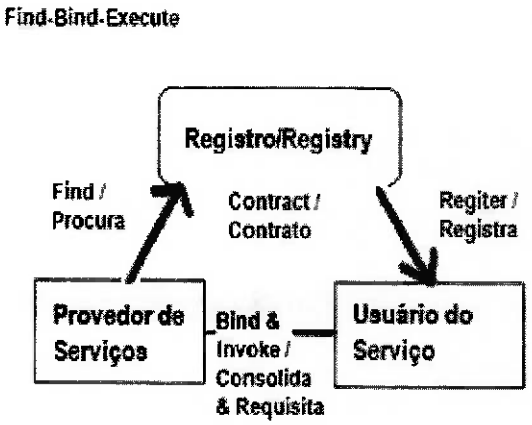

Figura 3 - Esquema adaptado do paradigma "*find-bind-execute*"

Do ponto de vista técnico, o processo considera que os provedores de serviços registrem informações em um registro central, com suas características, indicadores, e aspectos relevantes a tomadas de decisões. O registro é utilizado pelo cliente para

determinar as características dos serviços necessários, e se o mesmo estiver disponível no registro central, como por exemplo, por um catálogo de serviços, o cliente poderá utilizá-lo, sendo este oficializado através de um contrato que enderece este serviço.

TECNOLOGIA

O SOA expressa um conceito onde aplicativos ou rotinas são disponibilizadas como serviços em uma rede de computadores (Internet ou Intranet) de forma independente e se comunicando através de padrões abertos. A maior parte das implementações de SOA se utilizam de *web services* (SOAP-*Simple Object Access Protocol* , REST-*Representational State Transfer* e WSDL-*Web Service Description Language*). Entretanto, uma implementação de SOA pode se utilizar de qualquer tecnologia padronizada baseada em *web*.

DEFINIÇÕES DOS TERMOS RELACIONADOS A SOA

Em um estudo (SCHULTE; ABRAMS, 2006), o termo arquitetura orientada a serviço é descrito como um estilo de arquitetura para aplicações de negócio que são modulares, passíveis de serem distribuídas, compartilhadas de uma forma de baixa ligação ou dependência.

Para que uma aplicação seja considerada aderente aos padrões SOA, os seguintes princípios precisam ocorrer:

1. A aplicação precisa ser modular.
2. Os módulos devem ser passíveis de distribuição.
3. Os desenvolvedores de software devem codificar ou gerar interfaces de meta-dados que especificam um contrato explicitamente, de forma que outros desenvolvedores possam encontrar e usar o serviço (para auxiliar o passo de baixa ligação).
4. A interface do serviço precisa ser separada da implementação (código e dados) do módulo provedor de serviço (este passo auxilia a baixa ligação).

5. Os provedores de serviço devem ser compartilháveis, ou seja, projetados e implementados de uma maneira que permita que sejam chamados sucessivamente por consumidores diversos.

Na figura 4, o termo serviço é explicado como um processo repetitivo dentro de uma empresa e que tem um comportamento bem definido e que pode ser compartilhado para uso comum de múltiplos sistemas. A orientação a serviço nasce justamente da condição múltipla de um ou mais sistemas em reaproveitar seus módulos de forma conhecida, e permitindo a integração de serviços para suporte a um processo completo de negócio.

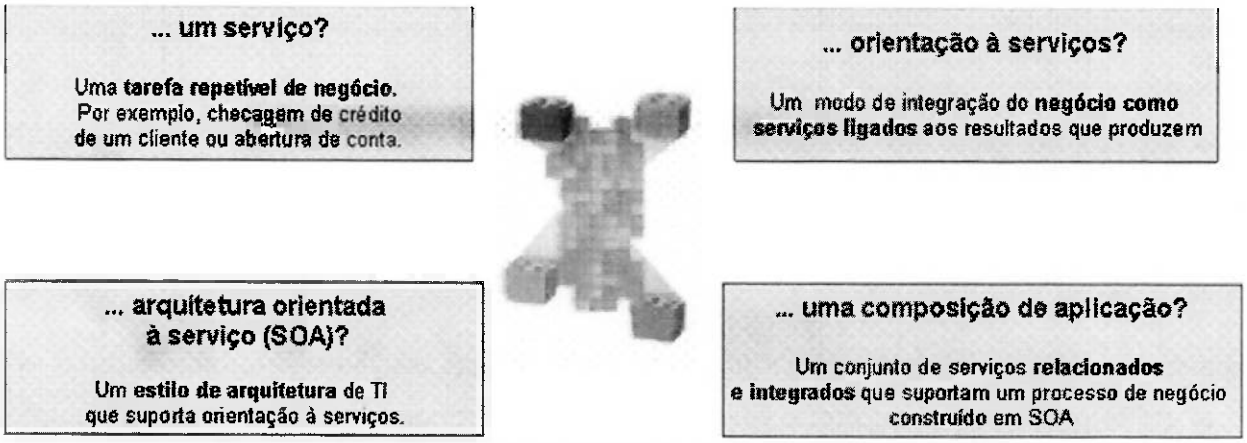

Figura 4 - Fundamentos SOA (adaptado de BUECKER et al., 2007)

SOA 2.0

Natis e Schulte (2006) abordam a arquitetura avançada SOA (referenciada como SOA 2.0) como o próprio nome sugere, uma versão mais madura do conceito original de SOA. A diferença fundamental é que o SOA avançado adiciona eventos de negócio e uma arquitetura orientada a eventos (EDA – event-driven architecture) ao modelo original *request/reply* interativa de serviço original SOA. Existem aplicativos orientados à eventos, embora representem a minoria de todos os novos projetos de software. Além disso sua abordagem para eventos ainda é primitiva, assim como as primeiras implementações de RPC representaram a abordagem *request/reply* de SOA.

O centro original de SOA considera um estilo de arquitetura que utiliza servidores, mensagens independentes de negócio e componentes, acessados de uma forma

interativa por meio de interfaces de programação remotas documentadas. No centro do projeto de SOA interativo está o serviço de chamada de interfaces. Ao contrário de objetos e componentes de interfaces, esta foi projetada para ser útil e usável para outros aplicativos, inclusive com outras empresas. Para atingir isso, os módulos de serviços SOA devem representar o escopo de funcionalidade do negócio, deste caso a arquitetura dos componentes de negócio.

O EDA é um estilo de arquitetura para distribuição modular de software aplicativos nos quais os módulos são projetados para enviar e receber eventos de notificações. Um evento é o centro do projeto de aplicações EDA. Eventos (tal como métodos em uma interface SOA) devem ter um escopo de negócio para ser usável entre aplicativos e empresas. EDA, assim como SOA tradicional é uma arquitetura de componentes de negócio, embora a seqüência dos componentes de negócio em uma transação, fluxo e unidade lógica de trabalho sejam diferentes entre os estilos interativo e de eventos de notificação. Atividades de monitoração de negócio e algoritmos de análise financeira são exemplos de aplicativos EDA.

O SOA avançado compreende o SOA interativo tradicional e o padrão baseado em notificações do EDA em uma única estrutura, em um único ambiente e contexto operacional. O SOA avançado representa a fusão natural dos princípios fundamentais de ambos os estilos. Uma vez implementado, o processamento de eventos e o *middleware* de processamento interativo compartilham facilidades de meta-dados, *middleware* de execução de transporte de recursos (assim como ESB), segurança, gerenciamento e outras facilidades comuns.

A indústria desenvolveu um conhecimento sofisticado dos méritos e deméritos da arquitetura de serviço interativo e por eventos. A tabela 3.1 declara estas diferenças.

	SOA interativo	**EDA**
Modelo de processamento principal	Cliente (que envia mensagem original) endereça a função do receptor (provedor de serviço) por nome e normalmente espera por uma resposta	Origem envia uma mensagem contendo os dados que descrevem a ocorrência de um evento de certo tipo. O Receptor (*sink*) toma conhecimento da chegada deste evento e processa-o. O receptor e a origem não tem conhecimento direto um do outro.
Modularidade	Sim (entre serviços e consumidores)	Sim (entre gerenciadores de eventos – *sink* e originadores de eventos)
Dependência	Fortemente dependente	Minimamente dependentes

	(consumidor chama um serviço por nome, os necessário receptores ativos e disponíveis para o sucesso do processo)	(origem nunca se associa ao receptor, toda comunicação entre as partes é intermediária)
Descentralizado	Sim	Sim
Principal fundamento	Serviço e interface de serviço	Evento e mensagem de evento
Padrão típico de comunicação	Interativo (request/reply) e (raramente) em um sentido apenas	Publicação e subscrição ou passagem de mensagem ponto-a-ponto
Escopo da unidade lógica de trabalho	Cliente e serviço combinados em uma única unidade lógica de trabalho	Origem e receptor (gerenciador de eventos) como duas unidades independentes de trabalho
Controle da unidade lógica de trabalho	Cliente	Origem e receptor, cada um com sua unidade de trabalho independente
Tipos de midleware de comunicação	RPC, ORB, SOAP RPC (como em servidores de aplicação, ESB e serviços web)	Publicação e subscrição ou mensagens ponto-a-ponto (como em middleware orientado a mensagens MOM, servidores de aplicação, ESB e serviços web avançados futuros)
Modelos de programação dedicados atualmente	J2EE EJB, .NET, Web services, CORBA	J2EE MDB e JMS, JSLEE, outros MOM (por exemplo WebSphere MQ, MSMQ) serviços avançados web futuros.
Passagem de dados	Lista de parâmetros em um buffer, documentos	Documentos, outras mensagens
Relacionamento dos participantes	Cliente conhece os serviços por nome ou sinônimo, o serviço deve estar disponível no momento da chamada	As origens de eventos não conhecem os receptores, nem mesmo se estão disponíveis ou se existem
Dependência entre participantes	Definição de interface e versão, serviço nomeado, disponibilidade em tempo de chamada	Estrutura de mensagem e sua versão

Tabela 3.1 – Diferenças entre SOA e EDA (adaptado de Natis; Schulte, 2006)

Para efeitos deste estudo, não haverá distinção entre SOA tradicional e EDA ou da combinação para SOA avançado pelo fato da proposta deste estudo ser de independência da estrutura de segurança da lógica de negócio, seja ela em um modelo interativo ou de evento de notificação. O estudo será genérico, mas consistente com a proposta inicial tendo aspectos de segurança no contexto SOA, sem prejuízo às variações existentes.

EVOLUÇÃO DOS CONCEITOS DE APLICAÇÃO

Orientação a serviços é vista constantemente como uma maneira de melhor alinhar os objetivos de negócio e TI, suportando os níveis de flexibilidade e mudanças solicitadas pelo negócio.

Seguindo a orientação a serviço, a figura 5 exemplifica processos de negócios que são decompostos em unidades discretas de funções de negócio denominadas "serviço". Tais serviços são então recombinados em processos de negócio de uma forma mais flexível. Esta decomposição viabilizou o surgimento de sistemas colaborativos (eco-sistemas), onde um processo reconstruído geralmente integra serviços de parceiros, provedores externos e até clientes.

Segundo (BUECKER et al., 2007), ao se mover funcionalidades tais como controle de fluxo, tradução de formatos de dados e protocolos, propagação de credenciais entre serviços para fora da lógica da aplicação para dentro de serviços de infra-estrutura se ganha flexibilidade em relação a como os serviços podem ser interconectados, uma vez que é necessário apenas conhecer os serviços de infra-estrutura.

Outra situação declarada é que a orientação a serviço não representa uma mudança radical nas abordagens anteriores de projetos de sistemas distribuídos e de tentativa de alinhamento de objetivos de TI e negócio. Ao contrário, ela representa o próximo passo na abordagem evolutiva de conexão entre componentes de aplicação. Cada passo progressivo provê flexibilidade e reuso ao mover componentes considerados não-negócio para fora dos componentes de aplicação.

Figura 5 - Serviços de infra-estrutura (adaptado de BUECKER et al.,2007)

Um dos primeiros passos da evolução foi a introdução de uma infra-estrutura de mensagens, com abstração de conexão entre os componentes de aplicação com o uso de filas. Dentro do espírito de filas ao invés de uma aplicação se comunicar diretamente com outra aplicação, o processo se dá por meio de uma fila, que por

sua vez enfileira a mensagem para outro componente destino. O uso desta estrutura permitiu que o desenvolvedor não tivesse que se preocupar com as particularidades de lidar com plataformas heterogêneas, assegurando ainda garantia de entrega quando a aplicação destino estivesse temporariamente indisponível.

