MARIO FREITAS DA SILVA

UMA ABORDAGEM PARA MONITORAMENTO DE CONTRATOS ELETRÔNICOS BASEADA EM ASPECTOS

MARINGÁ
2008

MARIO FREITAS DA SILVA

UMA ABORDAGEM PARA MONITORAMENTO DE CONTRATOS ELETRÔNICOS BASEADA EM ASPECTOS

Dissertação apresentada ao Programa de Pós-graduação em Ciências da Computação da Universidade Estadual de Maringá, como requisito parcial para obtenção do grau de Mestre em Ciências da Computação.

Orientadora: Profª. Drª. Itana Maria de Souza Gimenes

MARINGÁ
2008

Dados Internacionais de Catalogação-na-Publicação (CIP)
(Biblioteca Central - UEM, Maringá – PR., Brasil)

S586a

Silva, Mario Freitas da
Uma abordagem para monitoramento de contratos eletrônicos baseada em aspectos / Mario Freitas da Silva. -- Maringá : [s.n.], 2008.
104 p. : il. , figs., tabs.

Orientadora : Profa. Dra. Itana Maria de Souza Gimenes.
Dissertação (mestrado) - Universidade Estadual de Maringá. Programa de Pós-graduação em Ciências da Computação, 2008.

1. Engenharia de software. 2. Programação orientada à aspectos. 3. Monitoramento - Contratos eletrônicos. I. Universidade Estadual de Maringá. Programa de Pós-graduação em Ciências da Computação

Cdd 21.ed. 005.1

MARIO FREITAS DA SILVA

UMA ABORDAGEM PARA MONITORAMENTO DE CONTRATOS ELETRÔNICOS BASEADA EM ASPECTOS

Dissertação apresentada ao Programa de Pós-graduação em Ciências da Computação da Universidade Estadual de Maringá, como requisito parcial para obtenção do grau de Mestre em Ciências da Computação.

Aprovado em 25/08/2008

BANCA EXAMINADORA

Profa. Dra Itana Maria de Souza Gimenes
UEM/DIN

Prof. Dr Sérgio Roberto Pereira da Silva
UEM/DIN

Profa. Dra Maria Beatriz Felgar de Toledo
IC/UNICAMP

AGRADECIMENTOS

Primeiramente a Deus, que me guia pelos caminhos a serem seguidos, sempre presente em minha vida, ajudando-me em todos os momentos, auxiliando minhas decisões.

A toda minha família pelo apoio.

A todos aqueles que de forma direta ou indireta, influenciaram e incentivaram a realização deste trabalho.

Agradecimento especial à Profª. Itana, pessoa que Deus colocou em meu caminho para me ensinar que os obstáculos são superados um de cada vez. Agradeço também a ajuda e compreensão das minhas limitações e dificuldades.

Resumo

Processos de negócios são artefatos de grande valor para as organizações contemporâneas. Um processo de negócio contém um conjunto de atividades executadas em determinada seqüência para alcançar um objetivo de negócio dentro de uma organização. Essas atividades podem ser implementadas por serviços Web que constituem um tipo específico de serviço eletrônico. Os serviços Web podem ser disponibilizados pela própria organização ou por organizações terceiras. A utilização de serviços Web terceirizados pode necessitar de uma regulamentação em contrato para garantir a qualidade de serviços. Os contratos eletrônicos são formas de representar e monitorar o acordo entre as organizações para o uso de serviços eletrônicos. Percebe-se que existem dois interesses neste contexto: a realização do processo de negócio e o monitoramento do contrato eletrônico. O interesse principal é a realização do processo de negócio, pois este contém as atividades alvo da organização. O interesse secundário é o monitoramento do contrato eletrônico associado ao processo de negócio já que este serve para verificar a qualidade dos serviços eletrônicos. O monitoramento deve ocorrer de forma transparente ao processo de negócio. A separação de interesses em aplicações tem sido um alvo de pesquisa na área de sistemas de informação. A abordagem orientada a aspectos realiza a separação de interesses, identificando o interesse principal de um determinado processo e seus interesses secundários, também chamados de interesses transversais. Os interesses transversais são então encapsulados em estruturas chamadas aspectos e devem participar do processo de forma transparente. Pode-se perceber então que o monitoramento de contratos eletrônicos de um determinado processo de negócio é um candidato a interesse transversal de acordo com a abordagem orientada a aspectos. O objetivo da abordagem apresentada nesta dissertação é permitir o monitoramento de contratos eletrônicos, especificados em WS-Agreement, utilizando o paradigma orientado a aspectos. Os resultados obtidos mostram que, ao utilizar esta abordagem, não há necessidade de interferir na especificação do processo de negócio para incluir instruções de monitoramento, pois estas são modeladas como aspectos. Caso alguma cláusula do contrato seja alterada, a única estrutura que vai ser afetada é o aspecto de monitoramento, desta forma não há necessidade em redefinir o processo de negócio. O serviço de monitoramento recebe os dados capturados durante a execução e pode tanto simplesmente armazenar para posterior utilização e auditoria, como realizar alguma computação sobre esses dados no momento do monitoramento.

Abstract

Nowadays business processes are valuable artifacts for current organizations. A business process contains a set of activities executed in order to achieve a business objective within an organization. These activities can be implemented by Web services, which are a specific kind of electronic service. Web services can be available within an organization or can be provided by third parties. The use of third party Web services requires proper regulation in order to ensure the quality of them. Eletronic contracts are means to represent an agreement between organizations in such a way that they can be monitored throughout their execution. We can highlight two concerns in this context: the realization of the business process and the monitoring of the electronic contract. The main concern is the realization of the business process because it contains the target activities of the organization. The secondary concern is the monitoring of the electronic contract as it serves to control the quality of the electronic services. This monitoring should be transparent to the business process. The separation of concerns has been a research goal for information systems. The separation of concerns in the aspect oriented approach is carried out by distinguishing a main goal from the secondary ones, which are called transversal concerns. These concerns are than encapsulated within structures called aspects. They participated of the process in a transparent way. We can observe that the electronic contract monitoring of a given business process is a candidate to be a transversal concern according to the aspect oriented approach. The objective of the approach presented in this dissertation is to support the monitoring of electronic contracts, specified in WS-Agreement, based on the aspect oriented paradigm. The obtained results show that our approach reduces the necessity of interfering in the business process specification in order to include monitoring operations as they are modeled as aspects. In case some contract clause needs to be modified, the only structure to be affected is the corresponding monitoring aspect. Thus, it is not necessary to redefine the business process. The data captured throughout the process execution by the monitoring service can be either stored, for further auditing, or actions can be taken due to the results of their computation.

LISTA DE FIGURAS

LISTA DE LISTAGENS

LISTA DE SIGLAS

AO4BPEL	*Aspect Oriented extension to BPEL4WS*
BPEL	*Business Process Execution Language*
BPM	*Business Process Management*
BPMI	*Business Process Management Initiative*
BPMS	*Business Process Management System*
COS	*Computação Orientada a Serviços*
EAI	*Enterprise Application Integration*
GGF	*Global Grid Forum*
GRAAP	*Grid Resource Allocation and Agreement Protocol Working Group*
HTTP	*HyperText Transfer Prtocol*
OO	*Orientado a Objetos*
POA	*Programação Orientada a Aspectos*
QoS	*Quality of Service*
RPC	*Remote Procedure Call*
SLA	*Service Level Agreement*
SLO	*Service Level Objective*
SOA	*Service Oriented Arquitecture*
SOAP	*Simple Object Access Protocol*
UDDI	*Universal Description, Discovery and Integration*
WfMC	*Workflow Management Coalition*
XSD	*XML Schema Definitions*
WS-Agreement	*Web Services Agreement Specification*
WSDL	*Web Services Description Language*
WSLA	*Web Service Level Description*
XML	*eXtendible Markup Language*
XSD	*XML Schema Definitions*

SUMÁRIO

1. Introdução

Muitas organizações estão atuando de forma cooperativa para atingir objetivos comuns de negócio por meio da integração e automação de seus processos de negócio. As organizações estão se concentrando nas atividades chaves para sua área de negócio e contratando a realização de atividades secundárias (FANTINATO, 2007). Neste contexto, a Internet tem oferecido uma importante infra-estrutura para sediar esta cooperação, principalmente por meio da tecnologia de troca de serviços eletrônicos entre organizações. Alguns processos de negócio podem ter a participação de diversas organizações em que cada uma desempenha a atividade na qual é especializada. Esses processos de negócio são tipicamente de longa duração, envolvem a execução de muitas tarefas manuais e automáticas, requerem acesso a diversos bancos de dados diferentes, bem como a invocação de várias aplicações e envolvem regras de negócios complexas (DAYAL et al, 2001). Atualmente, muitos dos trabalhos realizados na área de processos de negócios interorganizacionais utilizam a arquitetura orientada a serviços (SOA) e a tecnologia de serviços eletrônicos para integração de aplicações (CASATI, 2002). Um serviço Web é um tipo específico de serviço eletrônico. Os serviços Web podem ser disponibilizados pela própria organização ou por organizações terceiras.

