WSLogA: Monitoramento de Processos de Negócio Baseados em Serviços Web

Sérgio Manuel Serra da Cruz

Universidade Federal do Rio de Janeiro

Instituto de Matemática

Núcleo de Computação Eletrônica

Mestrado

Maria Luiza Machado Campos, Ph.D

Paulo de Figueiredo Pires, D.Sc

Rio de Janeiro

2004

WSLogA: Monitoramento de Processos de Negócio Baseados em Serviços Web

Sérgio Manuel Serra da Cruz

Dissertação submetida ao corpo docente do Departamento de Ciências da Computação do Instituto de Matemática/Núcleo de Computação Eletrônica da Universidade Federal do Rio de Janeiro como parte dos requisitos necessários para a obtenção do grau de Mestre em Informática.

Aprovada por:

_____

Profa. Maria Luiza Machado Campos, Ph.D. - Orientadora

_____

Prof. Paulo de Figueiredo Pires, D.Sc. – Co-Orientador

_____

Profa. Marta Lima de Queirós Mattoso, D.Sc.

_____

Prof. Pedro Manuel da Silveira, Ph.D.

_____

Prof. Carlo Emmanoel Tolla de Oliveira, Ph.D.

RIO DE JANEIRO, RJ – BRASIL

DEZEMBRO DE 2004

Dedico este trabalho aos meus amados Ana Cláudia e Pedro.

C957 CRUZ, Sérgio Manuel Serra da Cruz

WSLogA: WSLogA: Monitoramento de Processos de Negócio Baseados em Serviços Web / Sérgio Manuel Serra da Cruz. - Rio de Janeiro, 2004.

xiv, 130 f.: il.
Dissertação (Mestrado em Ciências da Computação) – Universidade Federal do Rio de Janeiro – UFRJ, Instituto deMatemática – IM/NCE, 2004.

Orientador: Maria Luiza Machado Campos

1. Composição de Serviços Web. 2. Data Marts. 3. Processos de Negócio. 4. Teses

I – Campos, Maria Luiza Machado Campos (Orient.) II. Universidade Federal do Rio de Janeiro. Instituto de Matemática III. Título. CDD 004.6

AGRADECIMENTOS

Ao terminar esta dissertação e, olhar para trás, percebo que tenho uma dívida de gratidão para com muitas pessoas que contribuíram para que este trabalho se realizasse com sucesso.

Quero aqui registrar inicialmente meus agradecimentos a Deus. Aquele que é a nossa primeira origem e o nosso último fim.

À Maria, Mãe do Filho eterno de Deus feito homem, mãe caríssima que sempre me embala nas franquezas e alegrias.

À minha amada esposa Ana Claudia Vieira, que mesmo ocupada pelas atividades de sua tese de doutorado, soube graciosamente incentivar este trabalho.

Ao meu filho Pedro Vieira Cruz. Aproveito o momento para pedir-lhe perdão por ter subtraído horas preciosas da sua infância.

Aos meus sogros por ter contribuído ativamente na solução dos problemas do cotidiano.

À minha mãe e ao meu pai (*in memoriam)* pela minha formação.

Aos meus avós maternos pela amorosa acolhida nos momentos difíceis.

Aos professores Maria Luiza Machado Campos e Paulo de Figueiredo Pires pela confiança, amizade, dedicação e competência pela excelente orientação.

Aos professores Marta Lima de Queirós Mattoso, Pedro Manuel da Silveira e Carlo Emmanuel Tolla de Oliveira por terem participado da banca

Aos professores do curso de mestrado no IM-NCE/UFRJ, em especial à professora Lígia Barros pela excelente disciplina de metodologia de pesquisa.

À direção da NCE-UFRJ e da Área de Apoio Acadêmico que viabilizaram o afastamento das minhas atividades profissionais em regime de meio período.

À Selma, Marília, Silma, Raquel, Maria e Maria Rosa pela valiosa ajuda na organização das referências bibliográficas.

À Dona Deise pelo incomparável trabalho feito na secretaria acadêmica da Informática.

À leal amizade das amigas Simone Garcia e Tania Hiromi, que apesar da minha chatice sempre me ajudaram.

À Linair por ter colaborado na confecção de uma série de trabalhos.

À Yoko, hoje professora Maria Cláudia, pelo incentivo no início do curso.

Ao Fábio Heuseler, Camille Furtado e Rafael por terem implementado uma pequena parte da WSLogA no seu projeto final de curso.

Aos colegas do mestrado, e em especial à Alissandra Martins, Glenda Amaral, Maria Cristina, Kelli Cordeiro, Luciana Leal, Renata Wo, Alexandre Moriya, Arnaldo, Cássio Freire, Jorge Barboza, Edmundo Contar, Erik Praxedes, Eugênio Silva, Ilan Chamovitz, Marcos Rabelo, Maurício, Pedro Demasi, Pablo Mendes e tantos outros.

Aos meus familiares, amigos, colegas de trabalho e todas as pessoas que de algum modo contribuíram para a realização desta dissertação.

“Os dementadores estavam mais próximos...Formavam uma muralha sólida...
- EXPECTO PATRONUM!- berrou Harry”

J. K. Rowling

RESUMO

CRUZ, Sérgio Manuel Serra da. **WSLogA: Monitoramento de Processos de Negócio Baseados em Serviços Web**. Orientadores: Maria Luiza Machado Campos e Paulo de Figueiredo Pires. Rio de Janeiro: UFRJ/IM, 2004. Dissertação (Mestrado em Informática).

A tecnologia de serviços Web representa um significativo avanço na contínua evolução das iniciativas de comércio eletrônico. Para explorar completamente as oportunidades de negócio oferecidas por este paradigma, faz-se necessário monitorar sua utilização. Esta tarefa pode ser executada através dos registros em logs. Entretanto, as atuais iniciativas de log na Web não cobrem a utilização de serviços Web. Esta dissertação apresenta e discute como o uso da WSLogA, uma arquitetura de log de serviços Web, pode auxiliar a gerência de processos de negócio no ambiente Web. A arquitetura WSLogA disponibiliza informações necessárias para a análise de processos de negócio baseado no monitoramento da qualidade de serviços, de forma a detectar problemas, tais como os de desempenho e disponibilidade de serviços Web. Além disso, para viabilizar investigações mais refinadas, propomos uma abordagem multi-perspectiva através de um conjunto de *Data Marts* originados a partir dos logs de serviços Web e outras fontes de dados.

ABSTRACT

CRUZ, Sérgio Manuel Serra da. **WSLogA: Monitoramento de processos de negócio baseados em serviços Web**. Orientadores: Maria Luiza Machado Campos e Paulo de Figueiredo Pires. Rio de Janeiro: UFRJ/IM, 2004. Dissertação (Mestrado em Informática).

The emergence of Web services technology represents a significant advance in the continuing evolution of e-business initiatives. In order to fully explore business opportunities provided by this paradigm, it is important to track its utilization. This task can be accomplished through the use of logging facilities. However, current Web logging initiatives do not address Web services usage. This dissertation presents and discusses the use of WSLogA, a Web services logging architecture, as a basis for business processes management in the Web environment. WSLogA provides information for business process investigations and supports services quality monitoring, addressing administrative issues like performance and availability. Besides, to better support fine-grained analyses, we propose a multi-perspective, flexible *Data Mart* infrastructure generated from Web services log data and other data sources.

SUMÁRIO

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

ÍNDICE DE APÊNDICES

CAPÍTULO 1

INTRODUÇÃO

Para se manter competitiva, uma empresa precisa ser ágil e adaptar seus processos de negócio à dinâmica do mercado. Um processo de negócio é um elemento fundamental para o sucesso de qualquer organização. De acordo com a norma ISO8402-1994 (2000), um processo representa uma coleção de atividades de trabalho inter-relacionadas, iniciada em resposta a um evento, atingindo um determinado resultado para o cliente e demais interessados no processo.

As empresas reconhecem que a gerência dos processos de negócio desempenha um papel fundamental para que as estratégias de negócio sejam alcançadas (DAVENPORT, 1994). Essas empresas almejam, através da integração de processos de negócio, incrementar colaboração com parceiros e se adaptar rapidamente às mudanças dos requisitos de negócio.

Recentemente, um novo paradigma de desenvolvimento de aplicações denominado genericamente de *serviços Web*, firma-se como uma tecnologia inovadora para as empresas (BRUNNER *et al.*, 2002). Essa tecnologia apresenta uma série de vantagens, tais como: ubiqüidade, facilidade de uso, elevada reusabilidade, além de ser uma tecnologia não invasiva, isto é, os serviços Web podem ser utilizados como uma camada de software que se localiza acima dos sistemas legados e não requer a alteração destes, o que facilita a integração interna e externa de sistemas.

Tradicionalmente, grandes empresas que realizam comércio eletrônico estudam a audiência do seu sítio e, geralmente, utilizam ferramentas de mercado para analisar os arquivos de logs de servidores de páginas. Esses logs, inicialmente concebidos para fins de

auditoria e segurança, são semanticamente pobres e, registram grosseiramente os rastros que os clientes deixaram durante a navegação. Os logs são capazes de registrar quais páginas, documentos e *links* foram acessados. Kimball e Merz (2000) apresentam que os estudos mais refinados do *clickstream* - rastros deixados pelos clientes – podem ser realizados através de *Data Marts* especialmente projetados para esta finalidade. O principal objetivo desses *Data Marts* é o aumento da compreensão do comportamento eletrônico dos clientes e parceiros de negócio, permitindo também personalizar a iniciativa de comércio eletrônico.

Hoje em dia os processos de negócio podem ser implementados através de composições de serviços Web (SMITH, FINFAR, 2003). Para que processos sejam gerenciados, as suas atividades devem ser monitoradas. Logo, mecanismos de log aplicados a serviços Web podem ser utilizados como ferramenta básica de monitoramento de processos de negócio baseados em serviços Web. Acreditamos que esse monitoramento pode beneficiar a gerência de processos de negócio, mas até o momento, os logs de serviços Web não são contemplados nos padrões estabelecidos pelos órgãos normativos como a W3C (*World Wide Web Consortium*) e a WS-I (*Web Services Interoperability Organization*) (BRITTENHAM, 2003). Com o objetivo de suprir esta lacuna, apresentamos um mecanismo capaz de registrar as informações envolvidas nas interações que ocorrem entre os serviços e seus clientes.

1.1 Caracterização do problema

Esta dissertação foi elaborada com o objetivo de solucionar o seguinte problema: como monitorar e posteriormente interpretar, segundo múltiplas perspectivas, processos de negócio baseados em serviços Web. O monitoramento não pode interferir no funcionamento dos serviços Web, seja pela indução de retardos no processamento ou qualquer tipo de alteração nas fontes dos serviços.

1.2 Hipótese

Esta dissertação se baseou na seguinte hipótese. É possível criar uma ambiente flexível o suficiente para se integrar a serviços Web e que, simultaneamente, seja capaz de capturar e registrar, em repositórios de logs, os diversos tipos de interações que ocorrem entre os clientes e os processos de negócio baseados em serviços Web? A premissa inicial desta hipótese está baseada na necessidade de não alterar o código dos serviços Web já existentes em uma empresa.

Como extensão desta hipótese, consideramos que os dados capturados pelos logs gerados podem ser utilizados para dimensionar um ambiente analítico de monitoramento e utilização de processos de negócio baseados em serviços Web.

1.3 Objetivo da dissertação

O objetivo desta dissertação é a elaboração de uma proposta tecnológica que será utilizada para capturar e armazenar as interações entre atividades de processos de negócio implementadas através de serviços Web.

Para alcançar este objetivo, realizou-se inicialmente um levantamento dos problemas resultantes das interações entre clientes e serviços Web. Esse levantamento teve como propósito destacar aspectos e requisitos fundamentais para a solução satisfatória dos problemas relativos à captura do comportamento eletrônico dos usuários de serviços Web.

Partindo desse estudo inicial, uma arquitetura baseada em intermediários foi especificada e posteriormente um protótipo foi codificado para validar a viabilidade de sua construção.

Para explorar as informações dos logs de serviços Web, utilizamos uma abordagem baseada na proposta de Kimball e Merz (2000) e Sweiger *et al.* (2002), onde um ambiente analítico multidimensional é utilizado para a exploração dos logs.

1.4 Organização da dissertação

O Capítulo 2 descreve um breve estudo sobre o estado atual da tecnologia de serviços Web.

O Capítulo 3 apresenta os principais problemas relativos ao monitoramento de processos de negócio. Ele também apresenta os principais requisitos de monitoramento da qualidade de serviços Web.

O Capítulo 4 trata da solução proposta nesta dissertação, apontando os módulos e aspectos mais relevantes da arquitetura WSLogA.

O Capítulo 5 apresenta detalhes da implementação da arquitetura WSLogA e descreve os mecanismos de captura e as estruturas dos arquivos de log de serviços Web.

O Capítulo 6 apresenta um cenário de monitoramento de processos de negócio baseados em serviços Web e o projeto estruturas multidimensionais que podem auxiliar gestores na investigação de alguns aspectos de qualidade de processos de negócio e serviços Web.

Por fim, o Capítulo 7 enumera os resultados alcançados e apresenta contribuições para trabalhos futuros.

CAPÍTULO 2

SERVIÇOS WEB

A tecnologia de serviços Web traz consideráveis benefícios para o desenvolvimento de aplicações, uma vez que propicia a agilidade requerida pelas empresas frente às rápidas mudanças no ambiente de negócios (CASATI 2000; HEUVEL 2001). Indiscutivelmente, a principal vantagem do uso dos serviços Web está na sua elevada interoperabilidade, obtida graças à adesão a protocolos padrão amplamente difundido na Web.

Os serviços Web se consolidam a cada dia como a base das novas iniciativas de negócios eletrônicos, pois permitem construir redes intra e interorganizacionais de aplicações colaborativas e distribuídas, onde os serviços Web, na forma de módulos auto-contidos, são descritos, publicados, localizados e dinamicamente invocados através de uma arquitetura orientada a serviços.

2.1 Definição de serviços Web

Segundo Austin *et al.* (2002) um serviço Web é uma aplicação auto-contida, identificada por um URI (*Uniform Resource Identifier*), cujas interfaces e ligações são definidas, descritas e localizadas por artefatos que utilizam a linguagem XML (*Extensible Markup Language*). Um serviço Web deve ser capaz de interagir com outras aplicações através da troca de mensagens XML utilizando os protocolos de comunicação padrão atualmente disponíveis na Internet.

Os serviços Web podem estar associados a domínios de confiança. Por esse motivo, Kaye (2003) os classifica como serviços Web internos e externos. Os primeiros estão

relacionados a um único domínio, geralmente a própria empresa. Os serviços Web externos estão, normalmente, conectados a mais de um domínio ampliando as fronteiras da condução de negócios através da Internet. Em última análise, refletem processos de negócio dos parceiros através da infra-estrutura da rede, além de possibilitar a interoperação de sistemas produzidos por fabricantes distintos.

De acordo com Kaye (*l.c.*) e Gottschalk *et al.* (2002) os serviços Web apresentam inúmeros benefícios, dentre os quais destacamos: independência de plataforma de hardware e software; baixo acoplamento devido à elevada granulosidade dos módulos e reusabilidade dos módulos característicos que aumenta a velocidade de integração destes. Finalmente, a ubiqüidade, a padronização e a escalabilidade dos serviços Web são diferenciais importantes quando comparados a outras tecnologias, como por exemplo, CORBA, DCOM e RMI (BARRY, 2003).

2.2 Arquitetura de serviços Web

Os serviços Web são descritos como uma arquitetura baseada em serviços, também conhecida por SOA (*Service Oriented Architecture*), termo inicialmente definido pela IBM (GRAHAM *et al*., 2001; CHAPPEL, JEWELL, 2002; GOTTSCHALK *et al.*, 2002).

Os componentes da arquitetura SOA representam uma coleção de serviços que se comunicam através da troca de mensagens XML. Nesta arquitetura estão definidos três papéis que interagem entre si. Os papéis são:

1. **Provedor de serviço** – responsável pela descrição e publicação de um determinado serviço Web no registro dos serviços. O provedor também é responsável por descrever as informações de ligação do serviço usadas para sua chamada. As informações estão representadas em um documento XML

escrito na linguagem padrão WSDL (*Web Service Description Language*) (WSDL, 2001);

2. **Consumidor do serviço** – responsável por descobrir um serviço, obter a sua descrição e, usá-lo para se ligar a um provedor a fim de invocar um serviço Web;

3. **Registro dos serviços** - mantém um diretório com informações sobre serviços, como por exemplo, nome, provedor e categoria. O padrão adotado na SOA para registro é o UDDI (*Universal Description, Discovery and Integration*) (UDDI, 2001).

A interação entre os três papéis envolve: a *publicação* da informação sobre um determinado serviço, a *descoberta* dos serviços disponíveis e a *ligação* entre esses serviços (KREGER, 2001; GOTTSCHALK *et al*., 2002). A figura 2.1 ilustra como os papéis e as interações estão relacionados.

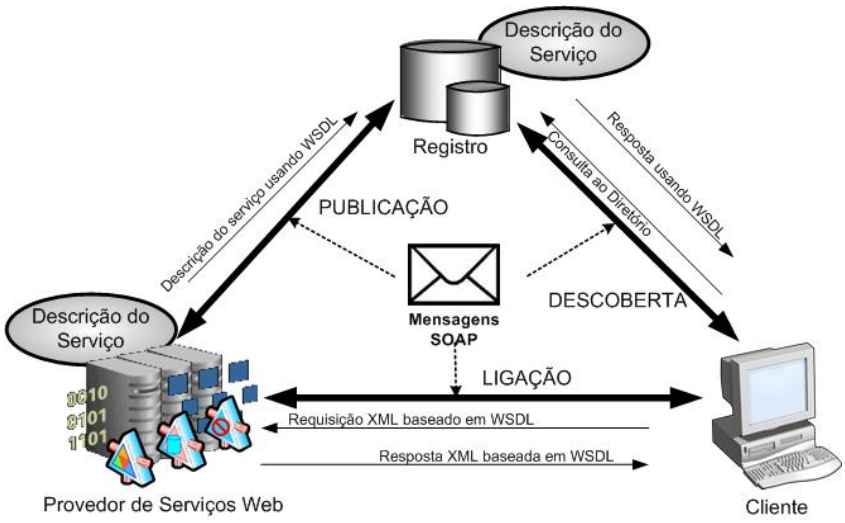


Figura 2.1 - Modelo de serviços Web, papéis e suas interações

Um dos maiores benefícios da arquitetura SOA é o desacoplamento das requisições do cliente ao serviço Web. A arquitetura ainda apresenta outras vantagens, dentre as quais se destacam o suporte para diferentes tipos de cliente; a elevada manutenibilidade; a facilidade

de reuso; a escalabilidade e disponibilidade. Essas vantagens contribuem decisivamente para a integração de processos de negócio intra e interorganizacionais além da capacidade de compartilhar e reutilizar serviços e recursos.

2.3 Classificação dos componentes dos serviços Web

Curbera, Mukhi e Weerawarana (2001) afirmam que a arquitetura de serviços Web é uma plataforma que tem por objetivo integrar aplicações na Web. Os autores usam como critério de classificação a *funcionalidade das especificações*. De acordo com este critério, a plataforma de serviços Web é classificada em quatro conjuntos de especificações que têm em comum o uso da linguagem XML. Esses conjuntos são:

1. **Descrição de serviços** - utilizada para definir as operações, mensagens e os tipos de dados de um serviço, também mantendo as informações sobre como acessar os serviços;
2. **Publicação e descoberta de serviços** - contém os protocolos que possibilitam a localização da descrição dos serviços.
3. **Descrição de composição de serviços** - contém os modelos e linguagens utilizadas para descrever como se dará a interação dos serviços;
4. **Protocolos de comunicação** - utilizados para definir, estabelecer e manter a comunicação entre as aplicações, também contendo a descrição dos formatos das mensagens utilizadas no estabelecimento da comunicação entre aplicações.

Os quatro conjuntos apresentam uma característica comum, que é o uso da linguagem XML, padrão aberto e extensível que permite a integração e troca de dados entre componentes distintos, garantindo a interoperabilidade necessária para a arquitetura.

Discutiremos os principais elementos da categoria de componentes padronizados nas próximas seções.

2.4 Descrição de serviços

A especificação WSDL (*Web Services Description Language*), atualmente na versão 1.2, é uma linguagem baseada em XML que descreve de forma padronizada e independente de plataforma como e onde os serviços podem ser conectados e utilizados através da rede (CHINNICI *et al*., 2003).

Um documento WSDL possui o mesmo papel que a IDL da OMG nos ambientes distribuídos (WSDL, 2001), representando um contrato entre o provedor de serviços e seus clientes e definindo os serviços como uma coleção de portas na rede. Os documentos WSDL apresentam uma dicotomia entre a definição da mensagem (*parte abstrata)* e sua implementação (*parte concreta)*, o que permite o reuso das definições das mensagens e *port types* (COYLE, 2002).

Um documento WSDL é composto por sete elementos XML que representam as partes abstrata e concreta.

Na parte abstrata temos quatro elementos:

1. **types** - fornece a definição dos tipos de dados para descrever as mensagens trocadas entre aplicações, normalmente representadas por um documento XSD (*XML Schema Definition*);
2. **message** - representa a informação que será trocada através das definições dos tipos de dados;
3. **porttype** - é um conjunto de operações suportadas por um ou mais *end points*, onde cada operação se refere a uma mensagem de entrada, saída ou erro;

4. **operation** - descreve a ação suportada pelo serviço.

Na parte concreta temos os outros três elementos:

1. **binding** - define uma especificação de protocolo e formato de dados para as mensagens definidas em um *port type*;
2. **port** - é um *end point*, representa a combinação de um *binding* e um endereço de rede;
3. **service** - é uma coleção de portas. Cada elemento *port* se relaciona com um elemento *binding* particular, indicando qual interface e qual protocolo de comunicação estão sendo utilizados nessa implementação.

2.5 Publicação e descoberta de serviços

A especificação UDDI, atualmente na versão 3.0, especifica mecanismos para a publicação, descoberta e integração de serviços. Tais mecanismos definem as estruturas de dados necessárias para sua descrição e classificação, bem como uma interface baseada no protocolo SOAP que permite o acesso a essas informações (BELLWOOD *et al*., 2002).

Resumidamente, o registro de serviços UDDI possui dois tipos de cliente. O primeiro envolve as aplicações que desejam publicar serviços e suas interfaces, o segundo tipo envolve os clientes que desejam obter e se ligar a serviços Web.

Conceitualmente, o protocolo UDDI apresenta três papéis, representados sob a forma de XML *Schemas* (NEWCOMER, 2002). Os papéis são:

1. **Páginas Brancas** - contém identificadores sobre o contato técnico do serviço oferecido;
2. **Páginas Amarelas** - contém informações genéricas sobre os tipos e localização dos serviços disponíveis;

3. **Páginas Verdes** - contém informações técnicas sobre um determinado serviço Web.

A especificação UDDI define quatro estruturas de dados, também descritas como documentos XML, onde cada elemento descreve o tratamento dado ao serviço Web, (KREGER, 2001; BELLWOOD *et al*., 2002; NEWCOMER, 2002).

1. **businessEntity** - estrutura de alto nível (páginas brancas) que contém, para cada serviço, as informações (nome, categoria, identificadores, entre outros) sobre a organização que publicou o serviço Web;
2. **businessService** - contém informações descritivas sobre serviços Web (páginas amarelas), tais como nome e descrição do serviço publicado;
3. **bindingTemplate** - contém informações técnicas sobre o serviço Web tais como forma de acesso e endereços dos pontos acesso ao serviço (páginas verdes);
4. **tModel** - mecanismo usado para a troca de definições abstratas (metadados) sobre um serviço Web, contém as descrições do serviço Web, opcionalmente aponta para documentos WSDL.

2.6 Descrição de composições de serviços

A composição de serviços Web se relaciona com a automação de processos de negócio e requer uma especificação que controle a troca de mensagens entre os serviços Web. Para Wohed *et al.* (2004), as especificações que descrevem a composição de serviços Web devem seguir uma série de princípios:

1. Estar em consonância com as demais especificações da W3C;
2. Permitir a composição de serviços mais complexos a partir de serviços Web mais simples;

3. Ser capaz de manter alta independência entre serviços com baixo acoplamento;
4. Facilitar a implementação de processos de negócio e a comunicação entre seus serviços; definir serviços obrigatórios e opcionais;
5. Indicar a ordem de execução e as rotas que serão seguidas no caso de sucesso ou falha na execução de um dado serviço Web.

Inicialmente, existiam duas especificações de descrição de composição de serviços: a WSFL (*Web Services Flow Language*) (LEYMANN, 2001) e a XLANG (THATTE, 2001). Entretanto, elas foram substituídas pela especificação BPEL4WS (*Business Process Execution Language for Web Services*). BPEL4WS une características das linguagens XLANG e WSFL tornando-se mais completa que suas antecessoras (ANDREWS *et al.*, 2003).

A especificação BPEL4WS, atualmente na versão 1.1, possui elementos que possibilitam a descrição de *workflows* baseados na interação de serviços Web. A linguagem também define um modelo e uma gramática para a descrição do comportamento de um processo de negócio e ainda define mecanismos para o tratamento de exceção (ANDREWS *et al*., 2003).

2.7 Protocolos de comunicação

A especificação SOAP (*Simple Object Access Protocol*) é um protocolo de transporte que rege a troca de mensagens entre aplicações em ambientes distribuídos e descentralizados. Seu conteúdo é composto por informações e estruturas de dados (KREGER, 2001; SEELY, 2001; COYLE, 2002).

A especificação SOAP 1.2 contém três partes principais (MITRA, 2003):

1. **Modelo de empacotamento** - *envelope SOAP* - define o conteúdo da mensagem SOAP;
2. **Mecanismo de serialização** - *conjunto de regras de codificação* - define os tipos de dados utilizados na aplicação;
3. **Mecanismo de comunicação** – *convenção* - define as chamadas e respostas através de procedimentos remotos.

Toda mensagem SOAP é um documento XML. Ela obrigatoriamente contém os elementos <*Envelope*> e <*Body*> e opcionalmente os elementos <*Header*> e <*Fault*>.

1. O elemento <*Envelope*> é a raiz do documento XML e representa a mensagem propriamente dita;
2. O elemento <*Body*> contém a carga de informações (operações e parâmetros) que são entregues ao destinatário da mensagem Este elemento, de acordo com a especificação SOAP, pode conter um elemento <*Fault*>, que, quando presente, pode ser utilizado no processamento de falhas do serviço Web. O elemento <*Fault*> descreve erros de chamada de métodos remotos ou mantém informações acerca do tipo de erro. Portanto, ele conta com os elementos <*FaultCode*>, <*FaultActor*>, <*FaultString*> e <*Detail*>;
3. O elemento <*Header*> expande uma mensagem SOAP. Ele define algumas características opcionais e acordos negociáveis entre as partes; seu conteúdo deve ser aceito pelas aplicações que estiverem se comunicando.

A especificação SOAP, a rigor, não depende do protocolo HTTP. Apesar de ser a combinação mais amplamente utilizada nos dias de hoje, a SOAP pode ser utilizada por outros protocolos da camada de transporte como, por exemplo, FTP, SMTP (COYLE, 2002).

Newcomer (2002) considera a especificação SOAP um tipo de extensão do protocolo HTTP, pois ela envia e recebe mensagens XML através das operações requisição e resposta do protocolo HTTP. Para que este processo ocorra, é necessário utilizar um servidor de páginas que seja capaz de atuar como um processador SOAP.

A figura 2.2 ilustra uma mensagem SOAP sendo enviada através do método POST do protocolo HTTP v 1.1.

```
POST /HopliasBooks HTTP/1.1
Host: localhost
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: http://equipe.nce.ufrj.br/serra
<soapenv:Header>
  <soapenv:criptograph
      soapenv:mustUnderstand="0" xsi:type="xsd:string">yes
  </soapenv:criptograph>
  <soapenv:priority
      soapenv:mustUnderstand="0" xsi:type="xsd:string">high
  </soapenv:priority>
 </soapenv:Header>
 <soapenv:Body>
   <validateInfo
     soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <Username xsi:type="xsd:string">SergioSerra</Username>
    <BirthDate xsi:type="xsd:string">03/02/1965</BirthDate>
    <IDCardType xsi:type="xsd:string">CPF</IDCardType>
    <IDCardNumber xsi:type="xsd:string">45608</IDCardNumber>
   </validateInfo>
 </soapenv:Body>
</soapenv:Envelope>
```

Figura 2.2 – Mensagem SOAP contendo os elementos <*Envelope*>, <*Header*> e <*Body*>.

