ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO

CHRISTIAN REIS MENEGUIN

APLICAÇÃO DA ARQUITETURA ORIENTADA A SERVIÇOS UTILIZANDO TÁTICAS BASEADAS EM ATRIBUTOS DE QUALIDADE

Monografia apresentada a Escola Politécnica da Universidade de São Paulo para conclusão do Curso de MBA em Tecnologia de Software.

Área de concentração:
Engenharia de software

Orientadora:
Profª. Drª. Jussara Pimenta Matos

São Paulo
Janeiro 2007

FICHA CATALOGRÁFICA

Meneguin, Christian Reis
Aplicação da arquitetura orientada a serviços utilizando táticas baseadas em atributos de qualidade / C.R. Meneguin. -- São Paulo, 2007.
87p.

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

1.Desenvolvimento de software (Recomendação) 2.Qualidade de software 3.Arquitetura orientada a serviços I.Universidade de São Paulo. Escola Politécnica. Programa de Educação Continuada em Engenharia II.t.

Aos pais, Pedro e Mônica
pelo apoio.
À namorada Priscilla pelo
carinho incondicional.

OK

Agradecimentos

À Profa. Dra. Jussara Pimenta Matos, pelo apoio, orientação, dedicação e paciência que sem os quais não seria possível a execução deste trabalho.

À Profa. Dra. Selma Shin Shimizu Melnikoff, pela oportunidade de realizar este curso.

Aos professores do curso de MBA em Tecnologia de Software, pelo conhecimento transmitido.

Aos amigos Arthur Utiyama e Rodolfo Miao, por colaborarem com o meu aprendizado ao trabalharmos em grupo durante este curso.

Aos amigos de classe, pela oportunidade da troca de experiências.

RESUMO

O objetivo deste trabalho é apresentar um conjunto de táticas relacionadas aos atributos de confiabilidade, de segurança e de desempenho. Por meio da análise desses atributos e avaliando os impactos em relação aos sistemas de software que utilizam a Arquitetura Orientada a Serviços, essas são usadas para auxiliar o desenvolvimento de um projeto.

A Arquitetura Orientada a Serviços (Service Oriented Architecture – SOA) pode ser utilizada de forma a integrar diferentes tipos de sistemas de software, disponibilizando suas funções através de serviços. Para este tipo de desenvolvimento é importante considerar não somente a necessidade de atendimento às regras de negócio, mas também a qualidade do software.

ABSTRACT

The objective of this work is to present a set of tactics related with the quality attributes like reliability, security and performance. The analysis of these attributes and evaluating the impacts in the software systems based on the Services Oriented Architecture are used to help on software system development project.

The Services Oriented Architecture (SOA) can be applied as a way to integrate heterogeneous software systems, to render possible yours functionalities through services. For this kind of development is important consider the necessity the quality of software instead just to attend the business rules.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CORBA – Common Object Request Broker Architecture

DMZ – Demilitarized zone

HTTP – HyperText Transfer Protocol

HTTPS – HyperText Transfer Protocol Secure

IBM – International Business Machines Corporation

IIOP – Internet Inter-ORB Protocol

J2EE – Java 2 Platform, Enterprise Edition

JMS – Java Message Service

SAML – Security Assertion Markup Language

SCM – Supply Chain Management

SOA – Service-Oriented architecture

SOAP – Simple Object Access Protocol

SSL – Secure Sockets Layer

UDDI – Universal Description Discovery and Integration

URL – Uniform Resource Locator

WSDL – Web Services Description Language

WSFL – Web Services Flow Language

XML – Extensible Markup Language

1. INTRODUÇÃO

Este capítulo apresenta considerações iniciais a respeito da presente monografia, desenvolvida com base na Arquitetura Orientada a Serviços (SOA) e Qualidade de Software. Nesta parte é mostrado o objetivo, a justificativa, a metodologia e a estrutura do trabalho.

1.1 OBJETIVO

O objetivo deste trabalho é apresentar diretrizes, denominadas de táticas, para o desenvolvimento de sistemas de software utilizando Arquitetura Orientada a Serviços (Service Oriented Architecture - SOA), tendo como foco o atendimento aos atributos de qualidade de confiabilidade, segurança e desempenho. Para isto, é realizado um mapeamento entre a Arquitetura Orientada a Serviços e esse conjunto de atributos (ENDREI et al, 2004; O'BRIEN; BASS; MERSON, 2005).

Além desse conjunto de atributos, também é adotada a tecnologia dos Web Services (KULKARNI et al, 2005). A adoção dessa tecnologia tem como um dos indicadores, o crescimento do número de sistemas de software que a utilizam, sendo também amplamente usada em computação distribuída e encontrada em diversos segmentos de negócios tais como, comércio eletrônico, instituições financeiras e multinacionais através de ERP(s) (DUAN et al, 2005; KESSLER, 2004; KULKARNI et al, 2005; PAPAZOGLOU; VAN DEN HEUVEL, 2005).

Para um melhor entendimento dos conceitos relativos à Arquitetura Orientada a Serviços, esta é descrita por meio de padrões, onde é mostrada a maneira como ocorrem as

interações entre entidades envolvidas neste tipo de sistema de software (BUSCHMANN et al, 1996).

1.2 JUSTIFICATIVA

A necessidade de trabalhar com informações originadas em sistemas de software heterogêneos é motivo de estudo por diferentes pesquisadores. Isto levou ao surgimento da Arquitetura Orientada a Serviços (SOA) que permite este tipo de integração com certa facilidade e de maneira satisfatória (BROWN; JOHNSTON; KELLY, 2002; DUAN et al., 2005).

A utilização de SOA é crescente no desenvolvimento de diferentes tipos de sistemas de software, sendo recomendado trabalhar com níveis de qualidade de software elevados na adoção desta arquitetura (KESSLER, 2004; KULKARNI et al, 2005; GANCI et al, 2006). Um estudo feito por Forrester Research (COMPUTERWORLD b, 2006), em 32 empresas globais, com 40 mil ou mais funcionários, aponta que SOA é prioridade para a maioria delas, sendo que 67% destas estão implantando sistemas de software baseados nesta arquitetura em 2006. Além disso, apresenta também que 83% das empresas estão utilizando SOA para integração interna.

As posições do instituto Gartner publicadas em 2005 (CEARLEY; FENN; PLUMMER, 2005; COMPUTERWORLD, 2005), dão conta que para o ano de 2008, a receita de muitos sistemas de software será fruto da utilização de SOA em seu desenvolvimento e em 2010, mais de 50% das grandes empresas terão estabelecido um portfólio de sistemas sob este padrão, em busca de uma plataforma unificada de negócios.

Além disto, organizações como W3C (2006) trabalham na elaboração de normas e de documentos com o objetivo de incentivar a utilização de boas práticas no desenvolvimento de sistemas de software baseados na Arquitetura Orientada a Serviços, facilitando sua aceitação em empresas distintas.

1.3 METODOLOGIA

Para a elaboração deste trabalho, a metodologia utilizada foi:

- Pesquisas bibliográficas reunindo livros, artigos e publicações de fontes como Software Engineering Institute (SEI) e Institute of Electrical and Electronics Engineers (IEEE). Também foram pesquisados documentos publicados na Internet, contendo informações relacionadas ao assunto.
- Revisão, análise e seleção do material reunido, selecionando entre as fontes pesquisadas, as que possuem material consistente para compor a monografia.
- Reunião dos conceitos relacionados à Arquitetura Orientada a Serviços, Qualidade de Software e meios para atingir os atributos de qualidade esperados.
- Desenvolvimento do trabalho com base no material pesquisado e o auxílio do orientador.

1.4 ESTRUTURA DO TRABALHO

O Capítulo 1 apresenta a Introdução, Objetivo, Justificativa e a Estrutura do Trabalho.

O Capítulo 2 apresenta os conceitos sobre Arquitetura Orientada a Serviços (Service Oriented Architecture SOA), considerando como os Padrões (Patterns) podem auxiliar no desenvolvimento dos sistemas de software, também discute aspectos da tecnologia de Web Services, utilizada para implantação dos princípios de SOA.

O Capítulo 3 apresenta os atributos de qualidade relacionados à confiabilidade, segurança e desempenho, também são apresentadas as táticas relacionadas aos atributos previamente selecionados.

O Capítulo 4 apresenta uma proposta para utilizar os atributos de qualidade selecionados, de maneira que estes possam colaborar no atendimento as necessidades e no desenvolvimento de sistemas de software baseados na Arquitetura Orientada a Serviços, por meio da utilização das táticas relacionadas.

O Capítulo 5 apresenta um estudo de caso para aplicação das táticas descritas neste trabalho em um sistema de software relacionado ao Gerenciamento da Cadeia de Suprimentos (Supply Chain Management – SCM).

O Capítulo 6 apresenta a conclusão deste trabalho, assim como, sugestões para trabalhos futuros.

2. ARQUITETURA ORIENTADA A SERVIÇOS

2.1 CONSIDERAÇÕES INICIAIS

O objetivo desta seção é apresentar a Arquitetura Orientada a Serviços (SOA), utilizada como base para o desenvolvimento deste trabalho. Também são mostradas as interações entre entidades envolvidas nessa arquitetura, juntamente com os elementos que compõe estas entidades. Estes elementos são agrupados em camadas, de maneira a facilitar a definição da responsabilidade de cada um em relação as suas funções (BROWN; JOHNSTON; KELLY, 2002; ENDREI et al, 2004).

Para colaborar com o desenvolvimento de sistemas de software baseados na Arquitetura Orientada a Serviços, quanto à descrição das interações envolvidas, são utilizados nesta seção os padrões de software (BUSCHMANN et al, 1996; ZDUN et al, 2006).

Nessa parte também é mostrada de forma mais detalhada a tecnologia dos Web Services (Korotkiy; Top, 2006), utilizada para demonstrar como os sistemas de software baseados em SOA podem ser implantados. Além dessa tecnologia, são mostrados outros meios para implantar a Arquitetura Orientada a Serviços, como Corba (OMG Corba, 2004) e Java Message Service (JMS) (Hapner, 2002).

2.2 DEFINIÇÕES SOBRE A ARQUITETURA ORIENTADA A SERVIÇOS

Arquitetura Orientada a Serviços (Service Oriented Architecture - SOA) pode ser definida como uma maneira de desenvolver sistemas de software, onde estes devem ser

criados em forma de conjuntos de serviços, que devem ser utilizados nas trocas de mensagens. Estes serviços consistem em encapsular as funcionalidades dos sistemas de software, devendo ser publicadas e acessadas somente através de interfaces consistentes (BROWN; JOHNSTON; KELLY, 2002; ENDREI et al, 2004; GANCI et al, 2006; HASHEMI; RAZZAZI; BAHRAMI, 2006; O'BRIEN; BASS; MERSON, 2005; ZDUN et al, 2006).

Em SOA, os serviços são disponibilizados por meio de entidade que pode ser definida como Provedor de Serviços. Também deve ser considerada a entidade Aplicação Cliente, denominada desta forma, quando se trata do sistema de software que acessa serviços disponíveis em outras entidades (BROWN; JOHNSTON; KELLY, 2002; ENDREI et al, 2004; GANCI et al, 2006; HASHEMI; RAZZAZI; BAHRAMI, 2006).

Na Arquitetura Orientada a Serviços, as entidades relacionadas podem ter os elementos que as compõem distribuídos em camadas como uma forma de apresentação (ENDREI et al, 2004). A figura 2.1 mostra estes elementos e sua organização em camadas.

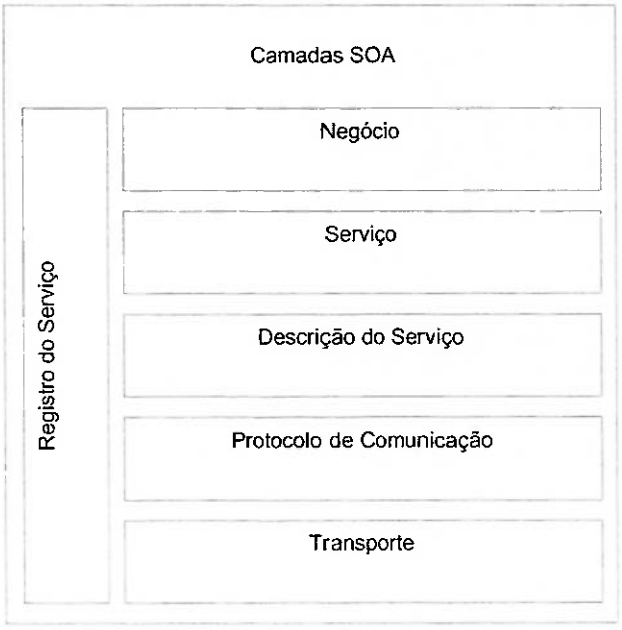

Figura 2.1 – Camadas de SOA
Baseada em: Endrei et al. (2004), ZDUN et al. (2006)

As camadas apresentadas na figura 2.1 são definidas a seguir:

- Transporte: esta camada é responsável por gerenciar as mensagens entre provedor e cliente, está relacionada aos mecanismos usados para encaminhar as requisições de serviços, feitas por aplicações clientes a seus respectivos provedores e as mensagens de respostas encaminhadas a partir destes provedores para as aplicações cliente.
- Protocolo de Comunicação: esta camada é responsável por definir as tecnologias e protocolos responsáveis pela realização da comunicação entre a aplicação cliente e provedor, facilitando o transporte de mensagens entre as entidades emissoras e receptoras.
- Descrição do Serviço: esta camada é responsável por concentrar as informações sobre os atributos de cada serviço, como este deve ser invocado e quais são os dados necessários para que o processo ocorra com sucesso. Isto possibilita que serviços provenientes de diferentes tipos de sistemas de software possam adequar o formato de suas requisições e aguardar o retorno destas, de acordo com as descrições apresentadas.
- Serviço: esta camada é responsável por apresentar os serviços desenvolvidos a partir das funcionalidades pertinentes ao sistema de software relacionado. Estes podem ser invocados com base nas definições apresentadas na camada de descrição dos serviços.
- Negócio: esta camada é responsável por apresentar o processo de negócio ou o fluxo de trabalho, composto por uma coleção de serviços, que podem ser solicitados em uma seqüência definida e sob um conjunto particular de regras. Este processo deve reunir os requisitos do negócio.
- Registro do Serviço: esta camada é responsável por gerenciar o repositório de serviços, usado pelo provedor para publicar as definições dos serviços oferecidos. Este repositório também é utilizado pela aplicação cliente, com o objetivo de obter informações sobre os serviços disponíveis.

Além dos elementos de SOA e de sua distribuição em camadas, é possível analisar nesta arquitetura, a forma como ocorrem interações entre as entidades. Estas interações podem ser representadas através do paradigma "find-bind-execute" (Mahmoud et al, 2005), também citado como "find, bind and invoke" (Endrei et al, 2004).

Este paradigma preconiza que uma aplicação cliente realiza busca de informações em provedores que disponibilizam registros sobre os serviços, considerando um conjunto de critérios pré-estabelecidos. Após obter as regras em seu contrato e o endereço do serviço, a aplicação cliente pode acessar o serviço através de sua interface. A representação deste paradigma está apresentada na figura 2.2, onde as interações envolvendo entidades que compõem a Arquitetura Orientada a Serviços são descritas (ENDREI et al., 2004; MAHMOUD et al, 2005; VO; WEINHARDT; WOJCIECHOWSKI, 2006).

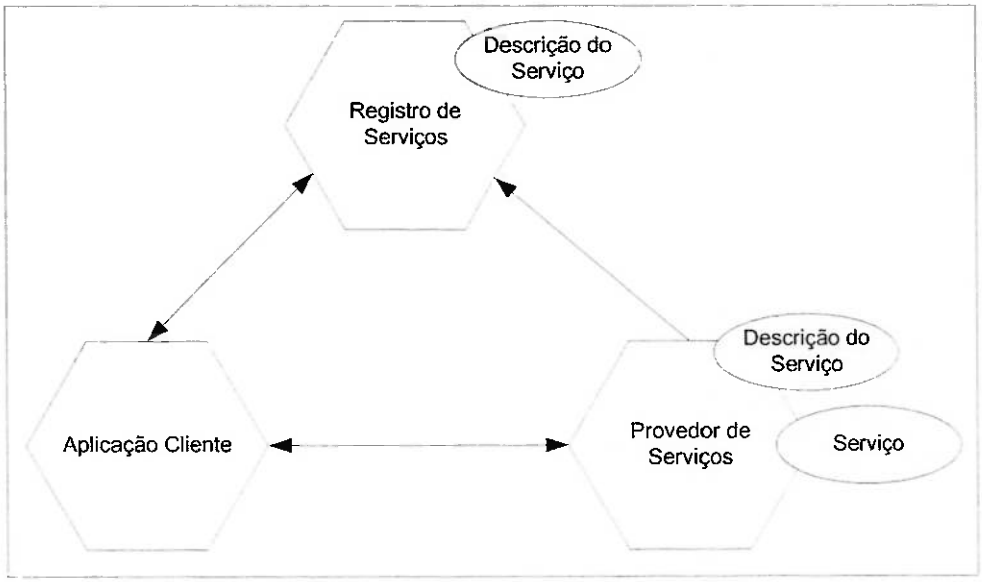

Figura 2.2 - Interação entre os elementos de SOA
Baseado em: Mahmoud, (2005); Endrei et al. (2004)

A figura 2.2 é formada pelos seguintes itens:

- Aplicação Cliente: esta entidade pode ser formada por outros serviços que enviam mensagens com requisições. A aplicação cliente realiza a busca por serviços e atendendo as restrições estabelecidas, utiliza o serviço.
- Provedor de Serviços: esta entidade publica os serviços e seus contratos de interface, possibilitando que estes sejam utilizados por aplicações solicitantes.
- Registro de Serviços: este item abriga o repositório de serviços disponíveis, permite que as informações sobre serviços sejam publicadas e possibilita a busca a estes por aplicações cliente (MAHMOUD, 2005; ENDREI et al, 2004).