As organizações parceiras em um determinado processo de negócio realizado por meio da composição de serviços Web podem firmar um contrato de uso desses serviços, denominado de contrato eletrônico. As organizações que contratam ou disponibilizam um serviço Web têm interesse em especificar quais as capacidades, requisitos e garantias que determinado serviço oferece, bem como em monitorar se essas capacidades estão sendo respeitadas (ANDRIEUX et al, 2004). Esses requisitos e capacidades são chamados de Acordo de Nível de Serviço (SLA[1]). Para a organização que está contratando um serviço Web é importante obter garantias acerca da qualidade do serviço contratado. A organização provedora por outro lado pode especificar as características que fazem parte de seus serviços, podendo inclusive criar diferentes níveis de serviços para diferentes tipos de organizações clientes.

O monitoramento de contratos eletrônicos consiste em coletar dados relativos a este contrato durante a execução do processo de negócio associado. Portanto, o monitoramento de contratos eletrônicos é essencial para que as organizações possam verificar se as garantias,

[1] Do inglês Service Level Agreement.

requisitos e capacidades que foram negociados estão sendo cumpridos por ambas organizações.

O *framework* WSLA (KELLER e LUDWIG, 2003) foi elaborado para definição e monitoramento de SLAs de serviços Web. Este *framework* foi desenvolvido para aplicação em ambientes de serviços Web. WS-Agreement (ANDRIEUX et al, 2004) é uma proposta mais abrangente que o WSLA para a definição de acordos entre organizações cliente e provedoras para a utilização de serviços que podem ser serviços Web, uso de recursos de agendador de tarefas de grandes sistemas distribuídos ou gerenciadores de requisitos aplicados em rede. WS-Agreement permite descrever os serviços oferecidos, suas propriedades e as garantias associadas ao serviço. WS-Agreement também possui a definição de um protocolo para criar, monitorar e terminar um SLA.

CREMONA (LUDWIG et al, 2004) é a primeira proposta de implementação do WS-Agreement existente na literatura. Ela proporciona aos provedores uma infra-estrutura para gerenciar os acordos, implementar interfaces, verificar disponibilidade da capacidade do serviço e expor os estados do acordo em tempo de execução. Ela também oferece aos clientes uma infra-estrutura para ler moldes de contratos, preenchê-los de modo a criar o acordo e assim monitorar seus estados e seu tempo de execução. A biblioteca Java do CREMONA implementa as interfaces do WS-Agreement de modo a oferecer funcionalidades de gerenciamento para as instâncias e moldes de contratos dos acordos. Também define abstrações do sistema provedor de serviço para moldes e instâncias de acordos. As abstrações do sistema provedor de serviços podem ser implementadas em um ambiente de aplicação específico. Mesmo diante dessa proposta, nota-se algumas lacunas como falta de modularização de interesses de monitoramento que são especificados por meio de instrumentação do processo de negócio com diversas funções de monitoramento. Essas propostas não apresentam seus resultados quando aplicados a processos de negócio complexos. As propostas se baseiam na interação entre apenas um cliente e um provedor na negociação de uso de apenas um serviço.

Percebe-se que existem dois interesses principais no contexto de processos de negócio com contratos eletrônicos: a realização do próprio processo de negócio e o monitoramento do contrato eletrônico associado a este. O interesse principal é a realização do processo de negócio, pois contém as atividades alvo da organização. O interesse secundário é o monitoramento do contrato eletrônico já que este serve para verificar se os requisitos, capacidades e garantias desejados estão sendo cumpridos. Entretanto, o monitoramento

modular de contratos eletrônicos não é uma tarefa trivial, pois o mesmo requer a instrumentação do processo de negócio. A instrumentação necessária ao monitoramento é espalhada em meio à especificação do processo de negócio tornando-se, assim, um interesse que entrecorta o interesse principal. A separação de interesses em aplicações tem sido um alvo de pesquisa na área de sistemas de informação. A abordagem orientada a aspectos realiza a separação de interesses, identificando o interesse principal de um determinado processo e seus interesses secundários, também chamados de interesses transversais. Os interesses transversais são então encapsulados em estruturas chamadas aspectos e devem participar do processo de forma transparente. Esses aspectos são na verdade trechos do processo que são executados em tempo de execução. Como não fazem parte do interesse principal do processo, os aspectos são inseridos em pontos determinados do processo chamados pontos de corte. Dessa forma, o monitoramento de contratos eletrônicos é uma propriedade candidata a ser modularizada com o uso do paradigma orientado a aspectos (KICZALES et al, 1997). Por outro lado, mecanismos convencionais da programação orientada a aspectos (KICZALES et al, 1997) não podem ser diretamente aplicados ao monitoramento de contratos eletrônicos, pois não considera os elementos arquiteturais e a dinâmica de execução desses. Além disso, poucos trabalhos, tais como Charfi e Mezini (2004) e Braem et al. (2006), têm explorado o uso de orientação a aspectos no contexto de composição de serviços Web.

Esta dissertação apresenta uma abordagem para monitoramento de contratos eletrônicos baseada no paradigma orientado a aspectos, chamada de AspectMonitor. É definida uma arquitetura para implementação da abordagem, as atividades e responsabilidades das organizações envolvidas e um conjunto de regras de mapeamento de contratos eletrônicos para aspectos de monitoramento. O tratamento do monitoramento é realizado por aspectos monitores que atuam no processo de negócio e enviam os dados de monitoramento para serviços Web monitores. Os pontos de corte e as atividades realizadas pelos aspectos de monitoramento são definidos baseando-se no que foi especificado no contrato eletrônico definido em WS-Agreement.

Além de ser transparente ao processo de negócio, o monitoramento realizado com aspectos também permite que sejam executadas atividades mais complexas durante o monitoramento como, por exemplo uma análise sobre os dados recolhidos pelos serviços monitores a fim de propor melhorias no relacionamento entre as organizações. Para a avaliação da proposta foi criado um exemplo de aplicação seguindo o passo a passo da abordagem proposta. Os resultados obtidos mostram que, ao utilizar esta abordagem, não há a

necessidade de interferir na especificação do processo de negócio para incluir instruções de monitoramento, pois estas são modeladas como aspectos. Caso alguma cláusula do contrato seja alterada, a única estrutura que vai ser afetada é o aspecto de monitoramento, desta forma não há necessidade em redefinir o processo de negócio. O serviço de monitoramento recebe os dados capturados durante a execução do processo de negócio e pode realizar diversas análises sobre os dados além de repassar às organizações participantes do contrato eletrônico informações acerca do que foi realizado durante o processo de negócio.

A dissertação se divide em 5 capítulos. O capitulo 2 descreve as tecnologias que são utilizadas para concepção da abordagem proposta. O capitulo 3 apresenta a abordagem de monitoramento com aspectos, explicando os passos necessários para a criação do processo de negócio, dos contratos associados e seu monitoramento. O capítulo 4 apresenta um exemplo de aplicação da abordagem proposta. O capítulo 5 apresenta as conclusões e trabalhos futuros.

2. Fundamentação Teórica

Este capítulo apresenta os conceitos e tecnologias necessárias para a criação da abordagem de monitoramento de contratos eletrônicos baseado no paradigma de aspectos proposta nesta dissertação. Inicialmente, é apresentado o paradigma orientado a aspectos. Depois são apresentados os conceitos de processos de negócio e serviços eletrônicos. Em seguida, como a computação orientada a serviços é associada à tecnologia de serviços Web para a criação de processos de negócio baseado na composição de serviços Web. Em particular, são apresentadas a linguagem de especificação de processo de negócio WS-BPEL e a extensão orientada a aspectos desta linguagem que é o AO4BPEL. A exigência por garantia da qualidade dos serviços envolvidos em processos de negócios demanda o estabelecimento e monitoramento de contratos eletrônicos, portanto este capítulo também apresenta o conceito de contratos eletrônicos, acordos de níveis de serviços e mecanismos de definição e monitoramento destes contratos e mecanismos.

2.1. Paradigma Orientado a Aspectos

Para se obter modularidade em um sistema, é necessário dividi-lo em partes para reduzir sua complexidade. Todo sistema de software lida com interesses[2] diferentes, sejam eles dados, operações, ou outros requisitos do sistema. O ideal é que a parte do sistema dedicada a satisfazer a um determinado interesse esteja concentrada em um único ponto separado de outros interesses, para que o interesse principal possa ser estudado e compreendido com facilidade.

O desenvolvimento estruturado realizou a separação de interesses orientando-se pelas diferentes funcionalidades oferecidas pelo software. Cada função era implementada em um único módulo, ou procedimento. Assim, surgiram conceitos que ajudam a manter a separação de interesses, como o baixo acoplamento e a alta coesão.

A orientação a objetos (OO) veio como forma de resolver uma das deficiências do desenvolvimento estruturado. Apesar de interesses relativos a funcionalidades ficarem separados, interesses relativos a dados ficavam distribuídos em diversos módulos. O paradigma OO definiu que a separação deveria acontecer em duas dimensões, primeiro dividido em termos de dados e depois em termos das funções que utilizam cada tipo de dados.