Na figura 2.2, o campo **SOAPAction**, é utilizado por servidores de páginas ou *firewalls* para filtrar ou rotear uma mensagem SOAP. Este campo pode ser nulo ou conter o nome de algum método SOAP. O cabeçalho da mensagem se refere a uma possível prioridade de execução, que ainda não foi estabelecida. Ele faz uso do atributo **mustUnderstand** que exige do receptor o suporte a transação solicitada pelo emissor da mensagem. Caso o receptor da mensagem não suporte a transação, ele originará uma mensagem SOAP de falha.

O corpo da mensagem possui uma operação definida pelo atributo **ValidateInfo** e um pequeno conjunto de parâmetros definidos pelos elementos <*UserName*>, <*Birthdate*>, <*IDCardType*>, <*IDcardNumber*>.

2.7.1 Intermediários SOAP

Podem existir diversos tipos de participantes em uma mesma transação SOAP, sendo que cada participante possui uma função conforme determinado na especificação SOAP. Nesta seção trataremos dos intermediários SOAP, já que eles desempenham papel central na proposta de arquitetura apresentada nesta dissertação.

Para Booth *et al.* (2003) um intermediário SOAP é um agente - programa que atua em benefício de uma pessoa, entidade ou processo - capaz de receber, enviar e retransmitir mensagens. Um intermediário é capaz de processar alguns aspectos de uma mensagem ou retransmitir mensagens para outros intermediários ao longo de uma determinada rota.

Um intermediário SOAP pode ser de dois tipos: *passivo* ou *ativo*. Os primeiros não alteram o conteúdo de uma mensagem SOAP, enquanto que outros são capazes de alterar exclusivamente o elemento <*Header*> de uma mensagem SOAP. A alteração se dá através da adição, alteração ou remoção de informações (COYLE, 2002; NEWCOMER, 2002; MITRA, 2003).

A alteração das mensagens SOAP ocorre sem que o remetente e receptor da mensagem tomem conhecimento e seus usos potenciais são: segurança, auditoria, manipulação de conteúdo e monitoramento (BOOTH *et al*., 2003).

De acordo com Coyle (2002) e Mitra (2003) a especificação SOAP v 1.2, permite a construção de intermediários que estejam em conformidade com o padrão de desenho *Pipe and Filter*, isto é, uma mensagem enviada por um emissor poderá ser interceptada seqüencialmente por um ou mais intermediários até que alcance seu destinatário.

Para Graham *et al.* (2001) existem três fatores que justificam a necessidade e a importância dos intermediários: domínios de confiança, escalabilidade e a capacidade de agregar valor ao negócio.

No âmbito desta dissertação, a capacidade dos serviços Web de agregar valor aos processos de negócio é fundamental. Os intermediários, por exemplo, podem ser utilizados no rastreamento das mensagens SOAP trocadas entre processos de negócio baseados em serviços Web, isto é, através do uso de intermediários é possível avaliar quantitativa e qualitativamente a troca de mensagens SOAP.

Do ponto de vista quantitativo é possível medir os volumes de chamadas de serviços Web, seus erros, tempos de execução e transporte das mensagens SOAP, entre outros. Do ponto de vista qualitativo é possível avaliar diversos aspectos de qualidade dos serviços Web.

2.7.2 Variações dos Intermediários SOAP

Para Brittenham *et al.* (2003) os intermediários SOAP podem ser construídos segundo algumas variantes arquitetônicas, tais como, *Man in The Middle* (MITM), *Proxy Interceptor* (PI) e *Same Machine Interceptor* (SMI).

A variante MITM monitora uma porta TCP/IP, intercepta mensagens SOAP e as repassa para os respectivos serviços Web. Como exemplos de implementação de MITM destacam-se as ferramentas: *tcpTrace*, (FELL, 2003) *tcpTunnelGUI* do Apache SOAP toolkit (SCHAFFNER, 2003) e *MSSOAP* do Microsoft SOAP Toolkit v.3, atualmente incorporado ao *framework* .NET (SOAP, 2004).

A principal vantagem desta variante quando comparada com as demais reside no fato de ser facilmente codificada em um ambiente de troca de mensagens RPC, isto é, ela requer somente conhecimentos básicos sobre como tratar leitura e escrita sobre *sockets* TCP/IP. A variação MITM, quando aplicada a arquiteturas que envolvem roteamento de mensagens

SOAP, apresenta alguns problemas, uma vez que o caminho percorrido pelas mensagens pode variar. Além disso, ela também apresenta problemas com relação às mensagens criptografadas, pois ela não pode ser capturada corretamente.

A variante PI funciona de modo análogo a um servidor Proxy HTTP. Ela filtra as mensagens HTTP/SOAP endereçadas para o serviço Web em uma única porta TCP. Como exemplo de implementação dessa variante destacamos a ferramenta *proxyTrace* de Simon Fell (FELL, 2003). Esta variante de intermediários é mais facilmente configurável que o MITM, porém, apresenta as mesmas limitações daquela abordagem, entre elas a dificuldade em tratar mensagens criptografadas, limitações em monitorar mensagens roteadas, além das restrições de funcionar sobre um único protocolo de rede.

A variante SMI impõe que o intermediário sempre esteja situado na máquina que recebe a mensagem de solicitação de execução do serviço Web. Para Brittenham *et al.* (2003), os intermediários SMI podem funcionar de duas formas distintas. A primeira forma envolve a mudança do serviço Web para interagir com uma porta TCP diferente da original. Por exemplo, um serviço Web que utilize a porta 80, deve ser movido para a porta 8080. O intermediário abre um *socket* para a porta 80, redireciona todas as solicitações para a porta 8080 e então registra todo o tráfego no arquivo de log. A segunda forma é mais utilizada. Ela requer que o intermediário monte e manipule uma pilha de mensagens, como por exemplo, nos serviços Web implementados com a tecnologia ASP.NET (FELL, 2003). Neste caso o intermediário deverá ser codificado sob a forma de um módulo HTTP que registrará todo e qualquer dado enviado ao serviço Web sem que seja necessário alterar o esquema de endereçamento do servidor.

A variante SMI também possui inconvenientes. Por exemplo, não há nenhuma garantia de que um remetente e um receptor tenham como sincronizar a geração e o armazenamento das mensagens nos arquivos de log. Conseqüentemente, uma mesma

mensagem pode ser registrada mais de uma vez nos logs de serviços Web. Esse fato acrescentará grande complexidade à ferramenta de análise, uma vez que ela terá que ser capaz de identificar e tratar mensagens duplicadas. Finalmente, é importante ressaltar que as variantes de intermediários utilizadas em serviços Web descritas na literatura não contemplam satisfatoriamente todos os requisitos necessários para a geração de logs de utilização de serviços Web.

No cenário atual, muitas empresas têm como objetivo gerenciar e monitorar seus processos de negócio e, com a popularização dos serviços Web como ferramenta de integração de infra-estrutura, esta tarefa pode ser facilitada. Neste sentido, os serviços Web desempenham um importante papel, pois apresentam a flexibilidade necessária para facilitar a troca de informações entre processos de negócio intra e inter-organizacionais.

Os processos de negócio, resumidamente, ditam como uma empresa coordena e organiza tarefas e informações para produzir um bem de valor, atualmente, existe uma forte tendência de implementar os processos de negócio sob a forma de serviços Web (LEYMANN *et al.*, 2001; SMITH, 2002).

No próximo capítulo apresentaremos os principais conceitos relacionados aos processos de negócio e seu gerenciamento, bem como as condições necessárias para o monitoramento dos processos de negócio baseados em serviços Web.

CAPÍTULO 3

MONITORAMENTO DE PROCESSOS DE NEGÓCIO

As empresas cada vez mais se preocupam em avaliar a qualidade de seus processos de negócio. Elas buscam maior competitividade, produtividade, eficiência e qualidade total. Para alcançar estes objetivos faz-se necessário trabalhar eficientemente tanto a modelagem quanto o monitoramento de processos de negócio internos ou externos.

Atualmente, no contexto da Web, apenas os processos de negócio internos mais simples (em ambientes B2C) são usualmente monitorados através de arquiteturas que utilizam os logs dos servidores de páginas, registrando as requisições de páginas Web (SWEIGER *et al*., 2002). Essas arquiteturas têm como principal objetivo capturar e registrar, em repositórios próprios, o comportamento navegacional (comportamento eletrônico) dos consumidores. As arquiteturas disponíveis na literatura carecem de mecanismos de monitoramento adequado para ambientes nos quais os processos de negócio são baseados em serviços Web. Processos de negócio implementados sob a forma de serviços Web necessitam de mecanismos capazes de monitorar sua utilização e o desempenho de forma a auxiliar analistas e gestores na tomada de decisões que envolvam a condução do negócio.

Com o objetivo de suprir essa deficiência, este capítulo tem como foco o levantamento dos requisitos de monitoramento de uma arquitetura que seja capaz de monitorar os processos de negócio, internos e externos, baseados em serviços Web.

3.1 Processos de negócio

O conceito de processos de negócio foi inicialmente incorporado à área de controle de qualidade e controle estatístico de processos por diversos autores (Usirono, 2003). Nos anos

90, o conceito foi aprofundado por Davenport (1994) e aplicado à área de Reengenharia de Processos. Desde então, as definições de processo variam sutilmente a cada autor analisado.

Davenport (1994) definiu processos de negócio como sendo conjuntos de atividades estruturadas de trabalho que geram resultados específicos para os clientes, isto é, cada processo constitui-se de uma série de atividades inter-relacionadas que cruzam as fronteiras de áreas funcionais com entradas e saídas próprias. De modo complementar, o autor também define algumas dimensões de desempenho que podem ser medidas e aprimoradas no que se refere ao custo, ao tempo, à qualidade e ao grau de satisfação do cliente.

De acordo com Porter (1998) um processo de negócio deve ter um objetivo definido além de entradas e saídas de dados específicas. Os processos utilizam recursos (computacionais ou não) e têm uma quantidade definida de atividades (também representadas como subprocessos) executadas segundo uma ordem, dependente de eventos e condições internas e/ou externas que acontecem durante o processo. Um processo de negócio é capaz de agregar valor para os clientes, sejam eles internos ou externos.

Como é possível constatar, as definições apresentadas anteriormente não diferem muito entre si. Nesta dissertação faremos uso do conceito estabelecido por Davenport (1994), pois além de ser o original, guarda muita semelhança com o conceito apresentado por Porter (*l.c.*). Além disso, Davenport (*l.c.*) ressalta a importância de monitorar o desempenho dos processos de negócio através de métricas apropriadas.

3.2 Gerenciamento de processos de negócio

Apesar do grande interesse das empresas pelo gerenciamento processos de negócio, é muito comum que sua média e a alta administração releguem a segundo plano tanto a modelagem dos processos, como a sua gerência. Para Gartner (2001, *apud* Smith 2002) a

gerência de processos de negócio é um caminho irreversível, pois o autor estima que aproximadamente 90% das grandes corporações utilizarão este conceito até 2005.

Segundo Smith e Fingar (2003) o gerenciamento de processos de negócio (do inglês BPM - *Business Process Management*) é um conceito que se refere a um conjunto de tecnologias utilizadas na construção de aplicações baseadas em processos de negócio, a exemplo de reengenharia de processos de negócio, EAI (*Enterprise Application Integration*), *Workflows* e mais recentemente serviços Web.

Tanto Smith, Fingar (2003) quanto Burlton (2001) consideram que as empresas precisam aprimorar sua gerência de processos de negócio para obter resultados satisfatórios no que tange à velocidade de resposta às mudanças dos requisitos de negócio, redução de custos operacionais e aumento da colaboração com parceiros de negócio, fornecedores e clientes.

Leymann (2001) foi um dos primeiros a associar serviços Web com BPM. O autor afirma que a tecnologia de serviços Web impulsiona o BPM isto é, graças à natureza tecnológica dos serviços Web é possível obter a agilidade requerida pelas organizações frente às necessidades apontadas por Smith e Fingar (*l.c.*) e Burlton (*l.c.*). Nesse contexto, os serviços Web são muito úteis, uma vez que podem promover a interoperabilidade de processos entre negócios internos e externos com alto grau de flexibilidade.

Padmanabhuni, Ganesh e Moitra (2004) ressaltam que existe na comunidade de BPM uma forte penetração de metalinguagens de descrição de processos de negócio baseadas em XML, tais como: BPML (*Business Process Modeling Language*), BPEL4WS e WSCI (*Web Service Choreography Interface*). Essas linguagens têm em comum a capacidade de definir a troca de dados e fluxos de controle entre serviços Web. Os serviços Web baseados em linguagens BPM tentam disponibilizar novas funcionalidades para aprimorar a utilização de serviços Web básicos e compostos voltados para a execução de processos de negócio. As

principais funcionalidades incorporadas são as execuções de processos, a manipulação de exceções e erros, a manutenção de estado dos processos e a orquestração de fluxos de dados e controles entre serviços.

3.3 Classificação dos processos de negócio

Os processos de negócio podem ser classificados de diversas formas e, sua classificação varia de autor para autor. Por exemplo, Llewelly, Armisted (2000) classificam os processos de negócio como processos de *operação* e de *suporte*. Segundo os autores os processos de operação relacionam-se com as atividades estratégicas da empresa, como, vendas, serviços, gestão da produção, cobrança e entregas. Os processos de suporte relacionam-se com gestão de recursos humanos, sistemas de informação, de finanças e gestão de ativos.

Para Lovelock, Wright (2001) os processos de negócio podem ser classificados como *processos externos de linha de frente (FrontOffice*) e *processos internos de retaguarda (BackOffice)*. Os primeiros envolvem as atividades de contato direto com o cliente, enquanto que os últimos envolvem atividades de apoio, sem o contato com os clientes. Nesta dissertação, seguiremos esta abordagem.

Os processos de negócio *internos* são formados por uma ou mais atividades internas e se relacionam exclusivamente com outros processos de negócio internos, sendo transparentes para os parceiros e clientes externos. Um processo de negócio *interno* pode ser materializado sob a forma de serviços Web básicos ou compostos utilizados por aplicações que rodam no contexto interno da organização.

Os processos de negócio *externos* podem agregar uma ou mais atividades internas e pelo menos uma atividade externa. Este tipo de processo pode se relacionar tanto com processos internos quanto com processos externos à empresa. Estes processos não são

transparentes para os parceiros e clientes, pelo contrário, fazem parte da estratégia de negócio da empresa. Um processo de negócio externo também pode ser implantado sob a forma de serviços Web, via de regra compostos, pois requerem controle de transação e algum grau de orquestração de serviços.

3.4 Ciclo de vida dos processos de negócio

Smith e Fingar (2003) complementam o conceito de processos de negócio apresentado por Davenport (1994). Os autores apresentaram o conceito de ciclo de vida de um processo de negócio. O ciclo de vida é composto por oito fases e está ilustrado na figura 3.1. Cada fase está associada a um determinado contexto. Os autores também ressaltam que a eficácia da gerência de processos de negócio repousa na compreensão de cada uma das fases.

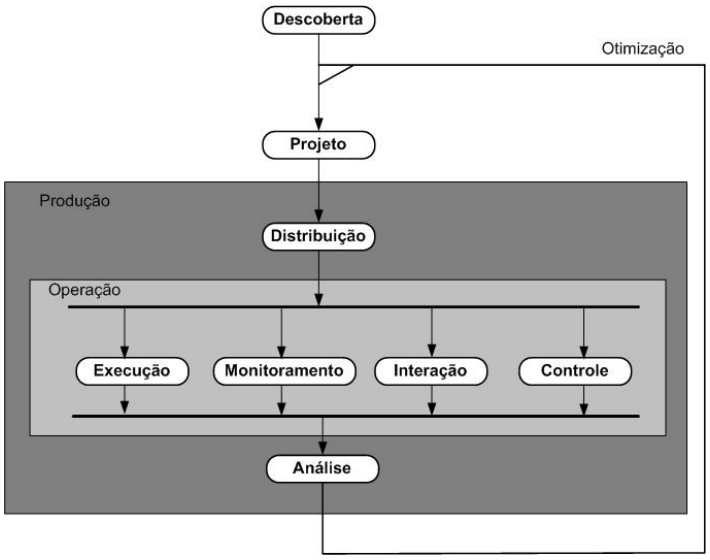


Figura 3.1 – Ciclo de vida de um processo de negócio, adaptado de Smith e Fingar (2003).

As fases de *Descoberta*, *Projeto*, *Interação* e *Distribuição* são as fundações para efetivar a implantação dos processos de negócio nas empresas. Entretanto, essas fases do ciclo de vida não serão discutidas nesta dissertação, pois não são as candidatas mais apropriadas para a captura e o registro da utilização nos processos de negócio. A fase de *Execução* implica

que um processo de negócio é consumido por todos os participantes (cliente, sistema computacional, ou mesmo outra organização) de uma cadeia de valor. Nesta fase controla-se o estado dos processos e a execução das transações.

A fase de *Análise* implica nas medidas do desempenho dos processos de negócio a fim de aprimorar estratégias e descobrir novas oportunidades que agreguem valor ao negócio, como por exemplo, avaliar os processos periodicamente para identificar ineficiências. A fase *Controle* se aplica tanto aos processos quanto ao gerenciamento das atividades que compõem um processo de negócio, ela focaliza o negócio e as intervenções técnicas necessárias para manter a integridade dos processos. O ciclo de *Otimização* relaciona-se com a melhoria de atividades dos processos de negócio, tal como, o suporte automático para a detecção de gargalos das atividades de um processo de negócio.

Dentre todas as fases, o *Monitoramento* é a fase que mais convém com objetivos desta dissertação. Ela está relacionada com a coleta de informações de execução dos processos de negócio. Desse modo, podemos usar arquivos de logs para registrar as informações que descrevem o funcionamento dos processos de negócio.

Posteriormente, esses arquivos de log podem ser processados e consolidados com outras fontes de dados. Essa abordagem abre perspectivas analíticas muito interessantes. Como exemplo, citamos as auditorias de segurança, exames de qualidade de processos, exame da qualidade dos dados, entre outros.

Apesar da importância do monitoramento de processos de negócio, na literatura existem poucas referências de ferramentas que desempenham este papel. Nossa proposta de arquitetura pode contribuir significativamente neste contexto, uma vez que ela é capaz de capturar e armazenar informações de utilização de processos de negócio baseados em serviços Web com diferentes níveis de granulosidade. Desta forma, ela pode ser acoplada, sem nenhum prejuízo, à fase de *Monitoramento* do ciclo de vida de um processo de negócio.

3.5 Camadas de monitoramento e métricas

Processos de negócio, suas atividades e serviços Web ocupam diferentes nichos dentro de uma organização. Cada nicho pode ser compreendido como uma camada de abstração distinta.

Partindo deste ponto de vista, definimos três camadas de monitoramento, onde cada uma permite uma visão do negócio. Para facilitar a compreensão do conceito, a figura 3.2 representa cada um destes níveis de abstração.

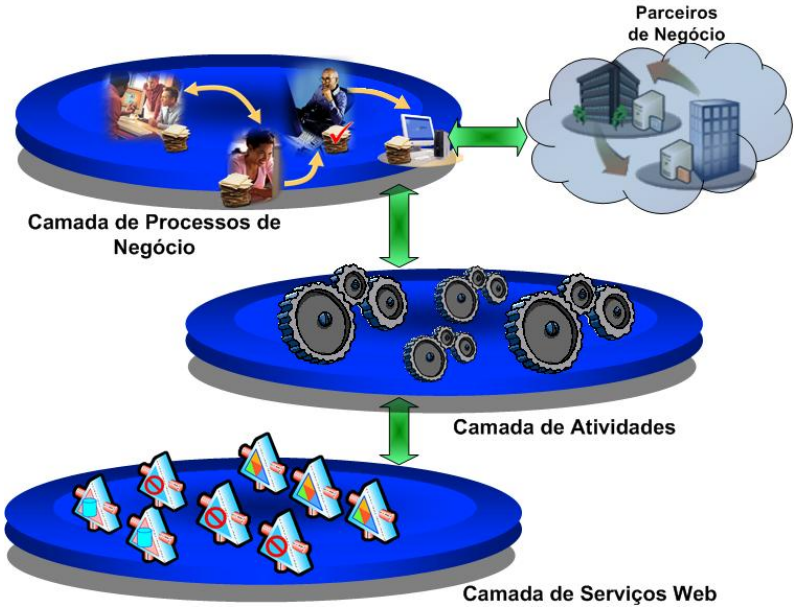


Figura 3.2 – Camadas de monitoramento

A *camada de processos de negócio*, de abstração elevada, abriga os processos de negócio definidos por Davenport (1994) e Porter (1998). Ela não possui mecanismos de log e relaciona-se diretamente com as atividades, sendo de especial interesse para os gestores de processos de negócio e para a alta gerência das empresas. As perspectivas analíticas viabilizadas por essa camada são interessantes, principalmente quando se busca compreender como os requisitos de negócio influenciam a dinâmica da composição de serviços Web. Através desta camada é possível avaliar processos de negócio internos e externos, isto é, a

camada oferece uma visão de alto nível, permitindo que analistas de negócio avaliem o desempenho dos processos de negócio. Esta camada pode ser desdobrada em outras camadas menos genéricas. Desta forma, pode existir uma camada para cada processo de negócio da organização.

A *camada das atividades*, de abstração intermediária, também é importante para as análises de processos de negócio, uma vez que ela permite a compreensão detalhada da dinâmica de cada processo. Através do estudo de cada atividade é possível detectar possíveis problemas nos processos de negócio e, ao mesmo tempo se pode avaliar o impacto dessas mudanças.

A *camada de serviços Web*, de baixa abstração, abriga os serviços Web propriamente ditos. Ele oferece uma visão operacional dos processos de negócio da empresa. De acordo com esse ponto de vista, é importante ressaltar que um provedor de serviços pode simultaneamente prover e consumir serviços Web. Logo, através desta camada é possível investigar o desempenho dos serviços, suas taxas de erros e possíveis causas. Ainda sob este ponto de vista, também é fundamental entender como os clientes (internos ou externos) consomem os serviços Web oferecidos. A partir desta perspectiva busca-se compreender ou até mesmo reconstruir o comportamento eletrônico dos clientes frente aos serviços Web e seus processos de negócio relacionados.

Para o provedor de serviços, que também pode consumir serviços Web de terceiros, é relevante definir requisitos funcionais e não funcionais de qualidade e, em função destes requisitos, empreender análises de custo e desempenho. O resultado dessas análises pode ter impacto direto sobre os processos de negócio internos e externos, uma vez que será possível avaliar e medir os custos operacionais e o grau de cooperação entre os parceiros de negócio.

Cabe ressaltar que as atividades de um processo de negócio são implementadas sob a forma de serviços Web. Portanto, existe um forte vínculo entre as camadas de serviços Web e

processos de negócio. Esse vínculo se refletirá na relação direta entre as análises associadas com essas camadas.

Estima-se que as visões oferecidas pelas diferentes camadas possam viabilizar análises que agreguem novas percepções do negócio, tanto no que se refere ao relacionamento com os clientes quanto no que tange à condução do negócio através das melhorias operacionais dos processos de negócio internos e externos. Todavia, para que seja possível definir os requisitos de monitoramento desejados, é importante estabelecer conjuntos de métricas para cada um dos níveis de monitoramento de processos de negócio e de serviços levantados esta seção.

O projeto de software é um processo que envolve riscos e decisões. Nestes projetos, as métricas são usadas para avaliar se as rotinas estão em conformidade com os objetivos pretendidos. Elas também auxiliam na elaboração de novas soluções que levem à melhoria da qualidade de um produto. Esse mesmo raciocínio pode ser empregado na definição das métricas das camadas de processos de negócio e de serviços Web.

As métricas usadas para analisar a camada de processos de negócio podem ser obtidas através de metodologias que avaliam o desempenho organizacional, como por exemplo, a *Balanced Scorecard* (KAPLAN, NORTON, 1992). Já as métricas para avaliar os serviços Web podem ser extraídas através do estudo dos quesitos de qualidade dos serviços Web (GRAHAM *et al*., 2002; MANI, NAGARAJAN, 2002; MENASCÉ 2002; CHATTERJEE, WEBBER, 2003; MENASCÉ, RUAN, GOMMA, 2004;).

Nesta dissertação, não exploraremos as métricas da camada de atividades. Esta decisão se deve à estreita relação de causa e efeito entre os processos de negócio e os serviços Web.

3.6 Avaliação do desempenho organizacional e monitoramento de processos

A avaliação do desempenho organizacional tem recebido crescente atenção nos últimos anos. No entanto, para avaliá-lo faz-se necessário utilizar modelos que permitam

quantificar, objetivamente, o desempenho de uma organização. Fernandes (2004) enumera quatro modelos de avaliação, dentre os quais se destacam:

1. Um modelo francês denominado *Tableau de Bord* que tem como viés principal as métricas financeiras;
2. Um modelo proposto por Brown (1996, *apud* Fernandes, 2004), que procura organizar as medidas de desempenho em função da seqüência de atividades de um processo de negócio, partindo dos insumos até chegar aos resultados;
3. Um modelo de avaliação integrada de desempenho proposto por Neely (2002, *apud* Fernandes, 2004). O modelo está baseado em um "prisma de desempenho" e tem como objetivo fornecer respostas aos gestores de processos, investidores e, em segundo plano, aos clientes;
4. Um modelo denominado BSC (*Balanced Scorecard*) proposto por Kaplan e Norton (1992). Ele fornece aos gestores informações abrangentes e essenciais sobre o desempenho organizacional, procurando equilibrar indicadores financeiros, de aprendizado e satisfação dos clientes.

Esta dissertação enfoca o último modelo, que segue a mesma filosofia dos anteriores, porém complementa-os em diversos aspectos, como por exemplo, incorporando novas medidas de desempenho que apontam relações de causa e efeito.

O BSC utiliza métricas de desempenho organizadas em torno de quatro perspectivas, a saber:

1. Perspectiva financeira;
2. Perspectiva dos clientes externos;
3. Perspectiva de aprendizado e da inovação;

4. Perspectiva dos processos internos.

A perspectiva financeira tem como principal objetivo agregar valores para a empresa. Nesta perspectiva, buscam-se métricas para a manutenção da imagem e reputação, através da redução de custos operacionais, aumento de receitas e melhoria da produtividade e, finalmente, garantia de que os projetos em curso agreguem valor à empresa.

A perspectiva dos clientes externos ocupa-se dos parceiros e mercados que a empresa se propõe a atender, as métricas têm como foco avaliar o relacionamento com os clientes através do aumento da satisfação, da retenção, aquisição de novos clientes e aumento da lucratividade e participação do mercado.

Na perspectiva de aprendizado e da inovação, o objetivo principal é a antecipação dos problemas, buscando-se identificar a infra-estrutura necessária para criar a longo prazo o aprendizado e alcançar o crescimento desejado.

A perspectiva dos processos internos tem como objetivos principais a produção eficaz e eficiente, buscando-se identificar os processos internos críticos que devem ser superados pela empresa ou unidade de negócio para que alcancem a excelência. Kaplan e Norton (1996) identificaram três níveis de processos de interesse nesta perspectiva:

1. Construir o negócio pela *inovação* de produtos e/ou serviços e conseqüentemente penetrar em novos mercados e segmentos;
2. Aprofundar o relacionamento com os clientes através do mecanismo de *pós-venda*;
3. Buscar excelência nas *operações* por meio da gestão do fornecimento, dos custos, da qualidade e do tempo.

3.7 Métricas de processos de negócio

Kaplan e Norton (1992; 1996) ressaltam que para avaliar cada uma dessas perspectivas, faz-se necessário a escolha de um conjunto de métricas para avaliar o desempenho organizacional. De modo complementar, Grembergen e Amelinckx (2002) também acrescentam a necessidade de métricas para avaliar os processos de negócio de projetos de *e-business* e apresentam uma metodologia para a definição de métricas BSC.

Neste sentido, faremos uso da metodologia BSC e discutiremos apenas as métricas que dizem respeito ao monitoramento de processos de negócio baseados em serviços Web sob as perspectivas dos processos internos, bem como algumas métricas relacionadas aos clientes externos, pois as demais perspectivas fogem ao escopo desta dissertação.

3.7.1 Métricas para a perspectiva dos clientes externos

Para a perspectiva dos clientes externos, definimos um conjunto de métricas de processos de negócio baseadas diretamente na interação entre clientes e os serviços Web; essas métricas podem ser chamadas de *métricas comportamentais*. As métricas comportamentais revelam informações sobre a interação dos clientes com a execução de serviços, suas invocações e erros. Essas métricas podem descrever o volume de chamadas e sua freqüência, tempos médios entre chamadas, entre falhas e de execução. As métricas comportamentais de processos de negócio também podem ser úteis para avaliar as invocações dos serviços ao longo do tempo. Com isso é possível avaliar quais serviços são mais ou menos invocados.