Essa estratégia demonstrou-se promissora na medida em que surgiram camadas intermediárias para absorver uma parte de código que não era reconhecida como necessariamente de negócio, mas que era indispensável para que as aplicações pudessem se comunicar de forma uniforme com outras aplicações. O conceito de broker (ORB) permitiu um avanço na abstração e segregação de funções, reduzindo o esforço total de codificação e efetivamente criando um canal de comunicação entre aplicativos.

A conclusão e a conseqüência do caminho de abstração resultaram em uma forma de pensar e modelar aplicativos, que era maior do que simplesmente retirar código das aplicações, mas tornou-se um ideal de arquitetura conforme demonstrado nos capítulos a seguir.

A figura 6 ilustra a evolução das arquiteturas de sistemas em relação a sua complexidade de desenvolvimento e as características de flexibilidade e reuso. Em um primeiro estágio, nota-se que a complexidade de segurança e rede faziam parte da estrutura principal do aplicativo, o que resultava em baixa capacidade de adaptação. Esta condição representada pela conectividade direta resulta em aplicações pouco flexíveis e com grande dependência da tecnologia em que foi implementada. Os sistemas de enfileiramento de mensagens (*message queues*) permitiriam abstração e separação da lógica de conexão e gerenciamento, mas ainda com certa dependência lógica para controles. A capacidade de evoluir para uma separação ainda maior forma a classe de aplicativos de segmentação de mensagens (*message brokering*). A proposta de orientação a serviço separa todas as camadas de integração e comunicação do código fundamental entendido como aplicação de negócio, que justamente permite reaproveitamento e flexibilidade para a dinâmica de negócio.

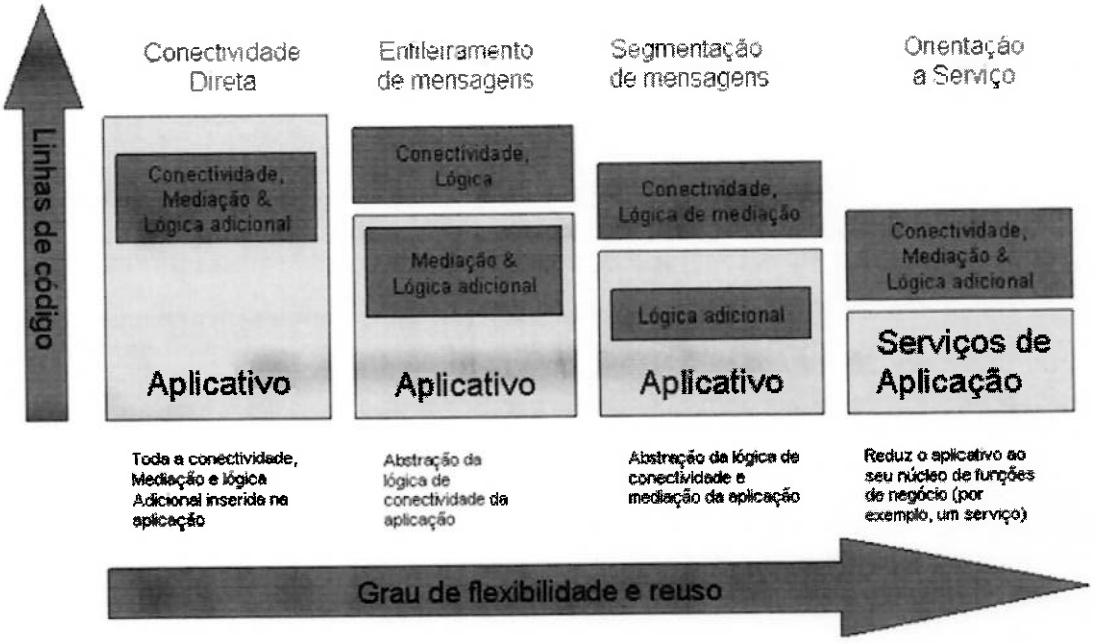

Figura 6 - Evolução da tecnologia dos aplicativos (adaptado de BUECKER et. al, 2007)

As previsões de mercado posicionam SOA como uma alteração durável na arquitetura de sistemas do futuro, e serão usadas em alguma parte em mais de 50% nos novos aplicativos de missão crítica e processos de negócio projetados em 2007, e em mais de 80% até 2010:

> "SOA will be used, in some part, in more than 50% of new mission-critical applications and business processes designed in 2007, and in more than 80% by 2010 (0.7 probability)" – Gartner Group.

SOA COMO TECNOLOGIA DE INTEGRAÇÃO

Um dos aspectos relevantes em uma arquitetura SOA é a capacidade de identificar os relacionamentos em um mesmo plano ou camada assim como entre camadas. Os planos de relacionamento incluem o plano de negócio, plano de serviço, plano de ambiente, plano de infra-estrutura e plano de aplicativos interdependentes. A figura 7 exemplifica três destes planos (negócio, serviço e infra-estrutura).

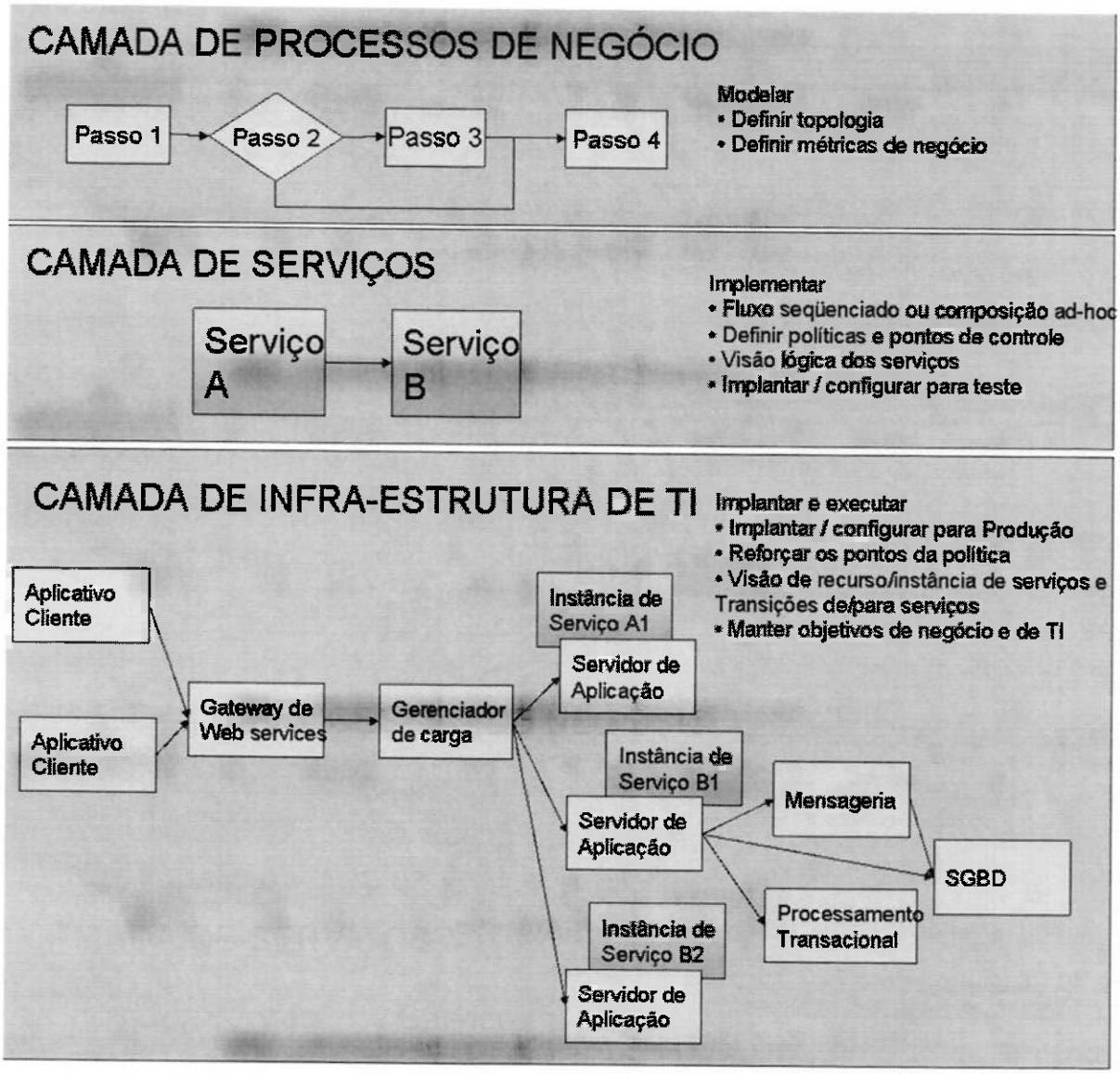

Figura 7 - Planos de SOA (adaptado de COX; KREGER, 2005)

Estrutura original de aplicação

A figura 8 ilustra a abordagem tradicional na implementação de um processo de negócio e seus aplicativos de TI associados, com cada unidade organizacional agindo isoladas. Cada processo de negócio possui sua própria implementação proprietária de uma atividade de negócio, as quais são freqüentemente reimplementadas em uma maneira ligeiramente diferente dos outros processos e unidades organizacionais.

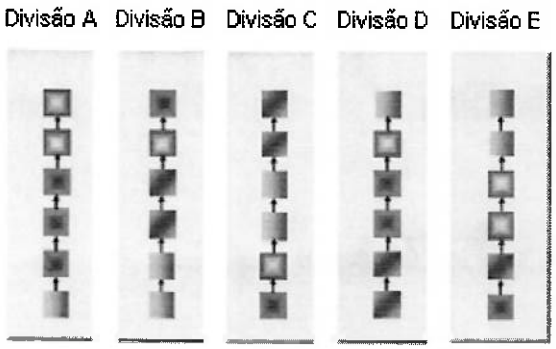

Figura 8 - Estrutura original de aplicação (adaptado de BUECKER et al., 2007)

Orientação SOA para aplicação

A figura 9 mostra o objetivo da orientação a serviço: Lógica de negócio comum e disponível de forma reutilizável para serviços que podem ser executados onde for mais apropriado, independente dos limites organizacionais.

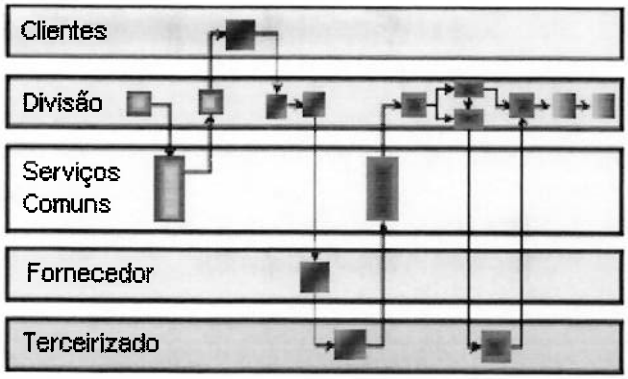

Figura 9 - Fluxo de negócio (adaptado de BUECKER et al., 2007)

ASPECTOS DE SEGURANÇA SOA DENTRO DA EMPRESA

3.1.1 Integração complica a segurança

Integração é o ato de unificar as capacidades de dois ou mais aplicativos independentes em uma mesma empresa ou empresas distintas. Por exemplo, em uma empresa de mercado financeiro podem existir sistemas diferentes, um para controlar depósitos e outro para controlar aplicações. Pensando em uma condição que necessite transferir valores de um sistema para outro, é necessário integrar ambos os aplicativos.

O nível de integração entre aplicativos pode ser diferente. Nem todos os aplicativos podem ser integrados de uma forma transparente. Por exemplo, a integração de

uma aplicação de conta corrente com outra aplicação de investimento requer que os dados internos (dinheiro, conta, cliente) sejam transferidos de um sistema para outro. Se houver necessidade de integração com um outro aplicativo que concentra vários tipos de operações financeiras, haverá outras informações a serem transferidas.