Em sua concepção, a Arquitetura Orientada a Serviços facilita a interoperabilidade, aplicando o conceito de baixo acoplamento e adotando um modelo padrão de comunicação. Esta arquitetura também possibilita o desenvolvimento de sistemas de software, disponibilizando os serviços com transparência quanto a sua localização física (BROWN; JOHNSTON; KELLY, 2002; TSAI et al., 2006).

Entretanto, construir serviços para sistemas de software existentes, a fim de obter benefícios da arquitetura SOA, não é automático ou muito fácil, tal migração pode representar uma tarefa complexa da engenharia de software. Isto devido à necessidade de uma análise, que pode apontar para um trabalho de reengenharia da aplicação existente (BROWN; JOHNSTON; KELLY, 2002; ENDREI et al, 2004; LEWIS; MORRIS; SMITH, 2006; SNEED, 2006).

Para o desenvolvimento de sistemas de software que utilizem a Arquitetura Orientada a Serviços, tecnologias como CORBA, Java Message Service (JMS) e Web Services podem ser adotadas (HAPNER, 2002; KOROTKIY; TOP, 2006; OMG CORBA, 2004; STAL, 2006). Uma visão geral destas tecnologias é apresentada seguir:

- CORBA (Common Object Request Broker Architecture): trata-se de tecnologia independente de infra-estrutura e utilizada por sistemas de software para comunicação em redes de computadores. Baseia-se no protocolo IIOP e seus sistemas de software são compostos por objetos que representam unidades individuais de softwares que combinam funcionalidades e dados (OMG CORBA, 2004).
- Java Message Service (JMS): trata-se de um padrão desenvolvido com a utilização da linguagem JAVA e a plataforma J2EE para troca de mensagens entre sistemas de software distintos. Por tratar-se de um padrão baseado em linguagem definida, é importante para os sistemas de software que pretendem adotar o JMS, ser compatíveis com a linguagem JAVA (HAPNER, 2002).
- Web Service: trata-se de tecnologia utilizada para troca de mensagens entre sistemas de software distintos e baseia-se na utilização de protocolos como XML e SOAP. Adicionalmente, sistemas de software que utilizam Web Services podem ser criados independentemente da linguagem de programação utilizada, do sistema operacional eleito para os servidores que devem abrigar os Web Services e independente do hardware utilizado (BROWN; JOHNSTON; KELLY, 2002; ENDREI et al, 2004, KOROTKIY; TOP, 2006).

2.3 PADRÕES UTILIZADOS NA ARQUITETURA ORIENTADA A SERVIÇOS

Os sistemas de software baseados na Arquitetura Orientada a Serviços (SOA) podem ser implantados utilizando um leque de tecnologias, conforme citado na seção 2.2. Devido a isto, um conjunto de atributos pertinentes a SOA, podem ser interpretados de maneira incerta por profissionais ligados ao processo de concepção do sistema de software. Para evitar este tipo de ocorrência, são utilizados padrões para descrição da Arquitetura Orientada a Serviços (STAL, 2006; ZDUN et al., 2006).

A utilização de Padrões (Patterns) é importante porque em sua concepção, eles descrevem um problema recorrente de projetos ou implantação de sistemas de software que acontece em um contexto específico, apresentando uma estrutura bem definida e devidamente provada para solução do problema. Para analisar a viabilidade de sua adoção, cada padrão é composto por três partes, as quais expressam a relação entre um determinado contexto, um determinado problema e uma solução para este (BUSCHMANN et al., 1996; GAMMA et al., 1998). Os padrões podem ser classificados basicamente em (BUSCHMANN et al., 1996; FERNANDEZ b; DELESSY, 2006; GAMMA et al., 1998):

- Padrões de Arquitetura (Architectural Patterns): que expressam a estrutura organizacional de um sistema de software.
- Padrões de Projetos (Design Patterns): que possibilitam a criação de um esquema para refinar os subsistemas ou componentes de um sistema de software.
- Padrões de Idiomas (Idioms): que se referem aos padrões específicos para programação, descrevendo como implantar aspectos particulares de um subsistema.

Através da adoção de Padrões usados na Arquitetura Orientada a Serviços, são descritas as interações entre as entidades relacionadas. Desta forma, o processo de busca de informações sobre serviços em provedores que disponibilizam os devidos registros, pode ser descrito usando Padrões. Esta descrição é mostrada a seguir, juntamente com o padrão utilizado.

Determinada aplicação cliente realiza busca de informações em provedores que disponibilizam registros sobre serviços. Estes registros são previamente inseridos com base nos serviços disponíveis em um determinado servidor. Se o registro possui o serviço solicitado, a aplicação cliente poderá utilizá-lo após obter as regras para acessar sua interface e receber o endereço para acesso ao serviço. Esta interação pode ser refletida no padrão *Lookup* (Kircher; Jain, 2004), que se refere ao padrão onde os serviços são

publicados em um diretório e as aplicações cliente podem realizar buscas para localizar os serviços desejados e obter seu contrato de uso, servindo como um ponto central de comunicação entre as aplicações cliente e os serviços. A figura 2.3 apresenta a interação entre a aplicação cliente e o serviço através o padrão Lookup. Esta interação consiste em (KIRCHER; JAIN, 2004; ZDUN et al, 2006):

1. A aplicação cliente busca nos registros de serviços por um serviço específico.
2. O registro de serviços, responde informando que possui as informações para acesso ao serviço.
3. A aplicação cliente envia requisição ao registro de serviços, visando obter informações sobre a interface e contrato de acesso ao serviço desejado.
4. O registro de serviços responde as informações solicitadas.
5. A aplicação cliente utiliza as informações recebidas para acessar o serviço desejado.
6. O serviço realiza o processamento da requisição.
7. O serviço envia a resposta da requisição feita para a aplicação cliente.

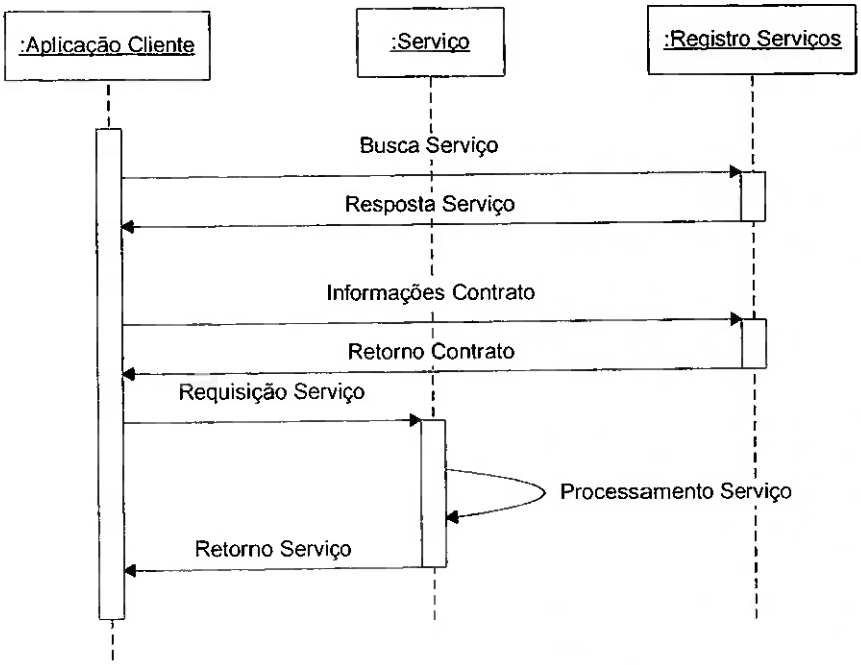

Figura 2.3 – Busca de Serviço
Baseado em Kircher; Jain. (2004)

Sendo finalizado o processo de localização do serviço desejado, ou possuindo as informações de acesso ao serviço, a requisição pode ser realizada através de interação direta entre as entidades: aplicação cliente e provedor de serviços. A descrição deste processo e definição do padrão utilizado é mostrada a seguir.

As interações entre a aplicação cliente e o provedor de serviços podem ser descritas utilizando o padrão *Broker* (Buschmann et al, 1996), que se refere ao padrão usado para estruturar sistemas de software distribuídos que interagem por meio de invocações remotas. Este padrão também pode ser responsável por coordenar toda a comunicação entre os sistemas, como as requisições enviadas por aplicações cliente e as respostas emitidas pelos serviços. Associado ao padrão Broker, pode ser utilizado também o padrão *Proxy* (Buschmann et al., 1996; Gamma et al., 1998) que consiste em criar representação da entidade (aplicação cliente ou serviço) original, apresentando o mesmo tipo de interface, onde determinada entidade utiliza para a interação.

O padrão Client-side Proxy (Buschmann et al, 1996), que pode ser considerado uma variação do padrão Proxy, é responsável por tornar invisíveis os detalhes da aplicação cliente e possibilitar que o serviço existente em provedores distintos seja acessado da mesma maneira que serviços locais. De maneira semelhante, os provedores que abrigam os serviços podem adotar o padrão Server-side Proxy (Buschmann et al, 1996), análogo ao padrão Client-side Proxy, com a diferença de que este permite o recebimento das mensagens de requisições e emite mensagens de retorno para as aplicações cliente. Os serviços são dispostos através de uma interface remota que emprega o padrão Interface Description (Zdun et al, 2006), o qual contém os detalhes sobre o serviço apresentado (BUSCHMANN et al, 1996; STAL et al, 2006; ZDUN et al, 2006). A figura 2.4 mostra a interação entre a aplicação cliente e o serviço, utilizando um padrão Broker. Esta interação consiste em:

1. A aplicação cliente envia uma requisição para o Client-side Proxy.

2. O Client-side Proxy empacota a requisição e outras informações relevantes, em seguida encaminha a requisição ao Broker.
3. O Broker busca o serviço solicitado e envia a requisição ao Server-side Proxy.
4. O Server-side Proxy desempacota a requisição e envia para o serviço.
5. O serviço processa a requisição e envia a resposta ao Server-side Proxy.
6. O Server-side Proxy empacota a resposta e outras informações relevantes, em seguida encaminha a requisição ao Broker.
7. O Broker busca a aplicação cliente solicitante e envia a resposta ao Client-side Proxy.
8. O Client-side Proxy desempacota a resposta e envia para a aplicação cliente.
9. A aplicação cliente processa a resposta recebida.

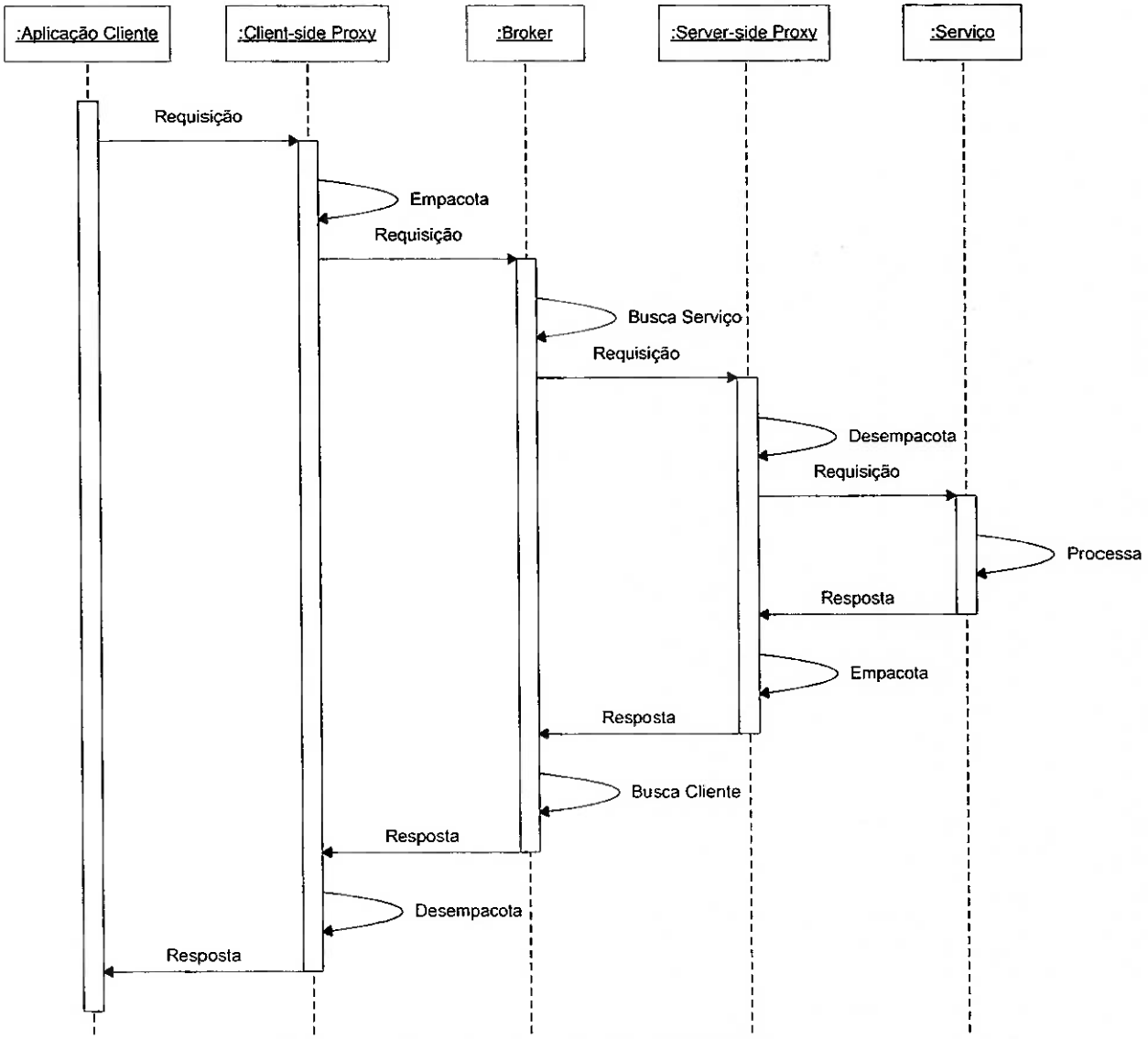

Figura 2.4 – Interação entre aplicação cliente e serviço
Baseado em Buschmann et al. (1996)

Os Padrões mencionados nesta seção estão relacionados com as interações entre os sistemas de software desenvolvidos com base na Arquitetura Orientada a Serviços. Através destes Padrões, os conceitos envolvendo a arquitetura são mostrados sem a necessidade de especificar uma tecnologia para o seu desenvolvimento (STAL, 2006; ZDUN et al., 2006).

2.4 A TECNOLOGIA DE WEB SERVICES

A tecnologia de Web Services pode ser utilizada para concretização de sistemas de software baseados na Arquitetura Orientada a Serviços. Características desta arquitetura, como promover a interoperabilidade entre sistemas de software ou mesmo a maneira de interação entre eles, podem ser empregados em Web Services (ENDREI et al., 2004; HUANG, 2006; KULKARNI et al., 2005; LEWIS et al., 2005; NEZHAD et al., 2006).

Ao observar os componentes que devem ser reunidos no desenvolvimento dos sistemas de software que utilizam Web Services, os seguintes padrões, especificações e protocolos podem ser encontrados (ISIS, 2006; MALLOY, 2006; W3C SOAP, 2000; W3C WSDL, 2001; W3C XML, 2004; UDDI, 2004; W3C WSA, 2004):

- eXtensible Markup Language (XML): foi padronizada para gerar linguagens de marcação para necessidades especiais, onde permite a troca de informações de forma estruturada através da Internet. Permite a criação de tags próprias, não existentes no HTML. Documentos XML são compostos de unidades de armazenamento contendo os dados. As marcações da linguagem são responsáveis por organizar a disposição dos dados armazenados e a estrutura lógica (W3C XML, 2004).
- Web Services Description Language (WSDL): WSDL é baseado em XML e usado para descrever os serviços que determinados Web Services disponibilizam, onde estão localizados e quais os parâmetros necessários para sua utilização. A comunicação ocorre através da troca de mensagens que possuem informações sobre documentos ou procedimentos (ISIS, 2006; W3C WSDL, 2001).
- Simple Object Access Protocol (SOAP): SOAP é definido como um mecanismo para transmissão de mensagens baseadas em XML entre pontos de um ambiente de rede

descentralizado. Este protocolo consiste basicamente de três partes. Um envelope responsável por abrigar as mensagens enviadas, regras para definir os mecanismos de serialização dos dados e as convenções que devem ser usadas para os procedimentos de chamada remota. (ISIS, 2006; W3C SOAP, 2000)

- Universal Description, Discovery and Integration (UDDI): trata-se de uma especificação baseada no formato XML para manter de forma padronizada diretórios, onde aplicações de sistemas de software podem listar os serviços disponíveis em Web Services e também buscar dinamicamente por outros serviços. As Web Services são categorizadas de acordo com o tipo de serviço disponível e localização (ISIS, 2006; UDDI, 2004).