De forma individualizada, as métricas, oferecem informações relevantes. Entretanto, ao associarmos mais de uma métrica por vez, podemos encontrar resultados ainda mais atraentes. Por exemplo, o número absoluto de invocações de um dado serviço por dia pode levar o analista a tirar conclusões errôneas. Para que ele obtenha resultados mais confiáveis

sobre a utilização de um determinado serviço, esta métrica deverá ser avaliada em conjunto com outras métricas, como por exemplo, volume de falhas e o tempo médio entre falhas.

Na tabela 3.1 apresentamos exemplos que ilustram como os logs de monitoramento de serviços Web auxiliam gestores e analistas de negócio nas atividades de gerenciamento de clientes externos.

Métrica	**Medida**	**Resultado**	**Dados requeridos**
Satisfação do Cliente	Número de reclamações	Percentual de clientes que realizam novas compras	Número de reclamações, dados sobre pedidos e clientes.
Conservação do cliente	Número de compras realizadas pelo mesmo cliente	Percentual de novos clientes ou Taxa de crescimento da base de clientes	Dados sobre pedidos e clientes.
Aquisição de novos clientes	Número de compras novas	Percentual de arrecadação com novos clientes	Dados sobre clientes.

Tabela 3.1 – Métricas e análises de BSC na perspectiva dos clientes externos suportadas pelos logs de monitoramento de serviços Web

3.7.2 Métricas para a perspectiva dos processos internos

Sob a perspectiva dos processos internos também é possível definir um conjunto de métricas baseadas nas atividades e nos processos de negócio. Essas métricas podem ser chamadas de *métricas de negócio*. Elas estão diretamente relacionadas com as taxas de sucesso ou falha das atividades ou dos processos de negócio; em suma, elas se relacionam com a condução do empreendimento.

As métricas de negócio representam uma forma de medir o desempenho dos complexos processos de negócio. Elas possuem elevado grau de abstração quando comparadas com as métricas de cliente. O alto grau de abstração deve-se ao fato de que as

medidas são colhidas sobre as operações realizadas por serviços Web compostos, isto é, de certa forma essas métricas refletem a composição de serviços Web.

Com o intuito de estruturar as métricas de negócio, elas serão divididas em dois grupos: *métricas para avaliação da efetividade do negócio* e *métricas para avaliação dos processos de negócio*. O primeiro grupo se relaciona com a eficácia[1] e eficiência[2] dos processos de negócio, enquanto que o segundo grupo se destina a avaliar fatores que influenciam o negócio.

3.7.2.1 Métricas para avaliação da efetividade do negócio

As métricas para avaliação da efetividade do negócio medem os resultados obtidos através da execução de um processo de negócio. Por exemplo, a taxa de conversão de vendas pode ser calculada em função do número de invocações de um serviço Web dividido pelo número efetivo de transações de vendas. Outra medida de efetividade é a taxa de retorno do investimento de uma campanha de vendas. Neste caso, a métrica levará em consideração os custos financeiros envolvidos na campanha divididos pelo valor arrecadado. Outras análises de efetividade também podem ser realizadas, tais como a avaliação das possíveis causas de baixas taxas de conversão de vendas. Por exemplo, para avaliar as possíveis causas de baixas taxas de conversão de vendas é possível investigar se este resultado é originado por indisponibilidade de serviços Web providos por parceiros de negócio, erros intrínsecos ao processamento do serviço Web ou ainda, simples abandonos de carrinhos de compra.

As métricas para avaliação da efetividade do negócio auxiliam na investigação de como um fator de negócio influencia uma decisão de compra. Por exemplo, como o frete grátis influencia uma efetivação de uma compra, ou ainda, como o parcelamento com juros

[1] *Eficácia* significa realizar as coisas certas, pontualmente e com os requisitos de qualidade especificados. A eficácia está na saída do processo de negócio.
[2] *Eficiência* se refere ao consumo de recursos. A eficiência é medida na entrada do processo de negócio.

através de cartão de crédito afeta a lucratividade das vendas de um determinado item de produto.

Para análises ainda mais detalhadas sobre um dado processo de negócio, apenas o simples cálculo de taxas de conversão pode não ser suficiente para avaliar o sucesso ou fracasso de uma campanha de vendas. Faz-se necessário utilizar técnicas de análise multidimensional para investigar se uma baixa taxa de retorno de investimento ou taxa de conversão é originada por problemas de divulgação ou definição incorreta do público-alvo ou, ainda, falhas dos serviços Web.

3.7.2.2 Métricas para avaliação dos processos de negócio

As métricas para avaliação dos processos de negócio medem a evolução do empreendimento através do monitoramento das atividades dos processos de negócio. Elas também são métricas de alto nível de abstração e são capazes de medir diferentes atividades de um processo de negócio. Elas se aplicam, por exemplo, ao processo de negócio venda *on-line* que é composto por uma série de atividades; ele tem início com a busca de um item, em seguida, ele é adicionado ao carrinho de compras e por fim, ocorre a atividade de *checkout*. Cada uma dessas atividades pode ser implementada por um ou mais serviços Web e, neste caso, as métricas para avaliação de processos podem ser utilizadas para avaliar o grau de sucesso de conversão de vendas ou identificar gargalos no referido processo.

Na tabela 3.2, apresentamos exemplos que ilustram como os logs de monitoramento de serviços Web podem ser úteis na gerencia de processos de negócio.

Métrica	Medida	Resultado	Dados requeridos
Processos concluídos	Tempo de entrega	Número de reclamações de clientes	Dados de fechamento de pedidos e clientes.
Disponibilidade do sistema	Tempo médio de disponibilidade do sistema	Número de execuções do serviço de identificação	Número de visitantes, tempo total de disponibilidade do sistema.
Segurança e confiabilidade	Numero de erros críticos	Número de carrinhos abandonados	Quantidade de erros, quantidade de carrinhos abandonados.

Tabela 3.2 – Métricas e análises de BSC na perspectiva dos processos internos suportadas pelos logs de monitoramento de serviços Web

3.8 Métricas de qualidade de software

Muitos trabalhos têm usado a teoria das medições para métricas de software (FENTON, NEIL, 2000). No entanto, no contexto da Web as métricas não são facilmente percebidas. A percepção de quão bom é o serviço varia de usuário para usuário e, de acordo com Pressman (2000), desta percepção resultará a aceitação ou rejeição de uma atividade. Um ponto de interesse desta dissertação é buscar métricas originadas no comportamento do usuário para avaliar aceitação e uso dos serviços Web.

Olsina *et al.* (1999) e Menascé e Almeida (2002) concordam que as características mais relevantes para os serviços Web são: usabilidade, funcionalidade, confiabilidade, eficiência e manutenibilidade. Todavia, cada autor dá maior ênfase a aspectos diferentes para seus conjuntos de métricas. Olsina *et al. (l.c.)* argumentam que as métricas centradas na Engenharia de Software podem ser aplicadas a sítios e serviços Web. Já Menascé e Almeida (*l.c.*) fazem uso das métricas para medir desempenho, custos e disponibilidade de servidores de Web e de aplicações.

Existem poucas propostas de definição de métricas relacionadas especificamente com sucesso do sítio ou dos seus serviços. Para Cutler e Sterne (2000), definir métricas que

avaliem o sucesso de um sítio é uma tarefa difícil. Os autores afirmam que faltam parâmetros confiáveis voltados para determinar a reatividade dos usuários frente a um sítio. Neste caso as métricas são de grande importância, pois estão centradas no comportamento do usuário quando este interage com o sítio ou com os serviços por ele oferecidos.

Cutler e Sterne (*l.c.*) cunharam o termo *E-metrics* para definir as métricas *indiretas centradas no comportamento do usuário*. As *E-metrics* ou *clickstream metrics*, estão baseadas no ciclo de vida de um consumidor na Web, que é constituído por inúmeros estágios ilustrados na figura 3.3.

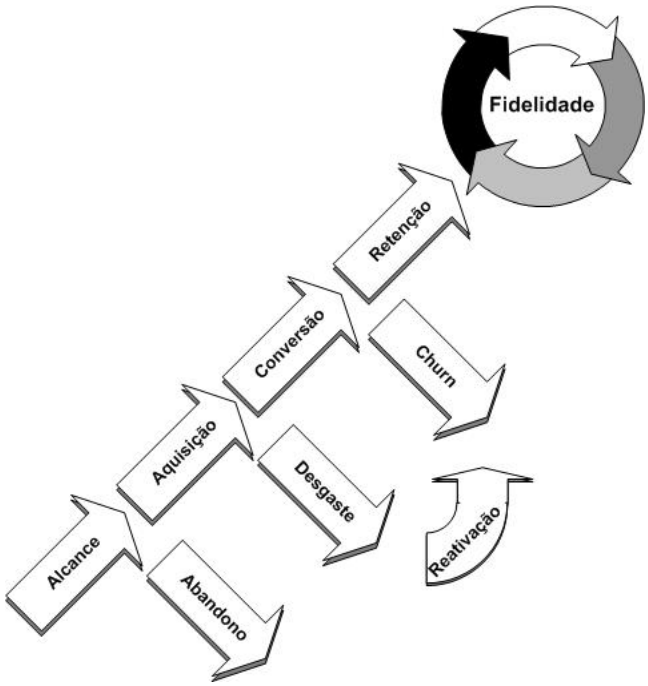


Figura 3.3 – Ciclo de vida de usuário Web segundo Sterne (2002)

O ciclo de vida de um consumidor se inicia na primeira visita de um cliente ao sítio e progride através de diversos estágios, a saber: Alcance, Aquisição, Conversão, Retenção e Fidelidade, quando então se estabelece a fidelidade do consumidor para com o sítio. Ao longo deste processo podem ocorrer fatores que interrompem o ciclo, tais como abandono, desgaste e "*churn*".

A tabela 3.3 apresenta as principais *E-metrics* definidas por Cutler e Sterne (2000).

Métrica	*Descrição*
Stickness	É uma medida composta que relaciona tanto a duração e a freqüência de visitas ao sítio. Captura a efetividade do conteúdo do sítio em termos da adesão dos usuários. Os autores ressaltam que é desejável que determinadas áreas do sítio apresentem alta *stickness.*
Slipperiness	É uma medida usada para medir partes específicas de um sítio, como por exemplo, a área de *checkout, login*, entre outras.
Focus	Mede o interesse do usuário em determinadas seções do sítio
Velocity	Mede a velocidade com que o usuário muda de estágios no ciclo de vida do consumidor.
Seducible Moments	Mede a suscetibilidade de um potencial consumidor em aceitar uma oferta.

Tabela 3.3 – Métricas centradas no comportamento do usuário.

3.9 Métricas para avaliação de qualidade de serviços Web

Os serviços Web podem ser avaliados de diversas formas, inclusive através do monitoramento da qualidade de serviços (*QoS*). Nesta seção discutiremos o que é *QoS*, seus principais quesitos e métricas.

Segundo a norma ISO 8402-1994 (2000), *qualidade* é um conjunto de funcionalidades e características de um produto ou serviço que lhe permite satisfazer necessidades implícitas ou pré-estabelecidas.

Este conceito é muito amplo e um tanto quanto vago, podendo ser aplicado a diversas áreas do conhecimento. Em computação, a *QoS* é utilizada para avaliar uma gama de recursos, sendo freqüentemente empregada nas áreas de redes de computadores, multimídia, sistemas operacionais, distribuídos e de missão crítica. Mais recentemente, este conceito também vem sendo amplamente aplicado nas áreas de comércio eletrônico e serviços Web.

A *QoS* é um aspecto que pode variar de acordo com a natureza do serviço Web, isto é, para algumas aplicações, severos quesitos de qualidade são pouco significativos, enquanto que para outras aplicações, estreitos níveis de qualidade de serviço são condições *sine qua non* para que o serviço seja consumido. Por exemplo, os serviços Web de previsão do tempo não requerem maiores garantias de qualidade de serviço, exceto pela correção da informação,

enquanto que os serviços de cotação de ações das bolsas de valores requerem níveis pré-determinados de qualidade como disponibilidade, desempenho e correção, caso contrário, consideráveis perdas financeiras podem ocorrer.

A forma mais usual de aferir a qualidade de serviços é através do uso de *frameworks* de monitoramento, mesmo assim estes sofisticados engenhos requerem a definição clara de quais quesitos de qualidade devem ser avaliados.

3.10 Quesitos de qualidade de serviços Web

Até o momento, não há um padrão claro para o conjunto de quesitos de descrição de *QoS* para serviços Web. Entretanto, existem várias iniciativas para tratar esta lacuna. Neste sentido destacam-se as contribuições de Menascé (2002), Mani e Nagarajan (2002), Graham *et al*. (2002), Chatterjee e Webber (2003), Menascé, Ruan, Gomma (2004). A tabela 3.4 apresenta os principais quesitos levantados por cada autor.

Autor	Quesitos
Menascé (*l.c.*) ,Menascé, Ruan e Gomma (*l.c.*)	Disponibilidade, segurança, tempos de resposta e processamento, *throughput.*
Mani e Nagarajan (*l.c.*)	Disponibilidade, acessibilidade, desempenho, confiabilidade, regulação, integridade, segurança.
Graham *et al*. (*l.c.*)	Disponibilidade, acessibilidade, desempenho, confiabilidade, regulação, autenticação, integridade dos dados.
Chatterjee e Webber (*l.c.*)	Disponibilidade, acessibilidade, desempenho, segurança, confiabilidade, conformidade.

Tabela 3.4 - Lista de quesitos de *QoS*.

Ran (2003), em recente trabalho, aponta vários quesitos adicionais de *QoS* para serviços Web e classifica-os em quatro grupos: execução, transação, segurança, configuração e custo. A tabela 3.5 apresenta os quesitos segundo a classificação de Ran.

Grupo	Quesitos
Execução	Escalabilidade, capacidade, desempenho, disponibilidade, robustez, exatidão.
Transação	Integridade
Segurança	Autenticação, autorização, confidencialidade, responsabilidade, auditabilidade, não repúdio, codificação.
Configuração e Custo	Regulação, suporte a padrões, estabilidade, completude, custo.

Tabela 3.5 - Lista de quesitos de *QoS* de Ran (2003).

Bochmann *et al.* (2001) também ressaltam que existem muitos trabalhos na área de *QoS* para sistemas de comércio eletrônico, entretanto, esses trabalhos relacionam poucos quesitos de *QoS* e que geralmente estão voltados para medir a disponibilidade de recursos do sistema. Os autores classificam os quesitos de qualidade de acordo com dois grupos: quesitos de *QoS* percebidos pelo usuário e quesitos de *QoS* internos. O primeiro grupo contém os quesitos importantes do ponto de vista do usuário do serviço, enquanto que o segundo grupa os critérios importantes para avaliação do serviço propriamente dito.

A tabela 3.6 apresenta os quesitos segundo a classificação de Bochmann *et al.* (*l.c*).

Grupo	Quesitos
QoS percebido pelo usuário	Tempo de reposta, disponibilidade, *throughput.*
QoS interno	Tempo de latência, Capacidade de acesso à rede, *throughput* máximo da rede, tempo de execução de consulta, tempo de enfileiramento de requisição, parâmetros de desempenho de baixo nível (utilização de memória, CPU).

Tabela 3.6 - Grupos de quesitos de *QoS* de Bochmann *et al*. (*2001*).

Devido à variada classificação dos quesitos de *QoS* apontadas por esses autores, fez-se necessário definir um conjunto de quesitos e métricas de *QoS* a serem utilizados nesta dissertação. Estes quesitos serão particularmente úteis nas discussões que envolveram as análises discutidas no capítulo 6.

A lista a seguir apresenta os principais quesitos disponíveis na literatura.

1. **Capacidade** – indica o limite máximo de atendimento de requisições concorrentes com garantia de desempenho. A métrica utilizada é uma medida direta que representa a capacidade de atendimento;

2. **Desempenho** – é a medida da velocidade de execução de uma requisição de serviço. Este quesito faz uso das métricas, tempo de resposta, *throughput* e latência que são medidas diretas. O tempo de resposta indica o tempo requerido para a execução de um serviço. Segundo Gunther (2000), o desempenho está relacionado com o quesito capacidade. O *throughput* indica o número de requisições de serviços completos em um dado período de tempo. Essa métrica está relacionada com a latência e a capacidade de um serviço. A métrica latência indica o intervalo de tempo transcorrido entre a chamada de um serviço e a resposta da chamada;

3. **Confiabilidade** – indica a habilidade de um serviço executar suas operações sob determinadas condições em um dado intervalo de tempo sem que ocorram falhas. A confiabilidade pode ser avaliada através das métricas tempo médio entre falhas (MTBF), tempo médio de falha (MFT), sendo que todas são medidas diretas. Segundo Gunther (*l.c.*), este quesito está relacionado com a capacidade de processamento de requisições;

4. **Disponibilidade** – indica a probabilidade da predisposição para execução imediata de um serviço. Baixas probabilidades indicam que a escolha do serviço pode não ser conveniente, isto é, baixa disponibilidade indica limitada utilidade do serviço. A disponibilidade é uma medida direta, entretanto sua interpretação pode ser cercada de polêmicas, pois o valor obtido varia em função de uma série de fatores ambientais, dentre os quais se destacam a capacidade de um servidor, tipo de rede, forma de implementação do serviço, versão de protocolos de rede, entre outros;

5. **Robustez** – indica quanto um serviço pode funcionar corretamente na presença de entradas de dados inválidas, incompletas ou conflitantes;
6. **Escalabilidade** – indica a habilidade de aumentar a capacidade computacional do provedor de serviços de modo que o sistema seja capaz de processar mais operações ou transações em um determinado período de tempo;
7. **Exatidão** – define a taxa dos erros produzidos por um serviço em função do número total de chamadas;
8. **Autenticação** – indica a garantia de que tanto o cliente quanto o serviço Web possam ser identificados;
9. **Autorização** - indica o grau de acesso do cliente a um dado serviço Web;
10. **Não-Repúdio** – expressa garantias de que, uma vez enviada uma mensagem, o remetente não pode negar seu envio;
11. **Confidencialidade** - é a garantia de que terceiros não podem ter acesso às informações trocadas entre o cliente e o serviço Web;
12. **Regulação** - Indica se o serviço está em conformidade com as leis, padrão de serviços Web e acordo por níveis de serviços (SLA) estabelecidos. Segundo Myerson (2002), SLA é um contrato formal entre um provedor de serviço e um cliente, garantindo desempenho de rede de acordo com alguns níveis e condições estabelecidas no contrato;
13. **Integridade dos dados** – é a garantia de que a mensagem não foi acidentalmente ou maliciosamente adulterada no caminho entre o cliente e o serviço Web;
14. **Estabilidade** – mede a freqüência de alterações de interface ou implementação de um serviço Web;

15. **Custo** – mede os custos monetários envolvidos na chamada de um serviço Web;
16. **Completude** – mede a diferença entre a especificação e a implementação de um serviço Web;

Os quesitos **Robustez**, **Escalabilidade** e **Exatidão** são medidos indiretamente, uma vez que sua avaliação também depende de fatores ambientais, isto é, eles dependem do ambiente computacional onde foram instalados.

Os quesitos **Autenticação**, **Autorização**, **Confidencialidade**, **Responsabilidade**, **Auditabilidade** e **Não Repúdio** estão relacionados características de segurança dos serviços Web, indicando níveis distintos de confiabilidade. Como a dissertação não é focada em segurança, o uso desses critérios não foi contemplado no Capítulo 6.

Os quesitos de qualidade de serviços Web bem como os relacionados a processos de negócio apresentados neste capítulo definem um conjunto básico de informações que podem ser obtidos a partir da arquitetura proposta nesta dissertação. O próximo capítulo apresenta os detalhes da arquitetura WSLogA e o ambiente necessário para a captura e geração dos logs de utilização serviços Web.

CAPÍTULO 4

A ARQUITETURA WSLogA

Este capítulo apresenta uma arquitetura, baseada em intermediários SOAP, denominada WSLogA (*Web Services Log Architecture*) que apresenta os mecanismos de captura e construção de logs de uso de serviços Web.

4.1 *Frameworks* de monitoramento de serviços Web

Na literatura existem poucos trabalhos que abordam a temática do monitoramento de serviços Web. Dentre eles destacamos os trabalhos de Dialani *et al*. (2002) e Cardoso, Sheth e Miller (2002). Nesses trabalhos, os autores apresentam *frameworks* baseados em *workflows* que monitoram os quesitos da *QoS* de serviços Web apontados por Mani e Nagarajan (2002). Entretanto, existe uma lacuna nesses trabalhos. Apesar de reconhecerem a importância dos arquivos de log, eles não definem nem como se dá a captura nem como é definida a estrutura do modelo de log de invocações de serviços Web.

Recentemente, Azevedo Junior (2003) propôs um *framework* denominado WebTransact-EM. Ele é usado para monitorar a execução dinâmica de serviços web semanticamente equivalentes. Embora sua proposta não seja, especificamente, voltada para avaliar muitos quesitos de QoS de serviços Web, o autor apresenta um modelo que contém critérios de qualidade e parâmetros de custo aplicáveis a serviços Web. O autor também faz uso de logs XML para monitorar a execução de serviços, contudo não é o foco desse trabalho

propor uma arquitetura flexível para monitoramento de processos de negócio em um ambiente de serviços Web.

A arquitetura WSLogA não foi, exclusivamente, concebida para ser um *framework* de monitoramento de qualidade de serviços Web. Ela pode auxiliar nesta tarefa, pois é capaz de capturar dados de utilização de serviços Web com diferentes granulosidades.

4.2 Papéis na arquitetura WSLogA

A fim de compreender a arquitetura *WSLogA* faz-se necessário definir os papéis dos produtores e consumidores de serviços Web. Por isso, representamos na figura 4.1 o modelo da interação de um processo do negócio baseado na composição de serviços Web, onde um consumidor interage com um ou mais serviços Web.

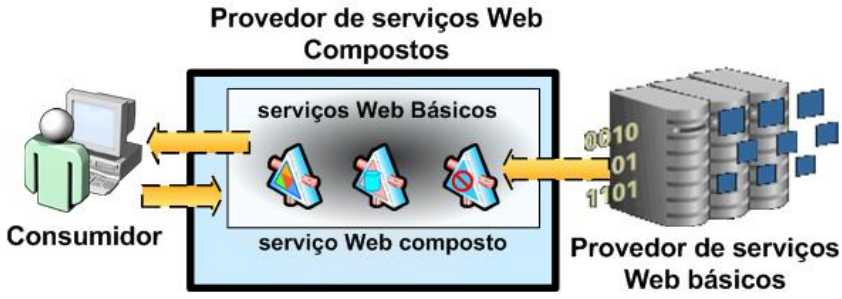


Figura 4.1 – Papéis da arquitetura WSLogA (Cruz *et. al*, 2004).

Na arquitetura WSLogA, existem três papéis:

1. **Provedor de serviços Web básicos** - este papel é desempenhado por um fornecedor de serviços Web. Os serviços Web básicos são aqueles que não são compostos por outros serviços, por isso, são considerados no escopo desta dissertação como serviços estanques e indivisíveis;

2. **Provedor de serviços Web compostos** - este papel é desempenhado por um fornecedor de serviços Web que agrega e publica os serviços Web básicos produzidos por um ou mais provedores de serviços Web;

3. **Consumidor de serviços Web** - este papel é desempenhado pelo elemento que consome um dado serviço Web. O consumidor pode ser uma aplicação local ou remota. O consumidor pode utilizar os serviços fornecidos tanto pelo provedor de serviços Web composto quanto pelo provedor de serviços Web básico;

Ao definir claramente os papéis na arquitetura WSLogA temos como objetivo principal não só reconhecer os principais atores envolvidos nas atividades de negócio da organização, como também, avaliar e medir a extensão das interações com os serviços Web nas camadas de monitoramento de processos de negócio e de serviços Web apresentadas na seção 3.5. Neste caso, para avaliar os serviços Web, temos como objetivo usar as métricas da seção 3.7.1(satisfação dos clientes) e também as métricas da seção 3.7.2 (melhoria da gestão de processos).

A gestão de processos de negócio, não é o objetivo inicial da arquitetura WSLogA, ela contudo, será materializada sob a forma de medidas de quesitos de *QoS* do serviços Web levantados no capítulo 3.

4.3 Módulos da arquitetura WSLogA

A arquitetura WSLogA foi inicialmente proposta por Cruz *et al*. (2003, 2004) e serve de base para esta dissertação. Nestes trabalhos, os autores ressaltam que existem ao menos duas alternativas para implantar um log de serviços Web.

A primeira envolve a mudança do código de fonte de cada serviço Web, onde um serviço Web poderá chamar um módulo ou componente responsável pelo processo de *logging.* Este módulo que pode ser codificado através de uso de *frameworks* de logs. Os *frameworks* de logs são pacotes de *software* que fornecem a infra-estrutura necessária para a criação de logs de aplicação. Normalmente, esses *frameworks* oferecem *interfaces* e *classes* que, necessariamente, devem ser incorporados ao código fonte das aplicações com a

finalidade de criar dos logs personalizados. Dentre os principais *frameworks* destacamos: JSDK - *Java Logging* (JSDK, 2001), l (L, 2004), IBM *Logging Toolkit for Java* - JLog (LOGJ, 2004).

Fizemos um estudo detalhado dos principais *frameworks* de logs, das suas características e limitações. Neste caso, observamos que eles simplificam a coleta de dados e geração dos logs, porém também agregam novos problemas, dentre os quais ressaltamos: sobrecarga adicional para gerar as mensagens de log; redução da legibilidade dos códigos fonte e conseqüente aumento do tamanho dos mesmos; baixa integração entre as diferentes frameworks, pois usam APIs específicas; ausência de padrão de logs; ausência de uma metalinguagem específica para a descrição do log, e não menos importante, todos os *frameworks* de log de aplicação exigem alterações nos códigos fontes das aplicações que se deseja monitorar.

No contexto de serviços Web, outras desvantagens devem ser consideradas como, por exemplo, o emprego de alguns desses *frameworks* ocasionaria interrupções nas operações dos serviços. Adicionalmente, nem sempre é possível acessar aos códigos fonte dos serviços Web desenvolvidos por terceiros. Finalmente, as alterações nos serviços Web além de consumir tempo estão sujeitas a introdução de erros.

A segunda alternativa, utilizada nesta dissertação, é menos invasiva, não requer nem a alteração de serviços Web, nem a interrupção da execução. Essa alternativa envolve o uso de intermediários SOAP. (COYLE, 2002; NEWCOMER, 2002; MITRA, 2003).

Esta alternativa confere simultaneamente à arquitetura WSLogA tanto a independência quanto a flexibilidade para a gerencia dos arquivos de log. Além disso, ela permite que a estrutura do log se ajuste a possíveis alterações nos requisitos de negócio sem a necessidade de qualquer tipo de intervenção no conjunto de serviços Web existentes.

A arquitetura WSLogA é composta por sete módulos principais, a saber: *provedores de serviços Web básicos e compostos, módulo de autoridade de testes, intermediários SOAP, arquivos de configuração e de parâmetros do log, parser do log do servidor Web* e *repositório XML de logs.* A figura 4.2 apresenta uma visão geral da arquitetura WSLogA.

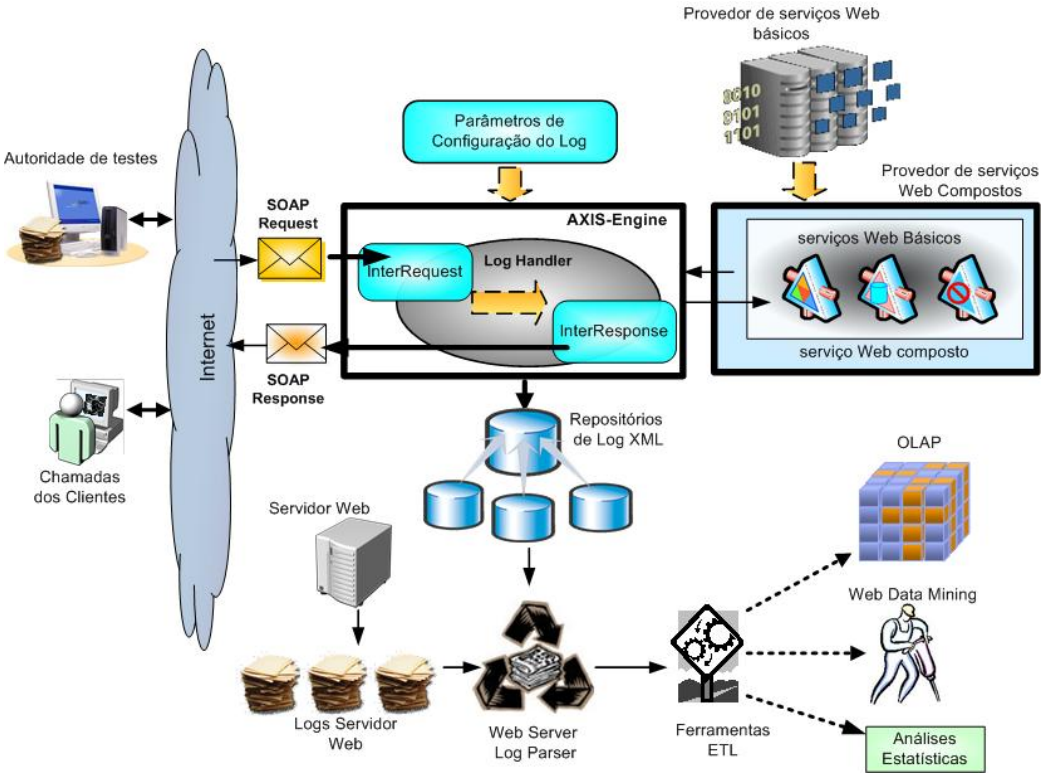


Figura 4.2 – Arquitetura WSLogA (CRUZ et. al, 2004)

4.3.1 Provedor de serviços Web básicos

O provedor de serviço Web básico é o módulo que fornece qualquer tipo de serviço Web que pode ter uma ou mais operações. Esse tipo de serviço é publicado pelos provedores de serviços Web básicos e podem ser utilizados por qualquer instância de um serviço Web composto.