A forma mais fácil de integração é reescrever os aplicativos, a qual não é uma opção na maioria dos casos. Ao invés disso, as técnicas de integração usadas passam pelas seguintes:

- Transferência de arquivos, por meio de padrões de *Electronic Data Interchange* (EDI).
- Troca de mensagens ponto-a-ponto com comunicação HTTP ou serviços de mensageria como Message Queues (MQ).
- Troca de mensagens usando o conceito de publicação e subscrição em uma camada unificada.
- Computação distribuída usando recursos de *Remote Procedure Call* (RPC).

A integração traz diversos desafios pois representa custos; é susceptível a erros; pode ser inflexível e por fim – pode ser um pesadelo para segurança. Independente de todos os problemas, os aplicativos ainda precisam ser integrados. Grandes empresas percebem o retorno de investimento (ROI) quando os aplicativos são integrados pelas empresas. O motivo que impulsiona as grandes empresas para integração é que embora existam sistemas mais complexos do que anteriormente, ainda existem sistemas antigos atendendo às suas necessidades. Em resumo, o comércio cresce com integração, que traz novas possibilidades e oportunidades. A integração oferece conveniência e reduz os custos da empresa, pois permite soluções incrementais que se apóiam nos sistemas atuais. A integração traz complexidades adicionais, sendo uma delas segurança e que não estão bem entendidas. Considerando um caso real como uma empresa de consultoria de investimentos que trabalha com clientes individuais, clientes corporativos e pequenos escritórios de investimento, cada uma com diferentes necessidades de apresentação de dados, níveis de sofisticação de relatórios e horários de acesso à base de dados, existe um conjunto de sistemas e processos de segurança inerentes a esta integração:

- O processo de entrada (*login*) no sistema deve ser feito uma única vez.
- Os pedidos de compra e venda de ações ou investimentos devem ser consolidados e finalizados com um processo de autorização no final do dia.
- Os usuários dos escritórios de investimentos devem ser capazes de efetuar operações em nome dos clientes, como se eles mesmos as tivessem feito.
- As transações que excederem um determinado limite devem ser reportadas para um sistema central, como uma forma regulatória de controle.
- Os dados de clientes e operações devem ser mantidos sob sigilo.

Algumas dessas preocupações são específicas ao tema de integração de sistemas. Por exemplo, se o sistema é monolítico, então o processo de login sempre será único, pois somente existe um sistema a considerar.

Tomando como base os tópicos conceituais de segurança descritos em 0 - Conceitos de Segurança, a implicação de integração de aplicativos passa pelo seguinte entendimento:

Autenticação: Enquanto que algumas empresas precisam de apenas um usuário e senha para efetuar operações, em grandes corporações provavelmente existirão múltiplos usuários. Tomando o exemplo da empresa de investimento, os escritórios isolados podem executar operações em nome dos clientes, mas ainda sim precisam de um processo de assinatura dos clientes para confirmar as operações. Isso sugere que existem diferentes meios de autenticação possíveis em uma empresa. Ao se construir um sistema de autenticação para um aplicativo normalmente armazena-se os dados relativos aos usuários e suas senhas de forma centralizada, ou eventualmente usando sistemas mais sofisticados como certificação digital. No entanto, para compartilhar estas credenciais com outros sistemas, as mesmas devem estar disponíveis para os outros aplicativos. Além disso, se tais dados forem compartilhados com outras empresas como ter certeza de que não serão mal utilizados?

Autorização: Em uma aplicação simples, os processos de autenticação e autorização estão em um único passo, embora estas duas práticas possam ser implementadas em módulos diferentes. Em um cenário simples, um aplicativo pode determinar quem tem acesso a qual recurso. No entanto, em uma condição de integração, o processo de autenticação pode ser separado da autorização, mas com

os dados de autorização disponíveis para todas as aplicações. O que dificulta esta abordagem é que pode não haver um modelo uniforme para autorização. Em uma única empresa, os usuários podem pertencer a determinados grupos e estruturas, mas com a proliferação destes grupos pode-se tornar um problema de gerenciamento. Para o caso de uma integração múltipla entre empresas, isso pode não ser possível. Por fim, existe a complexidade de mapear as permissões de um sistema em outro, fato este que surge meramente pela integração de múltiplos sistemas.

Confidencialidade: Para casos de comunicação ponto-a-ponto, a forma de confidencialidade é bem conhecida, mas este não é o caso quando múltiplas entidades estão envolvidas e cada uma tem necessidade de visualização de partes da mensagem (quando são destinadas para aplicativos específicos), evitando que outros aplicativos tenham chance de visualizar ou manipular seus conteúdos. Se o processo tiver apenas um remetente e um destinatário esta condição é facilmente resolvida. Se a mensagem tem o potencial de ser vista por diferentes aplicativos que precisam manipular a mensagem de alguma forma, deve-se assegurar que apenas as entidades corretas tenham acesso aos conteúdos específicos da sua necessidade. Este é um desafio comum na integração de aplicativos.

Integridade e não-repudiação: Em um cenário em que existe apenas um remetente e um destinatário, a integridade de dados pode ser atingida facilmente com assinatura digital, onde o destinatário saberá que a mensagem não foi modificada no caminho. O destinatário também poderá provar que ninguém mais poderia ter gerado a mensagem, exceto o remetente. A situação fica mais complexa quando existem vários remetentes e destinatários, uma vez que o fluxo de mensagem pode receber modificações de diferentes aplicativos, cada um precisando assinar a parte que lhe compete. Embora não seja um problema técnico, pode ser difícil gerenciar sem um padrão, pois os aplicativos terão que saber qual entidade assina por qual segmento de uma mensagem.

Privacidade: No contexto envolvendo duas entidades, a privacidade apenas surge por uma das duas partes. Se a mensagem for mantida e transmitida de forma segura, a privacidade é relativamente fácil de ser obtida. Quando existem vários remetentes e destinatários, a ameaça à privacidade pode vir de qualquer ponto. Cada sistema pode conter parte das informações sobre um investidor, a qual por si só não representa uma quebra de privacidade na ótica das leis. No entanto, a junção

de outros dados pode revelar mais do que a informação que é permitida, onde até restrições legais podem impedir a integração de aplicativos. Tecnicamente, é difícil modelar requisitos de privacidade em um ambiente complexo.

Ataques de Homem-no-meio: Na condição de apenas um remetente e um destinatário, a única forma passível de ataque é ouvir a linha de comunicação, mas com técnicas adequadas de criptografia, é possível restringir informações para outros. Adicionalmente, para evitar a retransmissão de uma mesma mensagem, basta usar controles de data e hora, ou números seqüenciais. Quando se têm múltiplos aplicativos, este tipo de ataque pode vir de qualquer parte, ou seja, um dos aplicativos integrados pode executar o papel de homem-no-meio já que todos os aplicativos têm acesso à mensagem, ele pode retransmiti-la a menos que se implementem formas mais sofisticadas de controle. Portanto, a necessidade de proteger contra aplicativos não confiáveis faz com que a proteção aos ataques homem-no-meio seja difícil.

Gerenciamento e rastreio: Existem certas condições que são mais simples de gerenciar quando não envolve aspectos de integração. Qualquer gerenciamento de segurança é difícil. Se existem vários aplicativos na empresa, gerenciar os usuários e senhas pode ser um pesadelo. Mesmo que todos os aplicativos estejam centralizados, o gerenciamento de políticas de segurança pode ser complexo. Esta questão torna-se ainda mais complexa com integração, pois os aplicativos terão que lidar não apenas com usuários, mas com outros sistemas. Ao passo que mais aplicativos participam do processo de integração, potencialmente, os aplicativos podem não ter muita idéia com quais usuários eles estão realmente lidando. Qualquer alteração em um aplicativo ou na estrutura de diretórios de usuários pode ter repercussões que são difíceis de entender, o que significa que a solução de gerência precisa lidar com este tipo de complexidade. Em adição a gerência, outra questão que surge é a capacidade de rastreio. Uma mensagem pode passar por diversos aplicativos, que por questões regulatórias, precisa ser adequadamente registrada para efeitos de auditoria. Além disso, existe a possibilidade de usar este registro para ajustar ou gerenciar os aplicativos, acompanhando as ações de dados e quais sistemas eles passaram.

3.1.2 Integração com sistemas legados

De acordo com (FELMAN, 2006), o eco-sistema de software ao redor de nós está principalmente populado de aplicativos legado, e apenas uma parte deles, a que está em desenvolvimento, não é legado ainda. Assegurar que os sistemas legado sejam seguros não é um nicho, mas sim um objetivo principal que precisa ser endereçado imediatamente.

Os sistemas legado tornaram-se uma fonte primordial para serviços web e arquiteturas orientadas a serviço. Neste sentido, elas são expostas ao mundo pelas camadas HTTP, HTML e SOAP. Mesmo as aplicações sendo extremamente seguras no seu contexto anterior, as mesmas ficam vulneráveis agora.

Fazer com que aplicações (especialmente do tipo legado) sejam resistentes a nova (agora mundo orientado *web*) segurança e seus ataques e ameaças requerem habilidades que fazem parte dos sistemas legado. Infelizmente, tais habilidades estão em contínua queda nos ambientes de TI, onde o conhecimento profundo de vários aplicativos e utilitários foi perdido, na medida em que os profissionais se aposentam ou atuam em novas funções. Fazer com que os aplicativos legado sejam aderentes às modernas regras e regulamentações é normalmente caro porque tais aplicações foram concebidas em uma era anterior a tais regras.

O desenvolvimento de segurança em sistemas legado atuais ou modernos pode ser atingido ao se entender o seu fluxo lógico, estruturas, dados externos, infra-estrutura e complexidade de código.

Os fatores críticos de sucesso para uma empresa são (1) entender as tecnologias e metodologias dos sistemas legado para modernização da segurança; (2) manter um conjunto confiável de habilidades em fontes de legado seja ela em COBOL ou *Visual Basic*; (3) desenvolver processos, aprender métodos e (4) adquirir ferramentas para proteger a crescente exposição dos sistemas legado.

Tomando como exemplo o fluxo de aplicativos de um hospital que necessita executar várias atividades para um cliente, ele pode necessitar de dados históricos de um outro hospital, verificar cobertura com o seguro de saúde ou enviar o pedido de receita médica para a farmácia mais próxima da residência do cliente para preparar uma manipulação de um medicamento. Cada uma dessas atividades pode ser atingida por diferentes aplicativos. Como o sistema do hospital prova para os demais aplicativos que efetivamente ele pode acessar tais dados e tem permissão

para executar tais tarefas? O elemento chave deste exemplo é que existe um serviço de alto nível e que também existe a possibilidade de brechas de segurança muito maior quando aplicativos são unidos para prover um serviço, ao invés de um único aplicativo atendendo a todas as necessidades. Em outras palavras, a integração de aplicativos faz com que seu projeto de segurança seja mais complexo do que seria.

Integração de aplicativos não é uma tecnologia nova, onde existem várias abordagens. No entanto, a arquitetura orientada a serviços é a maior delas, onde fornecedores, empresas e desenvolvedores estão adotando como uma nova forma de organizar capacidades de TI, que permite uma ampla integração de aplicativos de forma mais barata e fácil do que antes. Consequentemente, a segurança SOA é mais complexa do que a aplicativos tradicionais em ambientes de TI.

Mas ao contrário de outras tecnologias de integração que antecederam SOA, ela está bem posicionada para abordar os aspectos de segurança e integração. Por sua natureza e por ser uma abordagem padronizada, o SOA permite novas técnicas que podem auxiliar a resolver o problema antigo de integração de segurança em um nível maior do que antes. No entanto, as técnicas ainda não estão amplamente conhecidas aos praticantes de SOA (Kanneganto; Chodavarapu, 2006).

Embora a maioria dos praticantes entenda que SOA introduz novos desafios de segurança, eles normalmente não são claros como resolver tais desafios. Esta falta de clareza normalmente leva a uma das duas situações: Ou a implementação em SOA será errônea demais, colocando muita cautela e perderá os benefícios de uma integração simples e aberta ou no outro extremo, terminará com brechas de segurança esperando serem exploradas. Obviamente, ambas as situações são indesejáveis.

4 ROTEIRO DE SEGURANÇA DA INFORMAÇÃO PARA SOA

Os aspectos de segurança e a evolução da arquitetura para um modelo SOA é o principal do roteiro, através da revisão da camada de segurança dentro do novo contexto de serviço. O processo adotado para manipular uma aplicação SOA dentro da proposta desta monografia passa pelo seguinte fluxo geral, conforme a figura 10.