Conforme citado, os Web Services estão associados a Arquitetura Orientada a Serviços. Para mostrar esta associação, é realizado neste trabalho, o mapeamento entre os componentes utilizados em Web Services e os respectivos elementos que compõem SOA.

Este mapeamento é feito utilizando a estrutura mostrada na seção 2.2 (figura 2.1). A partir desta, são associados os padrões, especificações e protocolos ligados aos Web Services (ENDREI et al., 2004). A figura 2.5 mostra este mapeamento.

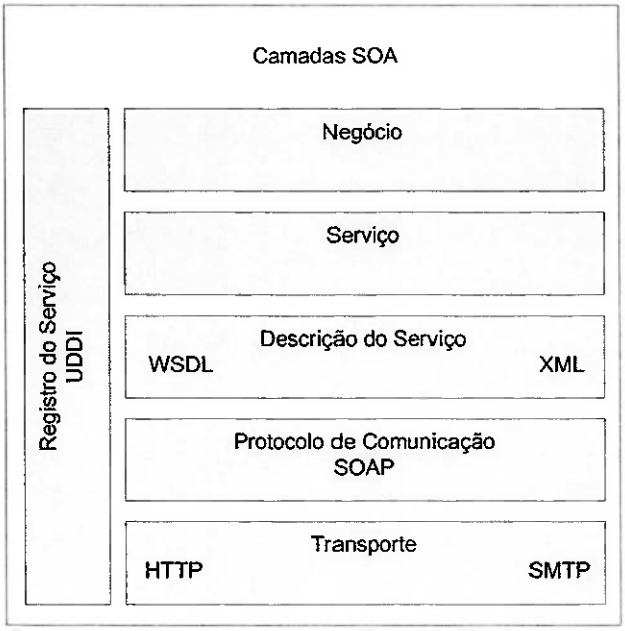

Figura 2.5 – Mapeamento entre componentes dos Web Services e camadas de SOA
Figura baseada em Endrei et al. (2004)

As camadas existentes na figura 2.5 são descritas a seguir (ENDREI et al, 2004).

- Camada de Transporte: esta camada agrupa os protocolos tais como HTTP (RFC 2616, 1999), ou SMTP (RFC 2821, 2001), utilizados em Web Services para transmissão de mensagens.
- Protocolo de Comunicação: esta camada abriga o protocolo SOAP, utilizado em Web Services para transmissão de mensagens.
- Descrição do Serviço: nesta camada deve ser inserido o padrão WSDL, que colabora na especificação dos atributos de Web Services.
- Serviço: nesta camada devem ser inseridos os códigos desenvolvidos para possibilitar o cumprimento dos serviços. Estes podem ser criados utilizando linguagens de desenvolvimento distintas. A localização e invocação dos serviços ocorrem com base nas informações disponíveis na camada de descrição de serviços.

- Registro do Serviço: nesta camada devem se inseridos os registros de serviços de Web Services baseados na especificação UDDI.
- Negócio: esta camada abriga documentos contendo políticas de utilização do sistema de software.

Este mapeamento possibilita o rastreamento entre os elementos da Arquitetura Orientada a Serviços e os componentes que devem ser reunidos para utilização dos Web Services, facilitando a análise da tecnologia mencionada quanto a adoção em sistemas de software (ENDREI et al, 2004).

2.3 CONSIDERAÇÕES FINAIS

Os conceitos e as características relacionadas a Arquitetura Orientada a Serviços, apresentados nesta parte, definem a arquitetura base para este trabalho. Os padrões citados facilitam o entendimento sobre a forma como ocorrem as interações entre as entidades relacionadas a SOA.

A tecnologia dos Web Services apresentada permite que seja analisada a maneira como podem sem implantados os sistemas de software baseados na Arquitetura Orientada a Serviços, considerando a proposta deste trabalho.

3. QUALIDADE DE SOFTWARE

3.1 CONSIDERAÇÕES INICIAIS

Esta seção tem como objetivo apresentar os critérios para seleção dos atributos de qualidade e as táticas adotadas para o desenvolvimento deste trabalho. De acordo com Matos (2005), faz-se necessário selecionar uma terminologia a ser utilizada quanto a qualidade de software. Entre os diferentes termos pode-se citar: características de qualidade, fatores de qualidade, requisitos não funcionais e atributos do sistema de software.

A terminologia adotada para qualidade de software tem como base as publicações de: Bass; Clements; Kazman, 2003; Dromey, 1995; Levinson; O'Brien, 2006; O´Brien; Bass; Merson, 2005, que são importantes para o trabalho, pois estes autores abordam a maneira como qualidade de software é tratada em relação a SOA. As publicações de: Levinson; O'Brien (2006) e O´Brien; Bass; Merson, 2005, estão diretamente relacionadas a Arquitetura Orientada a Serviços e por conseqüente, a proposta do trabalho.

3.2 ATRIBUTOS DE QUALIDADE SELECIONADOS

Os atributos de qualidade considerados neste trabalho são confiabilidade, segurança e desempenho (BASS; CLEMENTS; KAZMAN, 2003; DROMEY, 1995; LEVINSON; O'BRIEN, 2006; O´BRIEN; BASS; MERSON, 2005). Estes atributos são relevantes para o desenvolvimento de aplicações baseadas na Arquitetura Orientada a Serviços (SOA), devido aos seguintes aspectos.

- Confiabilidade: este é um atributo que se refere a capacidade de um software manter-se em funcionamento após um período de tempo. Este atributo é particularmente importante na Arquitetura Orientada a Serviços porque se refere as mensagens trocadas entre sistemas de software. As trocas de mensagens realizadas entre os serviços e aplicações cliente ou outros serviços que estão relacionados a SOA, podem apresentar falhas de entrega ou ocorrer casos onde a entrega de mensagens é feita mais de uma vez. Além disso, também podem apresentar problemas de provedores não confiáveis. Desta forma, é importante a utilização de técnicas para evitar problemas desta natureza (ENDREI et al, 2004; LEVINSON; O'BRIEN, 2006; O´BRIEN; BASS; MERSON, 2005).

  Sistemas de Software que atuam em diferentes ambientes podem ter distintas necessidades, entretanto, é necessário que as mensagens sejam entregues de maneira confiável (ENDREI et al, 2004; O´BRIEN; BASS; MERSON, 2005).

- Segurança: este atributo se refere a capacidade que o software deve possuir de evitar o acesso não autorizado, acidental ou deliberado, aos serviços e aos dados para pessoas ou aplicações. Esta capacidade é considerada relevante na Arquitetura Orientada a Serviços, devido ao aumento da preocupação em garantir que a troca de mensagens entre as entidades não seja facilmente interceptada e alterada. Além disso, o acesso aos serviços e dados deve ser permitido somente para entidades devidamente credenciadas, possibilitando uma relação de confiança entre as entidades envolvidas (BASS; CLEMENTS; KAZMAN, 2003; LEVINSON; O'BRIEN, 2006; O´BRIEN; BASS; MERSON, 2005).

  Desta forma, nos serviços disponibilizados em servidores e mensagens que são trocadas entre aplicações utilizando a Internet, este atributo deve ser analisado devido

aos aspectos de segurança (BASS; CLEMENTS; KAZMAN, 2003; DROMEY, 1995; O´BRIEN; BASS; MERSON, 2005).

- Desempenho: este atributo é uma constante preocupação no desenvolvimento de sistemas de software, devido a relação estímulo e tempo de resposta entre aplicações. Em SOA, geralmente é afetado negativamente com a perda de desempenho que pode estar ligada a fatores tais como, tempo de resposta das requisições afetado devido a demora na comunicação, transferência de mensagens em formatos que culminam em tamanho exagerado, uso de conexões sincronizadas que impedem novas requisições, ou ainda o número alto de requisições de serviços recebidos ao mesmo tempo. Com a finalidade de minimizar problemas desta natureza, cada sistema de software em SOA deve ser cuidadosamente projetado e avaliado (BASS; CLEMENTS; KAZMAN, 2003; DROMEY, 1995; LEVINSON; O'BRIEN, 2006; O´BRIEN; BASS; MERSON, 2005; MATOS, 2005).

3.3 TÁTICAS APLICÁVEIS AOS ATRIBUTOS DE QUALIDADE

Esta seção apresenta as táticas com o objetivo de auxiliar o desenvolvimento de sistemas de software baseados na Arquitetura Orientada a Serviços. De acordo com Bass, Clements e Kazman (2003), uma tática é considerada como decisão de projeto, podendo influenciar no controle do atendimento aos atributos de qualidade.

A seguir são apresentadas e agrupadas as táticas de acordo com o atributo de qualidade relacionado.

3.3.1 Confiabilidade

Este atributo de qualidade possui importante papel nos sistemas de software que utilizam a Arquitetura Orientada a Serviços, possibilitando troca de mensagens de maneira confiável e evitando que os provedores apresentem problemas de confiança quanto aos serviços disponibilizados (O'BRIEN; BASS; MERSON, 2005).

Com o objetivo de minimizar a ocorrência de problemas desta origem nos sistemas de software baseados em SOA, são apresentadas as táticas:

- **Provisão de servidores com redundância:** Esta tática consiste em aumentar a confiabilidade relacionada com a hospedagem e os servidores que provêem os serviços em SOA. Isto é obtido através da redundância de servidores, que aumenta a capacidade para processamento de serviços e dados. Também é recomendada a redundância em conexões via rede, para evitar que os serviços fiquem indisponíveis. Adicionalmente, é recomendada a divisão de carga entre servidores para aumentar a tolerância a falhas e adoção de clusters para evitar a existência de um único ponto para falhas (ERRADI; MAHESHWARI, 2005).

- **Utilização do Protocolo HTTPR:** Esta tática consiste na utilização do protocolo HTTPR, que se refere a um protocolo para o transporte confiável de mensagens entre aplicações sobre a Internet. O HTTPR é constituído de uma camada sobre o protocolo HTTP (RFC 2616, 1999) que define especificamente como os metadados e as mensagens enviadas pelos serviços são encapsuladas dentro das requisições e respostas do protocolo HTTP (RFC 2616, 1999). O protocolo HTTPR também possui regras para assegurar que cada mensagem é entregue exatamente para a aplicação de destino, caso contrário, é enviada uma mensagem comunicando a impossibilidade de entrega (BANKS et al, 2002; TODD et al, 2001).

Uma interação utilizando HTTPR é iniciada quando a aplicação cliente envia para o servidor uma requisição usando um comando POST que contém argumentos para sua identificação. Se o servidor validar a aplicação cliente, este envia resposta contendo a informação de status e caso tenha sido solicitado, um conjunto de mensagens é endereçado à aplicação cliente (BANKS et al, 2002; TODD et al, 2001).

- **Adoção da especificação WS-ReliableMessaging:** Esta tática consiste na adoção da especificação WS-ReliableMessaging (OASIS WS-RM, 2006), que tem por objetivo a criação de um mecanismo para transferência de mensagens de maneira confiável.

  Este mecanismo é definido como um protocolo encarregado de identificar, rastrear e gerenciar a entrega de mensagens de maneira confiável entre dois pontos, denominados na especificação de fonte e destino. Estes mecanismos garantem que as mensagens sejam entregues de maneira confiável entre os dois pontos estabelecidos, entretanto, estes mecanismos precisam ser desenvolvidos utilizando a tecnologia de Web Services, o que restringe a adoção desta tática na implantação de sistemas de software baseados em SOA (ENDREI et al, 2004; COHEN, 2006).

Essas táticas podem se empregadas em conjunto ou individualmente nos sistemas de software. A adoção de cada uma delas deve ser avaliada de acordo com as necessidades de cada aplicação, podendo ser usada em toda a aplicação ou somente em parte desta.

3.3.2 Segurança

Este atributo de qualidade possui grande importância em sistemas de software que utilizem a Arquitetura Orientada a Serviços, possibilitando maior segurança nos serviços

e transações (BASS; CLEMENTS; KAZMAN, 2003; O'BRIEN; BASS; MERSON, 2005). Com o objetivo de minimizar a ocorrência de problemas desta origem nos sistemas de software baseados em SOA, são apresentadas as táticas:

- **Autenticação:** Esta tática consiste no uso do processo de autenticação, que se caracteriza por verificar se a entidade (aplicação cliente ou outro serviço) que tenta acessar determinado serviço possui as credencias necessárias para esta operação. Estas credenciais podem ser compostas por uma senha e algum outro código identificador.

  Além da autenticação feita somente por uma aplicação cliente ao serviço, a autenticação mútua, onde a aplicação cliente precisa autenticar-se junto ao serviço e este também precisa autenticar-se junto a aplicação cliente, pode ser uma medida importante, devido as entidades (aplicação cliente e serviço) não permanecerem necessariamente conectadas todo o tempo (BASS; CLEMENTS; KAZMAN, 2003; MICROSOFT, 2005; W3C WSA 2004;). Segundo Microsoft (2005), a autenticação entre entidades pode ser agrupada em dois padrões distintos, sendo estes:

  - Padrão de Autenticação Direta: este padrão implica no serviço prestado atuar diretamente na autenticação da aplicação cliente ou outro serviço, para validar as permissões dadas a este e permitir ou não que determinado serviço possa ser utilizado.
  - Padrão de Autenticação Através de Broker: a utilização deste padrão possibilita a aplicação cliente ou outro serviço autenticar-se uma vez e utilizar os serviços disponíveis ao seu perfil, sem a necessidade de se autenticar em cada serviço a ser usado. Neste caso, a autenticação é realizada na entidade Broker, e não diretamente entre a aplicação cliente e o serviço a ser utilizado.

Além dos padrões de autenticação mencionados acima, também pode ser utilizado nesta tática, o *framework* baseado em SAML (Security Assertion Markup Language), que atua nas requisições/respostas para a autenticação. Seu uso é recomendado em aplicações que não possuam uma infra-estrutura compatível (NAEDELE, 2003; YU, 2006).

- **Autorização:** Esta tática implica em controlar o nível de acesso de aplicações cliente ou outros serviços de maneira a definir as permissões para utilização dos serviços pretendidos. Estes direitos de acesso podem ser atribuídos para entidades distintas ou para classes de entidades. Estas classes de entidades podem ser definidas por grupos, por regras ou ainda através de lista individual das entidades (BASS; CLEMENTS; KAZMAN, 2003; KARP, 2006; YU, 2006). Em Nakamur (2002), são citadas duas propostas para desenvolver o controle de acesso aos recursos. Estas são:

  - Lista de Controle de Acesso (ACL – Acess Control List): este modelo define um mapeamento entre as entidades caracterizadas por aplicações clientes e os recursos que podem ser devidamente acessados.
  - Controle de Acesso Baseado em Regras (RBAC – Role-Based Access Control): neste modelo é realizado um mapeamento entre a aplicação cliente e as regras definidas. Também é feito um mapeamento entre as regras estabelecidas e os recursos disponíveis.

- **Sigilo de dados:** Esta tática está relacionada com a proteção de dados contra acessos não autorizados. O sigilo dos dados pode ser obtido utilizando o processo de criptografia, que possibilita uma proteção extra para manter a persistência dos dados transmitidos através de mensagens e impedir o acesso de aplicações não autorizadas. Este processo de criptografia pode ser aplicado em duas etapas, são elas:

- Criptografia de mensagens: nesta etapa, a entidade remetente realiza a conversão da mensagem no formato original, seja texto ou XML (W3C XML, 2004), para o formato criptografado, tornando-a ilegível para entidades que não seja a de destino.
- Descriptografia de mensagens: nesta etapa, o processo realizado é o inverso, onde a mensagem é novamente convertida para o seu formato original.

O processo de criptografia pode ser realizado de maneira simétrica, onde a entidade remetente e a entidade destinatário dividem uma única chave que é utilizada para criptografar e descriptografar os dados. Outra opção de criptografia caracteriza-se pela realização deste processo de maneira assimétrica, também conhecida como chave pública de criptografia. Nesta opção, é utilizada determinada chave para criptografar as mensagens, e para descriptografar, é utilizada outra chave distinta (MICROSOFT, 2005; OASIS WSS, 2006).