Para avaliar a qualidade de serviços Web básicos é necessário escolher corretamente os requisitos de qualidade da seção 3.10. Estes requisitos devem ser escolhidos a partir de um contexto comum ao provedor de serviços Web básicos e ao consumidor de serviços.

Uma vez estabelecidos os requisitos de qualidade, a tarefa de medir um determinado aspecto do serviço Web torna-se simples, ela requer apenas um instrumento de medida e uma escala de valores para que as medidas dos valores aferidos possam ser compreendidas da mesma forma pelas partes.

Ao monitorar serviços Web básicos os provedores estão interessados em avaliar os quesitos de qualidade apresentados na seção 3.10 do capítulo anterior. Podemos citar como exemplo os quesitos da disponibilidade, do desempenho, da correção e da confiabilidade.

4.3.2 Provedor de serviços Web compostos

Provedor de serviços Web compostos é o módulo que faz a associação entre os diversos serviços Web básicos graças às características da arquitetura SOA. Os serviços Web compostos podem ser desenvolvidos por provedores de serviços independentes e os serviços Web básicos podem interagir entre si para executar uma determinada atividade mais complexa.

O serviço Web composto é publicado pelo provedor de serviços Web compostos e pode ser encarado como um processo de negócio. Eles têm por objetivo fornecer a infra-estrutura básica para construção e composição de novos serviços Web, de elevado valor agregado, a partir da composição de serviços Web básicos. A composição de serviços Web está estruturada como uma grande redoma que agrega múltiplos serviços Web básicos ou até mesmo outros serviços Web compostos, onde cada serviço interage de acordo com um determinado modelo de processos.

Os provedores de serviços Web compostos também têm interesse em aferir a qualidade de serviços Web compostos, entretanto, eles requerem quesitos qualidade adicionais quanto comparados ao provedor de serviços Web simples. Este tipo de provedor tem interesse de avaliar tanto o desempenho individual de cada serviço como também o

desempenho do serviço Web composto. Alguns dos critérios de qualidade de serviços apresentados o capítulo anterior que podem ser aplicados para os serviços Web compostos são a capacidade, o desempenho, a disponibilidade dos serviços compostos.

4.3.3 Módulo de autoridade de testes

O módulo de autoridade de testes é capaz de empreender chamadas identificadas. Ele é responsável por realizar os ensaios de teste dos serviços Web que se deseja monitorar. Os ensaios são invocações identificadas para serviços Web locais ou remotos que de tempos em tempos invocam um ou mais serviços Web. Através de análises destas chamadas, é possível verificar alguns atributos de *QoS* tais como a disponibilidade, a confiabilidade, os tempos de processamento e de resposta, entre outros.

Os ensaios, assim como as chamadas dos clientes do serviço, também são registrados no repositório de logs XML. Contudo, assim como nos logs de servidores de páginas, faz-se necessário distinguir os registros de chamadas reais dos registros das chamadas de teste. Essa prerrogativa está em acordo com os trabalhos de Casati, Ilnick, Lin (2000) e Casati, Shan (2001) onde os autores relatam que no ambiente de páginas Web, também existem chamadas de testes que são realizados pelos *Web robots* ou agentes.

Existem quatro razões para diferenciar as entradas dos ensaios das entradas dos serviços Web. Inicialmente, os provedores de serviço são reticentes quanto ao uso de ensaios não autorizados, uma vez que os testes são capazes de coletar informações sobre a lógica do processo de negócio. Em algumas situações um conjunto de ensaios poderia coletar, constantemente e sem autorização, informações sobre os lances de um serviço Web de leilão eletrônico.

A segunda razão diz respeito às interferências e distorções que tais entradas poderiam ocasionar nas análises de navegação e acesso a um dado servidor Web.

A terceira razão se relaciona com o uso indiscriminado de ensaios, o que pode ser muito oneroso uma vez que eles competem com os clientes pelos recursos de rede e dos servidores.

Finalmente, excessivos registros dos ensaios podem ser um indicativo de comportamento fraudulento, por exemplo, caso a cobrança ou consumo de um serviço seja calcada no número de chamadas por unidade de tempo, será necessário identificar e distinguir as chamadas de serviços das de teste.

4.3.4 Intermediário SOAP

O intermediário SOAP é o principal módulo da arquitetura WSLogA sendo posicionado entre o cliente e o provedor de serviços Web. Este módulo possui as funções a seguir:

1- O intermediário SOAP é capaz de receber e transmitir mensagens SOAP para serviços Web ou para outros intermediários. O intermediário obrigatoriamente deve preservar a integridade das mensagens trocadas entre os pares;

2 – O intermediário SOAP deve interceptar o fluxo de mensagens endereçadas a uma porta de comunicação, registrar a mensagem SOAP no repositório de logs XML e retransmitir a mensagem, sem alterações. Os arquivos de log podem se situar em diferentes pontos da arquitetura, isto é, no provedor de serviços Web básicos ou no provedor de serviços Web compostos. A localização de um arquivo de log pode implicar em problemas de segurança e desempenho que serão discutidos nas seções 5.3.1 e 5.3.2 do próximo capítulo.

Os intermediários da WSLogA são capazes de transpor domínios de confiança, uma vez que podem interagir com serviços Web e até mesmo outros intermediários posicionados em ambientes protegidos por *firewalls* ou redes virtuais privadas, ou mesmo em regiões

desprotegidas tais como a Internet. Essa flexibilidade deve-se ao uso de mensagens SOAP encapsuladas por mensagens HTTP em pacotes TCP.

Os intermediários também são flexíveis, isto é, sua operação leva em consideração não só a captura do conteúdo, cabeçalho e corpo das mensagens SOAP, como também capturar erros SOAP. O funcionamento dos intermediários é normalizado em função dos parâmetros indicados nos arquivos de configuração e parâmetros dos log.

Não menos importante é a capacidade que os intermediários têm de agregar valor aos processos de negócio de uma empresa. Por exemplo, os intermediários podem adicionar novos níveis de segurança na troca de mensagens SOAP em ambientes não confiáveis como, por exemplo, HTTP/SMTP. Essas características estão em sintonia com os objetivos do *Balanced Scorecard* apontados na seção 3.6.

Outra forma de agregar valor aos processos de negócio é a possibilidade de rastreamento das mensagens SOAP. Através do uso de intermediários é possível monitorar, diretamente, a fase de *Execução* do ciclo de vida de um processo de negócio. Neste caso é plausível monitorar quais os tempos consumidos em execução do serviço e no transporte fim a fim da mensagem.

Os intermediários SOAP da WSLogA se distinguem dos demais tipos de intermediários disponíveis na literatura, já que são capazes de capturar informações com diferentes níveis de granulosidade. Dados com baixa granulosidade apresentam elevado nível de detalhe, como por exemplo, os parâmetros passados através da chamada de uma operação de um serviço Web.

De forma inversa, dados de alta granulosidade, apresentam baixo nível de detalhe. Essa característica é de grande importância para o projeto do ambiente analítico proposto no Capítulo 6. A granulosidade da informação afeta tanto o volume de dados armazenados nos

logs quanto à natureza de consultas que podem ser executadas. Tomemos como exemplo um provedor de serviços Web compostos, que oferece um serviço de venda de livros. O conjunto de funcionalidades, de forma simplificada, é composto por dois serviços Web básicos, o primeiro consiste na validação das informações do cartão de crédito e o segundo é usado para consolidar a venda do livro. A figura 4.3 ilustra o conceito das diferentes granulosidades.

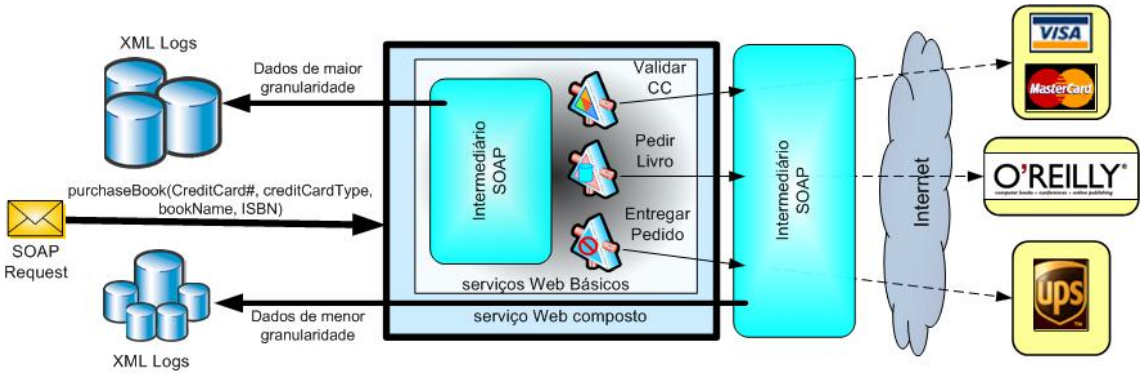


Figura 4.3 – Anatomia da execução de um serviço Web

As informações de elevada granulosidade permitem que o provedor de serviços Web compostos, por exemplo, investigue como está o volume de vendas dos livros de ação. Caso o volume de vendas esteja abaixo do projetado, é possível que ele estude as taxas de abandono das cestas de produtos.

As informações de reduzida granulosidade, aquelas não sumarizadas, permitem se façam estudos mais refinados, como por exemplo, o estudo dos motivos que levam ao abandono dos carrinhos de produtos.

As informações capturadas pelos intermediários SOAP, no exemplo citado anteriormente, são de grande utilidade quando se deseja monitorar as diversas perspectivas (apontadas na seção 3.6) de um processo de negócio baseado em serviços Web. Através dessas informações é possível monitorar alguns quesitos de qualidade de serviços Web, como também, apontar problemas relacionados à baixa disponibilidade do serviço ou problemas de

desempenho de um serviço Web. Adicionalmente, as informações podem ser úteis quando se consideram quesitos que indicam a efetividade do processo de negócio.

4.3.4.1 Requisitos de monitoramento dos intermediários SOAP

Os intermediários SOAP devem obedecer a um pequeno conjunto de requisitos funcionais para que executem suas funções de monitoramento de serviços Web. Um intermediário deverá:

1. Incorporar *listeners TCP/IP* concorrentes de tal forma que possam ouvir mais de uma porta TCP por vez. Um intermediário SOAP deve ser capaz de ouvir as portas TCP de número 80 e 8080, por exemplo;
2. Aceitar múltiplas conexões por porta; caso contrário ocorrerá um gargalo para atender as múltiplas requisições dos clientes;
3. Assegurar o mapeamento 1:1 das portas que estão sendo usadas para receber e retransmitir mensagens. Por exemplo, todas as mensagens recebidas na porta TCP 80 devem ser redirecionadas somente para a porta TCP 80 do provedor de serviços Web. A configuração das portas e seus mapeamentos devem ser carregados do arquivo de configuração dos logs;
4. Garantir a troca de mensagens inalteradas entre o cliente e o serviço Web enquanto perdurar a conexão TCP/IP.

4.3.5 Repositórios de logs XML

O repositório de logs XML consiste em um conjunto de documentos XML bem formados que reúnem as informações capturadas pelos intermediários SOAP. O repositório tem como principais objetivos armazenar as informações de utilização dos processos de

negócio baseados em serviços Web e, servir como uma das fontes de dados para a construção um ambiente de processamento analítico.

A arquitetura WSLogA prevê que sejam utilizados vários tipos de repositórios de dados, que podem ser classificados tanto com relação à estrutura de arquivo quanto à distribuição.

Com relação à estrutura, o repositório pode armazenar os dados em arquivos do tipo texto plano; sistemas gerenciadores de banco de dados relacionais, ou ainda em sistemas gerenciadores de bancos de dados XML nativo, porém cada tipo de repositório apresentará características tecnológicas peculiares que estão além do escopo desta dissertação.

Cruz e Campos (2002) relacionam algumas características dos arquivos de logs de páginas, seus tipos, estruturas e limitações. Os autores ressaltam que a proliferação de diferentes plataformas de software e hardware nas empresas cria desafios para a manutenção dos arquivos de log. Logo, fez-se necessário buscar uma solução genérica para a descrição e armazenamento de logs. A solução deverá ser a um só tempo simples, universal, de baixo custo e voltada para as diferentes perspectivas do monitoramento de sistemas. A fim de preencher esta lacuna, avaliamos algumas metalinguagens de descrição de log com o objetivo de padronizar uma sintaxe e uma semântica dos arquivos de log.

As metalinguagens de descrição de logs são especificações baseadas na linguagem XML e apresentam como vantagens a independência de domínio de aplicação e de plataforma e a extensibilidade do conteúdo.

Na literatura existem poucas metalinguagens de descrição de logs. Podemos destacar XLF (XLF, 1998), XLIF (XLIF, 2001), LOGML/XGMML (PUNIN, KRISHNAMOORTHY, ZAKI, 2001), DL Logs (GONÇALVES *et al*., 2002). Entretanto, nenhuma delas é suficientemente flexível para ser utilizada no contexto da computação orientada a serviços.

Para a arquitetura WSLogA, mais importante que deliberar sobre o tipo do repositório é definir qual será o conteúdo e o formato dos dados. Neste trabalho, optamos por armazenar os dados sob a forma de documentos XML bem formados e consolidados através de XML *Schemas*. Este formato permite registrar diferentes estruturas de dados cuja discussão será apresentada no próximo capítulo.

A arquitetura WSLogA, opcionalmente, permite a geração de logs compostos por fragmentos de XML. Segundo Grosso e Veillard (2001), fragmentos de XML são trechos de documentos XML bem formados que expressam inequivocamente uma informação através de um pequeno conjunto de elementos. Como vantagem os fragmentos XML apresentam facilidade de escrita, leitura e *parse*, além da simplicidade e portabilidade.

Trabalhar com fragmentos XML em detrimento de documentos completos pode ser desejável em alguns casos, uma vez que nem sempre é necessário manipular todo o documento XML (CARMEL *et al*., 2003). Como principal desvantagem para o uso de fragmentos XML nesta abordagem destacamos a dificuldade de contextualizar um fragmento XML de uma mensagem, por isso descartamos o uso deste tipo de documento na nossa implementação.

No que tange a distribuição dos arquivos de log de serviços Web, eles podem ser *centralizados* ou *descentralizados*. Nossa arquitetura também contempla a existência de um ou mais repositórios de dados, entretanto nosso protótipo fez uso exclusivo de repositórios de dados centralizados.

Na abordagem do repositório centralizado, fez-se uso de um repositório de dados único no provedor de serviços. Neste repositório são armazenadas todas as informações capturadas pelos intermediários SOAP locais.

Na abordagem do repositório descentralizado, fez-se uso de diversos repositórios de dados que podem estar localizados nos provedores de serviços Web simples, provedores de serviço Web Compostos ou até mesmo em sítios de terceiros que podem comercializar serviços de *logging* de serviços Web. Nestes repositórios são armazenadas as informações interceptadas pelos intermediários SOAP locais ou remotos.

4.3.6 Arquivos de configuração e parâmetros do log

Os arquivos de configuração e de parâmetros do log são arquivos XML que contém parâmetros que configuram o funcionamento dos intermediários SOAP e do módulo de autoridade de testes.

Os principais parâmetros de configuração do módulo de autoridade de testes são as especificações que definem quais os formatos de data e hora serão utilizados para gerar os logs, a freqüência de execução das chamadas de teste, o intervalo de tempo da janela de testes, além das especificações de quais serviços, métodos (operações dos serviços Web) e argumentos devem ser testados. Existe um conjunto de parâmetros que se relacionam com as falhas de uma mensagem SOAP, nesse caso, é possível filtrar quais os tipos de falhas serão capturadas.

Também podem ser definidas assinaturas que são associadas às chamadas de testes dos serviços Web. Essas assinaturas são elementos XML, do tipo string, definidos *a priori* pelo analista. As assinaturas são úteis para diferenciar, nos arquivos de log, as invocações de teste das invocações reais.

Outros parâmetros de configuração que auxiliam no funcionamento dos intermediários são: os nomes e regras de formação dos arquivos de log, portas TCP/IP que devem ser monitoradas, métodos HTTP filtráveis (GET, POST, entre outros), número máximo de conexões simultâneas, especificação dos arquivos *XML Schema* utilizados para validar os

arquivos de repositório de logs XML, intervalo de tempo de monitoramento, especificações de formatos de data e hora, entre outros.

Em suma, são os arquivos de configuração que estabelecem o que será capturado e armazenado nos logs de serviços Web.

4.3.7 *Parser* do log do servidor Web

O módulo *parser* do log de servidores Web é utilizado na transformação dos arquivos texto ASCII produzidos pelos servidores Web em arquivos XML. O *parser* é um programa executável capaz de identificar o padrão de arquivo de log de servidor Web e aplicar as transformações necessárias para produzir um documento XML bem formado que armazenará todos os campos especificados no arquivo de Web log.

Este componente da arquitetura WSLogA é de grande importância, pois assegura portabilidade e independência tanto em relação ao fornecedor de servidor Web quanto em relação ao padrão de log utilizado.

Após a transformação, todos os arquivos de dados, ficarão armazenados sob formato único, no caso XML, esse característica simplifica, em muito, a construção da ferramenta de ETL, uma vez que será necessário apenas implantar rotinas de carga de um único formato de dados para alimentar os bancos de dados do ambiente analítico de monitoramento de serviços Web apresentado no Capítulo 6.

No próximo capítulo apresentaremos as considerações mais relevantes sobre a implementação da arquitetura WSLogA.

CAPÍTULO 5

IMPLEMENTAÇÃO DA WSLogA

Neste capítulo apresentamos as principais características do protótipo do ambiente de monitoramento desenvolvido nesta dissertação. Discutiremos também as principais características dos intermediários e apresentaremos os esquemas de logs de serviços Web.

5.1 Descrição do ambiente de desenvolvimento

O desenvolvimento dos intermediários e serviços Web presentes nesta dissertação, foi baseado em padrões de software aberto. Utilizamos as seguintes tecnologias:

1. Linguagem Java e sua máquina virtual proveniente da JSDK versão 1.4.1 da Sun Microsystems. Eles foram utilizados tanto na codificação dos intermediários quanto dos serviços Web;
2. Tomcat foi o servidor que permitiu a implementação das especificações JSP e servlets. Este servidor foi desenvolvido pelo projeto Jakarta da Apache Software Foundation e, a versão utilizada em nossos experimentos foi a número 4.1;
3. ANT é um utilitário de linha de comandos escrito em Java e a versão utilizada foi a 1.6. Este utilitário é do tipo *makefile.* Ele utiliza um arquivo XML chamado *build.xml* para descrever as diversas etapas da criação dos serviços Web, como por exemplo, a compilação, a depuração e a distribuição dos serviços;
4. Xerces é um *parser* XML de código aberto Java que implementa classes e métodos utilizados para interpretar, validar e manipular documentos XML. O pacote Xerces

também implementa as APIs SAX e DOM, a versão incorporada ao Apache-Tomcat foi a de número 1.2;

5. AXIS atuou como o *SOAP Engine* e API para criação dos serviços Web, a versão utilizada nos experimentos foi a de número 1.1;

6. Apache é um servidor de páginas, compatível com o protocolo HTTP v. 1.1. Ele foi utilizado para registrar as chamadas HTTP.

5.2 Interação Tomcat - Axis SOAP *engine*

A associação Tomcat–Axis SOAP *engine* desempenha um importante papel no funcionamento dos intermediários SOAP desenvolvidos nesta dissertação. Segundo Irani e Basha (2002), o *AXIS Engine* é composto por vários componentes, cadeias e intermediários. Os autores representam o fluxo de mensagens SOAP no AXIS de acordo com a figura 5.1.

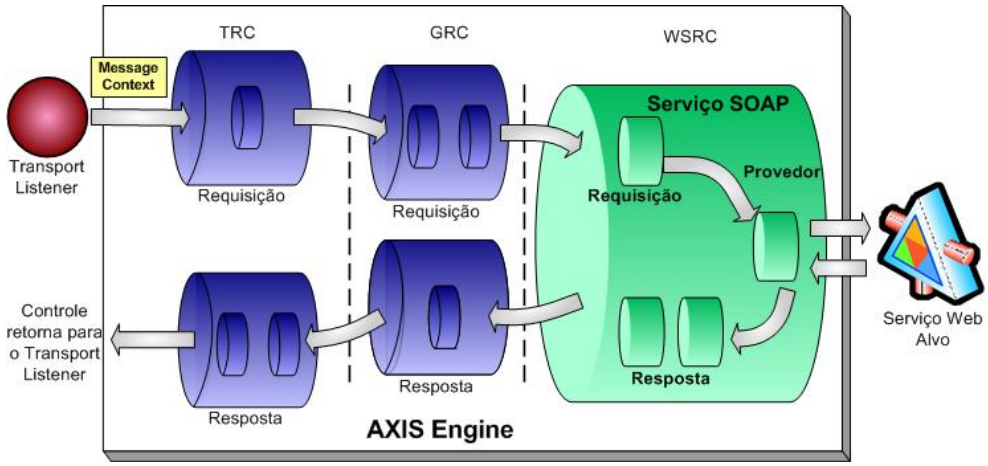


Figura 5.1 - Axis SOAP Engine IRANI, BASHA (*l.c.*) modificado

Uma mensagem SOAP, ao atingir o SOAP *Engine*, é interceptada pelo componente *Transport Listener* que encapsula a mensagem em um objeto *MessageContext* e redireciona-a para o componente AXIS *Engine*.

O AXIS *Engine* recebe os objetos *MessageContext* e verifica se existe alguma cadeia de requisição de pacotes configurada. Uma cadeia de requisição de pacotes é uma seqüência

pré-definida de intermediários onde uma mensagem SOAP deverá fluir até atingir um determinado serviço Web. A versão 1.1 do AXIS implementa três cadeias de requisição: *Transport Request Chain* (TRC) e *Global Request Chain* (GRC) e *Web Service Request Chain* (WSRC).

Caso exista alguma cadeia TRC configurada, os objetos *MessageContext* serão inicialmente interceptados pela TRC que verificará a existência de intermediários e, caso existam, eles serão invocados seqüencialmente. A seguir, o AXIS *Engine* verifica se existe alguma GRC configurada e, em caso afirmativo, todos os intermediários desta cadeia serão invocados.

Finalmente, após passar pelas cadeias TRC e GRC, o AXIS *Engine* verifica se existe alguma WSRC configurada e, se porventura esta cadeia existir, ela conterá um intermediário especial chamado *Pivot Handler*. Este intermediário desempenha um papel especial, pois é ele quem faz a chamada de um serviço Web. Este pivô pode ser visualizado como o ponto central onde as requisições são realizadas e as respostas são obtidas. Uma vez feita a requisição e obtida a resposta, o AXIS *Engine* irá chamar as cadeias na ordem inversa a anteriormente explicitada.

Resumidamente, a principal diferença entre uma cadeia de requisições e um intermediário é a seguinte: um intermediário é um objeto capaz de pré-processar ou pós-processar uma mensagem SOAP. Já uma cadeia consiste de uma coleção de intermediários que são invocados em uma seqüência pré-definida.

A figura 5.2 ilustra a relação cadeia-intermediário, além expressar sua ordem de execução ao longo do tempo.

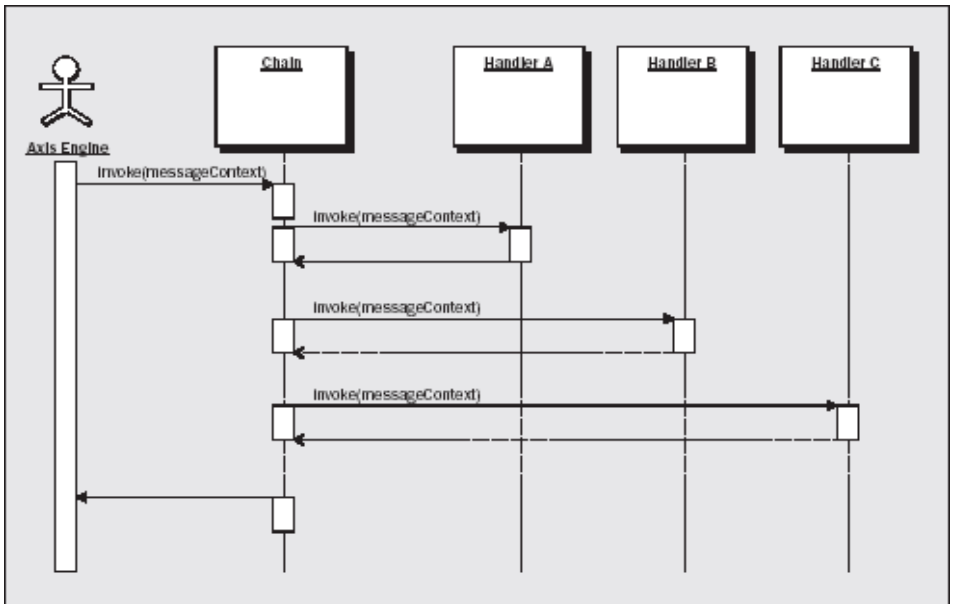


Figura 5.2 - Fluxo de mensagens SOAP (IRANI, BASHA *2002*) modificado.

De acordo com arquitetura do AXIS (2004), anteriormente apresentada, os intermediários SOAP da WSLogA podem ser classificados como sendo intermediários que fazem parte de uma cadeia de requisições do tipo WSRC.

5.3 Mecanismos de captura

Na implementação da arquitetura WSLogA existem dois mecanismos de captura de informações. O primeiro mecanismo se relaciona com os pacotes SOAP. Eles estão presentes nas requisições e respostas que fluem entre os serviços Web e os clientes. As informações, que posteriormente darão origem aos logs XML, são capturadas por dois intermediários que desempenham funções específicas na arquitetura WSLogA. Os intermediários, codificados sob a forma de classes Java, são: *InterRequest* e *InterResponse*.

O segundo mecanismo de captura diz respeito ao servidor Apache, uma aplicação modular, que de modo nativo é capaz de capturar as informações encapsuladas nos pacotes HTTP e armazená-las nos tradicionais arquivos de log HTTP. O formato padrão dos logs do servidor Apache é o CLF (*Common Log Format*).

O servidor Apache quando associado ao Tomcat, admite a configuração da classe *AcessLogValve*. Essa classe permite que se personalize o formato dos arquivos de log HTTP, além da regra de formação destes arquivos. Desta forma, sabe-se quais as informações serão capturadas, quais serão os nomes dos arquivos de log e qual a periodicidade de geração dos arquivos de log. Essas informações podem ser definidas através do arquivo *server.xml*.

A figura 5.3 apresenta, de forma simplificada, os fluxos de mensagens SOAP e a relação entre os componentes da arquitetura WSLogA que implementam os mecanismos de captura de dados.