1- Classificação das aplicações elegíveis: Tem por objetivo filtrar o conjunto de aplicativos para serem avaliados no processo definido nesta monografia.
2- Análise da camada de segurança existente: Esta fase trabalha com as recomendações de segurança para os aspectos de controle a serem investigados e entendidos na aplicação objeto de estudo.
3- Declaração dos requisitos de segurança : Esta fase consolida os itens faltantes ou de atenção para que a aplicação tenha melhor aderência a regras de segurança e tire proveito dos serviços complementares de segurança inerentes.

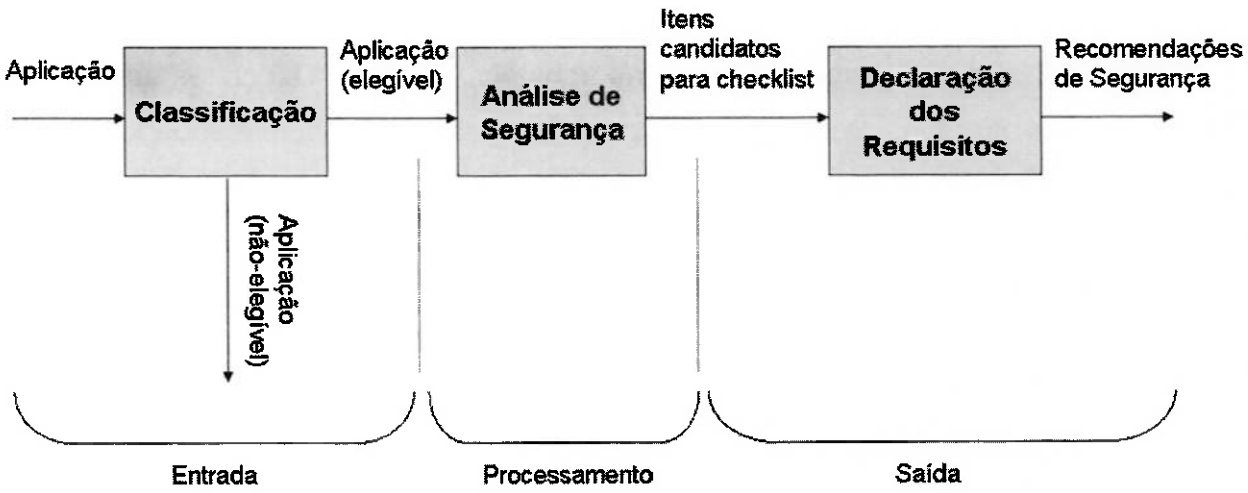

Figura 10 – Fluxo geral para classificação de aplicações

CLASSIFICAÇÃO DAS APLICAÇÕES ELEGÍVEIS

A primeira condição a ser satisfeita para o processo é a criação de uma forma de classificação de aplicações para o processo de segurança. Dentre os itens que normalmente definem os aplicativos com maior grau de segurança estão a

criticidade dos dados (clientes, fornecedores, formulas), volume financeiro envolvido, requerimentos legais ou padrões internos para aplicações.

Com o objetivo de sugerir um critério de classificação das aplicações, a categorização está dividida em três partes denominada (A) visão de aplicação; que aborda o grau de maturidade da aplicação para poder tirar proveito das regras e conceitos explorados nesta monografia; (B) visão de relevância para o negócio para estas aplicações e (C) aspectos regulatórios que devem ser ponderados para assegurar o desempenho da empresa. Considerando o critério fundamental com aplicações que tenham um teor técnico orientado à arquitetura SOA, a visão de negócio e os aspectos legais foram considerados menos relevantes para este tipo de classificação e portanto, terão um peso menos expressivo. Esta atribuição de pesos pode ser alterada para outras condições de uso se desejado. Em termos práticos, a regra de classificação de aplicativos descrita nesta monografia considera apenas os casos em que a componente técnica está aderente ao modelo SOA. A visão de negócio e legal, embora importantes dentro do contexto empresarial, não são determinantes para a escolha das aplicações para esta monografia.

A figura 11 demonstra o fluxo que uma aplicação deve seguir para aplicar os conceitos propostos nesta monografia.

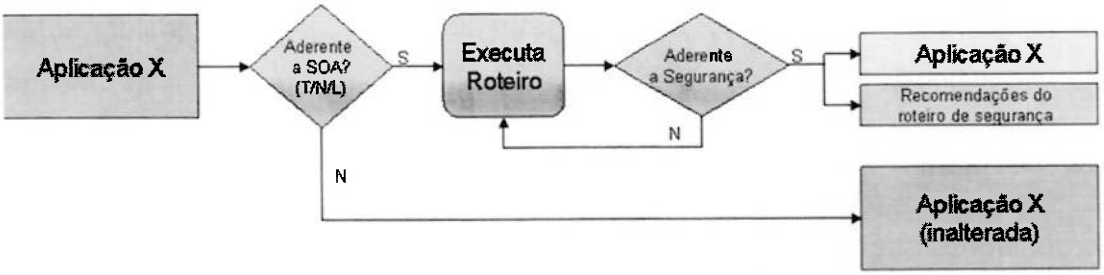

Figura 11 – Classificação de aplicações para o roteiro

É importante ressaltar que embora todas as aplicações possam fazer parte do processo de segurança, o esforço para sua adequação pode ser elevado. Como uma alternativa para assegurar cobertura plena, este processo pode ser inserido no fluxo de desenvolvimento de novos aplicativos e no processo de seleção de novas ferramentas de fornecedores, para assegurar que os novos processos tenham uma validação formal, enquanto que os sistemas internos são convertidos conforme a prioridade.

A tabela 4.1 representa os pesos e condições mínimas para análise de aplicações elegíveis, bem como os pesos atribuídos neste estudo.

Visão / Peso	**Categoria**	**Nota**
Segmentação de aplicação Peso: 5	Aplicação interna nova . aderente a SOA . não aderente a SOA	 5 1
	Aplicação interna antiga . aderente a SOA . não aderente a SOA	 5 1
	Aplicação externa . aderente a SOA . não aderente a SOA	 5 1
Negócio Peso: 2	Crítica Importante Apoio	5 3 1
Legal Peso: 1	Mandatória Desejável	5 1

Tabela 4.1 – Critério de avaliação para aplicações

O grau de eligibilidade será definido como sendo a média ponderada dos itens, levando em conta o perfil da aplicação escolhida para análise, conforme a formulação (3):

Fator de aderência = (Peso_Aplicação * nota_aplicação + Peso_negócio * nota_negócio + Peso_legal * nota_legal) / (soma dos pesos) **(3)**

A proposta desta monografia é que somente as aplicações que sejam orientadas a um modelo SOA devam ser submetidas a este processo. Sendo assim, o fator de aderência calculado deve ser superior a 3.0, o que significa que a aplicação está aderente aos padrões SOA para poder aplicar o roteiro. Os itens com peso 5 definem a categoria mais relevante para o processo, enquanto que os pesos inferiores são menos relevantes. As notas declaradas como 5 representam a maior relevância desejada para este processo, enquanto que notas inferiores demonstram itens menos desejados. Esta fórmula foi concebida pela sua simplicidade, podendo

ser refinada ou aumentada para outras necessidades de maior complexidade e com outras variáveis para tomada de decisão, caso deseje-se priorizar itens tais como aspectos legais ou estudos de caso de negócio em que a proposta do modelo seja mais relevante. Embora os pesos e notas de 1 a 5 tenham uma conotação subjetiva, os mesmos estão coerentes para o exercício desta monografia, suportando adequações para outras análises. A fórmula também pode ser expandida para suportar fatores qualitativos, funcionais e que sejam relevantes para composição objetiva do grau de elegibilidade.

ANÁLISE DA CAMADA DE SEGURANÇA EXISTENTE

O modelo de arquitetura SOA expõe um conjunto de desafios para as camadas de segurança tradicionais, em especial para o modelo monolítico das aplicações que até então concentram as operações de autenticação, autorização e trilhas de auditoria.

Tomando como base a evolução das aplicações, a primeira geração continha aspectos de segurança de forma não padronizada, cada uma com um entendimento de complexidade e qualidade nos aspectos fundamentais para atender a segurança de informação. Este raciocínio normalmente está ligado à percepção do desenvolvedor sobre os requisitos de segurança, o que faz com que o modelo seja fraco, ou até mesmo vulnerável, por não ter uma estrutura adequada de controles.

O primeiro passo nesta evolução é a criação de métodos específicos de segurança, onde o sistema insere pontos de controle na lógica de aplicação para garantir a validade de um processo. Eventualmente, este conjunto de rotinas pode evoluir para um módulo de segurança que pode ser trocado, permitindo a separação parcial de métodos de segurança dos dados e formas de controle. No entanto, esta arquitetura ainda mantém uma forte ligação entre o modelo de segurança e o código. Esta categoria de soluções representa um modelo referenciado como monolíticas, onde os aspectos de integração entre lógica de negócio e segurança são difíceis de separar, gerando normalmente rompimento no código ou recodificação.

A limitação imposta por esta condição deve ser trabalhada e separada em primitivas de segurança - identificação, autenticação, autorização e auditoria, para então ser tratada de forma específica e conforme as características desejadas na aplicação.

Um ponto importante é a capacidade de abstração e isolamento que estas primitivas precisam ter para assegurar sua reutilização e amplitude para diferentes casos de uso. Com o intuito de criar uma forma unificada de avaliação dos 4 aspectos de segurança, a justificativa levará em conta a seguinte divisão:

1. Uso atual: Conceitua o uso da tecnologia em condições normais, o comportamento do componente de segurança e seu papel dentro do aplicativo em relação à provisão de uma funcionalidade básica
2. Pré-requisitos para serviço: Demonstra as operações possíveis que a aplicação depende ou necessita para cumprir os aspectos de segurança (explícitos ou não) exigidos.
3. Funções primitivas do serviço: Elege um conjunto de operações fundamentais para que a aplicação tenha sucesso nas operações.
4. Controle e auditoria: Define um conjunto de regras para controle e rastreamento das operações de segurança.

Um dos objetivos é identificar itens para o roteiro de segurança da aplicação, de forma a torná-la independente e segura para a arquitetura SOA. O detalhamento de cada uma dessas condições está coberta a seguir, com a separação de itens-chave relevantes para o roteiro de segurança. É importante ressaltar que as 4 aspectos fundamentais (identificação, autenticação, autorização e auditoria) fazem parte do quadro de análise e recomendações.

4.1.1 Identificação

4.1.1.1 Uso atual

Uma aplicação convencional utiliza os serviços de identificação baseado em uma especialização e nomenclatura próprias, inerentes a necessidade do projeto. A forma mais comum é a representação de um *userid*, de forma alfanumérica (para os casos de nomes ou siglas) ou numérica (para os casos de números como RG, CPF ou matrícula). A semântica desta identificação está ligada diretamente a uma aplicação, fazendo com que o intercâmbio entre aplicativos seja mais difícil.

4.1.1.2 Pré-requisitos para serviço

Uma evolução deste processo é a criação de um padrão de identificação, que normalmente é válido no escopo de uma empresa ou conjunto de empresas, adotando-se uma forma de convenção para expressar o atributo de identificação do usuário. Formas diferenciadas envolvem também a biometria ou certificados digitais, que de certa forma confundem-se com o conceito de autenticação.

A capacidade de identificar um usuário de forma transparente entre várias aplicações está ligada ao principio de autenticação unificada (SSO) e de gerenciamento de identidade (*identity management*), onde a proposta é garantir a existência de relação unívoca de uma pessoa com o sistema em que ela necessita ter acesso, preferencialmente com uma identificação única, mas que pode ser diferenciada e manipulada pelo sistema de *single sign-on* para realizar a propagação da informação pertinente para aquele sistema. Desta forma, uma pessoa pode estar representada em múltiplos sistemas de forma diferenciada, com sintaxes e regras de formação de nome especiais e mesmo assim ser vinculada a uma única pessoa.

Este princípio permite que as facilidades de identificação possam agregar atributos complementares para uma pessoa, definindo assim um conjunto de informações que podem ser armazenadas para identificar e distribuir dados complementares conforme os sistemas de informação solicitarem. Um exemplo seria um repositório de uma empresa com a identificação e atributos pessoais como estação de trabalho, telefone residencial, e-mail, endereço atual, e outras informações que poderiam ser acessadas a partir da identificação indexada e unívoca.