Neste processo de criptografia, estão envolvidas as entidades (BASS; CLEMENTS; KAZMAN, 2003; MICROSOFT, 2005):

- Entidade remetente: a entidade (pode ser a aplicação cliente ou o serviço) assume esta característica no momento que envia uma requisição ou uma resposta para outra entidade (entidade destinatário).
- Entidade destinatário: a entidade (pode ser a aplicação cliente ou o serviço) assume esta característica no momento que recebe uma mensagem de requisição ou uma resposta para outra entidade (entidade remetente).

- **Assinatura digital:** Esta tática consiste em enviar uma assinatura digital, ou seja, uma chave criptografada com algoritmo específico, juntamente com a mensagem, para que a entidade destino (serviço ou ainda a aplicação cliente) possa verificar se a

mensagem sofreu algum tipo de alteração desde que foi emitida. Além disto, a assinatura digital também permite que seja feita uma autenticação da entidade remetente através da chave enviada. Desta forma, é possível aumentar a segurança quando nos referimos a melhoria da integridade dos dados e entidades relacionadas (BASS et al, 2003; ENDREI et al, 2004; MICROSOFT, 2005; OASIS WSS, 2006).

A utilização de assinatura digital pode ser simétrica, que corresponde ao uso de uma única chave para criptografar e descriptografar as mensagens, ou ainda a assinatura pode ser assimétrica. Neste caso é feito o uso de uma chave na criação da assinatura e outra em sua verificação (BASS et al, 2003; ENDREI et al, 2004; MICROSOFT, 2005; OASIS WSS, 2006).

- **Limite de Acesso:** Esta tática consiste em limitar o acesso de sistemas externos aos servidores que hospedam os serviços de um sistema de software baseado na Arquitetura Orientada a Serviços (SOA). Seu objetivo é conter ataques oriundos de fontes desconhecidas. Para limitar o acesso, podem ser utilizados firewalls. Estes se baseiam nos conteúdos das mensagens recebidas ou na porta de destino destas, impedindo o acesso não autorizado. O *firewall* pode ser configurado de acordo com a tecnologia utilizada para implantar o sistema de software em SOA.

  Além das restrições proporcionadas pelo uso de firewall, o acesso através da Internet é regularmente disponibilizado para determinados serviços, não devendo estender-se até a rede interna, onde estão localizados os servidores que hospedam os serviços ou outros sistemas de software. Este bloqueio é realizado através da criação e configuração de uma área de acesso desmilitarizada, conhecida como DMZ (Demilitarized Zone) (ALLEN, 2001; BASS; CLEMENTS; KAZMAN, 2003; O'BRIEN; BASS; MERSON, 2005).

- **Utilização do Protocolo SSL:** Esta tática consiste na utilização do protocolo SSL, que pode ser adotado para a transmissão de dados de maneira segura, utilizando métodos de criptografia de dados. O protocolo HTTPS consiste na utilização do protocolo HTTP (RFC 2616, 1999), onde é adicionada a camada SSL. Para estabelecer um canal de comunicação seguro, é necessário que as aplicações cliente e servidor realizem autenticação mútua para transmissão de mensagens. As aplicações baseadas em Web Services podem utilizar SSL na parte de infra-estrutura (NAKAMUR, 2002; O'BRIEN; BASS; MERSON, 2005).

Essas táticas podem se empregadas em conjunto ou individualmente nos sistemas de software. Sua adoção deve ser avaliada de acordo com as necessidades de cada aplicação, podendo ser usadas em toda a aplicação ou distribuídas dentro desta. Isto porque determinadas táticas podem se tornar redundantes, se aplicadas no mesmo ponto do sistema de software (BASS et al., 2003).

3.3.3 Desempenho

Este atributo possui importante papel em sistemas de software que utilizem a Arquitetura Orientada a Serviços, possibilitando troca de mensagens sem perda de desempenho e também evitando que aplicações apresentem problemas de desempenho devido ao alto número de requisições de serviços recebidas ao mesmo tempo. Com o objetivo de minimizar a ocorrência de problemas desta origem nos sistemas de software baseados em SOA, são apresentadas as táticas:

- **Utilização de Cache:** Nesta tática, os dados são replicados para diretórios distintos, de forma a reduzir a concorrência, cabendo ao sistema assumir a responsabilidade de manter os dados devidamente sincronizados. Seu uso reduz o tempo médio para

atendimento de requisições (BASS et al., 2003; MEIER et al., 2004; FERNANDEZ et al., 2005).

Esta tática pode ser utilizada em sistemas de software baseadas na Arquitetura Orientada a Serviços, entretanto, os seguintes fatos devem ser considerados para que a utilização de cache se torne viável (BASS et al., 2003; MEIER et al., 2004; FERNANDEZ et al., 2005):

- Deve existir um número grande o suficiente de requisições, de forma a justificar sua utilização.
- Os retornos emitidos por serviços que estejam utilizando cache devem possuir o mesmo conteúdo de retornos emitidos por serviços que não estejam utilizando cache.

Em Azim e Hamid (2002) esta tática é tratada de maneira diferenciada, onde é proposta a utilização de cache no lado da aplicação cliente com o objetivo de melhorar o desempenho pela redução de requisições aos serviços e conseqüente redução do tráfego em rede. Contudo, é necessário atentar-se a inconsistência que pode ocorrer entre os dados contidos em cache local e o serviço que fornece dados atualizados. A solução proposta para este problema de inconsistência, está relacionada a sincronização entre as entidades, de forma a as atualizações de dados sejam enviadas para a aplicação cliente.

- **Atribuição de Prioridade:** Esta tática consiste em definir e documentar uma política que trata a priorização ao utilizar os recursos da aplicação. Quatro formas para atribuição de prioridades são (BASS; CLEMENTS; KAZMAN, 2003):

  - Escalonamento através de fila (First-in/First-ou): FIFO é uma maneira de atender igualmente todas as requisições feitas aos serviços na ordem de recebimento. Sua

utilização pode acarretar em uma requisição simples e rápida, ter de aguardar o processamento de outra que demande maior tempo. Não havendo necessidade de priorizar o atendimento a determinadas requisições, a utilização desta maneira não se torna problemática.

- Escalonamento fixo de prioridade (Fixed-priority scheduling): esta maneira atribui a cada fonte requisitante ou serviço solicitado, um grau definido de prioridade e suas requisições são processadas obedecendo a esta ordem. Esta estratégia assegura melhor atendimento a requisições de alta prioridade, entretanto, admite a possibilidade de que uma requisição de baixa prioridade, mas importante, tenha de aguardar por um longo período de tempo até que as requisições de alta prioridade sejam atendidas.

- Escalonamento dinâmico de prioridade (Dynamic priority scheduling): esta maneira de definir prioridade consiste em atribuir dinamicamente a ordem de prioridade para o processamento de determinada requisição. Esta ordenação pode ser feita de acordo com o menor tempo definido para execução.

- Escalonamento de interrupções por tempo (Static scheduling): esta maneira possibilita que a execução de cada requisição seja realizada dentro de um período fixo de tempo.

A adoção de meios para atribuição de prioridades pode reduzir perdas de desempenho, entretanto, estes meios devem ser selecionados observando critérios como tipo de recurso utilizado, importância da requisição e número de recursos a serem utilizados. Desta forma, sua adoção implica na necessidade da uma análise de viabilidade (BASS; CLEMENTS; KAZMAN, 2003).

- **Transferência de grandes volumes de dados:** Esta tática pode ser aplicada para a melhoria de desempenho no processo de transferência de grandes volumes de dados. Ela compreende as seguintes técnicas (MEIER, 2004; W3C SOAP a, 2000; O'BRIEN; BASS; MERSON, 2005):

  - Envio de anexos: o protocolo SOAP 1.1 (W3C SOAP, 2002) permite anexar referência para dados em formatos binários (imagens, arquivos compactados em formato zip) no corpo da mensagem enviada.
  - Retorno de endereço para *download*: esta técnica pode ser usada para documentos ou arquivos maiores, e consiste no serviço retornar um endereço (URL) para que a aplicação cliente possa obter os arquivos realizando downloads destes.

- **Serialização de mensagens:** Esta tática pode ser aplicada em sistemas de software que utilizam XML para a transmissão de mensagens através da Internet, devido ao tamanho das mensagens no formato XML, que pode ser entre 10 e 20 vezes maior, se comparada a equivalente representação em binário. Desta forma, a mensagem XML pode, através do processo de serialização, ser transformada em código binário para realização da transferência entre o serviço e a aplicação cliente e, em seguida, novamente ser transformada em mensagem XML. Este processo é formado basicamente por três atividades (IBM a, 2005; O'BRIEN; BASS; MERSON, 2005):

  - Tradução: nesta atividade, a mensagem XML é traduzida de forma que possa ser interpretada pela aplicação cliente.
  - Validação: esta atividade é utilizada para garantir que determinada mensagem XML atende a estrutura pré-definida.
  - Transformação: esta atividade é definida como a transformação de uma estrutura XML para outra ou de um XML para outro formato.

- **Utilização de padrões Síncrono e Assíncrono:** Esta tática implica na utilização do padrão síncrono ou assíncrono. No síncrono, uma aplicação cliente utiliza determinado serviço através de uma transação sincronizada, onde é enviada uma requisição e a transação é finalizada somente após o processamento e retorno do serviço. No padrão assíncrono, determinada aplicação cliente faz uma requisição para um serviço distinto, este serviço recebe a requisição e finaliza a transação sem ter de permanecer conectado enquanto processa e responde para a aplicação cliente (MICROSOFT, 2005).

  Pode ser observada no padrão síncrono a ocorrência de problemas de desempenho, devido a conexão permanecer bloqueada até que o serviço envie a mensagem de retorno. Para reduzir esta perda, caso a aplicação cliente precise confirmar o recebimento da requisição, é possível utilizar um padrão Polling RM-Reply (Cohen, 2006) ou Callback RM-Reply (Cohen, 2006). Segundo estes padrões, em uma transação sincronizada e com requisição reconhecida, a aplicação cliente envia uma requisição e recebe uma resposta imediata. Esta resposta é uma mensagem de recebimento da requisição. Desta maneira, o serviço permanece bloqueado por um curto período de tempo (COHEN, 2006).

  Outra proposta para reduzir a perda de desempenho, consiste na utilização de padrões assíncronos, possibilitando a execução de outras tarefas em paralelo antes do serviço ter de responder a requisição feita (COHEN, 2006; MEIER, 2004).

Essas táticas podem se empregadas em conjunto ou individualmente nos sistemas de software. Sua adoção deve ser avaliada de acordo com as necessidades de cada aplicação, podendo ser usada em todo o sistema de software ou somente em parte deste.

3.4 CONSIDERAÇÕES FINAIS

Nesta seção, é apresentada a terminologia usada para qualidade de software, os atributos de qualidade selecionados e as táticas relacionadas. Estas táticas são utilizadas para atender a proposta deste trabalho, facilitando que confiabilidade, segurança e desempenho sejam atendidos no desenvolvimento dos sistemas de software baseados na Arquitetura Orientada a Serviços.

4. IMPACTO DOS ATRIBUTOS DE QUALIDADE SOBRE A ARQUITETURA ORIENTADA A SERVIÇOS

4.1 CONSIDERAÇÕES INICIAIS

Ao analisar as publicações ligadas a Arquitetura Orientada a Serviços e Web Services, é possível obter um conjunto de necessidades relacionadas aos atributos de qualidade (BASS; CLEMENTS; KAZMAN, 2003; COHEN, 2006; ENDREI et al, 2004; ERRADI; MAHESHWARI, 2005; MEIER et al, 2004; MICROSOFT, 2005; NAEDELE, 2003; O'BRIEN; BASS; MERSON, 2005; TODD et al, 2001).

Este conjunto de necessidades é analisado sob o foco dos atributos de confiabilidade, segurança e desempenho. Em seguida, é criada uma lista das necessidades e a partir desta, as táticas mostradas na seção 3 são adotadas e endereçadas ao atendimento das necessidades, através de tabela para mostrar seu relacionamento.

Com o objetivo de facilitar o rastreamento entre os elementos existentes em entidades relacionadas a SOA e as táticas discutidas nesta seção, é realizado um mapeamento entre estas táticas e a arquitetura mostrada na seção 2.2.

4.2 NECESSIDADES RELACIONADAS À ARQUITETURA ORIENTADA A SERVIÇOS

Esta seção tem por objetivo apresentar as necessidades da Arquitetura Orientada a Serviços que estão relacionadas aos atributos de confiabilidade, segurança e desempenho, discutidos neste trabalho. Estas necessidades são (BASS; CLEMENTS; KAZMAN,

2003; COHEN, 2006; ENDREI et al, 2004; ERRADI; MAHESHWARI, 2005; MEIER et al, 2004; MICROSOFT, 2005; NAEDELE, 2003; O'BRIEN; BASS; MERSON, 2005; TODD et al, 2001):

- Confiabilidade de serviços: Os serviços devem ser disponibilizados de maneira confiável, reduzindo falhas ocorridas nas tentativas de acesso devido a insuficiência de infra-estrutura para atender as requisições feitas.

- Entrega confiável de mensagens: Mensagens devem ser trocadas entre as entidades (aplicação cliente / serviço) de maneira confiável, reduzindo as falhas na entrega, casos de mensagens duplicadas ou ainda casos de entrega em seqüência errada.

- Segurança na troca de mensagens: As mensagens devem ser trocadas entre as entidades (aplicação cliente / serviço) de maneira segura, reduzindo o risco das mensagens serem interceptadas, roubadas ou ter seu conteúdo alterado.

- Autenticação de entidades (aplicação cliente / serviço): A autenticação de entidades (aplicação cliente / serviço) é importante para evitar que programas maliciosos utilizem serviços sem possuir as credenciais para tal.

- Autorização de acesso: os serviços disponíveis devem ser acessados somente por entidades (aplicações cliente / serviço) que possuam as devidas credenciais.

- Acesso limitado aos serviços: Os provedores que disponibilizem serviços para utilização através da Internet devem possuir o acesso limitado aos serviços, impedindo que programas maliciosos acessem outras partes dos servidores de hospedagem.

- Alto volume de requisições: A arquitetura deve possuir meios para atender ao volume elevado de requisições recebidas por serviços.

- Serviços prioritários: Os sistemas de software que possuem serviços com diferentes níveis de prioridade, devem dispor de meios para atender as requisições de maneira diferenciada, para minimizar problemas com perda de desempenho.

- Transferência de grandes volumes de dados: Os sistemas de software devem apresentar meios para evitar perda de desempenho durante a transferência de mensagens contendo grandes volumes de dados.

- Desempenho na transferência de mensagens: O sistema de software deve dispor de meios para reduzir as perdas de desempenho durante a transmissão de mensagens.

A tabela 4.1 organiza as necessidades da Arquitetura Orientada a Serviços de acordo com os atributos de qualidade selecionados. As táticas mostradas na seção 3.4 também são direcionadas ao atendimento das necessidades de SOA.

Atributos de Qualidade	Necessidades	Táticas Aplicadas
Confiabilidade	Confiabilidade de serviços	- Provisão de servidores com redundância.
	Entrega confiável de mensagens	- Utilização do Protocolo HTTPR. - Adoção da especificação WS-ReliableMessaging.
Segurança	Segurança nas trocas de mensagens	- Sigilo de dados - Assinatura digital - Utilização do protocolo SSL
	Autenticação de entidades (aplicação cliente / serviço)	- Autenticação
	Autorização de acesso	- Autorização
	Acesso limitado aos serviços	- Limite de acesso
Desempenho	Alto volume de requisições	- Utilização de Cachê - Serialização de mensagens - Utilização de padrões Síncrono e Assíncrono
	Serviços prioritários	- Atribuição de prioridade
	Transferência de grandes volumes de dados	- Transferência de grandes volumes de dados - Utilização de padrões Síncrono e Assíncrono
	Desempenho na transferência de mensagens	- Serialização de mensagens - Utilização de padrões Síncrono e Assíncrono

Tabela 4.1 – Mapeamento entre necessidades de software e táticas

Nesta seção, são mostradas as necessidades existentes nos sistemas de software baseados em SOA e para atender a cada necessidade, é citado um conjunto de táticas. Após este

mapeamento entre as necessidades e as táticas, a forma como estas podem colaborar com SOA é discutida na seção seguinte.

4.3 RELAÇÃO ENTRE OS ATRIBUTOS DE QUALIDADE E AS NECESSIDADES DA ARQUITETURA ORIENTADA A SERVIÇOS

Esta seção tem por objetivo analisar as táticas mostradas na tabela 4.1, quanto a sua utilização para suprir as necessidades da Arquitetura Orientada a Serviços, listadas na seção 4.2. Estas táticas são agrupadas de acordo com os atributos relacionados.

4.3.1 Arquitetura Orientada a Serviços e atributos confiabilidade

A seguir, as táticas endereçadas ao atributo de confiabilidade, são analisadas quanto ao seu uso para atender as necessidades relacionadas a este atributo. De forma a colaborar com o mapeamento proposto neste trabalho:

- **Provisão de servidores com redundância:** Esta tática pode colaborar com sistemas de software em SOA, permitindo o aumento na disponibilidade dos serviços através de redundância de servidores. Nestes casos, é recomendado a mesma configuração de hardware e rede para todos os servidores. Desta forma, condições onde podem ocorrer falhas na disponibilidade de serviços, são contornadas com maior facilidade. Esta redundância de servidores pode contribuir diretamente para a melhoria de desempenho, isto porque o atendimento as requisições pode ser realizado com a utilização de um número maior de provedores de serviços.