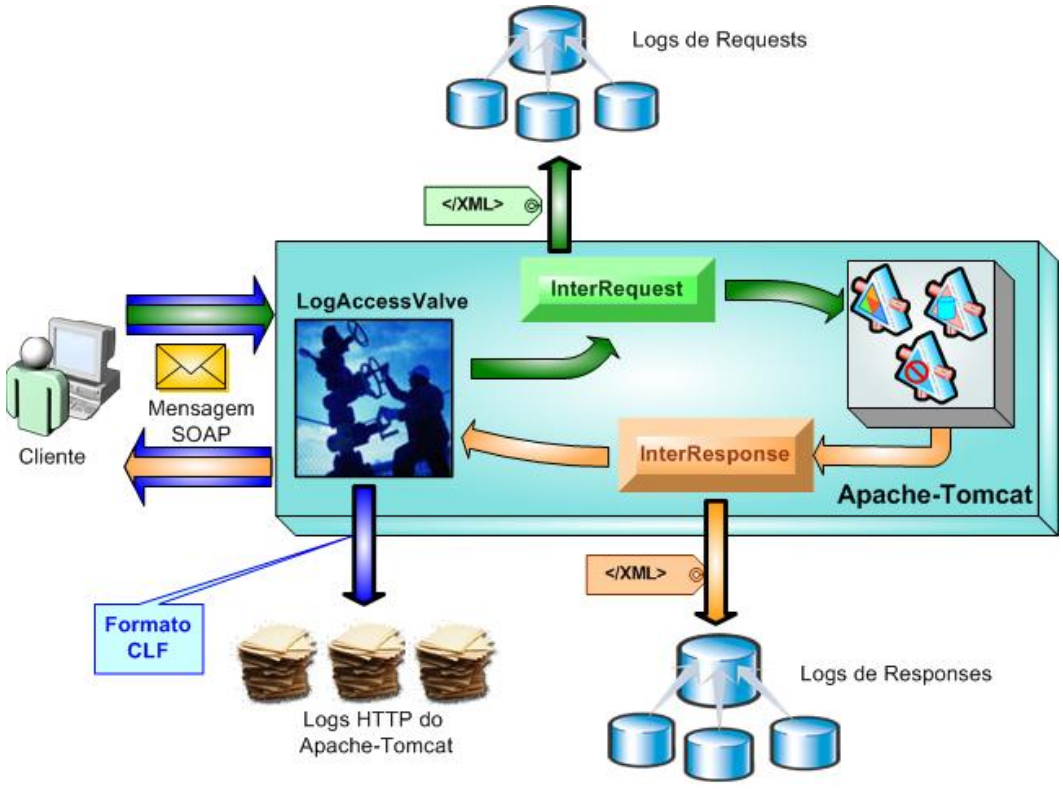


Figura 5.3 - Fluxo de mensagens SOAP e componentes de captura de dados

Os mecanismos de captura da WSLogA atuam na fase de monitoramento do ciclo de vida dos processos de negócio (seção 3.4) e conceitualmente, fazem parte da camada de monitoramento de serviços Web (seção 3.5).

5.3.1 Intermediário *InterRequest*

A codificação deste intermediário fez uso das APIs básicas da linguagem Java e das APIs de recursos avançados do AXIS, sua implementação se deu através da codificação da classe *InterRequest* que é uma extensão da classe *BasicHandler* do AXIS.

O intermediário *InterRequest* possui as seguintes funcionalidades:

1. Interceptação das mensagens SOAP do tipo requisição enviadas para qualquer serviço Web presente no servidor Apache-Tomcat. Se um repositório de log XML existir, uma nova entrada será produzida, caso contrário um novo repositório será criado. Ele também é responsável por definir, baseado nos parâmetros de configuração apresentados no item 4.3.6, o nome do repositório de logs e o cabeçalho do arquivo de log. As operações de entrada e saída do intermediário foram implementadas através dos métodos disponíveis no pacote *java.io*;
2. Geração de um identificador único para cada mensagem SOAP interceptada. O identificador é gerado pelo método *currentTimeMillis( )*[3] e, seu valor é calculado em função da hora do sistema (em milissegundos) em que a mensagem foi capturada. Este valor é incluído como uma parâmetro no **header** da mensagem SOAP e mais tarde será recuperado pelo intermediário *InterResponse*;
3. Reconhecimento das operações e parâmetros presentes nas chamadas dos serviços Web. O reconhecimento e captura dos elementos das mensagens SOAP são implementados através dos métodos disponíveis no pacote *org.apache.axis*;
4. Reconhecimento dinâmico dos elementos do cabeçalho ou do corpo da mensagem SOAP não é uma tarefa trivial. Para minimizar os problemas de desempenho, os intermediários fazem o *parse* da estrutura XML da mensagem SOAP em memória.

[3] Esse método calcula a diferença entre a data e hora atual, em milésimos de segundo, e o dia 01 de janeiro de 1970 às 00:00h.

i. Esta manipulação é implementada através dos métodos do pacote *java.wc3.dom* que permitem a carga da mensagem SOAP para memória principal e posterior manipulação da árvore, onde cada nodo é manipulado via DOM. O uso do DOM veio ao encontro de nossas necessidades de manejo de nodos de um documento XML, uma vez que é possível ter acesso irrestrito a todos os nodos do documento. Essa característica viabilizou a captura de todo o conjunto de informações da mensagem SOAP, independentemente do número e do tipo dos elementos;

ii. Essa funcionalidade é muito útil uma vez que a captura dos elementos XML da mensagem SOAP ocorrerá sem que seja necessário alterar o código dos intermediários.

5. Registro no repositório de logs XML as operações, parâmetros, nomes de serviços e demais elementos, além dos valores presentes no cabeçalho e corpo da mensagem SOAP,

5.3.2 Intermediário *InterResponse*

A implementação desse intermediário é muito parecida com a implementação do intermediário *InterRequest*. Ele também faz uso das APIs da linguagem Java e do AXIS e sua implementação se deu através da codificação da classe *InterResponse* que é uma extensão da classe *BasicHandler* do AXIS (2004).

O intermediário *InterResponse* apresenta as seguintes funcionalidades:

1. Interceptação das mensagens SOAP, do tipo resposta, remetida pelo serviço Web anteriormente invocado pelo cliente;

2. Recuperação do identificador único gerado pelo intermediário *InterRequest.* Para cada resposta o identificador deverá ser igual ao identificador da requisição correspondente.

    i. O valor do identificador possui duas funções. A primeira diz respeito à identificação unívoca do par de mensagens requisição-resposta. A segunda função diz respeito ao cálculo do tempo de processamento do serviço Web;

3. Reconhecimento dos elementos do cabeçalho, do corpo e das falhas, opcionais, da mensagem SOAP; A manipulação é idêntica à implementada no intermediário *InterRequest*;

4. Cálculo do tempo de processamento do serviço Web. O cálculo é realizado da seguinte forma: subtrai-se o momento da captura da mensagem pelo *InterRequest* do momento da captura da mensagem pelo *InterResponse*.

5. Transcrição e gravação dos elementos da mensagem SOAP para o repositório de logs XML;

5.4 Limitações dos intermediários SOAP

Apesar de serem muito importantes para a arquitetura WSLogA, os intermediários SOAP apresentam algumas limitações que podem influir na captura de dados. Cruz *et al.* (2003, 2004) destacam as principais limitações:

1. Os intermediários devem ser construídos para um protocolo de redes específico, como por exemplo, HTTP, FTP, SMTP. Esses protocolos, necessariamente devem fazer parte da camada de aplicação do modelo OSI da ISO, pois é nesta camada que ocorre o encapsulamento da mensagem SOAP (TANEMBAUN, 2004);

2. Os intermediários requerem uma porta de comunicação que será usada para interceptar e retransmitir uma mensagem SOAP;
3. Os intermediários podem registrar os dados capturados em repositórios locais ou remotos. Caso usem repositórios locais é necessário que eles sejam capazes de tratar os problemas relacionados à falta de espaço em disco. No caso da utilização de repositórios remotos é necessário considerar os problemas de conectividade com outros sistemas;
4. Os intermediários SOAP, localizados entre provedores de serviços básicos e compostos, são pouco eficientes quando interceptam mensagens SOAP roteadas. Neste tipo de roteamento não há garantias que as mensagens subseqüentes irão utilizar um mesmo caminho na rede nem que estas chegarão na mesma seqüência em que foram enviadas.
5. Os intermediários SOAP apresentam problemas ao manipular mensagens criptografadas. Os intermediários da arquitetura WSLogA, na atual versão, são capazes de capturar e registrar esse tipo de mensagem, porém, não são capazes de decodificá-las e extrair as informações contidas na mensagem SOAP.

5.5 Configuração da arquitetura WSLogA

Para garantir o pleno funcionamento da arquitetura WSLogA é necessário configurar, manualmente, tanto o servidor Apache-Tomcat quanto os serviços Web. Essas configurações habilitam tanto a captura dos pacotes HTTP no servidor de páginas quanto as mensagens SOAP.

5.5.1 Configuração do servidor Apache-Tomcat

Para que os pacotes HTTP sejam capturados corretamente pelo servidor Apache-Tomcat é necessário configurar o arquivo <*Tomcat_Home*>*conf**server.xml*.

O arquivo *server.xml* é composto por um grande conjunto de descritores que configuram o Tomcat. O descritor <*valve*> requer alterações, ele deve informar ao servidor a classe responsável pela captura dos pacotes HTTP a localização e regra de formação do nome do arquivo de log através de sufixos e prefixos. Ele também define qual será o conteúdo do log. O trecho XML abaixo representa a configuração utilizada no descritor <*valve*>.

```
<Valve className="org.apache.catalina.valves.AccessLogValve"
    directory="C:\Dissertação\logs" prefix="http_log." suffix=".txt"
    pattern="%a %A %b %B %h %H %l %m %p %s %t %u %U %v" resolveHosts="false"/>
```

Com relação ao conteúdo dos logs do Tomcat, optamos por defini-la através da configuração do atributo *pattern* de acordo com a tabela 5.1.

campo	descrição
%a	Endereço IP do cliente
%A	Endereço IP do servidor
%b	Quantidade de Bytes transferidos, excluído os dos header HTTP, se o tamanho for zero será representado por ' - '
%B	Quantidade de Bytes transferidos, excluído os dos header HTTP
%h	Nome do Host do cliente (ou endereço IP se resolve Hosts for falso)
%H	Protocolo de requisição
%l	Username lógico
%m	Método HTTP de requisição (GET, POST, etc.)
%p	Porta lógica que recebeu a requisição
%s	Código de status HTTP da resposta
%t	Data e hora, no formato *Common Log Format*
%u	Conta do usuário remoto se autenticado pelo servidor
%U	URL requerida
%v	Nome do servidor

Tabela 5.1 - Descrição do atributo *Pattern*.

Nossa abordagem utiliza o descritor <*valve*> que é válido somente para o contexto do AXIS. Essa configuração permite capturar as requisições destinadas, exclusivamente, ao SOAP *Engine* (as demais requisições de páginas não serão capturadas). Esse fato gera logs de dimensões mais reduzidas e mais limpas, pois só serão registradas as chamadas aos serviços Web.

5.5.2 Configuração dos serviços Web

A instalação dos serviços Web necessita de algumas configurações adicionais. Essas alterações garantem que todas as mensagens originadas ou destinadas ao serviço Web sejam capturadas pelos intermediários.

As alterações são, manualmente, inseridas nos arquivos que descrevem o funcionamento dos serviços Web, ou seja, os arquivos WSDD criados durante a etapa de distribuição (*deployment).* As alterações associam os fluxos de mensagem com os intermediários pertinentes através da classe *org.apache.axis.client.AdminClient* disponível através da API do AXIS. Uma vez efetuada a alteração executa-se, normalmente, a etapa de publicação dos serviços Web.

A figura 5.4 ilustra um trecho do arquivo alterado de descrição do serviço *ShoppingCart.*

```
<deployment
  xmlns="http://xml.apache.org/axis/wsdd/"
  xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
    <!-- Services from ShoppingCart WSDL service -->
    <service name="Cart" provider="java:RPC" style="rpc" use="encoded">
     <requestFlow>
        <handler name="InterRequest" type="java:intermediarios.InterRequest">
           <parameter name="fileName" value="c:\\Dissertação\\logs\\WSLogARequest_" />
           <parameter name="logErro"  value="c:\\Dissertação\\logs\\erro.log" />
        </handler>
     </requestFlow>
     <responseFlow>
        <handler name="InterResponse" type="java:intermediarios.InterResponse">
           <parameter name="fileName" value="c:\\Dissertação\\logs\\WSLogAResponse_" />
           <parameter name="logErro"  value="c:\\Dissertação\\logs\\erro.log" />
         </handler>
     </responseFlow>
     <parameter name="wsdlTargetNamespace" value="urn:cart"/>
     <parameter name="wsdlServiceElement" value="ShoppingCart"/>
     <parameter name="wsdlServicePort" value="Cart"/>
     <parameter name="className" value="cartws.CartSoapBindingSkeleton"/>
     <parameter name="wsdlPortType" value="Cart"/>
     <parameter name="allowedMethods" value="*"/>
    </service>
</deployment>
```

Figura 5.4 - Trecho alterado do arquivo WSDD do serviço *ShoppingCart*

Os fluxos de requisição e resposta das mensagens SOAP são representados pelos elementos <*requestFlow*> e <*responseFlow*> respectivamente. Para cada fluxo é necessário especificar um *handler* através do elemento <*handler*>.

Através do elemento <*handler*> se define quais intermediários são utilizados na captura das mensagens SOAP. Caso fosse definida uma cadeia de intermediários, faríamos uso do elemento <*chain*>. O elemento <*parameter*> especifica a localização dos repositórios de log XML que serão utilizados pelos intermediários.

5.6 Repositório de logs dos serviços Web

O repositório de logs é composto por vários arquivos de logs de utilização de serviços Web no formato XML. Os logs são de dois tipos: logs de requisição e logs de resposta dos serviços.

Os logs de requisição e de resposta são produzidos pelos intermediários *InterRequest* e *InterResponse*, respectivamente. Eles são gerados com a periodicidade indicada nos arquivos de configuração de parâmetros. Nosso protótipo produz novos logs a cada 24 horas.

Um problema decorrente da criação de novos arquivos de log, é a nomenclatura, por isso, utilizamos uma regra de formação dos nomes que está baseada no seguinte padrão:

- Logs de requisição são nomeados como *WSLogARequest_Ano_Mês_Dia.xml*;
- Logs de resposta são nomeados como *WSLogAResponse_Ano_Mês_Dia.xml*.

Onde:

Ano – Indica o ano de geração do arquivo;

Mês – Indica o mês de geração do arquivo;

Dia - Indica o dia de geração do arquivo.

As estruturas dos logs dos serviços são compostas de duas áreas principais, o cabeçalho e o corpo.

A primeira área diz respeito ao cabeçalho do arquivo log. Um cabeçalho será incluído sempre um que novo arquivo for criado. O cabeçalho está representado na figura 5.5.

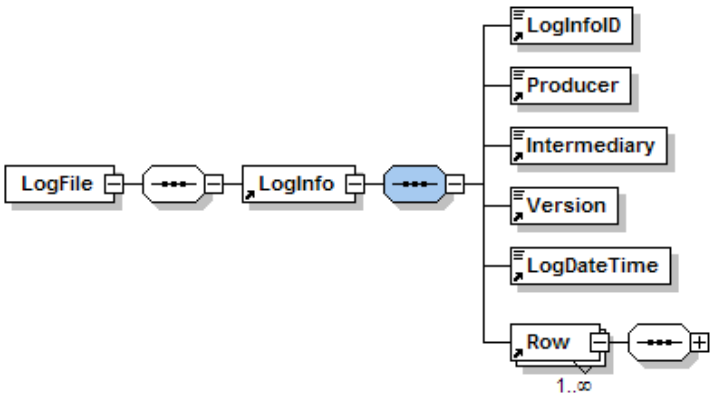


Figura 5.5 - Estrutura do cabeçalho de um arquivo de log

O elemento <*LogFile*> é o elemento raiz. O elemento <*LogInfo*> possui alguns elementos que descrevem o arquivo de log, por exemplo, o elemento <*LogInfoID*> identifica univocamente o arquivo de log no repositório; <*Producer*> identifica a arquitetura; <*Intermediary*> identifica o intermediário gerador do log; <*Version*> a versão da classe do intermediário e <*LogDateTime*> a data e hora de criação do arquivo.

A segunda parte do arquivo é o corpo do log. Ela armazena as informações capturadas pelos intermediários através do elemento <*Row*> que será explicado nas próximas seções.

5.6.1 Logs de requisição

Os logs de requisição registram todas as chamadas dos serviços Web através do elemento <*Row*> que é ilustrado pela figura 5.6.

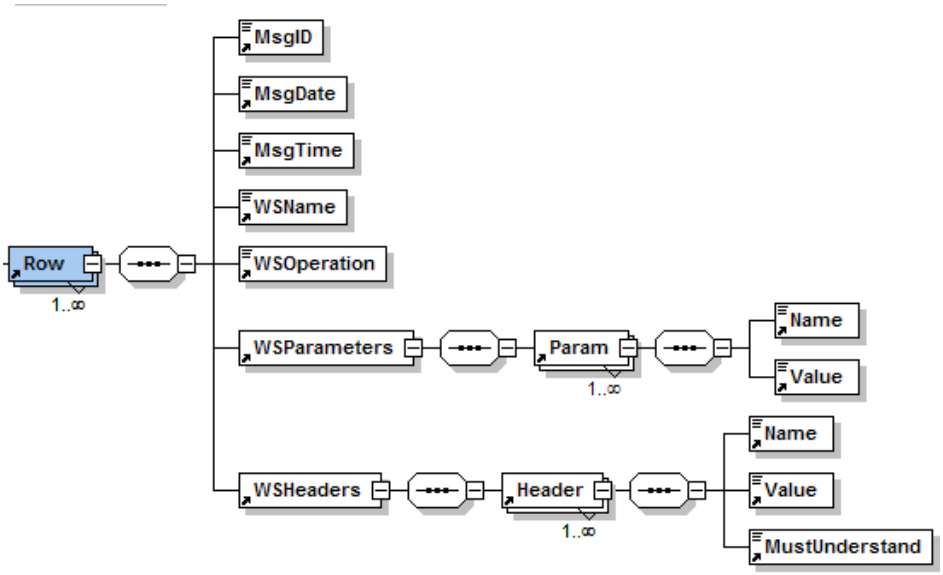


Figura 5.6 - Estrutura do elemento <*Row*> do log de requisição

O elemento <*Row*> é o elemento que registra as informações capturadas na chamada do serviço Web. Ele é composto por: <*MsgID*> que identifica univocamente a mensagem dentro do conjunto de mensagens capturadas; <*MsgDate*> que identifica a data em que ocorreu a captura; <*MsgTime*> que identifica a hora em que ocorreu a captura; <*WSName*> que identifica qual serviço Web foi invocado pelo cliente; <*WSOperation*> que identifica a operação do serviço Web.

O elemento <*WSParameters*> mantém a estrutura dos parâmetros passados para as operações de um serviços Web. O elemento <*Param*> é capaz de registrar o nome do parâmetro e o valor passado através dos elementos <*Name*> e <*Value*>.

O elemento <*WSHeaders*> mantém a estrutura dos cabeçalhos de uma mensagem SOAP na versão 1.1. O elemento <*Header*> é capaz de registrar o os elementos do header, seu valor e obrigatoriedade através dos elementos <*Name*>, <*Value*> e <*MustUndestand*>.

Um exemplo do log produzido pelo intermediário *InterRequest* é mostrado no apêndice A-1.

5.6.2 Logs de resposta

Os logs de resposta registram todas as respostas enviadas por um serviço Web e são ligeiramente diferentes dos logs de requisição. Cada resposta também é registrada pelo elemento <*Row*> que é ilustrado pela figura 5.7.

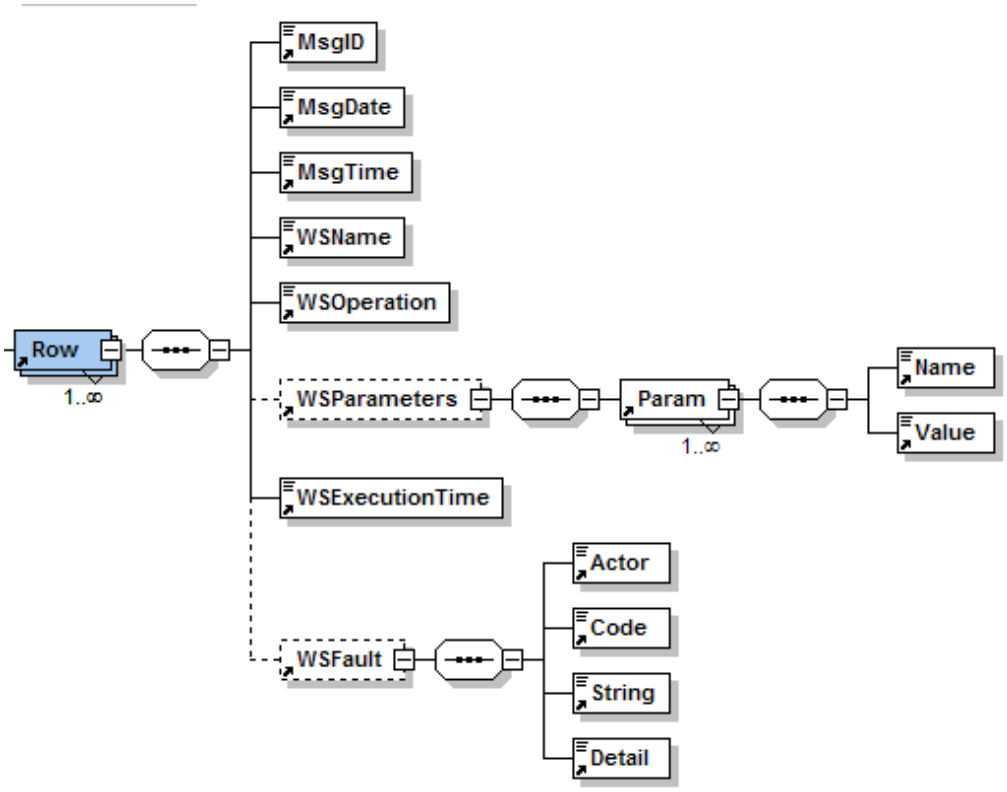


Figura 5.7 - Estrutura do elemento <*Row*> do log de resposta

O elemento <*WSExecutionTime*> armazena o tempo de processamento em milissegundos. O elemento <*WSParameters*> é opcional. Ele será representado no log quando não ocorrem erros na execução do serviço. Este elemento mantém a estrutura dos parâmetros retornados pelos serviços Web. O elemento <*Param*> é capaz de registrar o nome do parâmetro e o valor passado através dos elementos <*Name*> e <*Value*>.

O elemento <*WSFault*> também é opcional. Ele somente estará presente quando ocorrer um erro qualquer no processamento do serviço Web. Esse elemento mantém a mesma estrutura de erros da especificação SOAP (versão 1.1). O elemento <*Actor*> é capaz de registrar autor da falha. Os elementos <*Code*>, <*String*> e <*Detail*>, registram respectivamente, o código do erro, o erro propriamente dito e uma descrição detalhada do erro.

Um exemplo do log produzido pelo intermediário *InterResponse* é mostrado no apêndice A-2.

5.7 *Parser* dos logs do servidor Tomcat

O módulo *Parser*, da arquitetura WSLogA, opera a transformação dos arquivos texto ASCII produzidos pelos servidores Web em arquivos XML.

O *parser* foi codificado em Java e possui as seguintes funcionalidades:

1. Identificação do tipo e formato do Web log a ser transformado;
2. Carga do arquivo de configurações e o XML *Schema* que definem a estrutura e os nomes dos elementos que serão usados na transformação;
3. Identificação dos campos do arquivo original e transformá-los em elementos XML;
4. Inclusão um identificador único para cada registro, este identificador atuará como chave primária no ambiente relacional;
5. Inclusão de um identificador único para cada arquivo de log transformado.

A figura 5.8 representa um log padrão CLF produzido pelo servidor Tomcat antes de aplicar o *parser*.

```
200.73.63.59 146.164.248.21 445 -1 127.0.0.1 HTTP/1.0 - POST 8080 200 [05/Feb/2004:12:37:51 -0300] - /axis/services/Login localhost
200.73.63.59 146.164.248.21 470 -1 127.0.0.1 HTTP/1.0 - POST 8080 200 [05/Feb/2004:12:37:51 -0300] - /axis/services/BestSeller localhost
200.73.63.59 146.164.248.21 446 -1 127.0.0.1 HTTP/1.0 - POST 8080 200 [05/Feb/2004:12:37:52 -0300] - /axis/services/Busca localhost
200.73.63.59 146.164.248.21 458 -1 127.0.0.1 HTTP/1.0 - POST 8080 200 [05/Feb/2004:12:37:52 -0300] - /axis/services/DetalheItem localhost
200.73.63.59 146.164.248.21 457 -1 127.0.0.1 HTTP/1.0 - POST 8080 200 [05/Feb/2004:12:37:52 -0300] - /axis/services/Cart localhost
200.73.63.59 146.164.248.21 449 -1 127.0.0.1 HTTP/1.0 - POST 8080 200 [05/Feb/2004:12:37:52 -0300] - /axis/services/PedidoCompra localhost
200.73.63.59 146.164.248.21 478 -1 127.0.0.1 HTTP/1.0 - POST 8080 200 [05/Feb/2004:12:37:52 -0300] - /axis/services/ConfirmCompra localhost
201.74.64.60 146.164.248.21 445 -1 127.0.0.1 HTTP/1.0 - POST 8080 200 [05/Feb/2004:12:37:53 -0300] - /axis/services/Login localhost
201.74.64.60 146.164.248.21 470 -1 127.0.0.1 HTTP/1.0 - POST 8080 200 [05/Feb/2004:12:37:53 -0300] - /axis/services/BestSeller localhost
201.74.64.60 146.164.248.21 446 -1 127.0.0.1 HTTP/1.0 - POST 8080 200 [05/Feb/2004:12:37:53 -0300] - /axis/services/Busca localhost
201.74.64.60 146.164.248.21 458 -1 127.0.0.1 HTTP/1.0 - POST 8080 200 [05/Feb/2004:12:37:53 -0300] - /axis/services/DetalheItem localhost
201.74.64.60 146.164.248.21 457 -1 127.0.0.1 HTTP/1.0 - POST 8080 200 [05/Feb/2004:12:37:53 -0300] - /axis/services/Cart localhost
201.74.64.60 146.164.248.21 449 -1 127.0.0.1 HTTP/1.0 - POST 8080 200 [05/Feb/2004:12:37:53 -0300] - /axis/services/PedidoCompra localhost
```

Figura 5.8 - Trecho de log do servidor Tomcat antes da transformação

Os logs do servidor Apache-Tomcat também possuem uma regra de formação de nomes, por isso, utilizamos uma regra de formação dos nomes que está baseada no seguinte padrão:

- Logs de requisição são nomeados de *http_log.Ano_Mês_Dia.txt*;

Onde:

Ano – Indica o ano de geração do arquivo;

Mês – Indica o mês de geração do arquivo;

Dia - Indica o dia de geração do arquivo;

O log XML do servidor Apache-Tomcat mantém a mesma estrutura de campos que estão definidos no elemento <`valve`> do arquivo *server.xml* (seção 5.5.1). A estrutura dos elementos pode ser visualizada na figura 5.9.

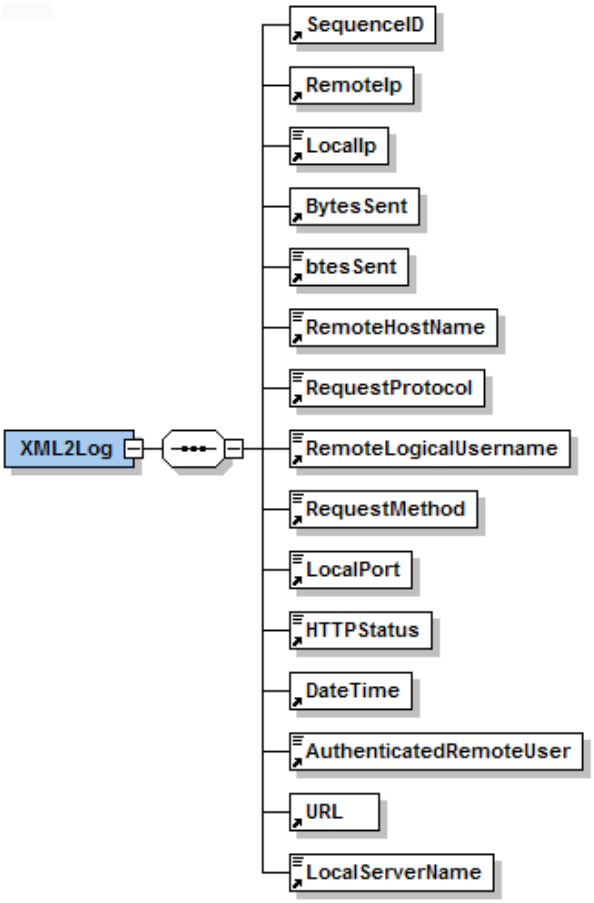


Figura 5.9 - Estrutura XML do Apache-Tomcat

O elemento raiz é <*XML2Log*>. Ele encapsula todos os elementos envolvidos em uma chamada HTTP de serviço Web.