[2] Interesse é utilizado como tradução do termo *concern*.

O paradigma OO melhorou as possibilidades de separação de interesses, mas ainda restaram deficiências. A Figura 1 (KERSTEN, 2002) mostra uma representação gráfica do código do sistema Tomcat, um servidor Web com capacidade de executar *servlets* Java. Cada coluna representa um módulo do sistema, sendo que o tamanho de cada coluna mostra o número proporcional de linhas de código daquele módulo. Como visto na Figura 1 na classe de cor mais escura o interesse de analisador de XML está bem separado.

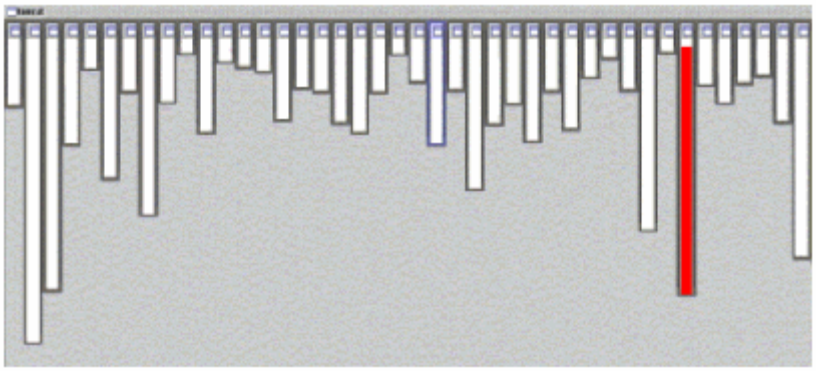

Figura 1: Analisador de XML no sistema org.apache.tomcat. Fonte: (KERSTEN, 2002)

No entanto, isso nem sempre é verdade. Se for considerada a funcionalidade de guardar registros para auditoria que inclui registrar as ações tomadas pelo servidor para se detectar violações de segurança e erros, então se percebe que o código responsável por esse comportamento está espalhado por quase todos os módulos, como apresentado na Figura 2. Os pontos mais escuros em cada classe representam os códigos necessários para auditoria.

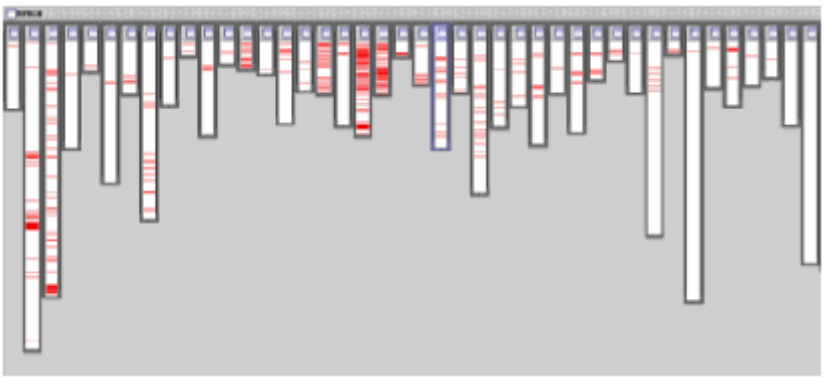

Figura 2: Registro de auditoria no sistema org.apache.tomcat. Fonte: (KERSTEN, 2002)

Na terminologia de orientação a aspectos, diz-se que a função de registro para auditoria é um interesse entrecortante, porque a sua implementação se espalha em diversos módulos do sistema. Praticamente todo programa OO não-trivial contém interesses entrecortantes.

O objetivo do desenvolvimento orientado a aspectos é encapsular interesses entrecortantes em módulos fisicamente separados do restante do código. Esses módulos são denominados aspectos (KICZALES et al, 1997). Pensando em termos abstratos, a orientação a aspectos introduz uma terceira dimensão de decomposição. Além de decompor o sistema em objetos (dados) e métodos (funções), decompõe-se cada objeto e função de acordo com o interesse relacionado e agrupa-se cada interesse em um módulo distinto chamado de aspecto. A Figura 3 apresenta uma comparação entre a abordagem OO e a abordagem orientada a aspectos. Nota-se que a abordagem OO possui os interesses entrecortando várias classes e que na abordagem orientada a aspectos os interesses transversais são encapsulados em apenas um aspecto que atua em diversas classes (CHAVEZ et al, 2003), representado na Figura 3 por um losango.

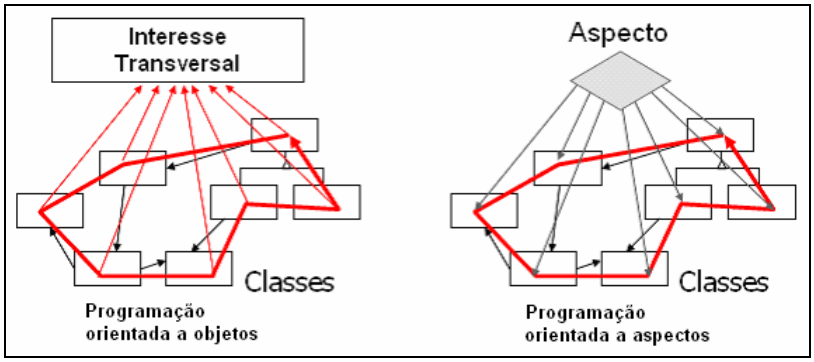

Figura 3: Interesse transversal orientado a objetos e aspectos. Fonte: (CHAVEZ et al, 2003).

A programação orientada a aspectos (POA) envolve duas etapas de trabalho. A primeira é a decomposição do sistema em partes não entrelaçadas e não espalhadas. A segunda consiste em juntar essas partes novamente para se obter o sistema desejado. O processo de juntar as partes se chama composição. A forma de composição das partes é o que realmente distingue linguagens orientadas a aspectos de outras linguagens. Em linguagens procedimentais ou OO, a composição é feita por meio de chamadas de procedimentos ou métodos. Assim, uma parte (por ex. uma classe) usa a funcionalidade de outra chamando um método. Em POA, não há chamadas explícitas de métodos entre partes. Ao invés disso, especifica-se, em uma parte separada, como uma parte deve reagir a eventos que acontecem em outra parte. Essa estratégia reduz o acoplamento entre as partes, pois elas não se interagem diretamente (KICZALES et al, 1997).

Os principais benefícios da abordagem orientada a aspectos são:

- Menos responsabilidade em cada parte: como interesses entrecortantes são separados em seus próprios módulos, as partes do sistema que lidam com a lógica de negócios não ficam poluídas com código que lida com interesses periféricos.
- Melhor modularização: como os módulos em AOP não se chamam diretamente, há uma redução no nível de acoplamento.
- Evolução facilitada: novos aspectos podem ser acrescentados facilmente sem necessidade de alterar o código existente.
- Mais possibilidades de reutilização: como o código não mistura interesses, aumentam-se as possibilidades de se reutilizar módulos em sistemas diferentes.

Programas orientados a aspectos precisam de um compilador específico. No caso da linguagem AspectJ (KICZALES et al, 2001), o compilador AJC transforma um programa escrito em AspectJ em um programa em bytecode Java, que pode ser executado por qualquer máquina virtual Java (JVM). O aspecto é o elemento principal do paradigma orientado a aspectos. Para a criação de aspectos devem ser utilizados os seguintes elementos:

- *Join point:* é qualquer ponto identificável pelo AspectJ durante a execução de um programa. O AspectJ permite diversos tipos de *joint points* como entradas e saídas de métodos, tratamento de exceções, acessos a variáveis de objetos, construtores, entre outros;
- *Pointcut*: são construções de programa que permitem a seleção de um ou mais *join point*. Podem ser utilizadas expressões regulares para se especificar os entrecortes, permitindo grande flexibilidade;
- *Advice*: é um trecho de código que deve atuar nos *join points* selecionados por um *pointcut*. O *advice* pode ser executado antes, depois, ou em torno do *join point*;
- *Inter-type declaration:* é uma forma de um aspecto introduzir mudanças em classes, interfaces e aspectos do sistema. Por exemplo, pode-se acrescentar uma variável ou método a uma classe existente;
- *Aspect*: é o elemento principal em AspectJ. Modulariza os interesses transversais. Contêm *join points*, *advices*, além de métodos e atributos locais.

A Listagem 1 apresenta um aspecto escrito em AspectJ. O aspecto `NormalizaCaixa` define um `pointcut` nos `joinpoints` identificados pela operação

`toString` de todas as classes * e um `advice` do tipo `around`. Significa que o código do `advice` será executado ao invés do `toString`. A palavra-chave `proceed` executa o método `toString` com o intuito de guardar na variável o retorno do `toString` e depois alterar para minúsculo a resposta da operação e retorná-la.

```
public aspect NormalizaCaixa {
      pointcut string() : call (* *.toString());
      String around() : string() {
            String s = proceed();
            return s.toLowerCase();
      }
}
```

Listagem 1: Aspecto escrito em AspectJ

Em resumo, o `pointcut` representa **ONDE** o aspecto deve atuar no código, o tipo de aspecto representa **QUANDO** o aspecto deve atuar e o `advice` representa **O QUE** o aspecto deve realizar.