Uma evolução da facilidade de mapeamento de uma pessoa em um *userid* para um sistema fica mais complexa quando se envolve outras empresas, Países e culturas, onde o alfabeto, as convenções de nomes e o uso da identificação podem ser diferenciados. Faz-se necessário que o padrão de identificação seja realmente forte, ou que permita o mapeamento de diferentes condições para grupos de pessoas (que necessitam de uma convenção de identificação).

4.1.1.3 Funções primitivas do serviço

Pensando no serviço de identificação, o fator relevante para assegurar o uso amplo deste serviço passa pela capacidade de gerenciar a diversidade de formas (nomes,

números, imagens, informações em bits), permitir o acesso de forma ágil e rápida, e assegurar que as variações de tamanho dos dados não seja problema (por exemplo, ter uma representação de um *userid* em um bloco de 64 KB ou mais).

Em termos de privacidade, a criptografia da identificação é pouco útil, mas para seus dados complementares isso é importante. A visibilidade destes dados para os diversos aplicativos deve ser controlada por chaves de criptografia que as aplicações de cada entidade dominem. O transporte da informação e a garantia da sua integridade dizem respeito não somente a identificação, mas também as demais primitivas de segurança abordadas neste documento.

A capacidade de esconder ou mascarar a identificação de um sistema para outro se torna um recurso interessante, na medida que existe segregação de administração e confiança parcial (não plena) entre as entidades ou sistemas. Este recurso permite, por exemplo, que um "usuário_A" seja convertido em uma identificação "usuário_B" montando uma relação condicional (DE-PARA) dinâmica e útil durante o vínculo entre dois sistemas particulares. Embora pouco útil para o usuário diretamente, este recurso é muito usado entre aplicações em que os sistemas possuem comunicação direta, usando um *token* de identificação aleatório ou mascarado.

4.1.1.4 Controle e auditoria

Não existem muitos processos de auditoria específicos para um serviço de identificação, pois sua finalidade geral é prover a primeira camada de serviço de segurança, encontrando as informações únicas de uma pessoa e preparando-se para passos mais elaborados dentro do processo de segurança de acesso. No entanto, deve-se atentar para a detecção de varredura e exploração da base de conhecimento dos usuários, a engenharia reversa e o mapeamento da base de usuários para descobrir algum padrão de funcionamento ou característica do sistema.

Os eventos de sucesso e falha de pesquisa do serviço de identificação devem ser registrados, e devem ser alertados se ocorrerem com uma freqüência de tempo excessiva, ou obedecendo a algum padrão seqüencial, de faixa de pesquisa, por combinação de letras, etc.

4.1.1.5 Itens relevantes para o roteiro

O serviço de identificação deve oferecer as seguintes funcionalidades de serviço, a saber:

- Busca de um usuário
- Recuperação de atributos de um usuário a partir da sua identificação chave
- Tradução de uma identificação em uma outra (uso de DE-PARA)
- Pesquisa de um usuário a partir de dados não completos (filtros)

Os aspectos não funcionais que devem ser suportados são:

- Atendimento a múltiplas solicitações de consumidores ao mesmo tempo.
- Independência entre pedidos de identificação (*stateless*)
- Capacidade de se registrar e divulgar o serviço de identificação e suas propriedades
- Ajustar-se em tempo de execução a novas regras (definidas em uma política)
- Criptografia (proteção de dados sensíveis durante o transporte de rede)
- Auditoria (registro da atividade e pesquisa de sucesso e falha)
- Isolamento e agrupamento (segregação de domínios de informação)
- Recursividade (capacidade de pesquisar bases indiretas, integração com serviços de identificação externos).

4.1.2 Autenticação

4.1.2.1 Uso atual

Considerando uma aplicação que deseja assegurar a correta identificação de cada um dos participantes em um sistema de informação, faz-se necessário que o sistema declare a identidade do sujeito participante, acrescido de um componente seguro e único para esta entidade, para um reforço desta declaração, que é caracterizada pela autenticação de usuários e sistemas. Desta forma, as entidades são conhecidas e atestadas como verificadas pelo sistema e, portanto, passam a ser confiáveis para interagir no sistema. É importante observar que esta confiança é aferida por uma aplicação, e que não necessariamente é reconhecida pelas outras aplicações como sendo confiáveis, ou estão no mesmo nível de confiança. A propagação das credenciais de autenticação, juntamente com a identificação do usuário faz com que

o conceito de *single sign-on* tenha mais força e permita neste estágio o aproveitamento do recurso combinado (identificação+autenticação) como os pilares para construção das camadas mais complexas de segurança para aplicações.

4.1.2.2 Pré-requisitos para serviço

Um aspecto desejado para autenticação é a capacidade do seu aproveitamento no tempo, onde uma aplicação possa utilizar e confiar na informação de autenticidade de uma entidade (usuário ou sistema) interagindo com a aplicação, e que várias aplicações possam compartilhar esta mesma informação de forma transparente e segura, permitindo que o sistema não incorra em processos computacionais redundantes e nem o usuário tenha que apresentar dados de sua credencial em múltiplos sistemas. Este conceito ainda pode ser expandido para integração de empresas, onde o domínio de segurança fica muito mais complexo, pois normalmente não existe um elo de confiança absoluta entre empresas ao ponto desta confiança de autenticação ser bidirecional e totalmente transparente.
A capacidade de usar os recursos e informações de autenticação de um domínio para outro foram inicialmente descritas como serviço de SSO, onde a proposta é que os aplicativos possam compartilhar suas credenciais, eliminando assim os múltiplos repositórios de usuários e senhas. O processo pode ser direto, com o acesso ao sistema primário e executando o sign-on sincronamente para todos os domínios secundários, pode ser imediato, com a chamada dos aplicativos secundários para executar os processos de sign-on durante o estabelecimento de sessão primária, indiretamente com a recuperação de dados do usuário armazenadas em um repositório central ou temporária, com o armazenamento de dados em memória conforme o usuário desejar se comunicar com outros aplicativos.

Embora o SSO seja uma primeira abordagem na linha de abstração de serviço, o conceito parte do pressuposto de que as aplicações já existem, com bases de dados de usuários existentes, mediando a comunicação e tradução de credenciais, e que a confiança entre os domínios não é absoluta.

4.1.2.3 Funções primitivas do serviço

Pensando na autenticação como orientada ao modelo de serviços, é importante que ela tenha a capacidade de receber dados de múltiplas aplicações e de formatos diferenciados, envolvendo, por exemplo, sintaxes de caracteres e biometria como fontes de verificação.

4.1.2.4 Controle e auditoria

Existe um risco maior na autenticação quando comparada com a identificação, pois como a autenticação visa abrir a porta para os sistemas aplicativos, ela pode ser subvertida ou forçada com ataques de senhas, forjamento de credencial ou ataques ao próprio repositório de credenciais, que normalmente também depende de algum sistema de autenticação. A capacidade de identificar eventos e acessos de autenticação são extremamente relevantes para detecção de violações e para processos de análise de comportamento de usuários (como por exemplo, montar um perfil padrão para usuários de um sistema comercial). O registro de tentativas com sucesso e fracasso no ciclo de autenticação garantem não somente controles internos e detecção de fraudes, mas também permitem melhor balanceamento dos serviços e análise de capacidade, estes últimos mais ligados à garantia do serviço.

4.1.2.5 Itens relevantes para o roteiro

O serviço de autenticação deve oferecer as seguintes funcionalidades de serviço, a saber:

- Uso do serviço de identificação
- Verificação de um usuário e de sua credencial

Os aspectos não funcionais que devem ser suportados são:

- Atendimento a múltiplas solicitações de consumidores ao mesmo tempo.
- Independência entre pedidos de autenticação (*stateless*)
- Capacidade de se registrar e divulgar o serviço de autenticação e suas propriedades
- Ajustar-se em tempo de execução a novas regras (definidas em uma política)
- Criptografia (proteção de dados sensíveis durante o transporte de rede)
- Auditoria (registro da atividade e pesquisa de sucesso e falha)
- Isolamento e agrupamento (segregação de domínios de informação)

- Recursividade (capacidade de pesquisar bases indiretas, integração com serviços de identificação externos).

4.1.3 Autorização

4.1.3.1 Uso atual

Via de regra, a autorização utiliza os serviços de identificação e autenticação para derivar o controle de acesso, tanto sua concessão como negação, e pode também utilizar a camada de auditoria e *log* para armazenar ou recuperar dados relativos ao comportamento de um usuário ou recurso computacional.

A primeira motivação para autorização é definir o nível de visibilidade (ou acesso) que uma entidade pode incorrer durante uma sessão de trabalho, obedecendo às restrições de tempo, localidade, permissão e regras de negócio que compõem a efetiva abrangência do grau de manipulação que a entidade usuária pode executar.

Normalmente, este nível de acesso está dividido em termos gerais (também conhecido como *coarse level*) ou específicos (*fine grained*), e podem estar representados em regras discretas (listas de acesso, matrizes de bits) ou em agrupamentos ou políticas de papéis ou funções (também conhecidas como roles). A tecnologia de autorização também pode levar em conta a classificação do dado e representar modelos mais sofisticados de segregação, aliando critérios de confidencialidade, níveis de privacidade e classificação (*security labels* ou *security levels*).

Uma aplicação que prevê um ou mais dos recursos mencionados tem, por definição, necessidade de um controle de acesso, podendo ser elaborados e muito específicos, ou simples e mais leves. Esta diferença de profundidade e amarração com a aplicação define em parte o grau de segurança efetivo que a aplicação terá, bem como a facilidade de mudar a lógica de negócio sem afetar o fluxo correto de checagem de autorização. Isso sugere que o código executável passa a ter duas funcionalidades bem distintas com dependência forte: De um lado a lógica de negócio e de outro os recursos de controle de autorização, inseridos de alguma forma no fluxo de execução.

A figura 12 expõe uma variedade de situações para uma aplicação isolada, onde as regras de controle de acesso estão dispostas de forma pouco estruturada ou

praticamente nula. Esta situação é encontrada em muitos aplicativos legado, principalmente em ambientes de baixa plataforma, onde o desenvolvimento teve sua origem em uma arquitetura *client-centric* e também *client-server*, mas com uma motivação de independência de qualquer outro item de infra-estrutura ou de aplicativos externos. Como o mercado e as empresas exigem algum tipo de segurança e controle, a decisão de desenvolvimento passa por internalizar facilidades de controle dentro da aplicação, gerando condições restritivas de crescimento ou díspares em relação a boas práticas de implementação e isolamento.

Pode-se dizer que os recursos de controle estão fracamente ligados, ou que existem formas de burlar a camada de segurança (AM) inicialmente prevista. Embora não seja o escopo desta monografia detalhar todas estas possibilidades, é suficiente considerar a codificação maliciosa ou erros de parametrização como duas das causas para acesso indevido, sem a confiança de que o controle será honrado plenamente nestes aplicativos.

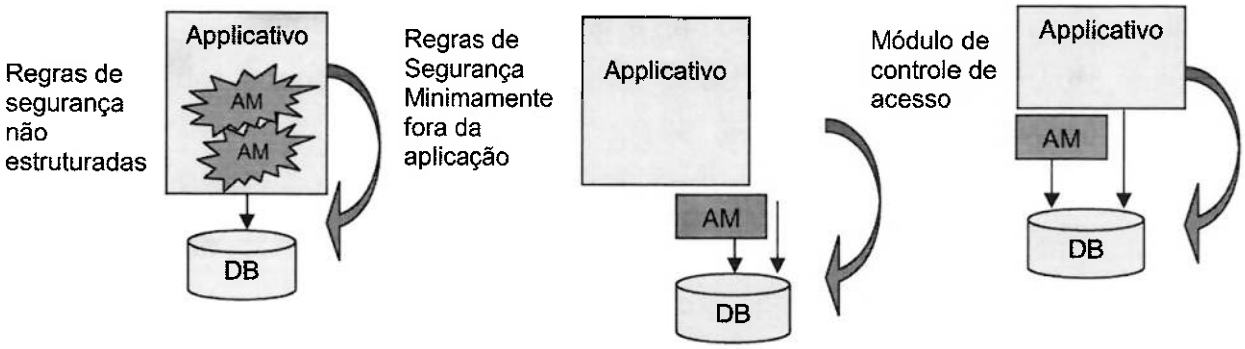

Figura 12 – Soluções de segurança monolíticas

A figura 13 explora a evolução do cenário das aplicações monolíticas, que é justamente da distribuição de componentes, onde a proposta de arquitetura é combinar recursos de uma ou mais componentes da aplicação para prover uma resposta para o usuário. Claramente, o aspecto de segurança e controle neste cenário precisou de ajustes e requer um balanço entre a capacidade de autenticação e autorização entre os componentes, em especial quando os mesmos estão executando em servidores distintos.