O elemento <*SequenceID*> originalmente não faz parte do conjunto de campos do log padrão do servidor Apache-Tomcat. Este elemento é incluído pelo componente *parser* com o objetivo de criar um identificador único para que, posteriormente, possa ser manipulado adequadamente no banco de dados relacional.

Os elementos <*RemoteIP*> e <*LocalIP*> registram os endereços IP do cliente e do servidor.

Os elementos <*BytesSent*> e <*btesSent*> armazenam a quantidade de bytes transferidos, excluindo-se os dos header http. Se o tamanho for zero será representado por '-'.

O elemento <*RemoteHost*> registra o nome da máquina do cliente (ou endereço IP se a variável *resolveHosts* do Apache for falsa)

O elemento <*RequestProtocol*> registra o protocolo da requisição, normalmente HTTP.

O elemento <*RemoteLogicalUsername*> é capaz de registrar, no caso de serviços que exijam autenticação, a conta utilizada pelo usuário no processo de autenticação.

Os elementos <*RequestMethod*>, <*LocalPort*>, <*HTTPStatus*> estão diretamente relacionados ao elemento <*RequestProtocol*>. O primeiro elemento diz respeito ao método de invocação do serviço Web. O segundo indica qual a porta TCP foi utilizada e o terceiro elemento indica a situação da resposta, no caso do protocolo HTTP sempre será registrado o status da mensagem.

O elemento <*DateTime*> indica o data e hora da requisição. O elemento <*URL*> registra o serviço Web invocado pelo cliente, enquanto que o elemento <*LocalServerName*> registra o nome do servidor Web.

Um pequeno exemplo de um trecho do log transformado pelo *parser* é mostrado no apêndice A-3.

Neste capítulo discutimos as principais características das estruturas dos logs de serviços Web. No capítulo seguinte apresentaremos um cenário de uma livraria virtual que descreve processos de negócio baseados em serviços Web. Também apresentaremos e discutiremos um exemplo de log de serviços Web produzido pela arquitetura WSLogA.

CAPÍTULO 6

DATA MARTS DE MONITORAMENTO E UTILIZAÇÃO DE PROCESSOS DE NEGÓCIO E SERVIÇOS WEB

Neste capítulo apresentamos o ambiente analítico de monitoramento de processos de negócio baseados em serviços Web a partir de um cenário de negócio. O ambiente analítico é composto por um conjunto de *Data Marts*. Optamos por trabalhar com pequenos *Data Marts* articulados entre si, compondo assim uma arquitetura com dimensões e fatos compartilhados.

6.1 Cenário de negócio

Para que se discuta, detalhadamente, o ambiente analítico para o monitoramento de processos, descrevemos um cenário de uma empresa que adota serviços Web para a comercialização de livros.

O cenário em questão, de escopo simplificado, descreve uma livraria fictícia denominada *Hoplias Books*. A livraria não mantém estoques e trabalha exclusivamente com cartões de crédito e, por questões estratégicas, a diretoria optou pela terceirização de alguns serviços, como por exemplo, serviços que envolvam transações seguras com as operadoras de cartão de crédito e serviços relacionados à remessa das encomendas.

Hoplias Books negocia livros com clientes e parceiros de negócio através de seu ambiente de comércio eletrônico que é representado, no nível operacional, por um provedor de serviços Web que mantém os seguintes serviços básicos e compostos: *Login, OrderInquiry, OrderDisplay, BestSeller, ProductDetail, SearchRequest, ShoppingCart,*

BuyRequest, BuyConfirm, CustomerRegistration, VerifyCreditCard, TrackOrder, DeliverOrder, Invoicing, RequestBook e *ReceiveBook.*

A livraria é líder de mercado no seu segmento e reconhece a importância de aprimorar a gerência dos seus processos de negócio para obter resultados satisfatórios no que tange à velocidade de respostas e mudanças imprevisíveis nos requisitos do negócio. A livraria também busca reduzir custos operacionais, aprimorar o relacionamento com os clientes e aumentar a colaboração com seus parceiros, sejam eles fornecedores ou clientes. Em suma, a empresa deseja aprimorar seu desempenho organizacional. A figura 6.1 ilustra o provedor de serviços Web da *Hoplias Books*.

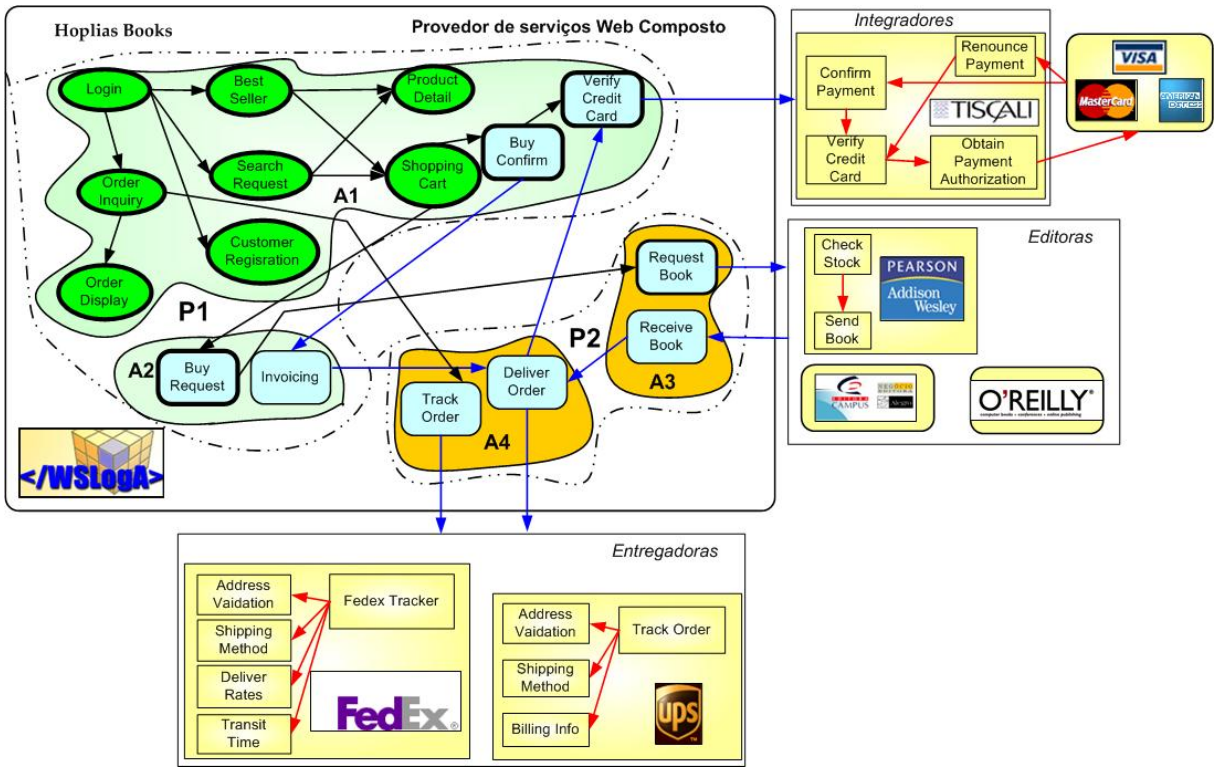


Figura 6.1. Cenário de utilização de serviços Web, atividades e processos de negócio da *Hoplias Books.*

Na figura 6.1 também podem ser observados três de grupos parceiros de negócio:

1. **Editoras** – Neste grupo se encontram as editoras que produzem os livros que são adquiridos pela livraria. Alguns exemplos de parceiros nesta área são as editoras Pearson & Addison Wesley, Campus, O'Reilly, entre outras;
2. **Entregadoras** – Este grupo constitui-se das empresas que fazem a entrega das encomendas. Destacam-se a UPS e a FedEx;
3. **Integradores** – Neste grupo se encontram as empresas que através de canais seguros de comunicação oferecem serviços Web com elevados níveis de segurança e confiabilidade.

O ambiente de comércio eletrônico é formado por serviços Web simples e compostos (os simples são representados na figura 6.1 sob a forma de elipses e os compostos sob a forma de quadrados de bordas arredondadas) sendo que apenas alguns foram codificados como parte do estudo de caso desta dissertação (os serviços Web codificados estão representados na figura com linhas mais espessas).

Os serviços Web aqui representados constituem a base para os processos de negócio. Neste cenário, mapeamos dois processos de negócio: *Venda* e *Controle*. Os processos de negócio estão representados na figura 6.1 sob a forma de linhas pontilhadas.

O processo de negócio *Venda* envolve a comercialização de livros, sendo composto por dois subprocessos: *Negociação* e *Cobrança*. O subprocesso *Negociação* diz respeito às interações do cliente. Neste subprocesso o cliente pesquisará livros e os comprará através do uso de cartão de crédito. O outro subprocesso se relaciona com a emissão das cobranças. O subprocesso *Cobrança* é composta pelo cálculo de custos (taxa de remessa, embalagem, preço livro, taxas de crédito, entre outros).

O processo de negócio *Controle* também é composto por duas atividades: *Ordem de Serviço* e *Transporte*. O subprocesso *Ordem de Serviço* requer interações com parceiros de

negócio, podendo se relacionar à requisição de livros produzidos por uma editora. A atividade *Transporte* relaciona-se com o procedimento de envio e acompanhamento de uma encomenda.

A tabela 6.1 apresenta os processos e subprocessos de negócio, representados por seus respectivos serviços Web.

Processo de negócio	**Subprocesso**	**Serviço Web**
Venda (P1)	Negociação (A1)	*Login, OrderQuery, OrderDisplay, BestSeller, SearchRequest, CustomerRegistration, ProductDetail, ShoppingCart, VerifyCreditCard, BuyConfirm*
	Cobrança (A2)	*BuyRequest, Invoicing*
Controle (P2)	Ordem de Serviço (A3)	*RequestBook, ReceiveBook*
	Transporte (A4)	*DeliverOrder, TrackOrder*

Tabela 6.1 - Processos de Negócio x Subprocessos x Serviços Web

6.2 Exemplos de log de serviços Web

Nesta seção apresentaremos um exemplo, para este cenário, de fragmentos de logs de processos de negócio baseados em serviços Web. Os fragmentos foram produzidos pelos componentes da arquitetura WSLogA apresentados na seção 4.3.

O primeiro fragmento representa o log produzido pelo intermediário *InterRequest*, descrito na seção 5.3.1. O segundo fragmento, ilustra o log produzido pelo intermediário *InterResponse*, descrito na seção 5.3.2.

6.2.1 Logs de invocações de serviços Web

No primeiro fragmento estão presentes as interações necessárias para a aquisição de um livro (os trechos de interesse são referenciados com setas numeradas). Nele, se identificam os serviços Web básicos (*Login*, *BestSeller*, *SearchRequest*, *ProductDetail*, *ShoppingCart*) e alguns serviços Web compostos (*BuyRequest*,*VerifyCreditCard, RequestBook* e *BuyConfirm*).

O fragmento inicia com a identificação do arquivo de log (trecho 1). Depois, a invocação de uma instância do serviço Web *Login* (trecho 2), isto é, quando determinado cliente faz uma solicitação de autenticação dando início à navegação.

Após a execução bem sucedida do serviço *Login*, ocorre a chamada do serviço *BestSeller* que busca livros da categoria “bebidas” (trecho 3). A seguir, o cliente recebe as informações solicitadas. Então, o cliente busca o livro cujo ISBN é 10021012021 (trecho 4), solicitando a seguir mais detalhes sobre o livro (trecho 5). Após estas ações ele insere uma unidade no carrinho de compras através do serviço *ShoppingCart* (trecho 6). A partir daí entram em cena os serviços Web compostos *BuyRequest* e *BuyConfirm*.

A aquisição propriamente dita se inicia quando o serviço Web *BuyRequest* gera um identificador do pedido de Compras para o cliente (trecho 7). A seguir, o serviço Web *BuyConfirm*, inicia dois serviços Web básicos (trecho 8). O primeiro é o *VerifyCreditCard*, que faz a validação do cartão de crédito do cliente (trecho 9). O segundo serviço Web é o *RequestBook* que solicita uma cópia para a editora (trecho 10). A compra só é finalizada ao término da execução dos serviços iniciados pelo *BuyConfirm.*

```
<?xml version="1.0" encoding="UTF-8"?>
<LogFile                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\Dissertação\WSLogs\WSLogARequest.xsd">
        <LogInfo>
                <LogInfoID>1076378395734</LogInfoID>
                <Producer>Web Services Log Architecture</Prod
                <Intermediary>InterRequest</Intermediary>
                <Version>1.0</Version>
                <LogDateTime>2004-Feb-09 23:59:40</LogDateTime>
...(outros elementos)
                <Row>
                        <MsgID>1076292004859</MsgID>
...(outros elementos)
                        <WSName>Login</WSName>
                        <WSOperation>validatePwd</WSOperation>
                        <WSParameters>
                                <Param>
                                        <Name>Username</Name>
                                        <Value>sps10g1</Value>
                                </Param>
                                <Param>
                                        <Name>Password</Name>
                                        <Value>ogbaserereri</Value>
                                </Param>
                        </WSParameters>
...(outros elementos)
```

Trecho 1

Trecho 2

```
            </Row>
            <Row>
                  <MsgID>1076292004906</MsgID>
...(outros elementos)
                  <WSName>BestSeller</WSName>
                  <WSOperation>seekBestSeller</WSOperati
                  <WSParameters>
                        <Param>
                              <Name>category</Name>
                              <Value>bebidas</Value>
                        </Param>
                  </WSParameters>
...(outros elementos)
            </Row>
            <Row>
                  <MsgID>1076292005000</MsgID>
...(outros elementos)
                  <WSName>SearchRequest</WSName>
                  <WSOperation>seekItem</WSOperation>
                  <WSParameters>
                        <Param>
                              <Name>ISBN</Name>
                              <Value>10021012021</Value>
                        </Param>
                  </WSParameters>
...(outros elementos)
            </Row>
            <Row>
                  <MsgID>1076292005078</MsgID>
...(outros elementos)
                  <WSName>ProductDetail</WSName>
                  <WSOperation>showDetail</WSOperation>
                  <WSParameters>
                        <Param>
                              <Name>Author</Name>
                              <Value>B. bubalis</Value>
                        </Param>
                        <Param>
                              <Name>Title</Name>
                              <Value>O aprendiz de cervejeiro</Value>
                        </Param>
                        <Param>
                              <Name>ISBN</Name>
                              <Value>10021012021</Value>
                        </Param>
                  </WSParameters>
...(outros elementos)
            </Row>
            <Row>
                  <MsgID>1076292005421</MsgID>
...(outros elementos)
                  <WSName>ShoppingCart</WSName>
                  <WSOperation>addItem</WSOperation>
                  <WSParameters>
                        <Param>
                              <Name>ISBN</Name>
                              <Value>10021012021</Value>
                        </Param>
                        <Param>
                              <Name>amount</Name>
                              <Value>1</Value>
                        </Param>
                        <Param>
                              <Name>Price</Name>
                              <Value>100</Value>
                        </Param>
                  </WSParameters>
...(outros elementos)
```

Trecho 3

Trecho 4

Trecho 5

Trecho 6

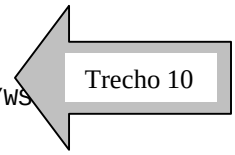
```
            </Row>
            <Row>
                  <MsgID>1076292005687</MsgID>
...(outros elementos)
                  <WSName>BuyRequest</WSName>
                  <WSOperation>demandItem</WSOperation>
                  <WSParameters>
                        <Param>
                              <Name>Client</Name>
                              <Value>ss1001</Value>
                        </Param>
                        <Param>
                              <Name>RequestID</Name>
                              <Value>1197</Value>
                        </Param>
                  </WSParameters>
...(outros elementos)
            </Row>
            <Row>
                  <MsgID>1076292005937</MsgID>
...(outros elementos)
                  <WSName>BuyConfirm</WSName>
                  <WSOperation>confirmation</WSOperation>
                  <WSParameters>
                        <Param>
                              <Name>RequestID</Name>
                              <Value>1197</Value>
                        </Param>
                        <Param>
                              <Name>TransactionCC</Name>
                              <Value>1430782548</Value>
                        </Param>
                        <Param>
                              <Name>DateTimeRequest</Name>
                              <Value>2003-11-17 22:53:37.657</Value>
                        </Param>
                        <Param>
                              <Name>CCardType</Name>
                              <Value>Visa</Value>
                        </Param>
                        <Param>
                              <Name>Number</Name>
                              <Value>2910123409871234</Value>
                        </Param>
                        <Param>
                              <Name>Value</Name>
                              <Value>100</Value>
                        </Param>
            </Row>
...(outros elementos)
            <Row>
                  <MsgID>10762920016789</MsgID>
...(outros elementos)
                  <WSName>VerifyCreditCard</WSName>
                  <WSOperation>checkVISACCard</WSOperation>
                  <WSParameters>
                        <Param>
                              <Name>CCNumber</Name>
                              <Value>2910123409871234</Value>
                        </Param>
                        <Param>
                              <Name>Value</Name>
                              <Value>100</Value>
                  </WSParameters>
...(outros elementos)
            </Row>
            <Row>
                  <MsgID>10762920234563</MsgID>
```

Trecho 7

Trecho 8

Trecho 9

```
...(outros elementos)
                    <WSName>RequestBook</WSName>
                    <WSOperation>requestItemToPublisher</WS
                    <WSParameters>
                           <Param>
                                  <Name>ISBN</Name>
                                  <Value>10021012021</Value>
                           </Param>
                           <Param>
                                  <Name>Amount</Name>
                                  <Value>1</Value>
                           </Param>
                    </WSParameters>
...(outros elementos)
              </Row>
       </LogInfo>
</LogFile>
```



O segundo fragmento de log, representado logo na próxima seção, ilustra as respostas das requisições de uma compra. Ressalta-se que cada resposta é identificada univocamente através do elemento <*MsgID*>.

6.2.2 Logs de respostas de serviços Web

Neste fragmento identificam-se as respostas dos serviços Web básicos (*Login*, *BestSeller*, *SearchRequest*, *ProductDetail*, *ShoppingCart*) e também dos serviços Web Compostos (*BuyRequest, VerifyCreditCard, RequestBook* e *BuyConfirm*).

O fragmento se inicia com o cabeçalho do arquivo de logs de respostas (trecho 11). A seguir, ele indica a resposta do serviço *Login* que foi realizado com sucesso (trecho 12), ainda neste trecho é possível verificar que a execução do serviço consumiu 110 milissegundos. As discussões das respostas subseqüentes são similares a essa explanação.

As respostas às requisições dos serviços Web *BestSeller, SearchRequest, ProductDetail, ShoppingCart, BuyRequest ,VerifyCreditCard, RequestBook* e *BuyConfirm* são indicadas pelos trechos 13, 14, 15, 16, 17, 18, 19 e 20 respectivamente.

O fragmento de log é capaz de registrar a composição dos serviços Web. Observa-se que existem registros que contém informações sobre as respostas das invocações dos serviços *VerifyCreditCard* e *RequestBook*. Nos casos de serviços Web compostos ainda é possível

obter o tempo de processamento do serviço através dos valores do elemento <*WSExecutionTime*> de cada serviço Web.

```
      <?xml version="1.0" encoding="UTF-8"?>
      <LogFile                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\Dissertação\WSLogs\WSLogAResponse.xdr">
            <LogInfo>
                  <LogInfoID>1076378390171</LogInfoID>
                  <Producer>Web Services Log Architecture</Prod
                  <Intermediary>InterResponse</Intermediary>
                  <Version>1.0</Version>
                  <LogDateTime>2004-Feb-09 23:59:50</LogDateTime>
      ...(outros elementos)
                  <Row>
                        <MsgID>1076292004859</MsgID>
      ...(outros elementos)
                        <WSName>Login</WSName>
                        <WSOperation>validatePwd</WSOperation>
                        <WSParameters>
                              <Param>
                                    <Name>validatePwdReturn</Name>
                                    <Value>OK</Value>
                              </Param>
                        </WSParameters>
                        <WSExecutionTime>0.110</WSExecutionTime>
                  </Row>
      ...(outros elementos)
                  <Row>
                        <MsgID>1076292004906</MsgID>
      ...(outros elementos)
                        <WSName>BestSeller</WSName>
                        <WSOperation>seekBestSeller</WSOperat
                        <WSParameters>
                              <Param>
                                    <Name>seekBestSellerReturn</Name>
                                    <Value>Ok!</Value>
                              </Param>
                        </WSParameters>
                        <WSExecutionTime>0.435</WSExecutionTime>
                  </Row>
      ...(outros elementos)
                  <Row>
                        <MsgID>1076292005000</MsgID>
      ...(outros elementos)
                        <WSName>SearchRequest</WSName>
                        <WSOperation>seekItem</WSOperation>
                        <WSParameters>
                              <Param>
                                    <Name>seekItemReturn</Name>
                                    <Value>Ok!</Value>
                              </Param>
                        </WSParameters>
                        <WSExecutionTime>1.044</WSExecutionTime>
                  </Row>
      ...(outros elementos)
                  <Row>
                        <MsgID>1076292005078</MsgID>
      ...(outros elementos)
                        <WSName>ProductDetail</WSName>
                        <WSOperation>showDetail</WSOperation>
                        <WSParameters>
                              <Param>
                                    <Name>showDetailReturn</Name>
                                    <Value>Ok!</Value>
                              </Param>
```

Trecho 11

Trecho 12

Trecho 13

Trecho 14

Trecho 15

```
                </WSParameters>
                <WSExecutionTime>1.281</WSExecutionTime>
        </Row>
...(outros elementos)
        <Row>
                <MsgID>1076292005421</MsgID>
...(outros elementos)
                <WSName>ShoppingCart</WSName>
                <WSOperation>addItem</WSOperation>
                <WSParameters>
                        <Param>
                                <Name>addItemReturn</Name>
                                <Value>Ok!</Value>
                        </Param>
                </WSParameters>
                <WSExecutionTime>0.932</WSExecutionTime>
        </Row>
...(outros elementos)
        <Row>
                <MsgID>1076292005687</MsgID>
...(outros elementos)
                <WSName>BuyRequest</WSName>
                <WSOperation>demandItem</WSOperation>
                <WSParameters>
                        <Param>
                                <Name>demandItemReturn</Name>
                                <Value>Ok!</Value>
                        </Param>
                </WSParameters>
                <WSExecutionTime>0.731</WSExecutionTime>
        </Row>
...(outros elementos)
        <Row>
                <MsgID>10762920016789</MsgID>
...(outros elementos)
                <WSName>VerifyCreditCard</WSName>
                <WSOperation>c]heckVISACCard</WSOperati
                <WSParameters>
                        <Param>
                                <Name>StatusOperationCC</Name>
                                <Value>Accept</Value>
                        </Param>
                </WSParameters>
                <WSExecutionTime>0.020</WSExecutionTime>
        </Row>
...(outros elementos)
        <Row>
                <MsgID>10762920234563</MsgID>
...(outros elementos)
                <WSName>RequestBook</WSName>
                <WSOperation>requestItemToPublisher</WS
                <WSParameters>
                        <Param>
                                <Name>StatusOperationRequest</Name>
                                <Value>Sucess</Value>
                        </Param>
                        <Param>
                                <Name>Publisher</Name>
                                <Value>Campus</Value>
                        </Param>
                        <Param>
                                <Name>QAmount</Name>
                                <Value>1</Value>
                        </Param>
                </WSParameters>
                <WSExecutionTime>0.031</WSExecutionTime>
        </Row>
...(outros elementos)
```

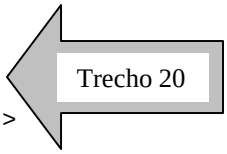
Trecho 16

Trecho 17

Trecho 18

Trecho 19

```
            <Row>
                  <MsgID>1076292005937</MsgID>
...(outros elementos)
                  <WSName>BuyConfirm</WSName>
                  <WSOperation>confirmation</WSOperation>
                  <WSParameters>
                        <Param>
                              <Name>confirmationBuyReturn</Name>
                              <Value>Ok!</Value>
                        </Param>
                  </WSParameters>
                  <WSExecutionTime>3.249</WSExecutionTime>
            </Row>
      </LogInfo>
</LogFile>
```



Nas próximas seções apresentaremos uma proposta de ambiente analítico de monitoramento de processos de negócio baseados em serviços Web.

6.3 Modelagem dos *Data Marts*

Os *Data Marts* propostos nesta dissertação foram desenvolvidos segundo a arquitetura *Data Warehouse Bus Architecture* (DWB) proposta por Kimball (1998). A arquitetura DWB retrata uma empresa composta por um conjunto de processos de negócio que constituem sua cadeia de valores. Usualmente, para explorar as propriedades dos processos são desenvolvidos vários *Data Marts*, ao menos um para cada processo, sendo que sua união lógica forma o *Data Warehouse*. Os *Data Marts* são articulados entre si, compondo assim uma arquitetura com dimensões e fatos compartilhados. Segundo o autor, um dos benefícios da arquitetura DWB é retratar uma empresa como sendo composta por um conjunto de processos de negócio e, no caso desta dissertação, os processos são materializados sob a forma de serviços Web.

O uso de dimensões compartilhadas pode ser compreendido como um “barramento” onde os processos de negócio podem se conectar para receber as informações que desejam. A metáfora do “barramento” pode ser vista no seguinte contexto. Uma matriz analítica cujas linhas são os *Data Marts* e as colunas são as dimensões, onde as marcações nas células dessa matriz indicam que um processo está conectado a uma dimensão compartilhada. Uma dimensão compartilhada é uma dimensão que possui o mesmo significado e estrutura em

diversos *Data Marts*. O fato compartilhado é uma definição de variável que é consistente nos diversos *Data Marts*.

Antes de apresentar os *Data Marts* de monitoramento e utilização é importante ressaltar uma questão de modelagem. É crucial estabelecer a noção de *Instância* e *Tipo.* Processos de negócio ou serviços Web são especificações definidas segundo um *tipo* que será instanciado quando da sua execução. *Instâncias* correspondem a estas execuções específicas, representadas pelo conjunto de informações da execução de determinado processo ou serviço em um determinado momento.

A importância da diferenciação entre tipo e instâncias está baseada em duas observações de ordem prática.

A primeira observação diz respeito ao nível de detalhe, isto é, quando o grão representado nas tabelas de fato é uma instância de um processo de negócio ou serviço Web. Com esse nível de detalhe é possível fazer análises individualizadas que envolvam aspectos operacionais, por exemplo, avaliar os quesitos de qualidade de serviços apresentados na seção 3.10. Quando o grão da tabela de fatos é representado pelo tipo do serviço Web ou processo de negócio, o nível de detalhe é menor, correspondendo às totalizações de chamadas, de erros, entre outras.

A segunda observação está relacionada com o tamanho dos *Data Marts* de instância. Se o analista registrar todas as instâncias de um dado serviço Web ao longo de grandes intervalos de tempo, os *Data Marts* irão armazenar imensos volumes de dados. Por estas razões os *Data Marts* de instância devem se restringir a janelas de tempo pequenas, como por exemplo, o último mês ou a última semana.

De qualquer modo, este tipo de análise operacional só tem sentido de ser realizado para períodos de tempo recentes, permitindo atuar por sobre fatores do ambiente que podem influenciar no desempenho ou na qualidade dos serviços.

Finalmente, é interessante notar que os *Data Marts* presentes nesta seção podem ser encarados como uma evolução dos *Data Marts* apresentados em trabalho anterior (Cruz *et al*. 2003).

6.3.1 *Data Mart* de monitoramento de instâncias de processos de negócio

À medida que os negócios da livraria se tornam cada vez mais complexos, faz-se necessário que gerentes e analistas compreendam como se desenrolam seus processos de negócio.

Para facilitar esta tarefa e, ir ao encontro das perspectivas dos processos de negócio internos apontadas pela metodologia BSC, propomos um *Data Mart* de monitoramento de instâncias de processos de negócio que está representado na figura 6.2.