A tecnologia de aspectos aplicada a processos de negócio será apresentada na seção 2.7. As próximas seções criam uma ambientação com os diversos conceitos envolvidos com aplicação de aspectos a processos de negócio. A próxima seção apresenta o primeiro conceito que é relacionado a processos de negócio.

2.2. Processos de Negócio

Em seus negócios diários, as organizações executam atividades para produzir produtos ou realizar serviços. Um processo de negócio é um conjunto de atividades executadas numa seqüência específica para alcançar um objetivo de negócio. O processo de negócio define a seqüência das atividades, os eventos externos que devem ser tratados, os requisitos de interação humana e o processamento condicional (SADTLER e KOVARI, 2004).

As atividades de um processo de negócio podem ser tanto automatizadas como manuais, realizadas com ou sem interação humana e podem envolver a cooperação com outras organizações. Uma atividade pode ser, inclusive, outro processo de negócio. Um processo de negócio pode ser composto de processamentos em sistemas informatizados, de preenchimentos de documentos, de atividades realizadas em série ou paralelas, de troca de informações, de trabalhos manuais, além de ser afetado por diferentes regras de negócio (DAYAL et al, 2001). Com esta variedade de tipos de atividades, um processo de negócio torna-se complexo. Um exemplo de processo de negócio é apresentado na Figura 4. Este processo envolve cinco (5) atividades. A atividade um (1) que é uma aplicação J2EE

(aplicação Java) é conectada à atividade dois (2) que representa um sistema legado; e, à atividade três (3) que representa um outro processo. Ao fim da atividade um (1), as atividades dois (2) e três (3) podem ser executadas em paralelo ou apenas uma delas pode ser executada baseada na condição especificada nos conectores. O fluxo de processo vai esperar na atividade quatro (4) por uma decisão humana para então seguir para a atividade cinco (5).

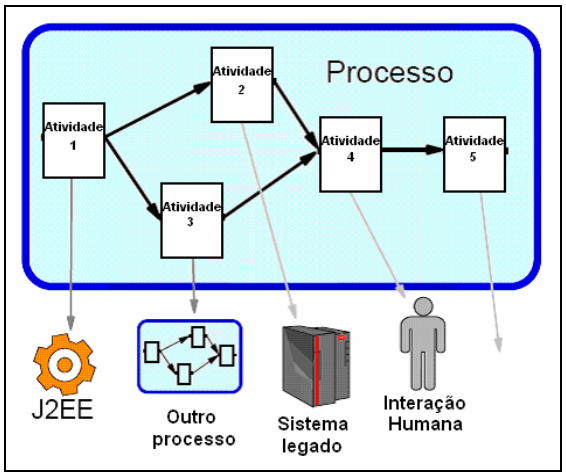

Figura 4 – Estrutura de um processo de negócio. Fonte: (SADTLER e KOVARI, 2004).

Um exemplo de processo de negócio simples é apresentado na Figura 5. Este processo envolve as atividades de análise de risco por um Aprovador (serviço *approver*) e um Assessor (serviço *assessor*). Quando o valor do empréstimo está acima de R$ 10.000,00 ou o risco do empréstimo for alto, a análise do Aprovador é obrigatória. Quando o valor é menor que R$ 10.000,00 e o risco é baixo, é necessária apenas a análise de um Assessor (MEDEIROS et al, 2007), (ACTIVEBPEL, 2006).

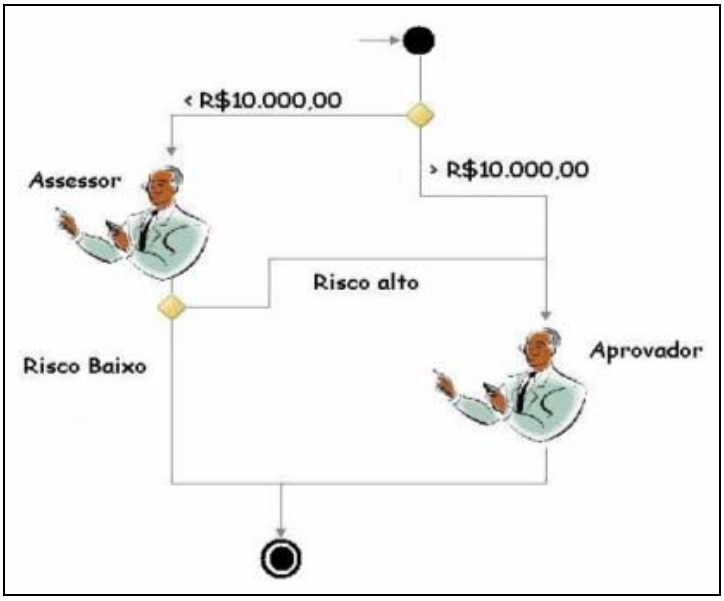

Figura 5: Processo de negócio de empréstimo. Adaptado de: (ACTIVEBPEL, 2006)

Processos de negócios podem ser investigados do ponto de vista administrativo em que organizações podem ser analisadas como centradas em processo ou não, ou em estrutura matricial (BALDAN et al, 2008). No contexto deste trabalho de mestrado, o foco está nas questões relacionadas ao apoio computacional necessário para gerenciamento de processos de negócio.

Um processo de negócio necessita de infra-estrutura computacional para que possa ser coordenado pelas organizações (DAYAL et al, 2001). Também existe necessidade de se integrar os processos de diversas organizações devido à crescente necessidade de terceirização de operações onde organizações clientes contratam serviços especializados de organizações provedoras. O gerenciamento de processos de negócio envolve atividades de definição, simulação, execução, manutenção, monitoramento e análise para melhorias. Sistemas gerenciadores de processo de negócio (WESKE, 2007) são os sistemas utilizados para apoiar as atividades de gerenciamento de processos de negócio.

A evolução de três principais tecnologias convergiu para formar o que atualmente é chamado de BPMS: *Workflow* (WfMC, 1995), *Enterprise Application Integration* (EAI) e Web. Assim, BPMS podem ser considerados como evoluções de WfMS (*Workflow Management System*) (MEDEIROS et al, 2007). A BPMI (Business Process Management Initiative) é uma organização independente voltada ao desenvolvimento de especificações abertas para o gerenciamento dos processos empresariais, exatamente como a WfMC no caso do worfkflow. Atualmente, estas duas iniciativas, têm mostrado que seus posicionamentos e objetivos estão convergindo. Além disso, fabricantes de diferentes segmentos, tais como Gerenciadores de workflow, Gerenciadores Eletrônicos de Documentos, Ferramentas Colaborativas e Gerenciadores de Conteúdo estão se orientando segundo os padrões de BPMS (SMITH e FINGAR, 2004).

BPMS não é apenas uma ligação entre aplicações, nem uma simples definição da lógica de transformações necessárias para extrair dados das aplicações. Mais do que isso, o BPMS tem a habilidade de definir e executar os processos de negócio independentemente das aplicações e da infra-estrutura envolvidas. A maioria dos BPMS disponíveis no mercado é orientada a componentes, o que permite que cada uma das partes possa ser implantada separadamente. Uma grande vantagem dos sistemas de BPMS sobre os WFMS, que também permitem automatizar atividades e definir regras de negócio, é sua capacidade de coordenar a passagem de tarefas de uma pessoa para outra. Com isso, os processos são otimizados, não

acontecendo apenas a execução de tarefas por meio de sua automação (KARAGIANNIS, 1995).

A associação de BPM com a computação orientada a serviços permite que processos de negócio possam ser compostos de serviços. A utilização de serviços em processos de negócio será explicada na seção 2.4. A próxima seção apresenta os conceitos de serviços e a tecnologia de serviços Web.

2.3. Serviços WEB

De acordo com Papazoglou e Georgakopoulos (2003), serviços eletrônicos são componentes abertos e auto-descritivos que oferecem apoio à composição de aplicações/soluções distribuídas de uma forma rápida e com baixo custo. A descrição de um serviço é usada para apresentar às organizações potencialmente interessadas: suas capacidades, sua interface, seu comportamento e sua qualidade.

A computação orientada a serviços (COS) é o paradigma da computação que utiliza serviços eletrônicos como elementos fundamentais para o desenvolvimento de aplicações distribuídas (PAPAZOGLOU e GEORGAKOPOULOS, 2003). Ela oferece uma infra-estrutura tecnológica, a partir da qual organizações interessadas podem se comunicar, via troca de mensagens, para alcançar os objetivos de um processo de negócio.

A COS utilizou inicialmente tecnologias de middleware e EAI, tais como CORBA (BEN-NATEN, 1995) e DCOM (HORSTMANN e KIRTLAND, 1997), para tratar da integração entre aplicações distribuídas utilizando serviços eletrônicos (CASATI, 2002). Embora muito tenha sido desenvolvido nessa área, essas tecnologias não chegaram a apresentar uma solução apropriada para a integração de aplicações em ambientes heterogêneos como a Internet, onde as interações ocorrem sem coordenadores e autoridades centrais. Um dos problemas associados às tecnologias EAI propostas é a alta complexidade de instalação, configuração e administração. Isto provoca muitos problemas em seu uso, além de serem tecnologias relativamente caras que não são facilmente encontradas na maior parte dos sistemas computacionais conectados à Internet (HSU, 2002).