Nestes casos, pode-se dizer que o sistema de segurança tem controle sobre a decisão inicial de acesso e registro de auditoria, mas ainda pode estar vulnerável ao tratamento interno da aplicação e portanto, sujeita a codificação maliciosa ou erros, mas pode assegurar controles mais fortes para uma ou mais aplicações.

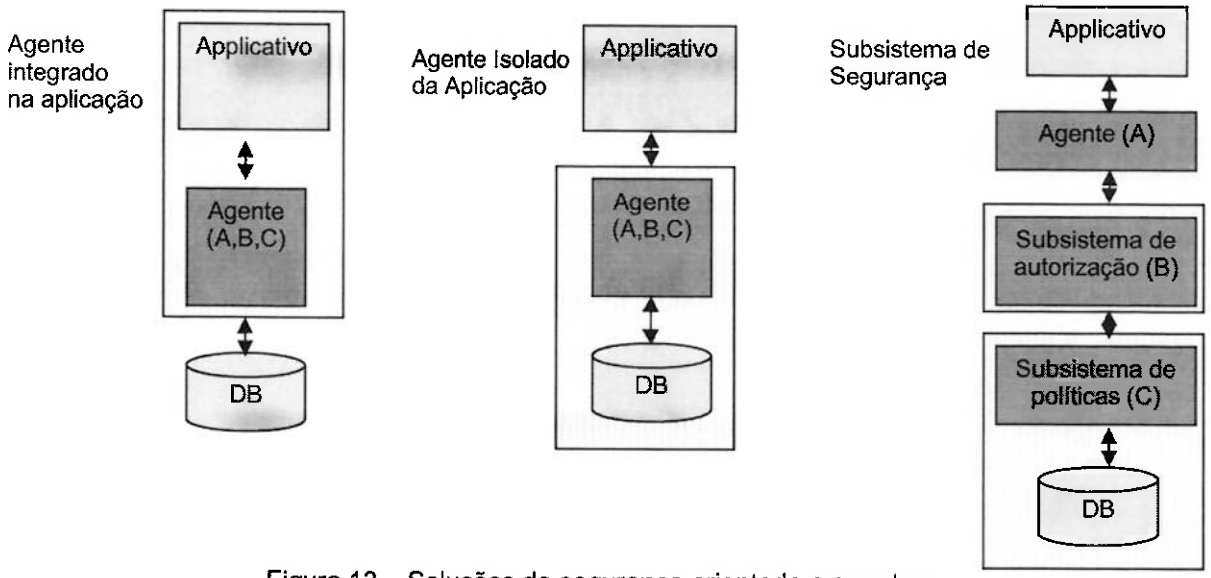

Figura 13 – Soluções de segurança orientada a agentes

4.1.3.2 Pré-requisitos para serviço

Os componentes de segurança criaram uma especialidade e uma complexidade maior, o que permite não somente ampliar a sua utilização como um subsistema robusto, mas também abre novas possibilidades de utilização destes mesmos componentes em múltiplas aplicações. Este última condição é justamente um dos pilares da arquitetura SOA, onde existe a proposta de abstração da estrutura de segurança em serviços de infra-estrutura.

Dentro do contexto de reuso e de serviço, faz sentido pensar na camada de segurança como um conjunto de serviços de TI, os quais são apresentados para as camadas de serviços de negócio como componentes ou serviços que podem ser combinados. A figura 14 exemplifica a separação conceitual que deve existir para aplicações SOA, onde a camada de segurança passa a fazer parte do subsistema de mensagens ou se integra a uma determinada via (*bus*) de serviço. Esta visão permite que a aplicação seja completamente isolada da aplicação, criando o primeiro passo efetivo para controle malicioso injetado na lógica de negócio.

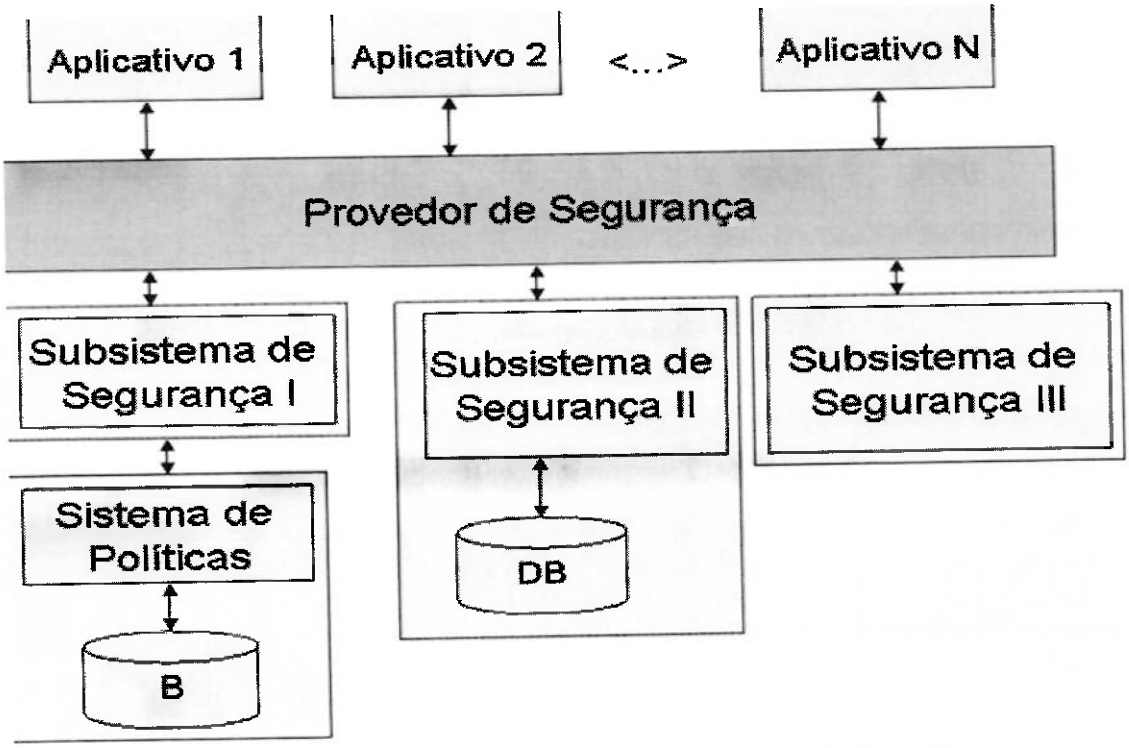

Figura 14 – Segregação entre componentes de aplicação

Aplicando o conceito de provedor de segurança como sendo a entidade (serviço) desejada no modelo final de uma estrutura segura, e usando o modelo de barramento para atender a funcionalidades distintas como serviços altamente configuráveis, têm-se uma primeira proposta de segurança que permite orientação a serviço, conforme a figura 15.

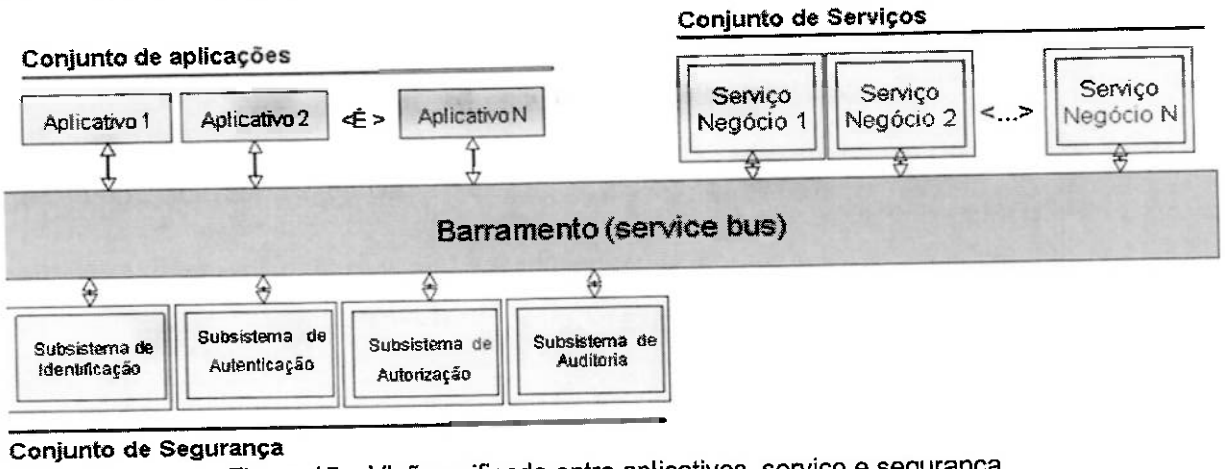

Figura 15 – Visão unificada entre aplicativos, serviço e segurança

4.1.3.3 Funções primitivas do serviço

Uma característica necessária em um serviço de autorização é poder confiar na informação recebida sobre a identidade do sujeito (usuário ou sistema) que solicitou

um recurso. A segunda condição importante é que a decisão produzida pelo sistema de autorização precisa ser honrada independente da lógica de aplicação, para evitar que o sistema seja burlado. Nesse sentido, o serviço de autorização precisa receber dados de identificação e precisa confrontá-los em tempo de execução no nível da operação transacional de negócio, para assegurar que nenhum componente intermediário produziu ou modificou a condição de segurança originalmente definida.

4.1.3.4 Controle e auditoria

Em se tratando de registro de atividade para pesquisa de auditoria, os eventos de sucesso e falha de uso de todos os itens elegíveis de controle precisam ser armazenados. Isso inclui a informação de usuário, data e hora do evento e informações sobre o que foi solicitado (qual recurso desejado), qual foi o nível de acesso solicitado e qual o máximo nível de acesso permitido para aquele sujeito no momento da checagem do sistema. Com estas informações, é possível validar a abrangência de atuação dos sujeitos (usuários e sistemas).

A auditoria deve considerar tanto os sujeitos como os objetos, com capacidade de pesquisa orientada aos agentes de acesso ao sistema bem como aos recursos disponíveis. Dessa forma, os filtros de controle e pesquisa podem refinar a visão de controle dependendo do tipo de condição pesquisada.

A granularidade para registro de eventos deve contemplar os grandes recursos tais como telas e formulários inteiros, recursos e operações tais como transações, bem como parâmetros e condições em tempo de execução tais como sessões.

4.1.3.5 Itens relevantes para o roteiro

O serviço de autorização deve oferecer as seguintes funcionalidades de serviço, a saber:

- Extração da credencial de um usuário
- Verificação de autenticidade e validade de uma credencial em tempo de execução
- Proteção de um recurso discreto (página web, transação, objeto de tela)
- Proteção de recursos genéricos ou agrupados
- Proteção paramétrica

- Proteção de privacidade

Os aspectos não funcionais que devem ser suportados são:

- Atendimento a múltiplas solicitações de consumidores ao mesmo tempo.
- Independência entre pedidos de autorização (stateless)
- Capacidade de se registrar e divulgar o serviço de autorização e suas propriedades
- Ajustar-se em tempo de execução a novas regras (definidas em uma política)
- Criptografia (proteção de dados sensíveis durante o transporte de rede)
- Auditoria (registro da atividade e pesquisa de sucesso e falha)
- Isolamento e agrupamento (segregação de domínios de informação)

4.1.4 Auditoria

4.1.4.1 Uso atual

A atividade executada pelos sistemas e usuários deve ser registrada por meio de eventos discretos, considerando a relevância dos dados, e o perfil de classificação da informação desejada. Normalmente, um sistema de auditoria registra os eventos de sucesso para algumas das operações internas tais como entrada e saída do sistema, troca de senha ou alteração de dados cadastrais (relativos a automação de processos de envio de senhas). Para os casos de tentativas sem sucesso, os sistemas de auditoria registram os recursos e os usuários que tentaram acessar o sistema, bem como o nível de acesso que foi negado. Isso serve para reconstruir o perfil de ataque no sistema. Eventualmente, um aplicativo pode também registrar a atividade cotidiana dos usuários, de forma a ter o registro dos acessos que foram executados e permitir que seja reconstruída a atividade normal para efeitos de rastreamento de operações. Existe ainda um refinamento possível que trata do registro orientado a um ou mais recursos, independente da ativação de trilhas para os usuários.