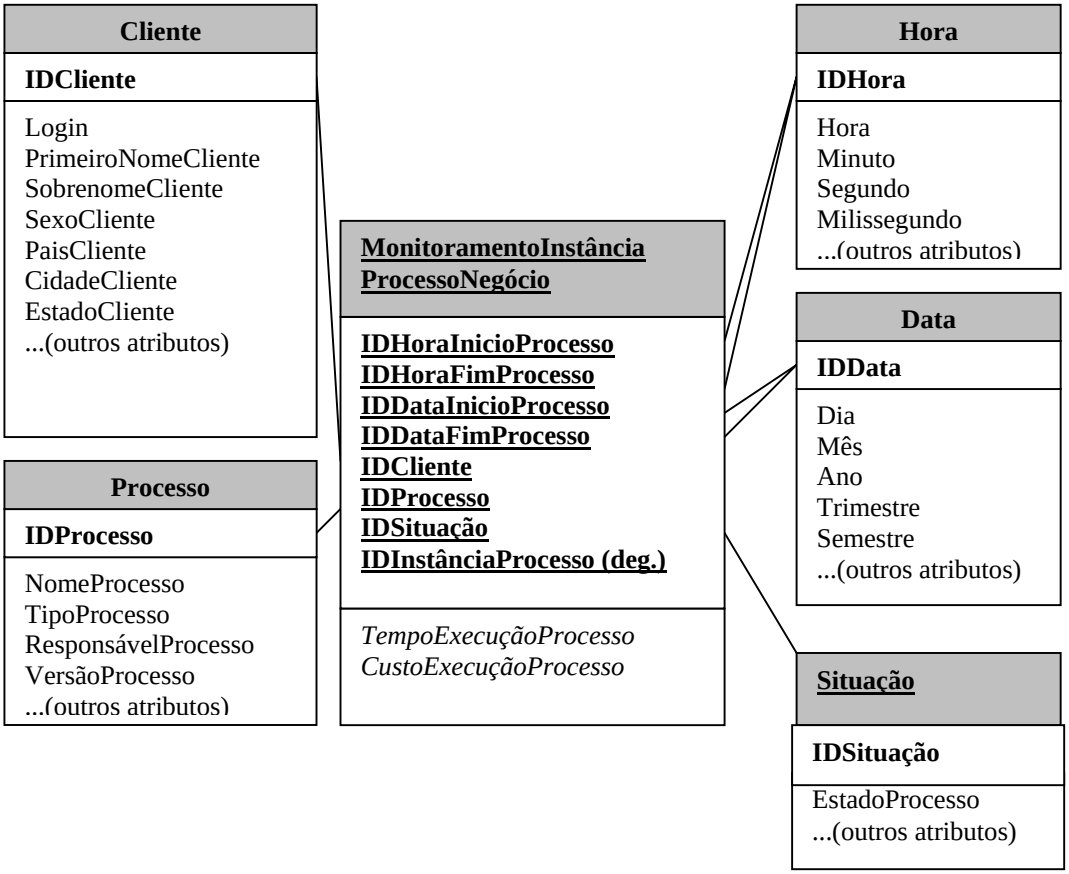


Figura 6.2 – *Data Mart* de monitoramento de instâncias de processos de negócio

A tabela 6.2 relaciona as dimensões do *Data Mart de monitoramento de instâncias de processos de negócio* e suas principais características. As dimensões presentes são: *Data, Hora, Processo, InstânciaProcesso, Cliente, Situação* e o grão da tabela de fatos *MonitoramentoProcessoNegócio* corresponde a um processo de negócio da empresa.

Dimensão	**Tipo**	**Tamanho**
Data	*Role Playing*	Mediana
Hora	*Role Playing*	Grande
Situação	Normal	Pequena
Processo	Compartilhada	Pequena
Cliente	*Slowly Changing*	Grande
InstânciaProcesso	Degenerada	Grande

Tabela 6.2 – Lista das dimensões do *Data Mart* de instâncias de processos

A dimensão *Processo* é a principal dimensão do diagrama. Ela é muito útil para a confecção de consultas relacionadas com a eficácia e a eficiência dos processos de negócio da

empresa. Através dela é possível avaliar os processos de negócio segundo as métricas da BSC para os processos internos. Ou seja, é possível avaliar a conversão de vendas em função do número de finalizações corretas do processo de negócio *Venda*.

Através da dimensão *Processo* o analista poderá saber qual o tempo médio de execução de uma instância de processo, isto é, por intermédio deste tipo de análise é possível descobrir indícios de problemas. Por exemplo, supondo que o tempo médio de execução (com sucesso) das instâncias do processo de negócio *Venda* tenha sido estipulado como 1 minuto e, um analista percebe que algumas demoram 3 minutos, é fácil detectar a existência de irregularidades.

A partir desse *Data Mart* ele poderá fazer operações de *drill-across* através da dimensão *Processo* e do *IDInstanciaProcesso*, avaliando qual o tempo consumido por cada atividade da instância do processo e descobrir a razão do atraso. Ele poderá avaliar os tempos médios de execução das instâncias dos serviços que compõem cada atividade. A técnica de *drill-across* consiste em ligar duas ou mais tabelas de fatos de *Data Marts* distintos, que possuam a mesma granulosidade e algumas dimensões compartilhadas (KIMBALL e ROSS, 2002).

Os conceitos de processos de negócio, atividades e serviços Web compõem uma hierarquia. Uma hierarquia pode ser a base do processo de agregação de dados, facilitando a exploração entre os diferentes níveis de detalhe em uma estrutura multidimensional. Através da dimensão *Processo* é possível explorar as atividades que compõem um processo de negócio e também explorar os serviços Web representados em outros *Data Marts*.

A dimensão *InstânciaProcesso* é uma dimensão do tipo degenerada. Uma dimensão é dita degenerada quando só existe um valor que corresponde a algum objeto do mundo real na tabela de fatos e, todos os demais atributos já estão representados na tabela de fatos ou em alguma outra dimensão (KIMBALL, 1998).

Neste *Data Mart*, *IDInstanciaProcesso* representa um identificador único para cada instância de processo de negócio e, através dele será possível correlacionar uma determinada instância de processo com suas instâncias de serviços Web. Vale a pena ressaltar que os identificadores de instâncias são gerados durante o processo de extração e carga (ETL) dos *Data Marts*.

A dimensão tempo é muito importante nesse *Data Mart* e requer uma tratamento diferenciado. Por esse motivo, optamos por dividi-la em duas dimensões (*Data* e *Hora*). As dimensões *Data e Hora* são dimensões com vários papéis (*Role Playing)* e aparecem mais de uma vez na tabela de fatos conforme ilustra a figura 6.2. A dimensão *Hora*, diferentemente da proposição de Kimball e Merz (2000) possui granulosidade de milésimo de segundo. Este nível de detalhe se faz necessário, pois a execução de um serviço Web pode consumir, em alguns casos, menos de um segundo.

Com relação ao tempo total de execução da instância de um processo de negócio é necessário esclarecer que ele é calculado através da seguinte fórmula:

$$\mathbf{T}_{\text{ExecuçãoProcesso}} = \mathbf{T}_{\text{FinalUltimoServiçoWebÚltimaSubprocesso}} - \mathbf{T}_{\text{InicalPrimeiroServiçoWebPrimeiraSubprocesso}}$$

$$\mathbf{T}_{\text{FinalUltimoServiçoWebÚltimaSubprocesso}} = \mathbf{T}_{\text{InicialUltimoServiçoWebÚltimaSubprocesso}} + \mathbf{T}_{\text{ExecuçãoServiçoWeb}}$$

Onde:

O tempo inicial $\mathbf{T}_{\text{InicalPrimeiroServiçoWebPrimeiraSubprocesso}}$ é capturado quando se inicia a instância do primeiro serviço Web da primeira atividade do processo. Essa medida pode ser obtida diretamente do log *InterRequest*.

O tempo final $\mathbf{T}_{\text{Final-UltimoServiçoWebÚltimaSubprocesso}}$ é obtido por intermédio da soma de duas parcelas. A primeira parcela $\mathbf{T}_{\text{InicialUltimoServiçoWebÚltimaSubprocesso}}$ é o momento em que se iniciou a execução da instância do último serviço Web do processo de negócio. Esta medida também pode ser obtida através do log *InterRequest*. A segunda parcela $\mathbf{T}_{\text{ExecuçãoServiçoWeb}}$ diz

respeito ao tempo de execução da instância do último serviço Web e pode ser obtido do log *InterResponse.*

A fórmula para calcular o tempo total de execução de uma instância de processo de negócio leva em consideração não só os tempos de processamento de cada serviço Web como também as diversas latências possíveis (rede, caches, filas, entre outros).

A dimensão *Cliente,* uma dimensão tradicional nos *Data Warehouses,* é do tipo *slowly changing,* o que implica que este registrará as mudanças de status do cliente que ocorrem ao longo do tempo. Os atributos dessa tabela podem variar em função da natureza das informações ali registradas. Por exemplo, se o analista deseja avaliar o perfil de compras de um determinado cliente, o *Data Mart* deverá ser capaz de registrar as variações que ocorrem ao longo do tempo. Uma alternativa para acomodar essa situação envolve a criação de novos campos que acomodem os valores do perfil original, do perfil atual, além da data de última atualização do perfil.

A dimensão *Cliente,* quase tão importante quanto a dimensão *Processo,* permite que se construam análises que se alinhem com a perspectiva dos clientes externos da metodologia BSC. Através dessa dimensão é possível fazer uso de métricas comportamentais e averiguar como os clientes interagem com os processos de negócio da empresa. Desse modo, um analista pode investigar o grau de satisfação de um cliente quando se estuda a freqüência de chamadas de um processo de negócio, seus tempos de execução e volume de erros das instâncias.

Algumas análises que envolvam a satisfação dos clientes requerem a participação da dimensão *Situação*. Ela é uma dimensão pequena que registra o tipo de finalização das instâncias dos processos de negócio. Por intermédio dessa dimensão podemos avaliar o tempo médio de execução de processos do tipo *Venda* que concluíram com situação “OK”. Outra

análise possível diz respeito ao custo mensal de execução do processo de negócio *Controle* que concluíram com situação “Erro”.

O *Data Mart* de monitoramento de instâncias de processos de negócio busca incrementar o desempenho dos processos para que resultem em maiores taxas de conversão. Neste ponto estamos em sintonia com a arquitetura DWB (KIMBALL, 1998) apesar de não representarmos um *Data Mart* isolado para os processos *Venda* e *Controle*. Nossa decisão de não apresentar os processos de negócio em Data Marts exclusivos e distintos repousa em uma seguinte observação de ordem prática. O *Data Mart* de monitoramento de processos de negócio permite que o analista tenha uma visão geral e de alto nível de todos os processos e suas atividades

Para concluir esta seção, ressaltamos que o *Data Mart* de monitoramento de instâncias de processos de negócio se alinha com a perspectiva analítica das camadas de processos de negócio apresentadas no capítulo 3 e as investigações deste *Data Mart* permitem que os analistas avaliem, sem se aprofundar em detalhes operacionais, como a composição de serviços Web pode afetar o desempenho da empresa.

6.3.2 *Data Mart* de monitoramento de instâncias de serviços Web

O *Data Mart* de monitoramento de instâncias de serviços Web, ilustrado na figura 6.3, se alinha com a perspectiva analítica da camada de serviços Web discutida na seção 3.4 desta dissertação. Ele tem como objetivo monitorar as instâncias dos serviços Web de um processo de negócio.

O *Data Mart* de monitoramento de instâncias é um pouco mais simples que o anteriormente apresentado e permite que um gestor monitore as atividades (materializadas como serviços Web) de um processo de negócio ao longo do tempo. Ele pode ser encarado como um refinamento do *Data Mart* de monitoramento de instâncias de processos de negócio.

A tabela 6.3 relaciona as dimensões do *Data Mart* e suas características. As dimensões presentes no *Data Mart* são: *Data, Hora, OperaçãoServiçoWeb, IDInstânciaServiçoWeb, Cliente e Situação* e o grão da tabela de fatos *MonitoramentoInstânciasOperacaoServiçoWeb* corresponde a um serviço de um processo de negócio da empresa.

Dimensão	**Tipo**	**Tamanho**
Data	*Role Playing*	Mediana
Hora	*Role Playing*	Grande
OperaçãoServiçoWeb	Normal	Pequena
Situação	Normal	Pequena
Cliente	*Slowly Changing*	Grande
IDInstânciaServiçoWeb	Degenerada	Grande
IDInstânciaProcessoNegócio	Degenerada	Grande

Tabela 6.3 – Lista das dimensões do *Data Mart* de instâncias de serviços Web.

As dimensões *Cliente e Situação* se valem das mesmas considerações do *Data M*art de monitoramento de instâncias de processos de negócio. Por isso, nenhum comentário adicional é necessário.

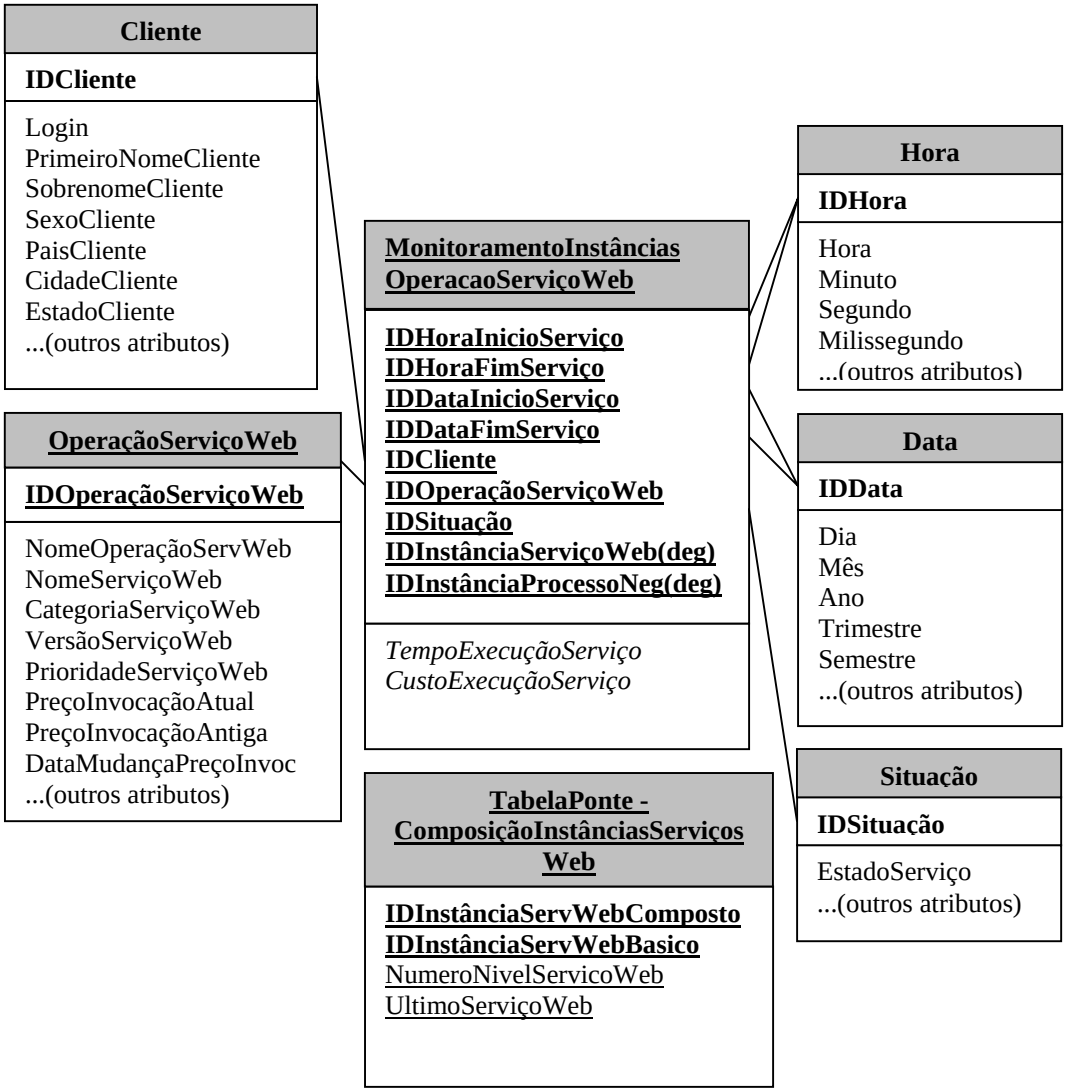


Figura 6.3 – *Data Mart* de monitoramento de instância de serviços Web

Neste *Data Mart*, *IDInstanciaServiçoWeb* e *IDInstânciaProcessoNeg* são dimensões degeneradas. Elas são representadas por um identificador único para cada instância de serviço e processo.

A tabela *ComposiçãoInstânciasServiçosWeb* é uma tabela do tipo ponte. As tabelas pontes são utilizadas para representar estruturas hierárquicas do tipo árvore, como por exemplo, uma empresa matriz com uma série de subsidiárias, onde cada uma deles pode fazer compras na matriz (KIMBALL, ROSS, 2002). Esse raciocínio é análogo ao da composição de serviços Web, onde um serviço Web composto pode invocar um outro serviço composto e este por sua vez invocar serviços Web básicos no final da hierarquia.

A tabela ponte funciona como um atalho. Ela permite que as consultas e junções SQL possam ser implementadas mais facilmente, por isso ela requer o uso de dois atributos. Um deles indica o número de níveis hierárquicos e o outro que indica se aquele nível é uma folha.

A tabela *ComposiçãoInstânciasServiçosWeb* representa a composição de instâncias de serviços Web compostos, isto é, para cada instância de serviço Web composto há uma associação de todas as instâncias de serviços Web básicos invocados pelo primeiro.Um exemplo de composição de invocações e respostas está representa no exemplo da seção 6.2 deste capítulo.

As dimensões temporais merecem as mesmas considerações das anteriormente apresentadas. Entretanto, é importante fazer a seguinte observação. Através dessas dimensões é possível determinar o tempo total de execução de uma instância de um serviço.

O tempo total de execução da instância de um serviço Web composto é calculado através da fórmula:

$$\mathbf{T}_{\text{ExecuçãoInstância}} = (\mathbf{T}_{\text{InicialUltimoServiçoWeb}} + \mathbf{T}_{\text{ExecuçãoServiçoWeb}}) - \mathbf{T}_{\text{InicalPrimeiroServiçoWeb}}$$

Onde o tempo inicial $\mathbf{T}_{\text{InicalPrimeiroServiçoWeb}}$ é capturado quando se inicia o primeiro serviço Web da composição. Essa medida pode ser obtida do log *InterRequest*.

O tempo $\mathbf{T}_{\text{InicialUltimoServiçoWeb}}$ é obtido diretamente do log *InterRequest*. A segunda parcela diz respeito ao tempo de execução do último serviço Web da composição. Este tempo também pode ser obtido do log *InterResponse*.

A dimensão *OperaçãoServiçoWeb* é a principal dimensão do diagrama, ela é muito útil para a confecção de consultas relacionadas com o desempenho das instâncias de um serviço Web (básico ou composto). Através dessa dimensão podemos fazer correlações interessantes entre os *Data Marts* de instância de processos de negócio. Por exemplo, supondo que um gestor acompanha o desenrolar do subprocesso *Negociação* e, ele percebe que custo

externo da execução de algumas instâncias é muito alto para alguns poucos clientes. Esse gestor poderá fazer operações de *drill-down* (nas instâncias de serviços Web compostos) a fim de descobrir qual operação ou serviço Web causa o embaraço. Eventualmente, ele pode descobrir que o serviço *VerifyCreditCard* foi instanciado diversas vezes e obteve como resposta “cartão bloqueado - roubo” para as invocações da operação de checagem do cartão. Neste caso, o gestor perceberá como o custo da instância do processo de negócio foi afetado pelo excessivo número de invocações do serviço Web de validação, onde cada invocação possui um custo associado.

Lembramos que análises equivalentes podem ser empreendidas em relação ao tempo total de execução do subprocesso, neste caso o tempo total será afetado pelo somatório das inúmeras parcelas de tempo das repetidas invocações do serviço Web.

Para finalizar esta seção, destacamos que o *Data Mart* de monitoramento de serviços Web tem como objetivo auxiliar os gestores de processos a acompanhar não só o desempenho dos serviços Web (básicos e compostos) como também compreender como cada uma delas influi no tempo e nos custos de um processo, isto é, através deste *Data Mart* podemos investigar o impacto financeiro que cada atividade finalizada com sucesso (ou não) causa na cadeia de valores da empresa.

6.3.3 *Data Mart* de monitoramento de qualidade de serviços Web

O *Data Mart* de monitoramento de qualidade, ilustrado na figura 6.4, se alinha com a perspectiva analítica da camada de serviços Web discutida na seção 3.4. Ele tem como objetivo monitorar alguns aspectos de *QoS* dos serviços Web.

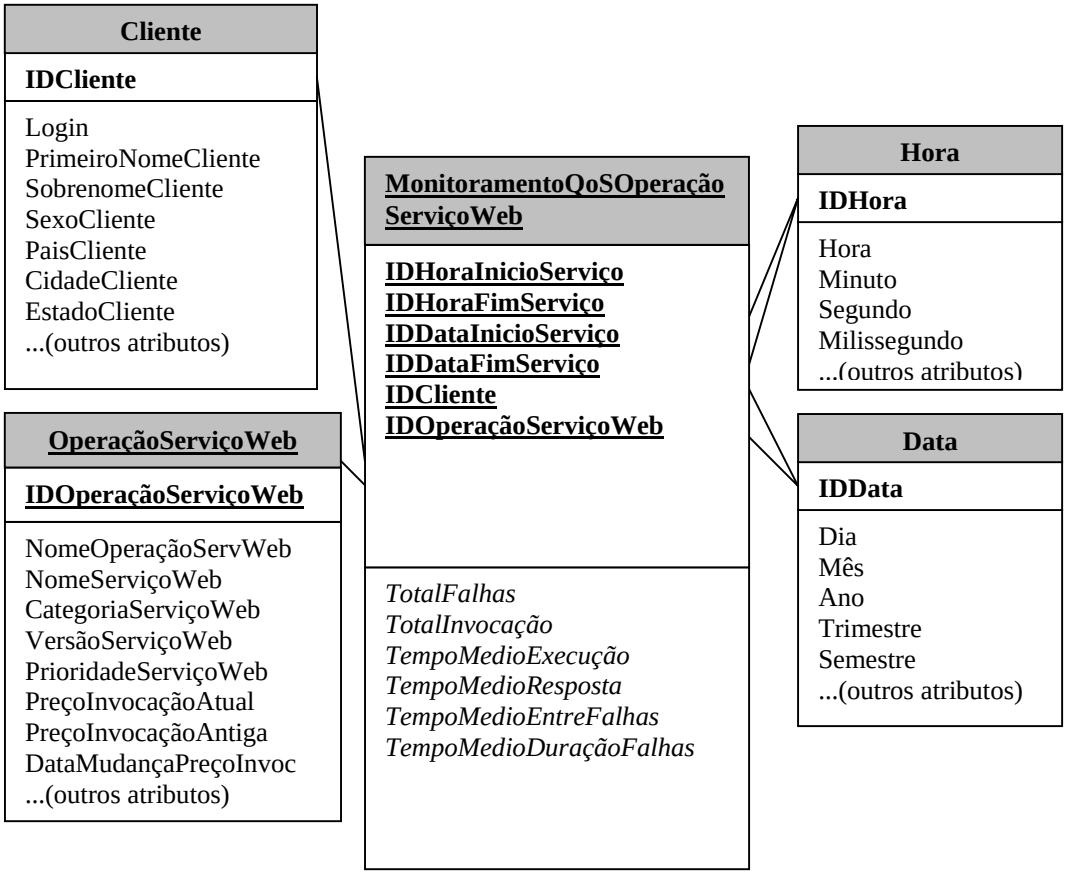


Figura 6.4 – *Data Mart* de monitoramento de QoS serviços Web

O *Data Mart* de monitoramento de QoS possui algumas características especiais. Ele é do tipo *instantâneo* (*Snapshot Schema*). Segundo Kimball e Ross (2002), essa categoria de *Data Mart* é capaz de armazenar o estado atual de uma transação de negócio. Isso quer dizer que o Data Mart de qualidade não tem como objetivo acompanhar as transações de negócio. Seu objetivo é armazenar a situação atual de um conjunto de transações.

A tabela 6.4 relaciona as dimensões e as características do *Data Mart*. As dimensões presentes no *Data Mart* são: *Data, Hora, OperaçãoServiçoWeb* e, *Cliente.* O grão da tabela de fatos *MonitoramentoQoSOperacaoServiçoWeb* corresponde a um serviço de um processo de negócio da empresa.

Dimensão	**Tipo**	**Tamanho**
Data	*Role Playing*	Mediana
Hora	*Role Playing*	Grande
OperaçãoServiçoWeb	Normal	Pequena
Cliente	*Slowly Changing*	Grande

Tabela 6.4 – Lista das dimensões do *Data Mart* de *QoS*

O *Data Mart* de monitoramento de qualidade é muito simples. O grão da tabela de fatos é a operação do serviço Web e os fatos são de natureza numérica. O conjunto das dimensões não requer muitos comentários, exceto pelo fato de podermos representar novas dimensões no *Data Mart*.

Por exemplo, para avaliar as falhas das operações dos serviços Web podemos representar uma dimensão *Falha*. A dimensão *Falha* mantém as possíveis falhas das operações monitoradas através dos intermediários *InterRequest* (as falhas e operações capturáveis são especificadas nos arquivos de configuração da arquitetura WSLogA, seção 4.3.6).

Ao *Data Mart* também podemos acrescentar uma dimensão do tipo *Causal*. Dimensões do tipo *Causal* descrevem as condições do mercado em um determinado momento. Elas podem fornecer explicações ou pistas sobre investigações de QoS em função de eventos temporais, por exemplo, o gestor poderá investigar se o aumento do número de falhas nas invocações de operações dos serviços Web está relacionado com causas temporais, tais como, Natal, Dia das Crianças, entre outros.

As variáveis da tabela de fatos guardam estreita ligação com as métricas de qualidade de negócio apresentadas na seção 3.10. Elas permitem avaliar, por exemplo, o número total de invocações de um determinado serviço Web e o número total de falhas. O resultado da divisão dessas medidas é uma indicação da confiabilidade de um serviço.

A métrica disponibilidade também guarda relação direta com os fatos do *Data Mart*. Por exemplo, o tempo médio de execução de um determinado serviço Web é uma medida que pode ser obtida diretamente nesse *Data Mart*.

No próximo capítulo apresentamos a conclusão deste trabalho, suas limitações e trabalhos futuros.

CAPÍTULO 7

CONCLUSÃO

O gerenciamento de processos de negócio requer procedimentos que garantam o progresso do ciclo de vida dos processos de negócio. Como resultado, será possível diagnosticar suas limitações. Os serviços Web provêm os mecanismos necessários para garantir a evolução dos processos de negócio e, com vistas a explorar as novas oportunidades oferecidas por esse paradigma, é importante que as empresas monitorem sua utilização. Neste sentido, as empresas podem contar com a arquitetura WSLogA, pois ela promove o monitoramento de processos de negócio baseados em serviços Web. A arquitetura pode ser utilizada para investigar aspectos funcionais de processos de negócio.

O projeto e desenvolvimento da WSLogA foi pautado por três pilares básicos:

1. Ser o menos invasiva possível, isto é, o uso da arquitetura não deve causar nenhum impacto na operação ou desempenho dos serviços disponíveis;
2. Capturar dados com diferentes níveis de granulosidade, característica que suportaria análises multidimensionais refinadas;
3. Prover um mecanismo simples e não proprietário de monitoramento e testes de serviços Web.

Processos de negócio, atividades e serviços Web ocupam diferentes camadas de abstração de uma empresa e seu monitoramento é viável graças aos dados capturados pela arquitetura WSLogA. Os dados podem alimentar *Data Marts* de processos de negócio e de

serviços Web que permitem que gestores avaliem o negócio do ponto de vista gerencial e estratégico.

Como estudo de caso, desenvolvemos um conjunto de intermediários que capturam a utilização de serviços Web. Desenvolvemos também um conjunto de serviços Web que compõem um cenário de uma livraria virtual, onde foram representados processos de negócio baseados em serviços Web, simulando as interações com os clientes e também com os parceiros de negócio.

Finalmente, modelamos *Data Marts* que permitem aos gestores observarem, a um só tempo e sob diversas óticas, a utilização e a *QoS* dos serviços Web e dos processos de negócio.

7.1 Contribuições

As principais contribuições deste trabalho são:

1. O estudo de mecanismos para o monitoramento de processos de negócio baseados em serviços Web em ambientes de comércio eletrônico;
2. Uma proposta de arquitetura, baseada em intermediários SOAP, para captura de dados de utilização de serviços Web;
3. Uma proposta de estrutura flexível de logs de serviços Web no formato XML, capaz de registrar, com diferentes granulosidades, dados de monitoramento de processos de negócio baseados em serviços Web;
4. A definição de critérios de qualidade para avaliar o desempenho de processos de negócio e dos serviços Web, sob as óticas dos gestores de processos e dos administradores de serviços Web;

5. A proposta de um ambiente analítico baseado em *Data Marts* que possibilitem sofisticadas análises multidimensionais, e permitam o cruzamento de dados oriundos de diversas fontes de dados.

Verificamos que as estruturas dos logs podem se adaptar para armazenar tanto uma requisição quanto uma resposta SOAP. Através do estudo da formação de mensagens neste formato, observamos que as estruturas dos logs acomodam qualquer conformação de uma mensagem SOAP. O modelo de log proposto foi capaz de registrar tanto as seções obrigatórias quanto opcionais de uma mensagem SOAP com qualquer o número de elementos.