Atualmente, os serviços Web estão sendo apresentados como uma tecnologia potencial para a efetiva automação das interações interorganizacionais na COS, por facilitar a descoberta e a invocação automática de serviços eletrônicos relevantes. A tecnologia de serviços Web oferece uma infra-estrutura apropriada para tornar processos de negócio acessíveis tanto em uma mesma organização quanto entre organizações. O benefício essencial

associado à tecnologia de serviços Web é a ampla padronização, já que os padrões podem ser utilizados de forma simples em qualquer plataforma de software e hardware, tornando mais fácil a integração de aplicações (ALONSO et al, 2004). A Figura 6 ilustra a comunicação entre serviços e os padrões utilizados. Fornecedores publicam serviços em repositórios. Consumidores buscam serviços nos repositório e os invocam de fornecedores.

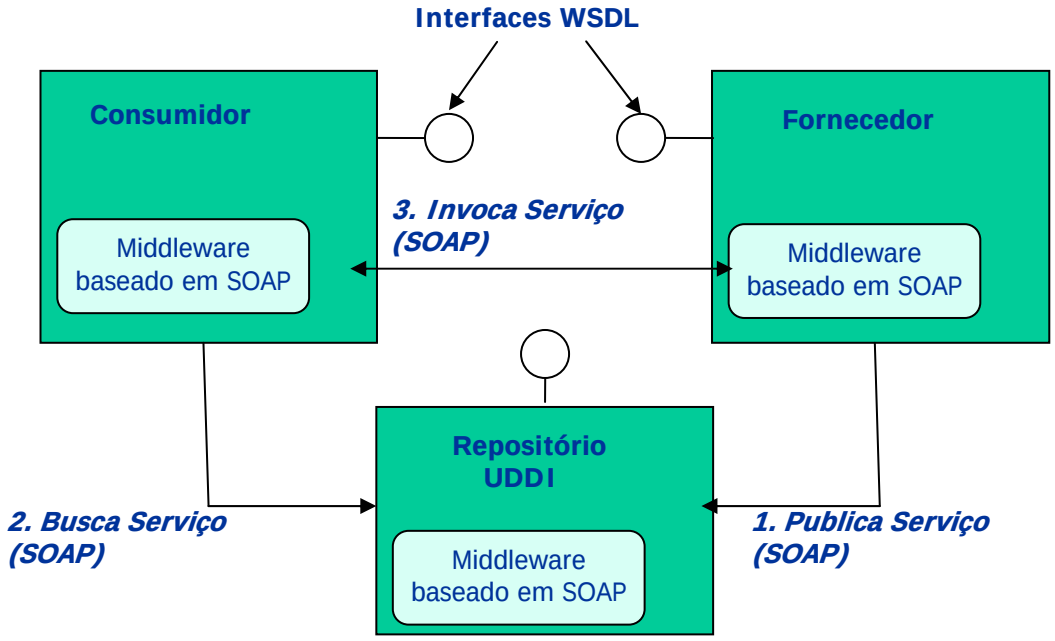

Figura 6: Comunicação entre serviços Web (Fonte: MEDEIROS et al, 2007).

Serviços Web utilizam alguns padrões que auxiliam na padronização de definição de suas funcionalidades. Os padrões utilizados são apresentados a seguir:

- *WSDL (Web Services Description Language)*: linguagem utilizada para descrever serviços Web. Uma descrição WSDL oferece as informações relacionadas a um serviço Web necessárias para sua publicação, descoberta e invocação. Essas informações incluem as operações, variáveis de entrada e saída e tipos de dados (CHRISTENSEN et al, 2001);

- *UDDI (Universal Description, Discovery and Integration)*: padrão que define a estrutura e o conteúdo dos diretórios de serviços que contém as descrições de serviços oferecidos. O padrão UDDI permite que provedores de serviços possam registrar seus serviços Web para que os clientes de serviços possam descobri-los (BELLWOD et al, 2002);

- *SOAP (Simple Object Access Protoco)l*: protocolo que define um mecanismo para a comunicação entre serviços Web por meio da Internet. O protocolo define o

formato das mensagens que são trocadas entre clientes, provedores e diretórios de serviços (BOX et al, 2000).

Múltiplas organizações podem realizar o gerenciamento de processos de negócio de forma centralizada, com o uso de um BPMS tanto como um gerenciador central ou descentralizado, utilizando-se vários BPMS.

As formas de interação entre serviços podem seguir o modelo hierárquico e ponto-a-ponto (LEYMANN et al, 2002), (FANTINATO et al, 2005):

Hierárquico: quando existe um único modelo de processo de alto nível que é estruturado em sub-processos de níveis mais baixos por meio de divisão hierárquica. O serviço composto de mais alto nível define as atividades necessárias para alcançar o objetivo geral do negócio e mapeia essas atividades para serviços oferecidos por outros parceiros de negócio. Nesse caso, existe a necessidade de uma organização coordenar a execução do serviço composto como um todo (orquestração).

Ponto-a-Ponto: quando existem dois ou mais sub-modelos de processo paralelos que trocam mensagens para alcançar um objetivo comum. As organizações envolvidas são organizações autônomas que compartilham o mesmo modelo de processo, porém elas podem ter dados de processos privados e sub-processos internos. Como não existe uma única organização realizando a orquestração do processo, é necessária uma sincronização entre os serviços envolvidos (coreografia).

Um ponto importante para a COS é medir propriedades referentes à QoS dos serviços Web. Os principais requisitos de QoS para serviços Web são: disponibilidade, acessibilidade, integridade, desempenho, confiabilidade, regulamentação e segurança (MANI e NAGARAJAN, 2006). Quando uma organização cliente utiliza serviços Web para realizar seus processos de negócio é importante que certos requisitos de QoS sejam garantidos. A negociação e definição dos requisitos de QoS são possíveis por meio de contratos eletrônicos entre organizações clientes e provedoras.

Os serviços são utilizados como forma de disponibilizar operações para realizar determinada atividade. Como os processos de negócio podem ser criados pela composição de serviços Web é apresentado na seção a seguir.

2.4. WS-BPEL

A especificação de um serviço Web é adequada para aplicações simples, porém aplicações mais complexas demandam a composição de serviços Web em processos de negócios, que inclusive podem envolver diversas organizações. A integração de processos de negócios em cenários reais envolve: interações de longa duração; gerenciamento de transações; invocações; e, são geralmente gerenciados por servidores que automatizam o fluxo de informação e as operações de negócios. Existem algumas propostas de linguagens de especificação de processos de negócios que permitem a composição de serviços Web, como WSFL da IBM (LEYMANN, 2001) e XLANG da Microsoft (THATTE, 2001). WSFL e XLANG deram origem a linguagem BPEL4WS (ANDREWS et al, 2003) que é atualmente a linguagem mais utilizada. BPEL4WS passou a ser chamada de WS-BPEL para uniformizar os padrões de serviços Web que iniciam com WS- (ARKIN, 2005).

WS-BPEL permite a especificação de aplicações que utilizam serviços Web, assim como também permite que uma aplicação seja especificada como um serviço Web. Isto porque um processo de negócio em WS-BPEL é publicado em forma de serviço Web, inclusive com seu próprio descritor WSDL.

Assim como uma linguagem de programação, WS-BPEL possui três componentes básicos:

- tipos de dados: XSD (*XML Schema Definitions*) que define os tipos usados no processo;
- entradas e saídas: WSDL descritor dos serviços Web utilizados na composição;
- lógica de programação: os elementos da linguagem WS-BPEL que une as variáveis às chamadas de serviços;

A Listagem 2 apresenta um exemplo simples de um processo `Hello World` em WS-BPEL.

```
1<?xml version="1.0" encoding="UTF-8"?>
2<process
       xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
       xmlns:print="http://www.eclipse.org/tptp/choreography/2004/engine/Print">

5      <import importType="http://schemas.xmlsoap.org/wsdl/"
       location="../../test_bucket/service_libraries/tptp_EnginePrinterPort.wsdl"
       namespace="http://www.eclipse.org/tptp/choreography/2004/Print" />
```

```
8      <partnerLinks>
9             <partnerLink name="printService"
                           partnerLinkType="print:printLink"
                           partnerRole="printService"/>
       </partnerLinks>

10     <variables>
              <variable    name="hello"
                           messageType="print:PrintMessage" />
       </variables>
11     <assign>
12     <copy>
              <from><literal>Hello World</literal></from>
              <to>.value</to>
       </copy>
       </assign>
13     <invoke partnerLink="printService" operation="print" inputVariable="hello" />
</process>
```

Listagem 2: Exemplo de Processo “Hello World” em WS-BPEL.