4.1.4.2 Pré-requisitos para serviço

Embora cada sistema tenha algum tipo de controle e capacidade de registro de eventos de auditoria, esta limita-se a eventos ocorridos dentro do domínio da aplicação, instância de banco de dados ou sistema operacional, sendo praticamente

inviável fazer o acompanhamento e o repasse destas informações entre sistemas distintos. Parte deste problema está ligado a forma de comunicação, onde um sistema pode controlar a execução da operação do outro, ou simplesmente delegar para que um outro sistema execute uma outra fase do processo. Neste último caso, o retorno para o sistema de origem não é necessário e portanto, os dados de auditoria acabam por ficar distribuídos em cada fase da execução de uma transação complexa entre sistemas.

4.1.4.3 Funções primitivas do serviço

A capacidade de centralizar os eventos de auditoria e correlacionar situações distintas conforme o tipo de usuário, padrões de execução normais e anormais, horários, operações executadas, volume financeiro e outras métricas são parte de sistemas centrais especializados em detecção de fraude, principalmente em sistemas de grande volume como transações de cartão de crédito. Isso sugere que um repositório central permite a tomada de decisão sobre padrões de comportamento e desvios em tempo de execução, o que é muito mais sofisticado do que coletar dados e elaborar relatórios mensais sobre desvios.

Em um ambiente em que os serviços podem chamar outros serviços, é necessário que todos tenham acesso ao mesmo repositório de dados para registrar a atividade dos usuários e viabilizar a pesquisa unificada. Esta característica deve fazer parte do sistema de identificação, autenticação e autorização como parte fundamental e primordial ao controle das primitivas de segurança. Além disso, as aplicações devem ter formas de registrar eventos específicos, de forma a reconstruir a seqüência de passos executados com detalhes suficientes para identificar quais dados foram alterados, inseridos ou consultados. Esta capacidade normalmente está além do registro de segurança, que se limita a preservar o usuário, o nome da regra que foi usada para conceder ou bloquear o acesso e controles de data e hora.

4.1.4.4 Itens relevantes para o roteiro

O serviço de auditoria deve oferecer as seguintes funcionalidades de serviço, a saber:

- Registro de eventos de segurança, tais como sucesso e falha
- Correlação de eventos simples e montagem de regras de comportamento

- Análise de ocorrências e padrões ao longo do tempo

Os aspectos não funcionais que devem ser suportados são:

- Atendimento a múltiplas solicitações de consumidores ao mesmo tempo.
- Independência entre pedidos de registro (*stateless*)
- Capacidade de se registrar e divulgar o serviço de auditoria e suas propriedades
- Ajustar-se em tempo de execução a novas regras (definidas em uma política)
- Criptografia (proteção de dados sensíveis durante o transporte de rede)
- Isolamento e agrupamento (segregação de domínios de informação)

4.1.5 Práticas de mercado

A abordagem de uma camada de segurança independente da aplicação, mas existente no contexto SOA para prover serviços seguros pode ser constatada em casos reais de implementação, como uma direção seguida por algumas empresas, pois além do apelo acadêmico deste trabalho, uma das constatações práticas é que estes conceitos encontram respaldo em situações de negócio da atualidade, com a representação destas idéias em modelos de referência comercialmente disponíveis.
Uma das propostas na linha de serviços de segurança é o modelo de referência apresentado pela IBM, exemplificado na arquitetura da figura 16. Neste modelo, existem três grandes grupos de serviços conhecidos como:

- Serviços de Segurança de negócio
- Infra-estrutura de políticas de segurança
- Serviços de segurança de TI

O Modelo considera os diferentes tipos e níveis de segurança que uma aplicação precisa ter nas diversas camadas. Tomando como exemplo o serviço de negócio para "segurança de identidade e acesso" deve-se definir as políticas apropriadas que serão gerenciadas pela infra-estrutura de segurança, que as distribuirá de forma consistente para todos os componentes relevantes. Estas políticas são reforçadas em pontos dentro dos componentes através de serviços de segurança, que podem estar localmente ou centralizados, aproveitando recursos de um sistema corporativo.

Figura 16 – Modelo de referência SOA da IBM (adaptado de BUECKER et al., 2007)

Para cada um dos serviços de segurança existe uma combinação entre as camadas do modelo e os sistemas, que na ótica do usuário final não é relevante. Esta capacidade de combinação, transparência e orquestração das entidades de TI para prover o serviço desejado é um forte aliado para este modelo, o qual poderá evoluir para outras soluções sem impactar as aplicações em uso. Esta condição é consistente com Crawford et al. (2005), que define o termo "infra-estrutura de commodities", onde os componentes novos, existentes ou de fornecedores serão consolidados em uma estrutura bem conhecida de SOA, onde tal agregação será apresentada como serviços e tais serviços utilizarão componentes agregados desta mesma infra-estrutura, caracterizando assim os elementos de segurança como elementos básicos e inerentes ao ambiente. O modelo IBM foi usado como uma referência conceitual para enriquecimento desta monografia, podendo ser usado outros modelos e até soluções que abordem segurança como uma camada de infra-estrutura.

DECLARAÇÃO DOS REQUISITOS DE SEGURANÇA

Nesta fase destacam-se as diretrizes de identificação, autenticação, autorização e auditoria que devem ser suportadas pela infra-estrutura de segurança e das capacidades que a aplicação deve utilizar a infra-estrutura de serviços de segurança disponíveis.

A tabela 4.2 resume as principais funcionalidades de segurança e as capacidades internas que a aplicação deve prever para ter um nível de segurança adequado e considerando as análises anteriores desta seção.

Aspecto de segurança	**Funcionalidades**	**Capacidades recomendadas para uma aplicação SOA**	**Observações**
Identificação	F1- Identidade /Tradução	Acesso a repositório de usuários locais e/ou remotos • Tradução de uma identificação em uma outra (uso de DE-PARA)	Serviço independente de aplicação ou modelo de dados
	F2 - Pesquisa	Coleta de listas de usuários com filtros de pesquisa genéricos ou específicos • Pesquisa de um usuário a partir de dados não completos (filtros) • Busca de um usuário • Recuperação de atributos de um usuário a partir da sua identificação chave	
	F3 - Manutenção Administrativa	Alteração de dados cadastrais locais ou remotos, dentro do domínio da aplicação	
	F4 – Aspectos não funcionais	Independência entre pedidos de identificação (stateless) • Capacidade de se registrar e divulgar o serviço de identificação e suas propriedades • Ajustar-se em tempo de execução a novas regras (definidas em uma política) • Criptografia (proteção de dados sensíveis durante o transporte de rede) • Auditoria (registro da atividade e pesquisa de sucesso e falha) • Isolamento e agrupamento (segregação de domínios de informação)	

		• Recursividade (capacidade de pesquisar bases indiretas, integração com serviços de identificação externos).	
Autenticação	F1 - Verificação	Checagem da credencial de um usuário de domínio local ou remoto. • Uso do serviço de identificação • Verificação de um usuário e de sua credencial Capacidade de transferir uma solicitação para outro sistema de segurança de autenticação (*function shipping*)	Serviço parametrizado com atributos e regras de múltiplos sistemas
	F2 - Manutenção Administrativa	Alteração de dados cadastrais locais ou remotos, dentro do domínio da aplicação	
	F3 – Aspectos não funcionais	Os aspectos não funcionais que devem ser suportados são: • Atendimento a múltiplas solicitações de consumidores ao mesmo tempo. • Independência entre pedidos de autenticação (*stateless*) • Capacidade de se registrar e divulgar o serviço de autenticação e suas propriedades • Ajustar-se em tempo de execução a novas regras (definidas em uma política) • Criptografia (proteção de dados sensíveis durante o transporte de rede) • Auditoria (registro da atividade e pesquisa de sucesso e falha) • Isolamento e agrupamento (segregação de domínios de informação) • Recursividade (capacidade de pesquisar bases indiretas, integração com serviços de identificação externos).	
Autorização	F1 - Controle de acesso discreto	Validar identidade e credencial de um sujeito em tempo de checagem e em tempo de uso. • Proteção de um recurso discreto	Serviço dependente do serviço de autenticação,

		(página web, transação, objeto de tela) • Proteção de recursos genéricos ou agrupados • Proteção de privacidade	para validar existência do acesso TOCTTOU
	F2 - Controle de acesso granular	Controle de acesso baseado em ACL, capacidades, matriz de acesso ou lista de atributos paramétricos esperados	
	F3 - Controle de acesso por contexto	Controle paramétrico por limites operacionais, freqüência, quantidade de operações, duração. • Extração da credencial de um usuário • Verificação de autenticidade e validade de uma credencial em tempo de execução	
	F4 - Controle de sessão	Controle de acesso baseado em classificação da informação Controle de acesso baseado em regras de contexto, tempo, histórico de operações / perfil dos últimos períodos	
	F5 - Pesquisa	Listar regras de segurança, agrupamentos e políticas em uso	
	F6 - Manutenção Administrativa	Alteração de dados cadastrais locais ou remotos, dentro do domínio da aplicação	
	F7 – Aspectos não funcionais	Os aspectos não funcionais que devem ser suportados são: • Atendimento a múltiplas solicitações de consumidores ao mesmo tempo. • Independência entre pedidos de autorização (stateless) • Capacidade de se registrar e divulgar o serviço de autorização e suas propriedades • Ajustar-se em tempo de execução a novas regras (definidas em uma política) • Criptografia (proteção de dados sensíveis durante o transporte de rede) • Auditoria (registro da atividade e	

		pesquisa de sucesso e falha) • Isolamento e agrupamento (segregação de domínios de informação)	
Auditoria	F1 - Registro de atividade orientada por usuário F2 – Registro de atividade orientada por recurso F3 - Correlação de eventos F4 - Consolidação de eventos no tempo F5 - Pesquisa F6 - Manutenção Administrativa F7 – Aspectos não funcionais	Registro de eventos de segurança, tais como sucesso e falha por usuário, grupo de usuários, recursos e grupos de recursos. Correlação de eventos simples e montagem de regras de comportamento Análise de ocorrências e padrões ao longo do tempo Filtro por período de eventos, tipo, usuário ou recurso Alteração de dados cadastrais locais ou remotos, dentro do domínio da aplicação Os aspectos não funcionais que devem ser suportados são: • Atendimento a múltiplas solicitações de consumidores ao mesmo tempo. • Independência entre pedidos de registro (*stateless*) • Capacidade de se registrar e divulgar o serviço de auditoria e suas propriedades • Ajustar-se em tempo de execução a novas regras (definidas em uma política) • Criptografia (proteção de dados sensíveis durante o transporte de rede) • Isolamento e agrupamento (segregação de domínios de informação)	Dependência de serviços de auditoria distribuídos para receber dados executados em outros ambientes

Tabela 4.2 – Recursos elegíveis de segurança para aplicações

5 ESTUDO DE CASO

O exemplo a seguir considera a aplicação simplificada da proposta de roteiro de segurança para uma aplicação de uma empresa. O método para geração do roteiro leva em conta os conceitos de segurança fundamentais, assumindo todos como necessários para a aplicação.

CLASSIFICAÇÃO

Esta fase define a seleção dos aplicativos elegíveis para o roteiro de segurança. Considerando o conjunto de *N* aplicativos (*N*>0), o processo de seleção para este exemplo resumiu-se na escolha de 1 sistema elegível, adotando o critério de quantidade (*M*=1). Em um contexto empresarial tradicional, a quantidade de aplicativos elegíveis para este processo pode ser aplicada para o limite total de aplicações disponíveis na empresa, (*N*>0 e 1<*M*<=*N*).

A aplicação denominada "CR1" consiste de um sistema de gestão de cartões de crédito com múltiplas funcionalidades, e que serão simplificadas para este processo em duas categorias e operações:

- (a) Transacional: Define as atividades vinculadas ao negócio, envolvendo diretamente o cliente final e seus dados na execução das operações permitidas pelo sistema. Nesta categoria estão operações de autorização de cartão de crédito, consulta de saldo, e validação de limite de crédito.
- (b) Administrativa: Define as operações que a equipe de apoio operacional deve ter acesso para assegurar o bom funcionamento e execução dos processos vitais do sistema, considerando a inclusão de cliente, mudança de dados cadastrais e manutenção limite de crédito.