A figura 4.1 mostra uma proposta básica relacionada com servidores de aplicações para um sistema de software em SOA. Nesta figura, o servidor utilizado para prover redundância possui as mesmas configurações de hardware e sistema operacional existentes no servidor de produção. O balanceamento de carga entre os servidores de aplicações é representado por um servidor específico, responsável por receber as requisições das aplicações através da Internet e encaminhar aos serviços de forma distinta (ERRADI; MAHESHWARI, 2005).

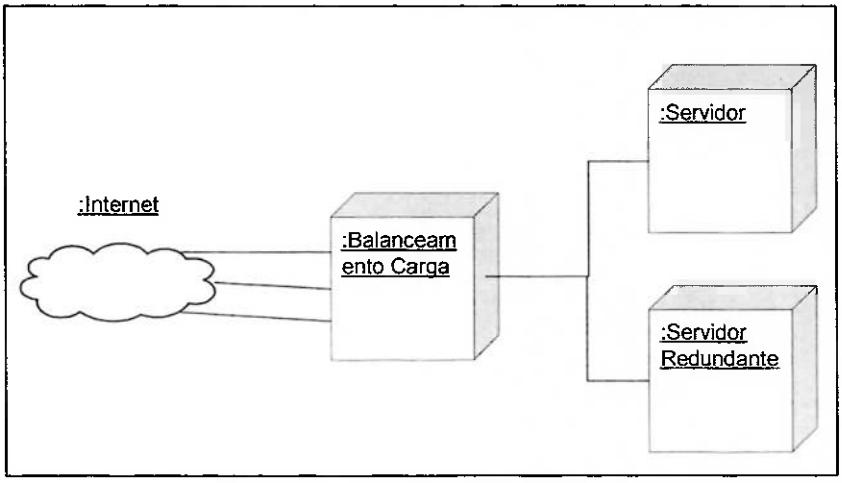

Figura 4.1 – Redundância de servidores

- **Utilização do Protocolo HTTPR:** O transporte de mensagens entre sistemas de software em SOA pode ser realizado com a utilização do protocolo HTTPR, mostrado na seção 3.3.1 e citado na tabela 4.1. A utilização deste protocolo em SOA, permite um controle sobre as mensagens enviadas, garantindo um retorno sobre o envio realizado, informando se houve a entrega da mensagem requisitada ou a falha na entrega. Este protocolo também pode ser utilizado para troca de mensagens de maneira sincronizada ou não. A figura 4.2 ilustra o fluxo de uma mensagem entre a aplicação cliente e o serviço selecionado, utilizando o protocolo HTTPR de maneira sincronizada. Este fluxo consiste em (BANKS et al, 2002; TODD et al, 2001):

1. A aplicação cliente encaminha uma requisição para a camada de suporte a comunicação.
2. A camada de suporte a comunicação da aplicação cliente, recebe a requisição e antes de enviar para a camada de suporte a comunicação do serviço, é feita a persistência da requisição.
3. A camada de suporte a comunicação da aplicação cliente, encaminha a requisição para a camada de suporte a comunicação do serviço.
4. Da mesma maneira, antes da camada de suporte a comunicação do serviço encaminhar a requisição, é realizada a persistência da requisição.
5. Em seguida a requisição é encaminhada ao servidor para o processamento pelo serviço.
6. Após o processamento, o servidor encaminha a resposta para a camada de suporte a comunicação do serviço.
7. A camada de suporte a comunicação do serviço realiza a persistência deste retorno.
8. Em seguida envia para a camada de suporte a comunicação da aplicação cliente.
9. Esta por sua vez, realiza a persistência da mensagem recebida.
10. Em seguida encaminha para a aplicação cliente.

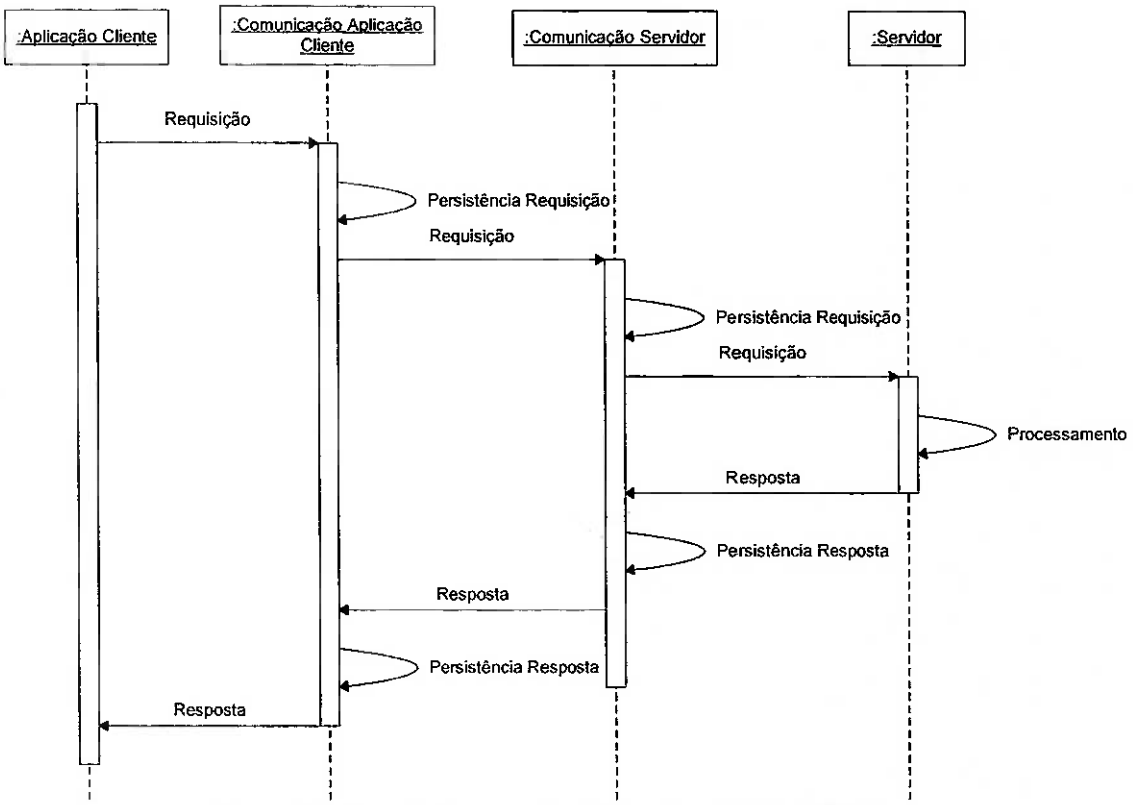

Figura 4.2 – Requisição de serviço utilizando protocolo HTTPR
Baseado em Todd et al, 2001)

- **Adoção da especificação WS-ReliableMessaging (WS-RM):** Esta tática possibilita a criação de um protocolo de comunicação para troca de mensagens utilizando Web Services, que também se refere a uma tecnologia baseada em SOA. Sua utilização também facilita a interoperabilidade entre sistemas de software desenvolvidos sobre diferentes infra-estruturas (COHEN, 2006; ENDREI et al, 2004; OASIS WS-RM, 2006).

  A utilização deste protocolo durante a comunicação entre a aplicação cliente e o serviço, inclui a criação de uma chave identificadora, usada no envio das mensagens. Ao final do envio, a entidade (aplicação cliente ou serviço) que está recebendo a mensagem retorna para a entidade emitente a relação de mensagens recebidas. Desta

forma, é possível verificar se as mensagens foram devidamente encaminhadas. Em caso de erros na entrega de mensagens, é possível que estas sejam novamente encaminhadas (OASIS WS-RM, 2006).

4.3.2 Arquitetura Orientada a Serviços e atributos segurança

A seguir, as táticas endereçadas ao atributo de segurança, são analisadas quanto ao seu uso para atender as necessidades relacionadas a este atributo. De forma a colaborar com o mapeamento proposto neste trabalho:

- **Autenticação:** Na Arquitetura Orientada a Serviços, a autenticação de aplicações que desejam utilizar os serviços disponíveis é uma medida importante para evitar que programas maliciosos utilizem serviços sem possuir as credenciais para tal. Na seção 3.4.2 foram mostrados dois padrões distintos de autenticação, a direta e utilizando Broker. Nesta parte, é analisada a autenticação realizada com a utilização de um Broker devido a sua característica de possibilitar que a aplicação cliente utilize os serviços sem a necessidade de uma nova autenticação a cada requisição.

  A adoção deste padrão implica na criação de uma entidade (Broker) responsável por realizar a autenticação das aplicações. A figura 4.3, mostrada a seguir ilustra o processo realizado. Este processo consiste em (BASS; CLEMENTS; KAZMAN, 2003; MICROSOFT, 2005):

  1. A aplicação cliente remete uma requisição de autenticação ao broker, passando as credenciais (ID e senha).
  2. O broker encaminha as credenciais recebidas para a autenticação, através de requisição para validação junto aos registros de clientes.

3. Após a validação, o broker retorna para a aplicação cliente um código de segurança.
4. A aplicação cliente envia uma requisição para o serviço juntamente com o código de segurança recebido.
5. O serviço valida a requisição através do código de segurança recebido juntamente com a mensagem.
6. O serviço retorna para a aplicação cliente a resposta de sua requisição.

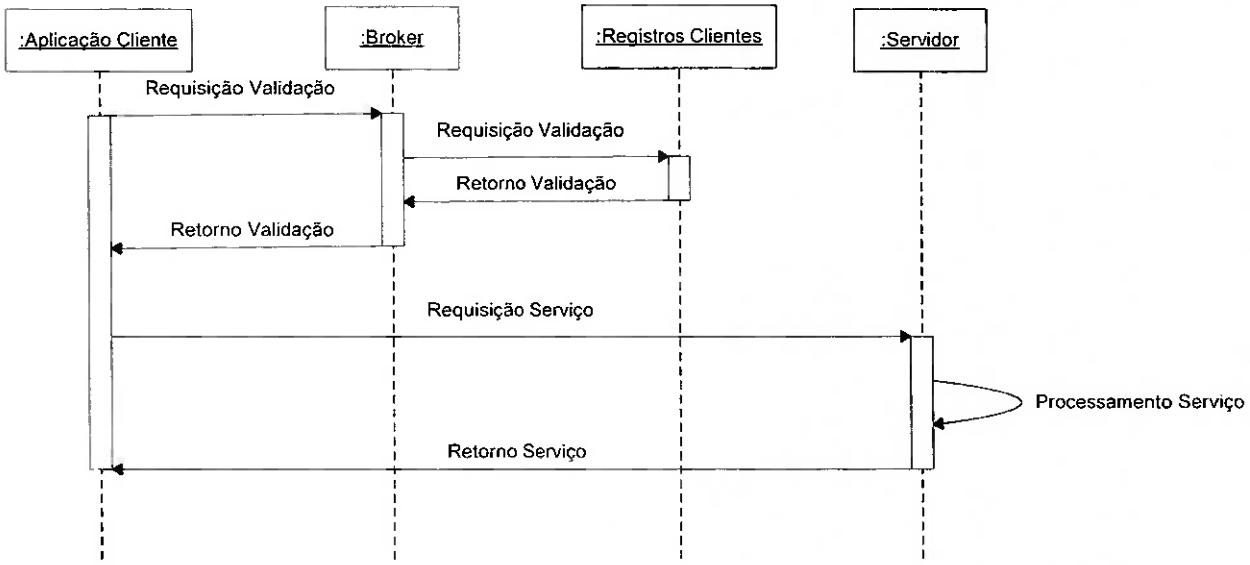

Figura 4.3 – Autenticação de aplicação

- **Autorização:** Esta tática pode ser empregada nos sistemas de software em SOA com o objetivo de garantir que os serviços sejam acessados somente por aplicações que possuam as credenciais para tal. Entre os modelos apresentados na seção 3.4.2 para o controle de acesso a recursos, deve ser considerado o modelo onde o controle de acesso está baseado em regras. Devido a opção da autenticação selecionada no sub-item acima, onde é utilizado um Broker, torna-se viável a adição deste controle (BASS; CLEMENTS; KAZMAN, 2003; NAKAMUR, 2002). Neste caso, a

verificação para autorizar acesso aos serviços, é realizada juntamente com a autenticação.

- **Sigilo de dados:** Como visto anteriormente, as mensagens transferidas entre aplicações cliente e serviços de um sistema de software baseado em SOA, podem ser interceptadas durante a sua transmissão por programas maliciosos. Para minimizar este risco, é proposto o uso da criptografia no caso de mensagens que utilizam XML (W3C XML, 2004) e também SOAP. Conforme apresentado na seção 3.4.2, a criptografia pode ser realizada com a utilização de chave simétrica ou assimétrica. Nesta parte, é adotado o uso de chave simétrica, devido ao tempo necessário para processamento ser menor, acarretando em ganho de desempenho (BASS; CLEMENTS; KAZMAN, 2003; MICROSOFT, 2005; OASIS WSS, 2006). A figura 4.4 mostrada a seguir ilustra o processo realizado. Este processo consiste em:

  1. O emissor (aplicação cliente/serviço) processa a mensagem, realiza a criptografia de seu conteúdo e encaminha a mensagem.
  2. O receptor (aplicação cliente/serviço) realiza o processo de descriptografia da mensagem utilizando chave simétrica e em seguida processa a mensagem.

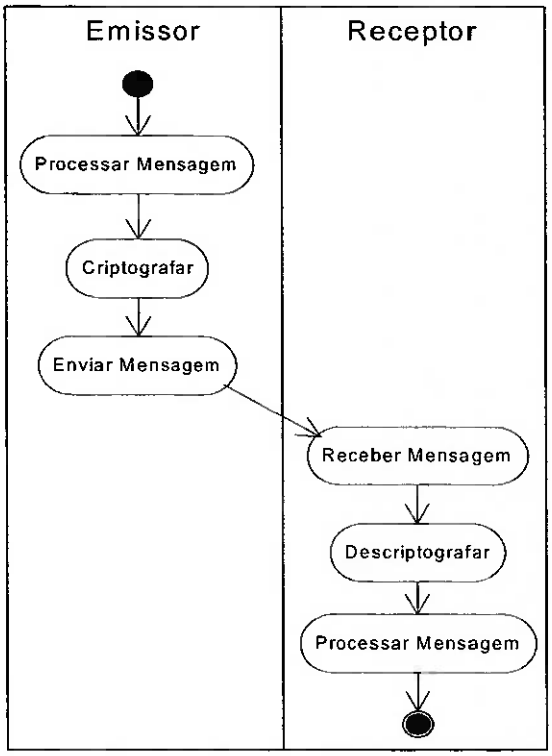

Figura 4.4 – Criptografia e descriptografia de mensagens
Baseado em: Microsoft (2005)

- **Assinatura digital:** A utilização da assinatura digital no desenvolvimento de sistemas de software baseados em SOA possibilita a verificação do conteúdo das mensagens trocadas, detectando possíveis alterações durante o seu envio. A assinatura digital também pode atuar como chave para autenticação da aplicação cliente junto ao serviço.

  As requisições ou respostas encaminhadas devem conter uma chave criptografada a partir de um algoritmo específico para criptografar/descriptografar. Esta chave é descriptografada na entidade de destino (aplicação cliente ou serviço) utilizando também um algoritmo específico para criptografar/descriptografar. Conforme citado na seção 3.4.2, este algoritmo pode ser simétrico ou assimétrico. Nesta parte, é adotado o uso de algoritmo simétrico, devido ao tempo necessário para

processamento ser menor, acarretando em ganho de desempenho (BASS; CLEMENTS; KAZMAN, 2003; ENDREI et al, 2004; MICROSOFT, 2005; OASIS WSS, 2006).

A figura 4.5 mostrada a seguir ilustra o processo realizado. Este processo consiste em:

1. O emissor (aplicação cliente/serviço) processa a mensagem, gera a chave criptografada e em seguida encaminha para o receptor.
2. O receptor (aplicação cliente/serviço) realiza o processo de descriptografia da chave recebida e em seguida a sua validação.