Na linha 5, o elemento `import` é uma diretiva para importar arquivos WSDL ou XSD. Utilizar arquivos XSD separados do arquivo WS-BPEL permite o uso comum de tipos de dados para um determinado serviço Web. Utilizar arquivos WSDL separados do arquivo WS-BPEL permite o uso comum de definições de serviços Web. O benefício de ter os arquivos XSD e WSDL separados é poder reutilizá-los em diferentes processos WS-BPEL.

Os parceiros definidos na linha 8, `partnerLinks`, são responsáveis pelos serviços Web a serem invocados. Os parceiros mapeiam como devem ser instanciados os serviços Web a serem utilizados pelo processo BPEL. Há um parceiro especial que identifica o próprio processo BPEL que se comportará como um serviço Web. O atributo `partnerRole` define as características de serviço Web que o processo WS-BPEL é publicado. Alternativamente, podem existir, ao invés do atributo `partnerRole`, o atributo `myRole` que define qual o serviço Web que o processo WS-BPEL implementa.

As variáveis definidas na linha 10 são utilizadas para armazenar dados do processo WS-BPEL. Uma variável pode conter valores XSD ou mensagens WSDL. No exemplo, a variável chamada `hello` é declarada como o tipo WSDL `print:PrintMessage`. As linhas 11 e 12 definem a atividade de atribuição e manipulação de variáveis. No exemplo, o valor literal “Hello World” é copiado para a parte `value` da variável `hello`. A sintaxe

utilizada na atribuição é criada em linguagem Xpath (CLARK e DEROSE, 1999). A linha 13 contém a atividade WS-BPEL `invoke` que define o momento em que o processo WS-BPEL envia o dado “Hello World” para o serviço Web `printService`. O parceiro especificado envia para o servidor WS-BPEL o endereço do serviço Web que será executado. Para utilizar a função `print` do serviço `printService` no comando `invoke` é necessário ter publicado o serviço `printService` em algum servidor que é descrito pelo arquivo WSDL.

A Figura 7 ilustra como é realizada uma interação para executar o processo descrito na Listagem 2. Nota-se que o início da interação é quando uma aplicação cliente realiza uma requisição ao servidor WS-BPEL. Esta requisição identifica qual processo WS-BPEL deve ser executado por meio de seu descritor WSDL. O processo WS-BPEL então é instanciado e sua atividade `receive` recebe os dados de entrada. Em seguida, é executada a atividade `assign` para atribuir os valores às variáveis corretas do processo. Então, a atividade `invoke` é executada e faz uma requisição a um servidor provedor de serviços por meio do seu arquivo WSDL. O serviço Web recebe os dados de entrada, executa a computação necessária e retorna a resposta à atividade invocada. A atividade `reply` finaliza a execução do processo WS-BPEL retornando os valores requisitados pelo cliente.

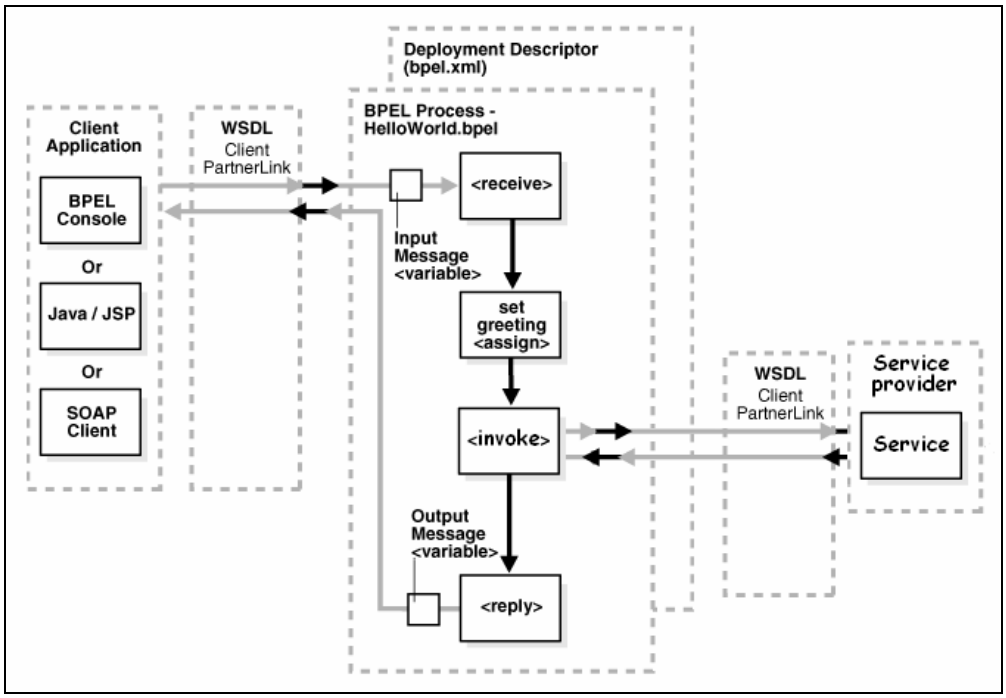

Figura 7: Exemplo de um processo de negócio WS-BPEL. Adaptado de: (ORACLE, 2006)

Alguns exemplos de implementações de ferramentas de apoio à especificação de processos em WS-BPEL são: BPWS4J (CURBERA et al, 2002); Oracle BPEL Process

Manager (ORACLE, 2007); e, ActiveBPEL (ACTIVEBPEL, 2006). A ferramenta ActiveBPEL é apresentada em mais detalhes na próxima seção.

2.4.1 ActiveBPEL

O conjunto de ferramentas ActiveBPEL é formado pelo: ActiveBPEL Designer; ActiveBPEL Engine e Active Endpoints SOAP Client.

ActiveBPEL Designer: é um ambiente para construção, teste e compilação de aplicações baseadas em WS-BPEL. Os serviços web são inseridos no processo WS-BPEL por meio de suas definições WSDL que podem ser importadas na ferramenta. A interface gráfica possui uma paleta com todas as operações WS-BPEL de modo que é necessário apenas clicar e arrastar para usá-las. Após definição do processo WS-BPEL, pode-se simular a execução do processo adicionando-se valores-teste para as respostas dos serviços web. Ao final do desenvolvimento, pode-se publicar o processo gerado exportando o arquivo descritor de processo para o servidor WS-BPEL. Na Figura 8 pode-se observar o ambiente ActiveBPEL Designer.

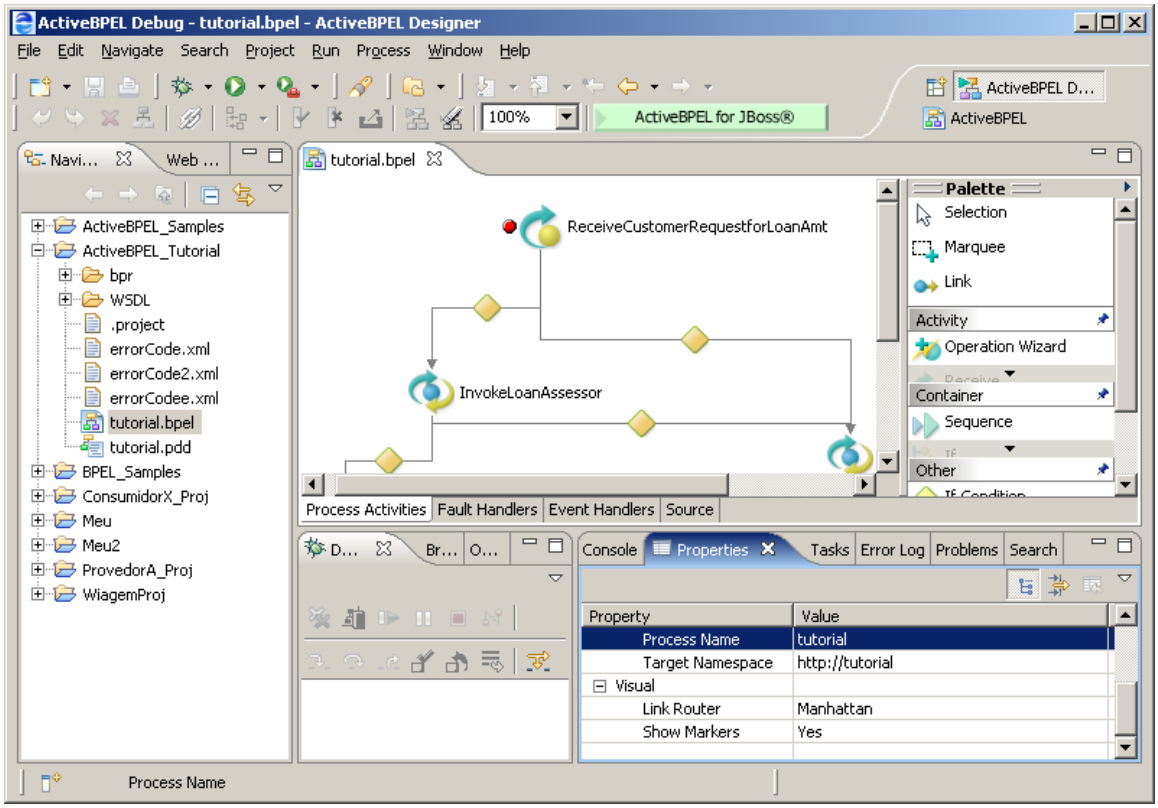

Figura 8: ActiveBPEL Designer

ActiveBPEL Engine: é o servidor WS-BPEL propriamente dito, onde se publicam os processos WS-BPEL. O servidor WS-BPEL é responsável por receber requisições de uso de processos WS-BPEL, instanciar o processo, executar os comandos contidos no processo e

retornar a resposta ao requisitante. Um exemplo do servidor WS-BPEL é mostrado na Figura 9.