A arquitetura WSLogA apresentou características interessantes. De acordo com nossos experimentos, ela introduziu sobrecarga mínima no ambiente de simulação e apresentou baixa necessidade de alteração ou interrupção dos serviços Web, uma vez que trabalha diretamente no SOAP *engine*.

A proposta de log no formato XML foi uma decisão que se mostrou acertada por uma série de fatores. Os logs XML representam as informações de forma mais rica e padronizada, pois armazenam os dados e seus descritores. Adicionalmente, este tipo de estrutura é capaz de registrar diferentes tipos de mensagens SOAP. Os logs apresentam razoável consumo de recursos de armazenamento, mas por outro lado, como esses recursos ficam cada vez mais baratos, o fator consumo de recursos é secundário quando se comparam com os benefícios.

O projeto do ambiente de captura de dados demonstrou que a proposta da arquitetura é factível e de simples implementação. Com o objetivo de comprovar a viabilidade de nossa proposta, realizamos alguns experimentos, que contemplaram as seguintes etapas:

1. Construção dos intermediários *InterRequest* e *InterResponse*, com capacidade de identificar individualmente as mensagens (requisição e resposta), requisito fundamental para o processamento no ambiente multidimensional;
2. Construção dos serviços Web que emularam alguns processos de negócio de uma livraria virtual. A motivação que levou a construção destes elementos foi à ausência de serviços Web reais disponíveis;
3. Especificação de um conjunto de *Data Marts* de utilização e de monitoramento de serviços Web e processos de negócio.

7.2 Limitações e dificuldades encontradas

As limitações da arquitetura dizem respeito ao relacionamento entre os intermediários SOAP e as mensagens criptografadas. Uma mensagem SOAP codificada não é registrada nos logs. Outra limitação dos intermediários diz respeito ao tratamento dos arquivos, do tipo MIME, anexados à mensagem.

Existe uma outra limitação herdada da atual infra-estrutura de serviços Web e redes. Apesar da associação HTTP-SOAP se revelar como a mais comum, ela possui algumas limitações. Por exemplo, o protocolo HTTP não guarda o estado da conexão e não oferece nem garantia de entrega nem ordem de entrega de pacotes. Além disso, em casos onde não há largura de banda suficiente na rede podem ocorrer descartes de pacotes. Essas características tecnológicas podem dificultar o monitoramento e gerência de processos de negócio baseados em serviços Web, uma vez que não é possível garantir que todas as interações são efetivamente registradas nos logs dos serviços Web.

A principal dificuldade encontrada no decorrer deste trabalho surgiu no momento do monitoramento de um ambiente real de comércio eletrônico que fizesse uso de serviços Web.

Mesmo com a crescente difusão de serviços Web, não havia na época da implementação, nenhum local onde fosse possível utilizar a arquitetura WSLogA. Por esse motivo, fomos obrigados a simular um ambiente de utilização processos de negócio baseados em serviços Web.

7.3 Trabalhos futuros

Para concluir esta dissertação, propomos alguns trabalhos futuros:

1. Utilizar a WSLogA em aplicações reais, para que as vantagens e limitações sejam mais bem compreendidas;
2. Desenvolvimento e implementação dos *Data Marts* propostos, com o respectivo tratamento de dados dos logs, permitindo avaliar o potencial deste tipo de solução;
3. Para contornar as limitações de garantia de entrega do HTTP-SOAP, sugerimos utilizar o protocolo HTTPR (*HTTP Reliable*) que é uma extensão do protocolo HTTP1/1 (HTTPR, 2004). O protocolo garante a entrega ordenada de pacotes e, se necessário, a retransmissão da mensagem. Ressalta-se que o protocolo HTTPR ainda está em fase de homologação e a principal vantagem da mudança de protocolo de rede não requer a alteração de nenhum serviço Web. Outra forma de contornar as limitações do HTTP-SOAP reside em explorar a especificação *WS-Reliability* que garante a entrega de mensagens SOAP de forma ordenada e não duplicada (WS-RELIABILITY, 2004);
4. Explorar como a especificação *WS-Security* (NADALIN, 2004), recomendada recentemente, pode interferir na captura das mensagens SOAP. A especificação fornece três mecanismos de segurança para as mensagens SOAP (integridade, confidencialidade e autenticação) que são inseridos nos

cabeçalhos das mensagens SOAP. Os elementos recém definidos na especificação podem contribuir para o monitoramento da qualidade de serviços Web;

5. Avaliar a viabilidade implementação da arquitetura WSLogA em ambientes de computação em *grid* que façam uso intenso de serviços Web. Nesses ambientes, as exigências de qualidade de serviço são críticas, por isso, o monitoramento dos serviços é um fator crucial para o uso racional dos recursos computacionais. Dos potenciais ambientes de computação em *grid* destacamos o Globus Toolkit e o GriPhyN (FOSTER, KESSELMAN, TUECKE, 2001; FOSTER, KESSELMAN, NICK, TUECKE, 2002).

REFERÊNCIAS BIBLIOGRÁFICAS

ANDREWS, T.; CURBERA, F.; DHOLAKIA, H.; GOLAND, Y.; KLEIN, J.; LEYMANN, F.; LIU, K.; ROLLER, D.; SMITH, D.; THATTE, S.; TRICKOVIC, I.; WEERAWARANA, S. **Specification: Business Process Execution Language for Web Services**. Version 1.1 - 05 May 2003. Disponível em: <http://www.ibm.com/developerworks/library/ws-bpel>. Acesso em: 10 jun. 2004.

AUSTIN, D.; BARBIR, A.; FERRIS, C.; GARG, S. **Web services architecture requirements. W3C working draft. 2002**. Disponível em: <http://www.w3.org/TR/2002/WD-wsa-reqs-20021114#id2604831>. Acesso em: 20 set. 2003.

AXIS Architecture Guide. Disponível em: <http://ws.apache.org/axis/java/architecture-guide.html>. Acesso em: 07 abr. 2004.

AZEVEDO JUNIOR, V. **Webtransact-Em: Um Modelo Para A Execução Dinâmica De Serviços Web Semanticamente Equivalentes**. Dissertação de Mestrado. COPPE/UFRJ – Brasil. 2003.

BARRY, D. K., **Web Services and Services-Oriented : The Savvy Manager´s Guide.** San Francisco: Morgan Kaufmann Publishers, 2003.

BELLWOOD, T.; CLÉMENT, L.; EHNEBUSKE, D.; HATELY, A.; HONDO, M.; HUSBAND, Y. L.; JANUSZEWSKI, K.; LEE, S.; JANUSZEWSKI, B.; MUNTER J.; RIEGEN C. **UDDI version 3.0 UDDI spec. technical committee specification.** 2002 Disponível em: <http://uddi.org/pubs/uddi-v3.00-published-20020719.htm#_Toc42047184>. Acesso em: 20 jun. 2003.

BRITTENHAM, P.; DURAND J.; KLEIJKERS L.; STOBIE K. **WS-I monitor tool functional specification.** Disponível em: <http://www.ws-i.org/Testing/Specs/MonitorFunctionalSpecification_BdAD_1.02.pdf>. Acesso em: 10 nov. 2003.

BRUNNER, J. R.; COHEN, F.; CURBERA, F.; GOVONI, D.; HAINES, S.; KLOPPMANN, M.; MARCHAL, B.; MORRISON, K. S.; RYMAN, A.; WEBER, J.; WUTKA M. **Java™ Web Services Unleashed**. Indianapolis: Sams Publishing, 2002.

BOCHMANN, G.; KERHERVÉ B.; LUTFIYYA H.; SALEN M-V. M.; YE, H. **Introducing QoS to electronic commerce applications**. ISEC 2001. Hong Kong: 2001. p. 138-147. Disponível em: <http://beethoven.site.uottawa.ca/dsrg/PublicDocuments/Publications/Boch01a.pdf>. Acesso em: 20 jul 2004.

BOOTH, D.; HAAS, H.; MCCABE, F.; NEWCOMER, E.; CHAMPION, M.; FERRIS, C.; ORCHARD, D. **Web services architecture - W3C working draft.** Disponível em: <http://www.w3.org/TR/ws-arch/>. Acesso em: 15 out. 2003.

BURLTON R. **Business Process Management: Profiting From Process**, Indianapolis: Sams Publishing, 2001.

CARDOSO, J.; SHETH, A.; MILLER, J. **Workflow Quality of Service**. Disponível em: <http://dme2.uma.pt/~jcardoso/Research/Papers/CSM02-TM.pdf>. Acesso em 04 fev. 2004.

CARMEL, D.; MAAREK Y. S.; MANDELBROD M.; MASS Y. SOFFER A., **Searching XML documents via XML fragments**. 2003. p. 151 – 158.

CASATI, F.; ILNICKI, S.; JIN, L. Adaptive and dynamic service composition in eFlow. In: INTERNATIONAL CONFERENCE ON ADVANCED INFORMATION SYSTEMS ENGINEERING, 12., 2000. Stockholm. **Proceedings of th 12th International Conference On Advanced Information Systems Engineering** Stockholm:. p. 13-31. 2000.

CASATI, F.; SHAN, M. Dynamic and adaptive composition of e-services. **Information Systems**, v. 26, n. 3, p. 143-163, 2001.

CHATTERJEE, S.; WEBBER J. **Developing enterprise web services: an architect´s guide**. New Jersey: Prentice Hall, 2003.

CHAPPELL, D.; JEWELL, T. **Java Web Services**. California: O'Reilly Books, 2002.

CHINNICI, R.; GUDGIN, M.; MOREAU, J-J.; SCHLIMMER, J.; WEERAWARANA, S. **Web services description language (WSDL) version 1.2. Part 1: Core language W3C working draft.** Disponível em: <http://www.w3.org/TR/wsdl12/>. Acesso em: 20 ago. 2003.

COYLE, F. P. **XML, Web Services, and the Data Revolution**. Boston: Addison-Wesley, 2002.

CUTLER, M.; STERNE, J. **E-metrics business metrics for the new economy.** Disponível em: <http://www.emetrics.org/articles/emetrics>. Acesso em: 30 mai. 2003.

CRUZ, S. M. S.; CAMPOS, M. L. M. Data mart de navegação: um ambiente baseado em XML. Simposio Brasileiro de Banco de Dados, 17. 2002. **Anais do I Workshop de Teses e Dissertações em Banco de Dados**. WTDBD, Gramado: 2002.

CRUZ, S. M. S.; CAMPOS, M. L. M.; PIRES, P. F., CAMPOS, L. M. Monitoring e-business web services usage through a log based architecture. In: Simposio Brasileiro de Banco de Dados, 18. 2003. Manaus **Anais Simpósio Brasileiro de Banco de Dados** Manaus: 2003. p. 267-280.

CRUZ, S. M. S.; CAMPOS, M. L. M.; PIRES, P. F., CAMPOS, L. M Monitoring e-business web services usage through a log based architecture. In: IEEE International Conference on Web Services 2,. 2004. San Diego. **Proceedings of the 2nd International Congress of Web Services** San Diego: 2004. p. 61-67.

CURBERA, F.; MUKHI N.; WEERAWARANA, S. On the emergence of a web services component model. In: 6 th INTERNATIONAL WORKSHOP ON COMPONENT-ORIENTED PROGRAMMING AT ECOOP, 2001. Budapest. **Proceedings of the 6th International Workshop on Component-Oeiented Programming** Budapest: 2001.

Disponível em: <http://research.microsoft.com/users/cszypers/events/WCOP2001/Curbera>. Acesso em: 01 jun. 2003.

DAVENPORT T. H. **Reengenharia de processos : como inovar na empresa através da ecnologia de informação.** 5 ed., Rio de Janeiro: Campus, 1994.

DIALANI, V.; MILES S.; MOREAU, L.; ROURE D.; LUCK M. Transparent Fault Tolerance for Web Services Based Architectures. In: **Proceedings of the 8th International Europar Conference**, Paderborn, Germany, 2002. pp. 889-898. Disponível em: <http://www.ecs.soton.ac.uk/~sm/dialani02transparent.pdf>. Acesso em: 03 fev. 2004.

FELL, S. **Pocket SOAP – tcptrace.** Disponível em: <http://www.pocketsoap.com/tcptrace/>. Acesso em: 20 nov. 2003.

FENTON, N. E.; NEIL, M. **Software Metrics: Roadmap**. Disponível em: <http://www.cs.ucl.ac.uk/staff/A.Finkelstein/fose/finalfenton.pdf>. Acesso em : 08 out. 2004.

FERNANDES, B. H. R. **Competência e Performance Organizacional: Um Estudo Empírico**. Tese de Doutorado. USP – Brasil. 2004. Disponível em: <http://www.teses.usp.br/teses/disponiveis/12/12139/tde-05042004-161002/publico/Tese.pdf >. Acesso em: 08 out. 2004.

FOSTER, I.; KESSELMAN, C.; TUECKE, S. **The Anatomy of the Grid: Enabling Scalable Virtual Organizations**. Disponível em: <http://www.globus.org/research/papers/anatomy.pdf>. Acesso em : 20 nov. 2004.

FOSTER, I.; KESSELMAN, C.; NICK, J.; TUECKE, S. **The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration**. Disponível em: <http://www.globus.org/research/papers/ogsa.pdf>. Acesso em : 20 nov. 2004.

GONÇALVES, M. A.; PANCHANATHAN G.; RAVINDRANATHAN U.; KROWNE A.; FOX, E. A.; JAGODZINKI F.; CASSEL L. **An XML log standard and tool for digital library logging analysis**. In: EUROPEAN CONFERENCE RESEARCH AND ADVANCED TECHNOLOGY FOR DIGITAL LIBRARIES, 6., 2002. Rome, p. 129-143.

GRAHAM, S. *et al*. **Building Web Services with Java™: Making sense of XML, SOAP, WSDL, and UDDI**. Indianapolis: Sams Publishing, 2001.

GREMBERGEN, W. V.; AMELINCKX, I. Measuring and Managing E-business Projects through the Balanced Scorecard. In: **Proceedings of the 35th Hawaii International Conference on System Sciences** – 2002. Disponível em: <http://csdl.computer.org/comp/proceedings/hicss/2002/1435/08/14350258.pdf>. Acesso em: 23 out. 2004.

GROSSO, P.; VEILLARD, D., **XML fragment interchange - W3C candidate recommendation.** Disponível em: <http://www.w3.org/TR/xml-fragment>. Acesso em: 20 ago. 2003.

GOTTSCHALK, K.; GRAHAN S.; KREGER H.; SNELL J. Introduction to web services architecture. IBM **Systems Journal**, v. 41, n. 2, 2002. Disponível em: <http://www.research.ibm.com/journal/sj/412/gottschalk.pdf>. Acesso em: 20 jun. 2003.

GUNTHER N., **The Practical performance analyst**. 2.Ed. Lincoln, iUniverse .com, 2000. Authors Choice Press. Disponível em <http://www.perfdynamics.com/>. Acesso em: 05 jul. 2004.

HAMMER, M., **The Reengineering Revolution**. New York: HarperCollins Publishers. 1995.

HEUVEL, W.; YANG, J.; PAPAZOGLOU M.P. Service Representation, Discovery, and Composition for E-marketplaces, in: **Proceedings of the 9th International Conference Cooperative Information Systems** (CoopIS 2001), Trento, Italy, 2001, pp. 270-284.

HTTPR - **A Primer for HTTPR: an Overview of the Reliable HTTP Protocol**. Disponível em: <http://uk.builder.com/whitepapers/0,39026692,60021908p-39000989q,00.htm>. Acesso em: 05 nov. 2004.

INMON, W. H. **Building the Data Warehouse**. New York: John Wiley & Sons, 3rd edition, 2002.

JSDK - Java logging overview. 2001. Disponível em: <http://java.sun.com/j2se/1.4.2/docs/guide/util/logging/overview.html>. Acesso em: 12 mai. 2004.

IRANI, R.; BASHA S. J. **AXIS the next generation Java SOAP**. Birmingham: WROX Press, 2002.

ISO-8402, ISO 8402-1994, **Quality management and quality assurance - vocabulary**. 2000. Disponível em: <http://www.iso.ch/iso/en/CatalogueWithdrawnDetailPage.CatalogueWithdrawnDetail?CSNUMBER=20115>. Acesso em: 22 jun. 2004.

KAPLAN, R. S.; NORTON, D. P. **The Balanced Scorecard - Measures That Drive Performance.** Harvard Business Review, Boston, v. 70, n. 1 pp.71-79. jan-feb 1992.

KAPLAN, R. S.; NORTON, D. P. **The Balanced Scorecard: Translating Strategy into Action**. Boston: Harvard Business School Press. 1996.

KAYE, D. **Loosely coupled: the missing pieces of web services**. Califórnia: RDS Press, 2003.

KIMBALL, R.; REEVES, L.; ROSS, M.; THORNTHWAITE W. **The data warehouse lifecycle toolkit: expert methods for designing, developing and deploying data warehouses**. New York: John Wiley & Sons, 1998.

KIMBALL, R.; MERZ, R. **The data webhouse toolkit: building the web- enabled data warehouse**. New York: John Wiley & Sons, 2000.

KREGER, H. **Web services conceptual architecture (WSCA 1.0) 2001**. Disponível em: <http://www-3.ibm.com/software/solutions/webservices/pdf/WSCA.pdf>. Acesso em: 15 jun 2003.

LEYMANN, F.; ROLLER, D.; SCHMIDT, M.-T. **Web services and business process management**. In: New Developments in Web Services and E-commerce Volume 41, Number 2, 2002. Disponível em <http://www.research.ibm.com/journal/sj/412/leymann.html>. Acesso em: 15 mai. 2003.

LLEWELLY, N.; ARMISTED, C. Business Process Management: Exploring Social Capital within Processes. **International Journal of Service Management Review**., 2000, v 11. n. 3. pp 205-243.

LOG4J – Logging services. 2004. Disponível em: <http://logging.apache.org/log4j/docs/index.html>. Acesso em: 10 mai. 2004.

LOGJ Alphaworks – logging toolkit for Java. 2004. Disponível em: <http://www.alphaworks.ibm.com/tech/loggingtoolkit4j>. Acesso em: 15 mai. 2004.

LOVELOCK, C.; WRIGHT L. **Serviços, marketing e gestão**. São Paulo: Saraiva, 2001.

MANI, A.; NAGARAJAM, A **Understanding quality of service for Web services**. Disponível em <http://www-106.ibm.com/developerworks/library/ws-quality.html>. Acesso em: 12 mai. 2003.

MENASCÉ, D. A.; ALMEIDA, V. A. F. **Capacity planning for web services, metrics, models and methods.** New Jersey: Prentice Hall, 2002.

MENASCÉ, D. A. QoS issues in web services. **IEEE Internet Computing online**, v. 6, n. 6, p. 72-75, Nov./Dec. 2002. Disponível em: <http://cs.gmu.edu/~menasce/papers/IEEE-IC-QoSIssuesWebServices-November-2002.pdf >. Acesso em: 18 jun. 2004.

MENASCE, D. A; RUAN, H.; GOMMA, H. A. Framework for QoS-aware software components. In: WORKSHOP ON SOFTWARE AND PERFORMANCE ARCHIVE, 4., 2004, Redwood Shores. **Proceedings of the 4th international workshop on Software and performance** Redwood Shores : 2004. p. 186 - 196.

MITRA, N. **SOAP version 1.2. Part 0: Primer W3C recommendation 24 June 2003**. Disponível em: <http://www.w3.org/TR/soap12-part0/>. Acesso em: 30 jun. 2003.

MYERSON, J.M., **Guarantee your Web service with an SLA** – Introduction, architecture, and testing mechanisms. Disponível em: <http://www-106.ibm.com/developerworks/webservices/library/ws-sla/>. Acesso em: 02 set. 2003.

NADALIN, A.; KALER, C.; HALLAM-BAKER, P.; MONZILLO, R. **Web Services Security: SOAP Message Security 1.0**. Disponível em: <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf>. Acesso em: 05 nov. 2004.

NEWCOMER, E. **Understanding web services:** XML, WSDL, SOAP and UDDI. Boston: Addison-Wesley, 2002.

OLSINA, L.; GODOY, D.; LAFUETE, G.J.; ROSSI G. Specifying quality characteristics and attributes for websites. In: MURUGESAN, S.; DESPHANDE, Y. eds. **Web Engeneering**: software engineering and web application development. Springer-Verlag, 2001. p. 266-278. (Lecture Notes in Computer Science, v. 2016). Disponível em: <http://gidis.ing.unlpam.edu.ar/downloads/pdfs/Olsina_WebE.pdf>. Acesso em: 30 mai. 2003.

PADMANABHUNI, S.; GANESH, J.; MOITRA, D. **Web Services, Grid Computing and Business Process Management**: In: IEEE International Conference on Web Services 2,. 2004. San Diego. **Proceedings of the 2nd International Congress of Web Services** San Diego: 2004. pp. 666-673.

PORTER, M. E. **Competitive Advantage:** Creating & Sustaining Superior Performance. New York, The Free Press, 1998.

PRESSMAN, R. S. **Software engineering: a practitioner's approach**. 5. ed. McGraw Hill, 2000.

PUNIN, J. R.; KRISNAMOORTHY, M. S.; ZAKI, M. J. **LOGML - log markup language for web usage mining.** In: WEBKDD WORKSHOP 2001: MINING LOG DATA ACROSS ALL CUSTOMER TOUCHPOINTS 2001. San Francisco. **Proceedings of the webkdd workshop.** San Francisco: 2001. Disponível em: <http://www.cs.rpi.edu/~puninj/LOGML/WEBKDD01.ps.gz>. Acesso em: 02 ago. 2002.

RAN, S. A Model for web services discovery with QoS. **ACM SIGECOM Exchanges**. New York, v. 4, n. 1, p. 1-10, Spring 2003.

SCHAFFNER, B. **Debug SOAP apps with Apache TCP Tunnel/Monitor**. Disponível em: < http://builder.com.com/5100-6389-1049605.html>. Acesso em: 5 dez. 2003.

SEELY, S. **SOAP: cross platform web service development using XML**. New Jersey: Prentice Hall, 2001.

SMITH, H. **Making business processes manageable: the change to provide a new solution to an old challenge**. Disponível em: <http://www.fairdene.com/processes/BPM-WSJ-Smith1.pdf>. Acesso em: 14 set. 2004.

SMITH H.; FINGAR, P. **Business Process Management – The third Wave**. Florida: Meghan-Kiffer Press, 2003.

SOAP Toolkit 3.0. Disponível em:<http://www.microsoft.com/downloads/details.aspx?familyid=c943c0dd-ceec-4088-9753-86f052ec8450&displaylang=en>. Acesso em: 12 jan. 2004.

SWEIGER, M.; MADSEN, M. R.; LANGSTON, J.; LOMBARD, H. **Clickstream data warehousing**. New York: John Wiley & Sons, 2002.

TANEMBAUN, A. S. **Redes de Computadores - Tradução da 4ª edição americana**. Rio de Janeiro: Campus, 2004.

THATTE, S. **XLANG : Web Services for Business Process Design**. Disponível em: <http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm>. Acesso em: 10 jun. 2004.

UDDI, Specification, 2001. Disponível em: <http://www.uddi.org/specification.html>. Acesso em: 22 abr. 2004.

URISONO, C. H., **Tecnologia de Workflow: O impacto de sua utilização nos proessos de negócio. Um estudo de casos múltiplos.** Dissertação de Mestrado. USP – Brasil. 2003. Disponível em: <http://www.teses.usp.br/teses/disponiveis/12/12139/tde-08122003-233842/publico/Dissertacao_Carlos_Hiroshi.pdf>. Acesso em: 10 out. 2004.

WOHED, P.; AALST W. M. P.; DUMAS, M.; HOFSTEDE, A. H. M. **Analysis of Web Services Composition Languages**: The Case of BPEL4WS. Disponível em: <http://tmitwww.tm.tue.nl/research/patterns/download/bpel_er.pdf>. Acesso em: 10 jun. 2004.

WS-REALIABILITY - **Web Services Reliability (WS-Reliability) Version 1.0**. Disponível em: <http://developers.sun.com/sw/platform/technologies/ws-reliability.html>. Acesso em: 05 nov. 2004.

WSDL Specification, 2001 – W3C. Disponível em: <www.w3.org/TR/wsdl>. Acesso em: 19 abr. 2003.

XLIF - XML log interchange format. 2001. Disponível em: <http://openebxml.sourceforge.net/realization/components/comp_xlif/doc/xlif-20011015.html>. Acesso em: 20 jun. 2002.

XLF - extensible log format (XLF) initiative. 1998. Disponível em: <http://xml.coverpages.org/xlf.html>. Acesso em: 20 jun. 2002.

APÊNDICES

A-1 – Exemplo de log produzido pelo intermediário *InterRequest*

```
      <?xml version="1.0" encoding="UTF-8"?>
      <LogFile                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\Dissertação\WSLogs\WSLogARequest.xsd">
            <LogInfo>
                  <LogInfoID>1076378395734</LogInfoID>
                  <Producer>Web Services Log Architecture</Producer>
                  <Intermediary>InterRequest</Intermediary>
                  <Version>1.0</Version>
                  <LogDateTime>2004-Feb-09 23:59:55</LogDateTime>
                  <Row>
                        <MsgID>1076292004859</MsgID>
                        <MsgDate>2004-Feb-09</MsgDate>
                        <MsgTime>00:00:04</MsgTime>
                        <WSName>Login</WSName>
                        <WSOperation>validatePwd</WSOperation>
                        <WSParameters>
                              <Param>
                                    <Name>Username</Name>
                                    <Value>PeVdCCP</Value>
                              </Param>
                              <Param>
                                    <Name>Password</Name>
                                    <Value>ogbaserereri</Value>
                              </Param>
                        </WSParameters>
                        <WSHeaders>
                              <Header>
                                    <Name>criptography</Name>
                                    <Value>No</Value>
                                    <MustUnderstand>1</MustUnderstand>
                              </Header>
                              <Header>
                                    <Name>priority</Name>
                                    <Value>high</Value>
                                    <MustUnderstand>1</MustUnderstand>
                              </Header>
                        </WSHeaders>
                  </Row>
      ... Outros registros de requisição
            </LogInfo>
      </LogFile>
```

A-2 – Exemplo de log produzido pelo intermediário *InterResponse*

```
      <?xml version="1.0" encoding="UTF-8"?>
      <LogFile                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\Dissertação\WSLogs\WSLogAResponse.xdr">
            <LogInfo>
                  <LogInfoID>1076378390171</LogInfoID>
                  <Producer>Web Services Log Architecture</Producer>
                  <Intermediary>InterResponse</Intermediary>
                  <Version>1.0</Version>
                  <LogDateTime>2004-Feb-09 23:59:50</LogDateTime>
                  <Row>
                        <MsgID>1076292004859</MsgID>
                        <MsgDate>2004-Feb-09</MsgDate>
                        <MsgTime>00:00:04</MsgTime>
                        <WSName>Login</WSName>
                        <WSOperation>validatePwd</WSOperation>
                        <WSParameters>
```

```
                                    <Param>
                                          <Name>validatePwdReturn</Name>
                                          <Value>Ok!</Value>
                                    </Param>
                              </WSParameters>
                              <WSExecutionTime>0.110</WSExecutionTime>
                        </Row>
      ... Outras registros de respostas
            </LogInfo>
      </LogFile>
```

A-3 – Exemplo de um trecho de log do Apache-Tomcat tratado pelo *parser*

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- FileName : C:\Dissertação\Randomize\log0502.txt -->
<!-- WebServer: Apache/Tomcat 4.1.12 -->
<!-- Valve Pattern: %a %A %b %B %h %H %l %m %p %s %t %u %U %v -->
<LogWebServer                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\Dissertação\WSLogs\LogTomcatSchema.xsd">
      <XML2Log>
            <SequenceID>1</SequenceID>
            <RemoteIp>191.64.54.57</RemoteIp>
            <LocalIp>146.164.248.21</LocalIp>
            <BytesSent>445</BytesSent>
            <btesSent>-1</btesSent>
            <RemoteHostName>127.0.0.1</RemoteHostName>
            <RequestProtocol>HTTP/1.0</RequestProtocol>
            <RemoteLogicalUsername>-</RemoteLogicalUsername>
            <RequestMethod>POST</RequestMethod>
            <LocalPort>8080</LocalPort>
            <HTTPStatus>200</HTTPStatus>
            <DateTime>[05/Feb/2004:12:37:49-0300]</DateTime>
            <AuthenticatedRemoteUser>-</AuthenticatedRemoteUser>
            <URL>/axis/services/Login</URL>
            <LocalServerName>localhost</LocalServerName>
      </XML2Log>
      ... Outras entradas de log Apache
</LogWebServer>
```