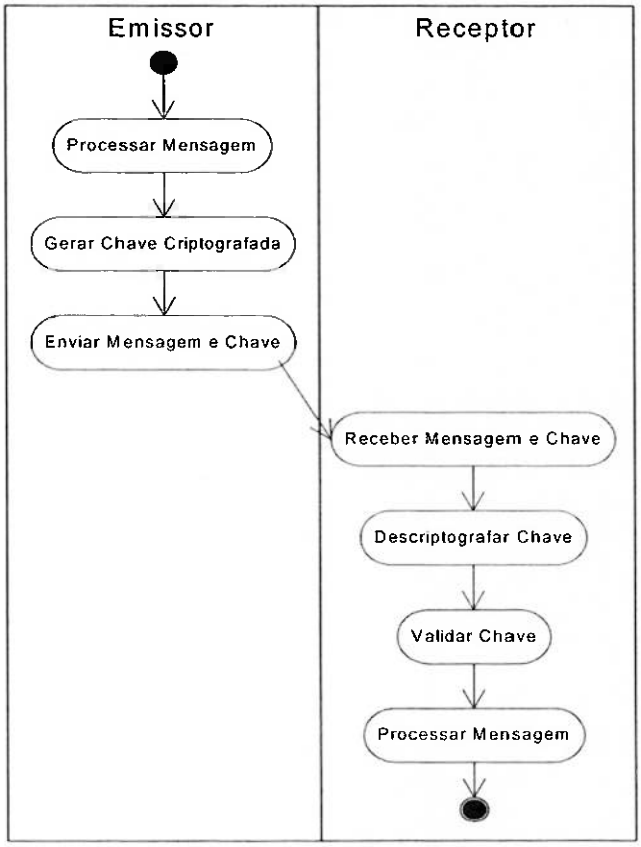

Figura 4.5 – Assinatura digital
Baseado em: Microsoft (2005)

- **Limite de acesso:** Além de proteger as mensagens trocadas entre sistemas de software baseados em SOA, faz-se necessária a proteção dos servidores que hospedam as aplicações cliente e os serviços contra ataques oriundos de aplicações maliciosas que utilizam a Internet para este fim. A proposta sugerida nesta tática aponta para o uso de firewall, conforme mostrado na seção 3.4.2, este atua como uma barreira, bloqueando o recebimento de mensagens indevidas. Além desta medida, também é possível adotar uma configuração para a criação de uma área desmilitarizada, conhecida como DMZ. Esta área desmilitarizada possibilita que sistemas de software possam acessar os serviços e ao mesmo tempo impede o acesso às demais áreas do servidor e também da rede (ALLEN et al, 2001; BASS; CLEMENTS; KAZMAN, 2003; O'BRIEN; BASS; MERSON, 2005).

A figura 4.6 mostrada a seguir ilustra a utilização de um firewall e DMZ para um sistema de software baseado em SOA.

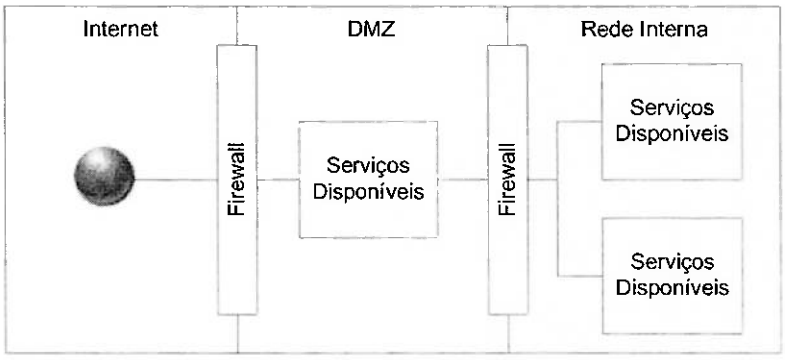

Figura 4.6 – Firewall e DMZ

Nesta figura, a parte denominada como Internet representa todo o meio externo aos servidores que hospedam o sistema de software.

O firewall, mostrado entre Internet e DMZ, representa o meio utilizado para filtrar o acesso externo de aplicações cliente aos serviços.

A parte denominada DMZ representa a área onde devem ser disponibilizados os serviços.

O firewall, mostrado entre DMZ e Rede Interna, representa o meio utilizado bloquear o acesso a rede interna.

A Rede Interna representa a área onde são abrigados outros serviços e outros sistemas de software contidos no servidor.

- **Utilização do protocolo HTTPS:** Com o objetivo de aumentar a segurança durante a troca de mensagens, é possível a utilização do protocolo HTTPS, onde as mensagens são enviadas de forma criptografada. Além disto, para sistemas de software baseados em SOA, é necessário a autenticação do serviço e também a autenticação da aplicação cliente. Neste caso, faz-se necessária a certificação das entidades por parte de uma autoridade capaz de avaliar a segurança da aplicação e emitir o respectivo certificado (NAKAMUR, 2002; O'BRIEN; BASS; MERSON, 2005).

4.3.3 Arquitetura Orientada a Serviços e atributos desempenho

A seguir, as táticas endereçadas ao atributo de desempenho, são analisadas quanto ao seu uso para atender as necessidades relacionadas a este atributo. De forma a colaborar com o mapeamento proposto neste trabalho:

- **Utilização de Cache:** Esta tática pode ser empregada com o objetivo de reduzir o tempo de processamento das requisições por parte dos serviços e conseqüente redução do tempo médio em seu atendimento.

Para sua utilização, é preciso avaliar os serviços a serem disponibilizados, com base no custo de processamento das requisições em termos de tempo e na freqüência de uso. Também é necessário definir a estratégia usada para a atualização de conteúdo dos serviços, que estejam utilizando cache, para garantir sincronismo entre as respostas emitidas nos serviços originais e os respectivos serviços em cache (BASS; CLEMENTS; KAZMAN, 2003; FERNANDEZ; FERNANDEZ; PAZOZ, 2005; MEIER et al., 2004).

O padrão Proxy (Buschmann et al, 1996; Gamma et al, 1998) que consiste em criar representação do serviço original, apresentando o mesmo tipo de interface, para onde a aplicação cliente submete as requisições, possui variações e entre estas o Cache Proxy que permite a utilização de cache no desenvolvimento do sistema de software.

A figura 4.7 mostrada a seguir, ilustra a interação de um sistema de software utilizando proxy durante uma requisição realizada. Nesta ilustração, é assumido que o serviço não possui a resposta atualizada em cache, sendo necessária a devida atualização. Este processo consiste em:

1. A aplicação cliente envia uma requisição a determinado serviço.
2. O proxy do sistema recebe a requisição e verifica se o serviço solicitado possui cache e se está devidamente atualizado (em casos onde o cache está atualizado, a resposta é encaminhada para aplicação cliente com base em seu conteúdo).
3. Neste caso, o cache não está atualizado, desta forma, o serviço original recebe a requisição, processa e retorna ao cache.
4. Este recebe a resposta e após realizar a atualização do cache, emite a resposta para a aplicação cliente.

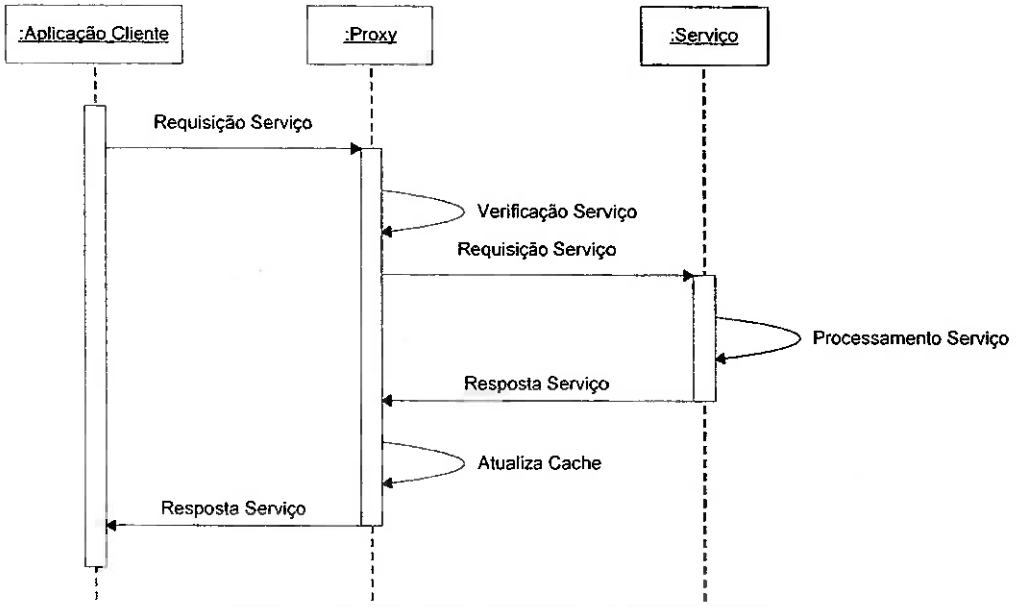

Figura 4.7 – Requisição de serviço utilizando Proxy
Baseado em: Buschmann et al, 1996

- **Atribuição de prioridade:** O alto volume de requisições feitas a um sistema de software baseado em SOA pode resultar em um enfileiramento, onde as respostas são emitidas na mesma seqüência de recebimento, entretanto, podem existir requisições com um nível maior de importância e estas precisam aguardar o processamento das requisições recebidas anteriormente. Este tempo de espera pode resultar em perda de desempenho, em decorrência da demora no atendimento a serviços prioritários. Com o objetivo de minimizar estes casos, esta tática possibilita a priorização no uso dos serviços a partir da documentação de uma política de utilização. Esta política pode ser implantada no sistema de sofware, de forma a estabelecer uma ordem no atendimento de requisições (BASS; CLEMENTS; KAZMAN, 2003).

Em Kim et al. (2005), é proposta uma técnica para a implantação de sistemas de software baseados em SOA, que está direcionada para atribuir prioridades na alocação de serviços. Através de uma política de utilização, criada a partir de acordos entre aplicações cliente e serviços, cada requisição recebida é classificada e sua execução realizada segundo a classificação. Quando é recebido um número grande de

requisições com a mesma classificação, a ordem de execução segue basicamente a forma FIFO, apresentada na seção 3.4.3.

- **Transferência de grandes volumes de dados:** O tamanho das mensagens trocadas, nos sistemas de software em SOA, durante as interações entre aplicações cliente e serviços incide diretamente no desempenho destes. Determinados formatos de mensagens podem resultar em um grande volume de dados a serem transferidos e conseqüente sobrecarga na rede.

  Além disto, aplicações cliente e serviços precisam manter sincronia durante a transmissão das mensagens. Para reduzir a perda de desempenho referente aos itens levantados, são apresentadas na seção 3.4.3 as táticas: Enviar mensagens com anexos e retornar URL para download (MEIER et al., 2004; O'BRIEN; BASS; MERSON, 2005; W3C SOAP a, 2000).

  Em sistemas de software que utilizem o protocolo SOAP 1.1 (W3C SOAP, 2000) é possível encaminhar mensagens como arquivos anexos, o que pode ocorrer com imagens ou outros documentos binários como mensagens compactadas (zip). Desta maneira, as mensagens podem ter seus conteúdos reduzidos através da compactação destes e o envio dos mesmos anexados.

  Outra prática recomendada para transferência de grandes volumes de dados consiste no serviço retornar para aplicação cliente um endereço eletrônico para download (URL), para que a aplicação possa obter os dados através desta forma.

- **Serialização de mensagens:** Conforme apresentado na seção 3.4.3, mensagens em formatos XML (W3C XML, 2004), possuem um tamanho muito maior que seu equivalente em binário. Isto reflete diretamente em perda de desempenho por sobrecarga na rede e a necessidade de maior tempo para processamento da requisição

por parte do serviço. Para minimizar perdas desta natureza, é possível a utilização de processo para serializar / deserializar as mensagens (NG, 2006; O'BRIEN; BASS; MERSON, 2005).

Para facilitar o processo de serializar mensagens, pode ser adotado o padrão XML-binary Optimized Packaging (XOP), que possibilita a transformação da mensagem em código binário de maneira mais eficiente em um documento XML. Em sistemas de software que utilizam tecnologias como Web Services e XML, este processo de transformação da mensagem em código binário pode ser realizado pelo serviço e então encaminhado com o protocolo SOAP (W3C XOP, 2005).

Neste processo de serializar mensagens usando XML, também pode ser utilizado o padrão Differential Serialization, que possibilita a transformação total da mensagem em código binário somente na primeira requisição. A mensagem é então salva em sua forma binária após atender a primeira requisição. Para atender as requisições subseqüentes, são transformados em código binário somente os elementos que sofreram alterações (ABU-GHAZALEH, 2004; ABU-GHAZALEH b, 2004; NG, 2006).

- **Utilização de padrões Síncrono e Assíncrono:** As interações entre aplicações cliente e serviços podem ocorrer através de dois padrões, síncrono e assíncrono. Conforme mostrado na seção 3.4.3, a diferença básica entre estes padrões se resume na necessidade da conexão manter-se bloqueada durante a interação para o processo sincronizado ou estar liberada no processo assíncrono. Com o objetivo de melhorar o desempenho, é recomendado o uso do processo assíncrono, entretanto, faz-se necessária uma análise para a seleção do processo que melhor atender a necessidade do negócio.

  Segundo Microsoft (2005), as duas situações mostradas a seguir podem colaborar na seleção do padrão:

- Considerar o uso do padrão assíncrono quando é necessária a utilização de processamento paralelo.
- Utilizar o padrão assíncrono quando é necessário realizar requisições a múltiplos serviços não relacionados.

Podem ser aplicados nos sistemas de software em SOA padrões como: Polling RM-Reply ou Callback RM-Reply, contudo, é necessário avaliar a viabilidade destes padrões de acordo como o serviço disponibilizado (COHEN, 2006; MEIER, 2004).

Após a apresentação de como as táticas relacionadas aos atributos de qualidade selecionados podem colaborar com os sistemas de software baseados em SOA, é mostrada em seguida a maneira como estas táticas podem ser agrupadas em camadas, nas entidades ligadas a Arquitetura Orientada a Serviços. Estas táticas são agrupadas de acordo com a forma como estão relacionadas aos elementos pertinentes a SOA.

4.4 ARQUITETURA ORIENTADA A SERVIÇOS E TÁTICAS DE QUALIDADE

Esta seção tem como objetivo mostrar o mapeamento entre táticas e a estrutura dos elementos que compõem SOA. As táticas citadas estão em conformidade com a tabela 4.1. A estrutura dos elementos que compõem SOA está apresentada na seção 2.2. Desta forma, a seleção da(s) tática(s) para aplicação em determinado elemento da arquitetura pode ser feita diretamente.

A figura 4.8 mostra as táticas divididas de acordo com a camada respectiva.

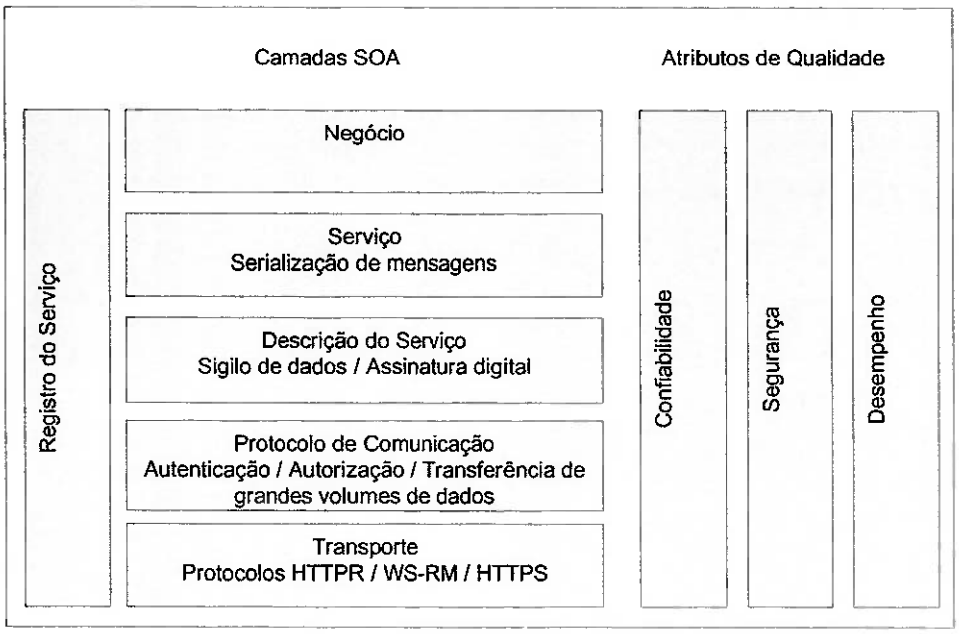

Figura 4.8 – Camadas SOA e atributos de qualidade relacionados

As táticas utilizadas no mapeamento são (ENDREI et al, 2004; FERNANDEZ; FERNANDEZ; PAZOZ, 2005; GANCI et al, 2006):

- Utilização do Protocolo HTTPR: Esta tática é relacionada com confiabilidade e com a camada de transporte. Pode ser analisada no momento de definir o protocolo a ser utilizado no transporte das mensagens.

- Adoção da especificação WS-ReliableMessaging (WS-RM): Esta tática é relacionada com confiabilidade e com a camada de transporte. Pode ser considerada no momento de definir o protocolo a ser utilizado no transporte das mensagens.

- Autenticação: Esta tática é relacionada com segurança e com a camada de protocolo de comunicação. Pode ser considerada no momento de definir a arquitetura do sistema de software.

- Autorização: Esta tática é relacionada com segurança e com a camada de protocolo de comunicação. Pode ser considerada no momento de definir a arquitetura do sistema de software.