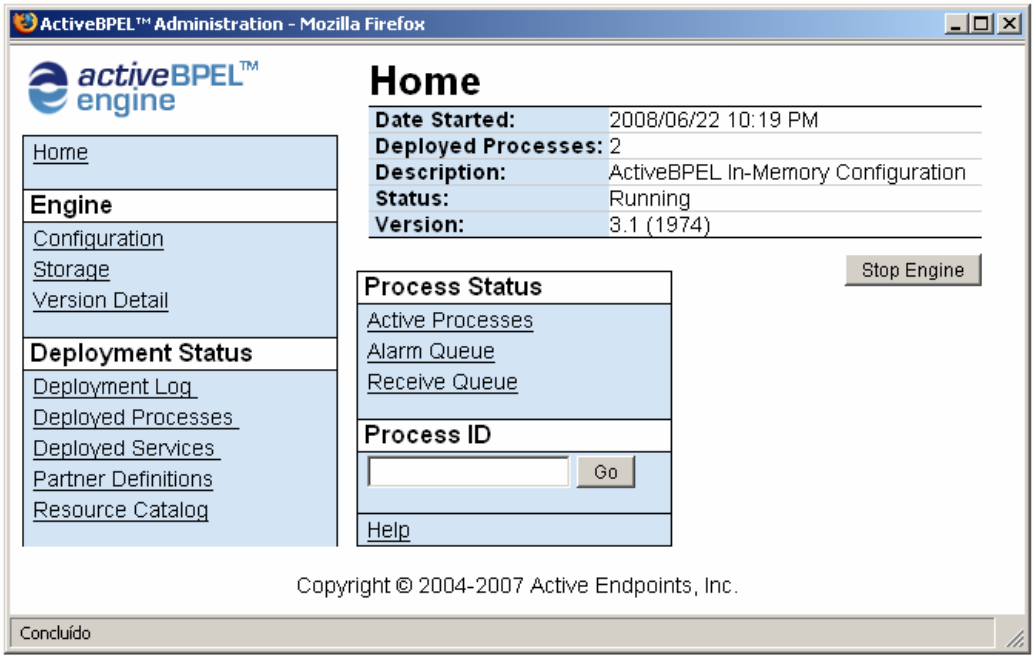

Figura 9: ActiveBPEL Engine

Active Endpoints SOAP Client: permite que se gerencie e configure o ActiveBPEL Engine e todos os artefatos contidos nele. Esta ferramenta disponibiliza diversas formas de selecionar e inspecionar instâncias de processos e seus arquivos descritores e recursos. Ela permite verificar gráficos de execução de um processo e analisar quais foram os passos executados neste processo. A Figura 10 apresenta a ferramenta ActiveBPEL Engine Console.

Figura 10: Active Endpoints SOAP Client

A utilização de serviços Web para compor um processo de negócio pode necessitar de um acordo entre a organização cliente e a provedora para definir as capacidades, requisitos e garantias de um determinado serviço Web. A próxima seção apresenta o conceito de contratos eletrônicos utilizados como forma de regulamentação de uso dos serviços Web.

2.5. Contratos Eletrônicos

As transações realizadas entre organizações diferentes são acompanhadas, implícitas ou explicitamente por contratos (GRIFFEL, 1998). Este contrato representa o comprometimento de cada parte em cumprir com as obrigações definidas. Uma organização geralmente recebe o direito de uso de algum serviço e a outra organização deve pagar pelo serviço (GRIFFEL, 1998).

Dependendo do contexto a que se aplica e da forma como é tratado o contrato pode ser apenas um documento de texto tornando a edição do contrato uma atividade simples. Por outro lado o contrato pode conter uma elaborada semântica o que envolve a criação de um sistema especialista em realizar contratos (GRIFFEL, 1998). O contrato geralmente se divide em três partes:

- Partes: representam os parceiros envolvidos no negócio, que exercem diferentes papéis no contrato;
- Atividades: descrevem os serviços a serem executados, incluindo informações necessárias para o fornecimento e o consumo de tais serviços;
- Cláusulas: representam restrições que precisam ser cumpridas durante a execução das atividades que estão previstas no contrato.

Com a COS, os processos de negócio interorganizacionais podem utilizar serviços de diversas organizações. Essas organizações podem ter interesse em restringir ou regulamentar o uso de seus serviços, ou então exigir garantias de uso quando utilizar um serviço de uma organização provedora. Um contrato eletrônico é um documento usado para representar acordos entre organizações parceiras que realizam negócios por meio da Internet, nas quais os serviços negociados são serviços eletrônicos (FANTINATO et al, 2005). Contratos eletrônicos contêm detalhes a respeito do processo de negócio a ser realizado de forma cooperativa entre as organizações, servindo de base para a execução e o acompanhamento das ações envolvidas no processo. Entre os possíveis detalhes contidos em contratos eletrônicos

estão informações de sintaxe e semântica sobre os serviços eletrônicos a serem executados, dados a serem trocados durante a execução dos serviços, atributos de qualidade definidos para estes serviços, custos envolvidos e possíveis operações de controle e monitoramento.

O ciclo de vida de um contrato eletrônico possui as seguintes etapas (DAN et al, 2004): implementação dos serviços eletrônicos, publicação, busca e descoberta dos serviços eletrônicos, negociação e definição de contratos eletrônicos, preparação para execução e execução do contrato.

Os principais requisitos de QoS para serviços Web são (MANI e NAGARAJAN, 2006):

- disponibilidade: requisito que especifica se o serviço Web está presente para ser executado ou não;
- acessibilidade: requisito que especifica se o serviço Web pode ser acessado;
- integridade: requisito que especifica se a operação executada pelo serviço Web foi realizada completamente ou em caso de falha se foi totalmente desfeita;
- desempenho: requisito que especifica o número de vezes que um serviço Web pode ser requisitado em determinado tempo, ou seu tempo de resposta;
- confiabilidade: requisito que especifica a quantidade de erros durante a execução de um serviço Web;
- regulamentação: requisito que especifica os padrões seguidos por um serviço Web;
- segurança: requisito que especifica os artifícios de segurança utilizados com criptografia ou uso de conexão segura.

Quando uma organização cliente utiliza serviços Web para realizar seus processos de negócio é importante que certos requisitos de QoS sejam garantidos. A negociação e definição dos requisitos de QoS são possíveis por meio de contratos eletrônicos entre organizações clientes e provedoras.

Um mesmo serviço pode ser oferecido por uma organização com diferentes níveis de QoS para diferentes organizações clientes de acordo com a necessidade. Portanto o contrato deve prever quais os parâmetros de QoS que devem ser cumpridos, formando assim um SLA.

A seguir são apresentadas algumas iniciativas que buscam definir modelos para a especificação de políticas associadas aos serviços Web.

WSLA: uma linguagem precisa e detalhada usada para definição de contratos eletrônicos (KELLER e LUDWIG, 2003). Permite a definição dos parceiros e das cláusulas a serem contratadas. Entre os parceiros é possível definir cliente, provedor e monitor do contrato. É considerada antecessora da linguagem WS-Agreement.

WS-Policy: o objetivo do *Web Services Policy Framework* é fornecer os mecanismos necessários para que aplicações desenvolvidas com Web Services possam especificar políticas (BAJAJ et al, 2006). WS-Policy fornece uma gramática flexível e extensível para expressar os requisitos, capacidades e características gerais de entidades, em um ambiente de serviços Web.

WS-Agreement: *Web Services Agreement Specification* é um documento público do *Global Grid Forum* (GGF), desenvolvido pelos participantes do *Grid Resource Allocation and Agreement Protocol Working Group* (GRAAP), que especifica um protocolo para o estabelecimento de um acordo entre um solicitante e um fornecedor de serviço utilizando uma linguagem XML extensível de acordo com o domínio de aplicação ao qual se aplica o contrato (ANDRIEUX et al, 2004).

A próxima seção apresenta mais detalhes da especificação WS-Agreement que é a linguagem de especificação de contratos eletrônicos utilizada na abordagem proposta desta dissertação.

2.6. WS-Agreement

WS-Agreement define uma linguagem para publicar as capacidades de um fornecedor de serviços, criar acordos entre clientes e fornecedores sobre o uso dos serviços e monitorar os acordos em tempo de execução (ANDRIEUX et al, 2004). WS-Agreement toma como base a linguagem WSLA (KELLER e LUDWIG, 2003).