Aplicando o critério de classificação para esta aplicação, temos os dados apresentados na tabela 5.1.

Visão	Avaliação quantitativa	Justificativa
Aplicação (Peso 5)	Nota: 5	Este sistema é um sistema já existente na empresa e utiliza um *middleware* de comunicação para transmissão de mensagens de autorização e verificação de crédito. Outros sistemas podem utilizar estas funções para lançar operações de crédito ou combinação de limite para outros produtos. Existe independência deste sistema e aplicando-se os critérios SOA, pode-se dizer que o sistema atende aos requisitos necessários de orientação a serviço.
Negócio (Peso 2)	Nota: 5	Este sistema representa 30% do lucro anual direto da empresa, e de 15% do lucro indireto, quando propicia serviços de integração com outros produtos (ex: financiamento parcelado)
Legal (Peso 1)	Nota: 5	Sujeito a regras de empresas externas, Banco central e regulamentação de lavagem de dinheiro

Tabela 5.1 – Demonstração do cálculo da aplicação

Fator aderência para avaliação = (5*5+2*5+5*1) / (5+2+1) = 5.0 (elegível)

ANÁLISE E IDENTIFICAÇÃO

As operações transacionais requerem que os seguintes fluxos de segurança sejam atendidos:

- Autorização de crédito (Identificação de cliente; Autenticação de ponto de venda, Autorização de limite; registro de evidência).
- Consulta saldo (Identificação de cliente, Autenticação de ponto de venda, registro de evidência)
- Validação de limite (Identificação de cliente, Autenticação de ponto de venda, registro de evidência)

Resumo dos requisitos necessários para este caso: 1-Identificação de cliente; 2-Autenticação de ponto de venda, 3-Autorização de limite; 4-Registro de evidência.

RESULTADOS

A tabela 5.2 demonstra os itens do roteiro para que a aplicação possa se beneficiar da infra-estrutura de segurança:

Aspecto de segurança	Funcionalidades	Capacidades recomendadas para uma aplicação SOA	Observações
Identificação	F1- Identidade	Garantir uso do serviço de identificação de segurança. Serviço *SOA_ctx_identify_user()*. Coleta de listas de usuários com filtros de pesquisa genéricos ou específicos • Pesquisa de um usuário a partir de dados não completos (filtros) • Busca de um usuário • Recuperação de atributos de um usuário a partir da sua identificação chave	Atender requisito 1- Identificação de cliente
	F4 – Aspectos não funcionais	Independência entre pedidos de identificação (*stateless*) • Capacidade de se registrar e divulgar o serviço de identificação e suas propriedades • Ajustar-se em tempo de execução a novas regras (definidas em uma política) • Criptografia (proteção de dados sensíveis durante o transporte de rede) • Auditoria (registro da atividade e pesquisa de sucesso e falha) • Isolamento e agrupamento (segregação de domínios de informação) • Recursividade (capacidade de pesquisar bases indiretas, integração com serviços de identificação externos).	
Autenticação	F1 - Verificação	Checagem da credencial de um ponto de venda. Serviço *SOA_ctx_authenticate()*.	Atender requisito 2- Autenticação de ponto de venda
	F3 – Aspectos não funcionais	Os aspectos não funcionais que devem ser suportados são: • Atendimento a múltiplas solicitações de consumidores ao mesmo tempo. • Independência entre pedidos de autenticação (*stateless*) • Capacidade de se registrar e divulgar o serviço de autenticação e suas propriedades	

		• Ajustar-se em tempo de execução a novas regras (definidas em uma política) • Criptografia (proteção de dados sensíveis durante o transporte de rede) • Auditoria (registro da atividade e pesquisa de sucesso e falha) • Isolamento e agrupamento (segregação de domínios de informação) • Recursividade (capacidade de pesquisar bases indiretas, integração com serviços de identificação externos).	
Autorização	F3 - Controle de acesso por contexto F4 - Controle de sessão F7 – Aspectos não funcionais	Controle paramétrico por limites operacionais, freqüência, quantidade de operações, duração. • Extração da credencial de um usuário • Verificação de autenticidade e validade de uma credencial em tempo de execução Controle de acesso baseado em regras de contexto, tempo, histórico de operações / perfil dos últimos períodos. Os aspectos não funcionais que devem ser suportados são: • Atendimento a múltiplas solicitações de consumidores ao mesmo tempo. • Independência entre pedidos de autorização (stateless) • Capacidade de se registrar e divulgar o serviço de autorização e suas propriedades • Ajustar-se em tempo de execução a novas regras (definidas em uma política) • Criptografia (proteção de dados sensíveis durante o transporte de rede) • Auditoria (registro da atividade e pesquisa de sucesso e falha) • Isolamento e agrupamento (segregação de domínios de informação)	Atender requisito 3- Autorização de limite

Auditoria	F1 - Registro de atividade orientada por usuário	Registro de eventos de segurança, tais como sucesso e falha por usuário.	Atender requisito 4- Registro de evidência
	F3 - Correlação de eventos	Correlação de eventos simples e montagem de regras de comportamento	
	F7 – Aspectos não funcionais	Os aspectos não funcionais que devem ser suportados são: • Atendimento a múltiplas solicitações de consumidores ao mesmo tempo. • Independência entre pedidos de registro (*stateless*) • Capacidade de se registrar e divulgar o serviço de auditoria e suas propriedades • Ajustar-se em tempo de execução a novas regras (definidas em uma política) • Criptografia (proteção de dados sensíveis durante o transporte de rede) • Isolamento e agrupamento (segregação de domínios de informação)	

Tabela 5.2 – Itens do roteiro identificados para a aplicação CR1

Considerando o conjunto de 4 requisitos de aplicação, o roteiro identifica 10 campos de ação para este perfil de aplicação, sendo que os aspectos não funcionais de controle são parecidos entre as categorias, o que sugere um esforço menor do que a soma de todos os itens identificados na tabela 4.2.

6 CONSIDERAÇÕES FINAIS

De acordo com um estudo (RAMARO, 2006), integração de segurança é algo difícil de cuidar e até mesmo questionar como a maturidade da tecnologia e fornecedores cobrem estas questões. Até recentemente, as tecnologias de integração não tentavam cobrir a segurança de forma abrangente, freqüentemente implementando uma solução adicional como segurança de transporte ou adicionando uma forma limitada de segurança. Com a ampla adoção de SOA tudo isso tende a mudar, pois se trata de uma tecnologia que possui várias utilidades e é uma solução que viabiliza a integração. Além disso, SOA traz várias padronizações para fazer a integração fácil.

SOA como modelo de arquitetura, não define qual ferramenta usar, mas formaliza os aspectos importantes que neste estudo, concentrou-se na capacidade de integração dos sistemas e sugere o uso de uma infra-estrutura de segurança como facilitadora de uma arquitetura orientada a serviços.

CONCLUSÃO

Uma constatação que vários autores e particularmente esta monografia ressalta é que as aplicações corporativas dependem de aspectos de segurança para uso confiável, registro seguro ou análise de risco. Os padrões de segurança mundiais apontam para os aspectos de identificação, autenticação, autorização e auditoria como necessários para o sucesso dos sistemas computacionais. A conclusão que uma camada de segurança é necessária dentro de uma arquitetura SOA fica evidente e sugere que as empresas estruturem seus modelos de aplicação para tirar proveito desta forma de trabalho. Independente do nome que se use, SOA ou outro, os serviços de segurança são peça fundamental para flexibilização e reuso das aplicações comerciais, devendo ser o caminho natural para o próximo ciclo de aplicações empresariais colaborativas. Usar uma camada de segurança flexível e independente deve fazer parte desta estratégia.

NOVAS LINHAS DE PESQUISA

Em se tratando de SOA, seu uso e entendimento ainda tende a aumentar e alterar a forma como as aplicações e os negócios são realizados nos meios de pagamento, serviços e rede de colaboração existente.

Uma evolução construída no princípio de SOA é o conceito de aplicações de negócio orientadas a serviço (do inglês SOBA - *service-oriented business applications*) que representa a nova geração de software estendendo as funcionalidades dos aplicativos tradicionais. As grandes empresas (como SAP, Oracle, IBM e Microsoft), outros fornecedores, provedores de serviços externos e usuários estão desenvolvendo os conceitos "SOBA" através do desenvolvimento de aplicações orientadas a serviço (do inglês SODA - *service-oriented development applications*).

O termo arquitetura de componente de serviço (do inglês SCA - *service component architecture*) é um padrão emergente que tem potencial para aumentar significativamente o desenvolvimento e a implementação de aplicativos SOA. O SCA baseia-se em interface gráfica, e em metadados para especificar soluções SOA utilizando políticas de serviços web.

Uma outra linha de pesquisa para o sucesso das estratégias SOA passa por aspectos de governança. A experiência demonstra que SOA requer um investimento maior em governança de TI, governança em processos de negócios e melhores práticas na integração de aplicativos do que antes. A governança de processos SOA é fácil de projetar, mas difícil de convencer e implementar.

Considerando o trabalho atual e seu potencial para aprofundamento, uma linha relevante de continuação deste trabalho seria refinar a fase de classificação de aplicações elegíveis desta monografia, utilizando por exemplo, modelos formais de análise como *balance score card*, ou ferramenta de análise de risco.

CONTRIBUIÇÕES

Este trabalho permitiu aprimorar o conhecimento da tecnologia SOA, e confirmar que os princípios de arquitetura segura encontram respaldo em SOA, pois permite que as aplicações façam bom uso dos recursos comuns (reuso) e deleguem a responsabilidade de execução de serviços não negócio para camadas distintas e especializadas.

Com relação às funcionalidades de segurança, os desenvolvedores de aplicações e os times que tem a responsabilidade de converter ou criar aplicativos numa arquitetura SOA podem (e devem) disseminar a cultura colaborativa, em que os serviços de segurança estejam disponíveis como parte da infra-estrutura independente da aplicação.

Este trabalho também abre oportunidade de refinamentos não somente teóricos como também práticos, em especial para a criação de serviços de segurança que implementem as funções descritas nesta monografia.

7 REFERÊNCIAS BIBLIOGRÁFICAS

AMOROSO, EDWARD G. **Fundamentals of computer security technology**. Prentice-Hall, 1994.

BUECKER, AXEL; ASHLEY, PAUL; BOUYSSOU, JULIEN; GARGARO, GIANLUCA; MUPPIDI, SRIDHAR; NEUCOM, RAY; READSHAW, NEIL; SCHINKE, GREGOR. **Understanding SOA Security**. IBM REDBOOKS - SG24-7310-00, 2007.

COX, D. E.; KREGER, H. **Management of the service oriented architecture life cycle**. IBM Systems Journal, VOL 44, NO 4, 2005.

CRAWFORD, C. H.; BATE, G. P.; CHERBAKOV, L.; HOLLEY, K.; TSOCANOS, C. **Toward an on demand service-oriented architecture**. IBM Systems Journal. VOL 44, NO 1, 2005.

FELMAN, JOSEPH. **Findings from security community meeting: Securing legacy systems**. Gartner Group G00140814, 2006.

ISO/IEC 27002:2005. **Information technology - Security techniques - Code of practice for information security management**. 2005.

ISO/IEC 9126-1:2001(E). **Software engineering — Product quality — Part 1: Quality model**. 2001

KANNEGANTO, RAMARO; CHODAVARAPU, PRASAD. **SOA Security in action. Manning Publications**, 2006. no prelo.

NATIS, YEFIM V.; SCHULTE, ROY W. **Advanced SOA for Advanced Enterprise Projects**. Gartner Group - G00141940, 2006.

NATIS, YEFIM V.; PEZZINI, MASSIMO; SCHULTE, ROY W.; IIJIMA, KIMIHIKO. **Predicts 2007: SOA Advances**. 2006. ID Number: G00144445

SCHULTE, ROY W.; ABRAMS, CHARLES. **SOA Overview and Guide to SOA Research**. Gartner Group G00144894, 2006.

SUMMERS, R. C. **An overview of computer security**. IBM Systems Journal. VOL 23, No 4., 1984.

OECD. **Guidelines governing the protection of privacy and transborder flows of personal data**. Computer Networks, NO. 2 1981.