- Sigilo de dados: Esta tática é relacionada com segurança e com a camada de descrição do serviço. Pode ser considerada no momento de definir a arquitetura do sistema de software.

- Assinatura digital: Esta tática é relacionada com segurança e com a camada de descrição do serviço. Pode ser considerada no momento de definir a arquitetura do sistema de software.

- Utilização do protocolo HTTPS: Esta tática é relacionada com segurança e com a camada de descrição do serviço. Pode ser considerada no momento de definir a arquitetura do sistema de software.

- Utilização de Cache: Esta tática é relacionada com desempenho e com a camada de protocolo de comunicação. Pode ser considerada no momento de definir a arquitetura do sistema de software.

- Atribuição de prioridade: Esta tática é relacionada com desempenho e com a camada de negócio. Pode ser considerada no momento de definir a arquitetura do sistema de software.

- Transferência de grandes volumes de dados: Esta tática é relacionada com desempenho e com a camada de protocolo de comunicação. Pode ser considerada no momento de definir a arquitetura do sistema de software.

- Serialização de mensagens: Esta tática é relacionada com desempenho e com a camada de serviços. Pode ser considerada no momento de definir a arquitetura do sistema de software.

As táticas provisão de servidores com redundância, limite de acesso e utilização de padrões síncrono e assíncrono não são citadas no mapeamento devido ao relacionamento com SOA referir-se a entidade e não mostrar-se inserida nos elementos da arquitetura. Entretanto, estas também podem ser consideradas no momento de definir a arquitetura do sistema de software.

Além das camadas mostradas, que representam os elementos de SOA discutidos na seção 2.2, são adicionadas três camadas e estas apresentadas na vertical, representando os atributos de qualidade discutidos neste trabalho. Esta maneira de apresentação, reflete a necessidade de analisar a forma como cada um dos atributos de qualidade pode contribuir nas camadas estruturadas de SOA.

4.5 CONSIDERAÇÕES FINAIS

O mapeamento apresentado no trabalho facilita o desenvolvimento de sistemas de software baseados em SOA, considerando os atributos de qualidade, onde as táticas que se relacionam com os elementos da arquitetura são agrupadas nas respectivas camadas. Desta maneira, além da possibilidade de se avaliar o uso de uma tática a partir de uma necessidade distinta, conforme mostrado na seção 4.2. Também é possível avaliar o uso de táticas relacionadas a determinado elemento de uma entidade pertinente a SOA.

5. ESTUDO DE CASO

5.1 CONSIDERAÇÕES INICIAIS

O objetivo deste capítulo é apresentar a utilização das táticas propostas por meio de um estudo de caso. O estudo de caso apresentado é enriquecido a partir da discussão apresentada em Endrei et al., (2004). O cenário deste estudo de caso refere-se a um sistema de software de Gerenciamento da Cadeia de Suprimentos (Supply Chain Management - SCM) tendo como base o comércio varejista.

5.2 O GERENCIAMENTO DA CADEIA DE SUPRIMENTOS

Neste estudo, é assumido que o sistema de software deve ser desenvolvido utilizando tecnologia de Web Services, onde os protocolos e padrões utilizados são:

- XML: formato utilizado para mensagens.
- WSDL: utilizado para descrição dos serviços.
- SOAP: protocolo de comunicação utilizado para troca de mensagens.
- HTTP: protocolo utilizado para o transporte de mensagens.

Além disso, é assumido que as fases de levantamento de requisitos, análise e validação destes junto aos stockholders estão finalizadas. Desta forma, é previsto que o sistema de software encontra-se na fase de definição da arquitetura e projeto. A utilização básica para o sistema de software tratado neste estudo de caso é descrita resumidamente a seguir.

Os clientes (usuários) podem acessar a interface Web do Sistema de Varejo, visualizar o catálogo de produtos e realizar compra de produtos. Este Sistema de Varejo também utiliza um serviço para enviar a requisição do pedido contendo o(s) produto(s) solicitado(s) para o Sistema de Armazenagem, que contém informações sobre os produtos armazenados. O Sistema de Armazenagem deve responder, informando se possui o(s) produto(s) solicitado(s) em estoque.

Quando a quantidade de itens de determinado produto fica abaixo do limite mínimo estabelecido, o Sistema de Armazenagem deve enviar uma requisição, através de um serviço, para o sistema da empresa fabricante (Sistema do Fabricante) com o objetivo de realizar a reposição do produto. O Sistema do Fabricante deve responder ao Sistema de Armazenagem, dentro de um período de tempo definido. Esta resposta deve ser realizada de forma assíncrona. Após análise do sistema, são considerados para o estudo os seguintes sistemas contendo serviços:

- Sistema de Varejo (Serviços de Varejo): disponibiliza os serviços para visualizar os catálogos de produtos e realizar os pedidos de compra.
- Sistema de Armazenagem (Serviços de Armazenagem): dá suporte para a entrega de produtos vendidos e verifica a quantidade de itens dos produtos em estoque. Quando esta quantidade fica abaixo dos níveis determinados, este sistema envia um pedido de reposição ao Sistema do Fabricante utilizando serviços.
- Sistema Fabricante (Serviço Fabricante): recebe os pedidos de reposição de produtos, inicia o processo de manufatura destes e responde ao Sistema de Armazenagem.

A figura 5.1 mostrada a seguir, ilustra as interações entre os serviços citados.

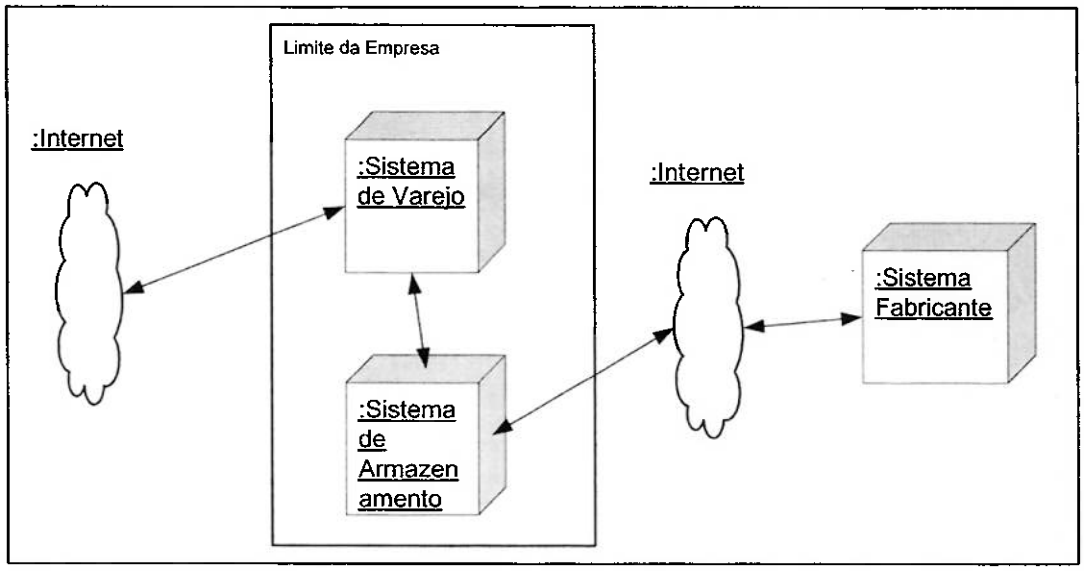

Figura 5.1 – Serviços definidos no SCM Varejista

A partir das informações básicas sobre as funcionalidades, o possível sistema de software é analisado utilizando o conjunto de táticas mostradas na seção 4.3. Informações adicionais são passadas na medida em que o tema é abordado no trabalho.

A tabela 5.1 apresenta um mapeamento entre as táticas discutidas na seção 4.3, para atender as necessidades mapeadas na tabela 4.1, e os sistemas utilizados no SCM deste estudo de caso. Este mapeamento possibilita uma análise dos sistemas a partir das características pertinentes a determinada tática.

Atributo de Qualidade	Tática	Sistema Analisado
Confiabilidade	Provisão de servidores com redundância.	Sistema de Varejo Serviço de Armazenagem
	Utilização do Protocolo HTTPR	Os sistemas não utilizam esta tática.
	Adoção da especificação WS-ReliableMessaging.	Sistema de Varejo Serviço de Armazenagem Sistema Fabricante

Atributo de Qualidade	Tática	Sistema Analisado
Segurança	Autenticação	Sistema de Armazenagem Sistema Fabricante
	Autorização	Sistema de Armazenagem Sistema Fabricante
	Sigilo de dados	Sistema de Armazenagem Sistema Fabricante
	Assinatura digital	Os sistemas não utilizam esta tática.
	Limite de acesso	Sistema de Varejo Serviço de Armazenagem
	Utilização do protocolo SSL	Os sistemas não utilizam esta tática.
Desempenho	Utilização de Cache	Sistema de Varejo
	Atribuição de prioridade	Sistema de Armazenagem
	Transferência de grandes volumes de dados	Sistema de Varejo Sistema de Armazenagem Sistema Fabricante
	Serialização de mensagens	Sistema de Varejo Sistema de Armazenagem Sistema Fabricante
	Utilização de padrões Síncrono e Assíncrono	Sistema de Varejo Sistema de Armazenagem Sistema Fabricante

Tabela 5.1 - Mapeamento entre atributos de qualidade e sistemas

A seguir, as táticas apresentadas na tabela 5.1 são analisadas quanto a sua aplicação para auxiliar no desenvolvimento do sistema de software proposto neste estudo de caso. Estas táticas são agrupadas de acordo com os respectivos atributos de qualidade.

Com relação ao atributo de qualidade confiabilidade são considerados.

- **Provisão de servidores com redundância:** Os serviços do Sistema de Varejo e Sistema de Armazenagem, situados dentro dos limites da empresa podem utilizar-se das considerações mostradas nesta tática. Desta forma, os usuários (aplicações cliente) que precisam utilizar os serviços contidos no Sistema de Varejo e Serviço de Armazenagem, podem contar com maior confiabilidade contra falhas decorrentes de problemas da infra-estrutura. A confiança por parte da empresa fabricante dos produtos (Sistema Fabricante) também aumenta devido a redução da probabilidade de falhas.

- **Adoção da especificação WS-ReliableMessaging (WS-RM):** A troca de mensagens entre as entidades: Sistema de Varejo – Sistema de Armazenagem e Sistema de Armazenagem – Sistema Fabricante, podem ser realizadas em concordância com a tática Utilização do Protocolo HTTPR ou Adoção da especificação WS-ReliableMessaging.

  Neste sistema de software, de acordo com Endrei et al (2004), é considerada a segunda opção devido a aplicação utilizar Web Services e existir a necessidade de obter retorno sobre as mensagens encaminhadas (informando sucesso ou falha na entrega da mensagem).

  Para a troca de mensagens entre a empresa varejista e outra empresa fabricante, faz-se necessário um acordo referente ao protocolo utilizado, podendo ser adotada esta mesma tática ou a Utilização do Protocolo HTTPR.

Com relação ao atributo de qualidade de segurança são considerados:

- **Autenticação e Autorização:** Ao analisar o sistema de software observando estas táticas, é possível definir:

  As trocas de mensagens entre: Sistema de Varejo – Sistema de Armazenagem não necessitam de autenticação e autorização para acesso aos serviços. Neste sistema de software, os serviços mencionados estão sob a mesma rede interna, possuem as devidas informações sobre as interfaces para acesso e o nível de permissão necessário para utilização dos serviços.

  A troca de mensagens entre: Sistema de Armazenagem – Sistema Fabricante deve utilizar os mecanismos de autenticação e autorização (táticas: Autenticação e Autorização). Esta solicitação é apresentada porque a necessidade de comunicação entre os serviços envolve a utilização da Internet como meio para troca de mensagens.

  Neste sistema de software, o padrão de autenticação utilizado é o "Padrão de Autenticação Através de Broker", mencionado nas seções 3.4.2 e 4.3.2. Após a realização da autenticação, a autorização deve considerar as permissões cedidas aos grupos em que a aplicação cliente está inserida. Desta maneira, a autorização é dada aos grupos, e passam a ser consideradas na utilização dos serviços.

- **Sigilo de dados:** Ao analisar o sistema de software observando esta tática, é possível definir:

  As trocas de mensagens entre: Sistema de Varejo – Sistema de Armazenagem não necessitam de criptografia. Neste sistema de software, os serviços utilizados estão sob a mesma rede interna e esta possui proteção contra acessos provenientes de redes externas (Internet). Também não é utilizado entre estes serviços o protocolo SSL

(tática: Utilização do protocolo SSL) ou mesmo a assinatura digital (tática: Assinatura Digital).

As trocas de mensagens entre: Sistema de Armazenagem – Sistema Fabricante deve utilizar esta tática (Sigilo de dados) para a criptografia de mensagens, devido a necessidade da comunicação entre os serviços envolver a utilização da Internet como meio para troca de mensagens. Neste sistema de software, é utilizado o mesmo código para criptografar / descriptografar as mensagens. Nas trocas de mensagens entre: Sistema de Armazenagem – Sistema Fabricante também não é utilizado o protocolo SSL (tática: Utilização do protocolo SSL) ou mesmo a assinatura digital (tática: Assinatura Digital).

- **Assinatura digital:** Esta tática não é utilizada neste sistema de software de acordo com as definições. Conforme mencionado anteriormente, as trocas de mensagens entre: Sistema de Varejo – Sistema de Armazenagem e Sistema de Armazenagem – Sistema Fabricante não devem adotar esta tática.

- **Limite de acesso:** Ao observar o sistema de software proposto a partir das recomendações mencionadas nesta tática, é possível definir:

  Os servidores que abrigam as entidades: Sistema de Varejo e Sistema de Armazenagem atendem as recomendações desta tática e utilizam em sua infra-estrutura padrões de firewalls e configuração de área desmilitarizada, conforme recomendações sugeridas nesta tática.

  Os servidores que hospedam o Sistema Fabricante, não são analisados nesta parte por motivo de apresentar-se como uma entidade externa ao sistema de software.

- **Utilização do protocolo HTTPS:** Esta tática não é utilizada neste sistema de software de acordo com as solicitações. Conforme mencionado anteriormente, as trocas de mensagens entre: Sistema de Varejo – Sistema de Armazenagem e Sistema de Armazenagem – Sistema Fabricante não devem adotar esta tática.

Com relação ao atributo de qualidade de segurança são considerados:

- **Utilização de Cache:** Ao observar o sistema de software proposto a partir das recomendações mencionadas nesta tática, é possível definir:

  A entidade Sistema de Varejo pode utilizar esta tática para disponibilizar o serviço responsável por mostrar a lista de produtos a serem acessados para venda. Para estes produtos, também devem ser disponibilizadas informações sobre valores, peso e descrição. Os demais serviços não devem utilizar esta tática por motivo de um volume menor na taxa de uso e o conteúdo destes passar por atualizações constantes.

- **Atribuição de prioridade:** O sistema de software foi analisado utilizando esta tática e as solicitações feitas pelo cliente. Neste ponto, é possível definir como é utilizada esta tática.

  Os serviços utilizados para a troca de mensagens entre as entidades Sistema de Varejo – Sistema de Armazenagem devem possuir prioridade de utilização em relação a troca de mensagens entre as entidades Sistema de Armazenagem – Sistema Fabricante. Desta forma, a troca de mensagens deve seguir a maneira "Escalonamento fixo de prioridade", descrita na seção 3.4.3.

  Mesmo com esta definição, quanto a maneira de atribuir prioridade para a troca de mensagens, no caso de receber várias requisições com a mesma prioridade, a maneira FIFO (Escalonamento através de fila), descrita na seção 3.4.3, deve ser seguida.

- **Transferência de grandes volumes de dados:** A troca de mensagens entre os sistemas envolvidos no sistema de software pode utilizar esta tática. Neste caso, deve ser aplicada juntamente com o protocolo SOAP 1.1 (W3C SOAP, 2002), onde as mensagens com conteúdo extenso podem ter estes convertidos em dados binários e anexados nas mensagens transferidas.

- **Serialização de mensagens:** As entidades: Sistema de Varejo, Sistema de Armazenagem e Sistema Fabricante devem utilizar as recomendações desta tática. Independente do meio utilizado para troca de mensagens, estas devem ser serializadas e utilizar o protocolo SOAP para transmissão.

- **Utilização de padrões Síncrono e Assíncrono:** Ao observar o sistema de software proposto a partir das definições contidas nesta tática, é recomendada sua utilização para atuar na redução de perdas de desempenho durante a troca de mensagens.

  A troca de mensagens entre as entidades: Sistema de Varejo – Sistema de Armazenagem deve ocorrer de maneira sincronizada, onde os serviços requisitados são processados e em seguida retornam para o elemento requisitante. Entretanto, as mensagens enviadas a partir do Sistema de Armazenagem para o Sistema Fabrica, requisitando a reposição de produtos faltantes, deve utilizar o padrão Callback RM-Reply, mostrado na seção 3.4.3. O motivo deste acordo entre as empresas origina-se na impossibilidade do Sistema Fabricante retornar de forma sincronizada a mensagem com o conteúdo devido, desta forma o Sistema de Armazenagem envia a mensagem discriminada e recebe como retorno a informação de recebimento da mesma. Após o processamento, o Sistema Fabricante envia novamente a resposta com o conteúdo devido para o Sistema de Armazenagem.

Observando o sistema de software proposto através das táticas mostradas, é possível utilizar as recomendações destas para auxiliar no seu desenvolvimento.

Este estudo de caso representa parte de um sistema de software de Gerenciamento da Cadeia de Suprimentos (Supply Chain Management - SCM), onde são descritas de maneira simplificada as entidades, aqui denominadas de sistemas, e funcionalidades que devem estar presentes nos serviços a serem desenvolvidos. A partir das necessidades apresentadas por estes serviços, é realizada uma análise baseada nos atributos de qualidade de confiabilidade, de segurança e de desempenho.

Com esta análise, a partir dos atributos de qualidade utilizados, as táticas relacionadas são avaliadas quanto a sua aplicação nos sistemas. De acordo com as necessidades destes, a adoção de determinada tática pode ou não apresentar resultados positivos.

Esta análise, na fase do projeto do sistema de software, contribui para que as necessidades relacionadas aos atributos de qualidade, discutidos neste trabalho, possam ser atendidas. Com este propósito, é mostrado através das táticas, um conjunto de possibilidades para atender a estas necessidades.

5.3 CONSIDERAÇÕES FINAIS

Os sistemas de software baseados na Arquitetura Orientada a Serviços podem atender as necessidades de comunicação entre empresas distintas. Contudo, estas aplicações podem apresentar deficiências relacionadas à falhas na transmissão dos dados, na segurança do conteúdo trocado e no desempenho alcançado. Uma maneira de evitar imprevistos desta natureza é o projeto do sistema de software considerar o atendimento dos atributos de qualidade de confiabilidade, de segurança e de desempenho, durante a sua concepção.

Para viabilizar este atendimento, é demonstrado nesta seção como o conjunto de táticas relacionadas aos atributos de qualidade selecionados, pode ser utilizado na definição do projeto a ser desenvolvido para o sistema de software.

6. CONCLUSÃO

Este trabalho apresenta um conjunto de táticas para apoiar o desenvolvimento de sistemas de software baseados na Arquitetura Orientada a Serviços (SOA), sendo selecionado um conjunto de atributos de qualidade.

Estas táticas podem atuar na concepção destes sistemas de software, propondo ações para auxiliar na maneira como ocorre a comunicação entre as entidades, na forma como os conteúdos das mensagens devem ser construídos e no aumento dos níveis de confiança e segurança na infra-estrutura dos sistemas.

A avaliação para utilização das táticas pode ser realizada a partir de uma necessidade distinta da arquitetura SOA, também pode ser feita a avaliação do uso destas táticas a partir de seu relacionamento com determinado elemento de uma entidade pertinente a SOA.

Ao utilizar as táticas discutidas neste trabalho para analisar os sistemas de software propostos, é possível realizar um levantamento sobre as práticas a serem adotadas para contribuir com o atendimento aos atributos de confiabilidade, de segurança e de desempenho. Como resultado desta análise, pode ser criado um relacionamento entre os serviços que devem estar presentes no sistema de software, e a maneira como as táticas podem ser usadas para apoiar este sistema, conforme mostrado no estudo de caso deste trabalho.

6.1 TRABALHOS FUTUROS

Como trabalhos futuros consideram-se a necessidade de desenvolver um número maior de táticas relacionadas aos atributos selecionados.

Além dos atributos de qualidade confiabilidade, a segurança e ao desempenho, esta análise pode ser expandida a outros atributos tais como interoperabilidade, disponibilidade, usabilidade, escalabilidade, adaptabilidade, testabilidade e modificabilidade (BASS; CLEMENTS; KAZMAN, 2003; DROMEY, 1995; O'BRIEN; BASS; MERSON, 2005).

Os aspectos relativos à mensuração do atendimento aos atributos de qualidade por estas táticas por constituir-se também como mais um trabalho a ser realizado.

A adoção de medidas para atender a determinado atributo de qualidade pode impactar de maneira negativa no atendimento a outro atributo, desta forma, pode ser considerado como trabalho futuro uma análise quanto aos possíveis conflitos causados na adoção de medidas ligadas a um atributo de qualidade específico.

REFERÊNCIA BIBLIOGRÁFICA

ABU-GHAZALEH, N.; LEWIS, M. J.; GOVINDARAJU, M., **Differential Serialization for Optimized SOAP Performance,** To appear in the 13th IEEE International Symposium on High Performance Distributed Computing (HPDC-13), June, 2004, Honolulu, Hawaii.

ABU-GHAZALEH b, N.; GOVINDARAJU, M; LEWIS, M. J., **Optimizing Performance of Web Services with Chunk-Overlaying and Pipelined-Send,** International Conference in Internet Computing (ICIC), pp: 482-485, June, 2004.

ALLEN, J. **The CERT Guide to System and Network Security Practices.** Boston, MA: Addison-Wesley, 2001.

AZIM O.; HAMID, A. K, **Cache SOAP services on the client side - Design patterns let you cache SOAP services and improve Performance.** March 8, 2002

BANKS, A.; CHALLENGER, J.; CLARKE, P.; DAVIS, D; KING, R.; WITTING, K.; DONOHO, A.; HOLLOWAY, T.; IBBOTSON, J.; TODD, S., **HTTPR Specification.** April, 2002, Disponível em: http://www.ibm.com/developerworks/library/ws-httprspec/ (Data Acesso 30/09/2006)

BASS, L; CLEMENTS, P; KAZMAN. R., **Software Architecture in Practice.** Second Edition, Addison Wesley, 2003.

BROWN, A; JOHNSTON, S.; KELLY, K., **Using Service-Oriented Architecture and Component-Based Development to Build Web Service Applications.** Rational Software Corporation, 2002. Disponível em: http://www-106.ibm.com/developerWorks/rational/library/4860.html.

BROWNSWORD, L. et al., **Current Perspectives on Interoperability (CMU/SEI-2004-TR-009, ADA421613).** Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2004. Disponível em : http://www.sei.cmu.edu/publications/documents/04.reports/04tr009.html

BUSCHMANN, F; MEUNIER, R; ROHNERT, H; SOMMERLAD, P; STAL, M., **Pattern-Oriented Software Architecture, A System of Patterns Volume 1**, Wiley, 1996.

CEARLEY D. W.; FENN J.; PLUMMER D. C.; **Gartner's Positions on the Five Hottest IT Topics and Trends in 2005.** Data de Publicação 12 de maio de 2005. Gartner. Disponível em: http://www.gartner.com/resources/125800/125868/gartners_positi.pdf

COHEN F., **FastSOA: The way to use native XML technology to achieve Service Oriented Architecture governance, scalability, and Performance.** Morgan Kaufmann, Dezembro, 2006.

COMPUTERWORLD; ÂNGELO F. K., **SOA: incessante busca por agilidade.** Data Publicação 29 de novembro de 2005. Data registro 07 de setembro de 2006. Disponível em: http://computerworld.uol.com.br/infra_estrutura/2005/11/29/idgnoticia.2006-03-29.9302141371/IDGNoticia_view

COMPUTERWORLD b, **Pesquisa: SOA é implementada por 67% das empresas em 2006.** Data Publicação 24 de agosto de 2006. Data registro 07 de setembro de 2006. Disponível em: http://computerworld.uol.com.br/mercado/2006/08/24/idgnoticia.2006-08-24.0058792133/IDGNoticia_view.

DROMEY, R. G., **A Model for Software Product Quality, IEEE Transactions on Software.** Engineering, Vol. 21, No. 2, February 1995.

DUAN, Z.; BOSE, S.; STIRPE, P.A.; SHONIREGUN, C.; LOGVYNOVSKIY, A., SOA without **Web services: a pragmatic implementation of SOA for financial transactions systems**, 2005 IEEE International Conference , July, 2005.

ENDREI, M; ANG, J; ARSANJANI, A; CHUA, S; COMTE, P; KROGDAHL, P; LUO, P; NEWLING, T., **Patterns: Service-Oriented Architecture and Web Services**. IBM RedBooks, April 2004.

ERRADI A; MAHESHWARI P., **A Broker-based Approach for Improving Web Services Reliability.** IEEE Computer Society, 2005

FERNANDEZ, J; FERNANDEZ, A.; PAZOZ J., **Optimizing Web Services Performance using Caching.** Next Generation Web Services Practices, 2005. NWeSP 2005. International Conference, Publication Date: 22-26 August. 2005

FERNANDEZ b, E.B.; DELESSY, N., **Using Patterns to Understand and Compare Web Services Security Products and Standards.** IEEE, Feb. 2006

GAMMA, E; HELM, R; JOHNSON, R; VLISSIDES, J., **Design Patterns Elements of Reusable Object-Oriented Software.** Addison-Wesley Professional Computing Series, 1998.

GANCI, J.; ACHARYA, A.; ADAMS, J.; EUSEBIO, P.; RAHI, G.; STRACHAN, S.; UTSUMI, K.; WASHIO, N, **Patterns: SOA Foundation Service Creation Scenario.** IBM RedBooks, June 2006.

HAPNER, M. et al., **Java Message Service.** Version 1.1, Sun Microsystems Inc April 12, 2002

HASHEMI, S.M.; RAZZAZI, M; BAHRAMI, A., **ISRUP E-Service Framework for agile Enterprise Architecting**. ieee, April, 2006

HUANG, D., **Semantic Descriptions of Web Services Security Constraints.** sose, pp. 81-84, Second IEEE International Symposium on Service-Oriented System Engineering (SOSE'06), 2006.

IBM a; MYERSON, J., **Work with Web services in enterprise-wide SOA, Part 7: Speed-up Web services applications with the XML-binary Optimized Packaging Specification.** October, 2005

ISIS, **Guide to Interoperability - Web Services.** Disponível em: http://www.sei.cmu.edu/isis/guide/technologies/web-services.htm, Last Modified: 13 January 2006.

KARP, A. H., **Authorization-Based Access Control for the Services Oriented Architecture.** Hewlett-Packard Laboratories, Fourth International Conference on Creating, Connecting and Collaborating through Computing (IEEE), 2006.

KESSLER, K.; **Web service Technology for SAP Netweaver**, 2004
Disponível em: https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/f65ecf90-0201-0010-94b0-c9983be54c67, Acessado em: 10/10/2006

KIRCHER, M.; JAIN, P., **Pattern-Oriented Software Architecture, Volume 3: Patterns for Resource Management**. John Wiley & Sons, April, 2004

KIM, D.; SANGKYU, LEE, S.; HAN, S; ABRAHAM, A.,**Improving Web services performance using priority allocation method**, IEEE Software, Aug. 2005.

KOROTKIY, M.; TOP, J., **Onto-SOA: From Ontology-enabled SOA to Service-enabled Ontologies.** IEEE, February, 2006.

KULKARNI, N.; KUMAR, S.; MANI, K.; PADMANABHUNI, S. **Web Services: E-Commerce Partner Integration.** IEEE, IT Professional, Mar-Apr, 2005.

LEFFINGWELL, D.; WIDRIG D.; **Managing Software Requirements, A Use Case Approach**: Addison Wesley, Second Edition, 2003.

LEVINSON, H.L.; O'BRIEN, L., **Acquiring Evolving Technologies: Web Services Standards.** SEI Institute, February, 2006

LEWIS, GRACE & WRAGE, LUTS., **Approaches to Constructive Interoperability (CMU/SEI-2004-TR-020).** Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2005. Disponível em: http://www.sei.cmu.edu/publications/documents/04.reports/04tr020.html

LEWIS,G.; MORRIS, E.; SMITH,D., **Analyzing the Reuse Potential of Migrating Legacy Components to a Service-Oriented Architecture.** Software Eng. Inst, March, 2006

MAHMOUD Q.H.., **Service-Oriented Architecture (SOA) and Web Services: The Road to Enterprise Application Integration (EAI).** April, 2005; Disponível em: http://java.sun.com/developer/technicalArticles/WebServices/soa/

MALLOY, B.A.; KRAFT, N.A.; HALLSTROM, J.O.; VOAS, J.M.; **Improving the predictable assembly of service-oriented architectures.** IEEE, March-April, 2006.

MATOS, J.P., **Concepção de Arquitetura de Software com Base nos Estilos de Arquitetura e Requisitos Não Funcionais.** Tese de Doutorado- Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo, 2005.

MEIER J.D at al., **Improving .NET Application Performance and Scalability, Chapter 10 — Improving Web Services Performance.** Microsoft, May, 2004. Disponível em: http://msdn2.microsoft.com/en-us/library/ms998562.aspx

MICROSOFT; HOGG, J at al., **Web Service Security - Scenarios, Patterns, and Implementation Guidance for Web Services Enhancements (WSE) 3.0.** Microsoft, November, 2005

NAEDELE, M.; ABB Corporate Res.; Switzerland, **Standards for XML and Web Services Security.** IEEE, April, 2003

NAKAMUR, Y.; HADA, S.; NEYAMA, R.; IBM, **Towards the Integration of Web Services Security on Enterprise Environments.** Applications and the Internet (SAINT) Workshops, 2002. Proceedings. 2002 Symposium on, 2002

NEZHAD, H.R.M.; BENATALLAH, B.; CASATI, F.; TOUMANI, F., **Web Services Interoperability Specifications.** IEEE, May, 2006.

NG. A., **Optimising Web Services Performance with Table Driven XML.** Department of Computing, Macquarie University, North Ryde, NSW 2109, Software Engineering Conference. Australia, 2006.

OASIS WSS, **Web Services Security: SOAP Message Security 1.1 (WS-Security 2004).** OASIS Standard Specification; February, 2006,

OASIS WS-RM, **Web Services Reliable Messaging (WS-ReliableMessaging),** Committee Draft 04, August 11, 2006, Disponível em: http://docs.oasis-open.org/ws-rx/wsrm/200608/wsrm-1.1-spec-cd-04.pdf, Acesso em: 20/10/2006

O'BRIEN, L; BASS, L; MERSON, P., **Quality Attributes and Service-Oriented Architectures (CMU/SEI-2005-TN-014).** Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, September, 2005.

OMG CORBA, **Common Object Request Broker Architecture: Core Specification,** Version 3.0.3, March, 2004. Disponível em: http://www.omg.org/docs/formal/04-03-12.pdf

PAPAZOGLOU, M.P.; VAN DEN HEUVEL, W.-J., **Web services management: a survey**, IEEE Internet Computing, December, 2005.

TODD S; HURSLEY; PARRL F; CONNER M., **A Primer for HTTPR - An overview of the reliable HTTP protocol,** July, 2001. Disponível em: http://www-128.ibm.com/developerworks/webservices/library/ws-phtt/ (Data Acesso 18/09/2006)

RFC 2616: **Hypertext Transfer Protocol -- HTTP/1.1**, 1999, Disponível em: ftp://ftp.isi.edu/in-notes/rfc2616.txt

RFC 2821: **Simple Mail Transfer Protocol**, 2001, Disponível em: http://tools.ietf.org/html/rfc2821

STAL, M.;SIEMENS CORPORATE TECHNOL.; **Using architectural patterns and blueprints for service-oriented architecture.** IEEE Computer Society, March/April, 2006.

SNEED, H. M., **Integrating legacy software into a service oriented architecture.** IEEE, March, 2006.

TSAI, W.T.; FAN, C; CHEN, Y.; PAUL, R.; CHUNG J, **Architecture classification for SOA-based applications.** IEEE, April, 2006.

VO, H.T.K.; WEINHARDT, C.; WOJCIECHOWSKI, R., **Corporate Portals from a Service-Oriented Perspective The CoFiPot Implementation.** IEEE, June, 2006

W3C SOAP, **Simple Object Access Protocol (SOAP) 1.1**. May, 2000. Disponível em: http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

W3C SOAP a., **SOAP Messages with Attachments.** December 2000. Disponível em: http://www.w3.org/TR/SOAP-attachments

W3C WSA, **Web Services Architecture.** Fevereiro, 2004. Disponível em: http://www.w3.org/TR/ws-arch/wsa.pdf

W3C WSDL, **Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language.** March, 2006. Disponível em: http://www.w3.org/TR/wsdl20/

W3C, **World Wide Web Consortium**. Diponível em: http://www.w3c.org; Acessado em: 05/09/2006

W3C XML, **Extensible Markup Language (XML) 1.0 (Fourth Edition).** Recommendation August 2006, Disponível em: http://www.w3.org/TR/2006/REC-xml-20060816/

W3C XOP, **XML-binary Optimized Packaging.** Janeiro, 2005. Disponível em: http://www.w3.org/TR/xop10/

YU, W.DAN, **Intelligent Access Control for Web Services Based on Service Oriented Architecture Platform.** IEEE, April, 2006.ZDUN, U; HENTRICH C.; WIL M.P.; AALST V. D., **A Survey of Patterns for Service-Oriented Architectures.** International Journal of Internet Protocol Technology 2006 - Vol. 1, No.3 pp. 132 – 143