Um acordo entre um solicitante e um fornecedor de serviço especifica um ou mais objetivos de serviço, expressando os requisitos do solicitante e as garantias do fornecedor quanto à disponibilidade dos recursos e quanto à qualidade dos serviços. Por exemplo, um

acordo pode fornecer garantias quanto ao tempo de resposta e a disponibilidade do serviço e pode, também, assegurar a disponibilidade de recursos mínimos como CPU e armazenamento de dados. Além disso, WS-Agreement permite a definição de contextos nos quais as garantias são válidas, além de multas ou recompensas.

A Figura 11 apresenta um diagrama de seqüência mostrando como é a interação para a criação e monitoramento de um acordo. No passo 1 o `Initiator` solicita a utilização de *templates* disponíveis para os serviços oferecidos. No passo 2 o `Initiator` sugere uma oferta baseada nos *templates* adquiridos. No passo 3 o `Responder` decide se aceita ou não a oferta baseando-se na atual situação de disponibilidade de recursos. No passo 4 são retornadas as referências do acordo para o `Initiator`. No passo 5, inicia-se o monitoramento dos estados do acordo solicitando-se a consulta de estados dos acordos, dos serviços Web e dos termos do contrato.

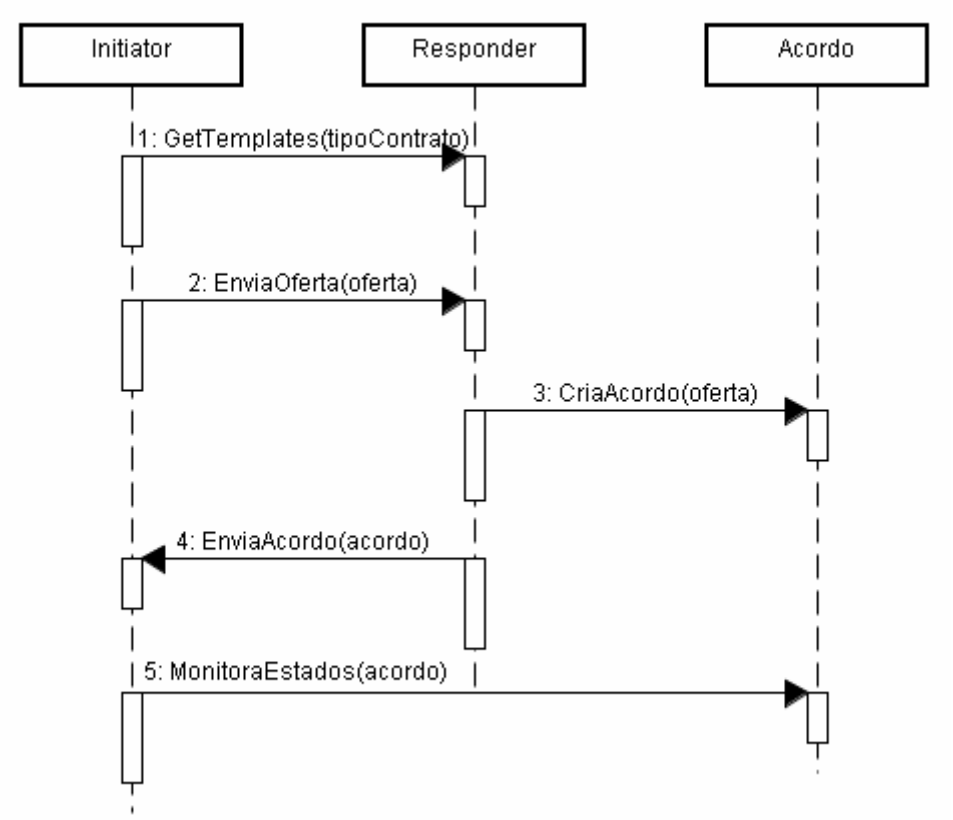

Figura 11: Diagrama de seqüência para criação e monitoramento de acordos. Baseado em (WALDRICH, 2006)

Uma especificação em WS-Agreement possui uma estrutura, chamada acordo, que consiste de dois componentes principais que são o `Contexto` e o `Termo` do acordo. A especificação contém operações tanto para criar e representar acordos sobre serviços oferecidos quanto para monitorar acordos.

A estrutura de um acordo WS-Agreement é formada por 3 principais elementos (ANDRIEUX et al, 2004):

- Um código identificador único e um nome descritivo;
- Uma seção de contexto: define as organizações participantes do acordo, tempo de expiração e outros elementos descritores do acordo;
- Termos: define as obrigações das duas partes no acordo. É o elemento principal do acordo. A especificação do acordo define dois tipos de termos: termos descritores de serviço e termos de garantia.
    - Termos descritores de serviços: descrevem os serviços Web que serão fornecidos pela organização provedora. Estes termos podem definir diferentes aspectos do serviço, por exemplo um arquivo WSDL de algum serviço Web. Um acordo WS-Agreement pode conter qualquer quantidade de SDTs. Os termos descritores de serviços podem ser de 2 tipos:
        - Propriedades do serviço: conjunto de propriedades mensuráveis e expostas, em termos de variáveis, associadas às características de serviços, para as quais serão definidas garantias de QoS. Cada propriedade de serviço deve ser referenciada por um termo de garantia posteriormente;
        - Referência do serviço: é um apontador para o serviço em questão ou um apontador para a descrição do serviço. Pode ser utilizado ao invés de descrever o serviço no acordo.
    - Termos de garantia: definem garantias mensuráveis individualmente. Definem as garantias para as propriedades de serviço. Os termos da garantia possuem 3 elementos:
        - `Service Scope`: define qual parte do serviço a garantia se aplica. Pode ser o serviço todo ou apenas uma operação do serviço;
        - `Service Level Objective`: define o que é garantido. A definição é escrita usando linguagem PMAC;

- `Business Value List`: define a valoração da garantia. Podem ser definidas multas ou recompensas.

A Listagem 3 apresenta um exemplo de contrato eletrônico em WS-Agreement. O nome `ContratoParaCredito` e o ID `contrato1` são especificados nas linhas 1 e 2 respectivamente. O contexto contém as organizações cliente VENDAS LTDA (linha 4) e provedora CRÉDITOS LTDA (linha 5) descritas por meio de seus endereços eletrônicos.

O termo descritor de serviço contém um nome identificador `TermoServicoCredito` e o nome do serviço a ser contratado `CreditoService` (linha 8). Para o exemplo, o termo descritor do serviço possui uma definição WSDL do serviço Web (linha 9). Esta definição WSDL especifica as características do serviço Web como operações e variáveis.

O tipo especial referência de serviço é utilizado para indicar qual a localização do serviço Web (linhas 13 a 15). Propriedades de serviço são as características do serviço que podem ser medidas como, por exemplo, `RequisicoesPorMinuto` e `DuracaoDaConsulta` (linhas 18 a 24) e que devem ser referenciadas posteriormente por um termo de garantia. Cada variável definida (linhas 18 e 21) representa uma propriedade de serviço a ser medida. O elemento `Location` (linhas 19 e 22) contém expressões que apontam para conteúdos dos termos descritores de serviço e definem qual o serviço Web ou a operação de serviço Web que a variável referencia.

O elemento `Location` deve obrigatoriamente conter o `portType` e a `operation`, pois estas informações serão utilizadas para criar o `pointcut` dos aspectos de monitoramento. Um operador especial "`*`" pode ser utilizado para indicar que todas as operações devem ter a mesma propriedade de serviço. A propriedade de serviço `DuracaoDaConsulta` (linha 21) utiliza o operador "`*`" (linha 22) para indicar que esta propriedade se aplica a todas as operações do serviço `CreditoService` (linha 17).

Os termos de garantia para o serviço Web `CreditoService` (linha 27). Este termo de garantia especifica o tempo médio de resposta de 5 segundos (linhas 31 e 32) para o serviço Web em questão. Também é definida uma multa para avaliações a cada 60 segundos (linha 38) de 100 reais (linha 41) se o especificado não for cumprido.

```
01<wsag:AgreementOffer AgreementId=”contrato1”>
02    <wsag:Name>ContratoParaCredito</wsag:Name>
03    <wsag:Context>
```

```
04        <wsag:AgreementInitiator>http://www.vendasltda.com.br/</wsag:A...>
05        <wsag:AgreementResponder>http://www.creditos.com/</wsag:A...>
06    </wsag:Context>
07    <wsag:Terms>
08 <wsag:SDT Name=”TermoServicoCredito” ServiceName=”CreditoService”>
09 <wsdl:definitions ...> ...
10 </wsdl:definitions>
11 </wsag:ServiceDescriptionTerm>
12 <wsag:SR Name=”ReferenciaServicoCredito” ServiceName=”CreditoService”>
13    <wsa:EndpointReference>
14       <wsa:Address>http://localhost/active-bpel/services/CreditoService</wsa:A>
15    </wsa:EndpointReference>
16 </wsag:ServiceReference>
17 <wsag:SP Name=”PropriedadesServicoCredito” ServiceName=”CreditoService”>
18    <wsag:Variable Name=”Requisicoes” Metric=”RequisicoesPorMinuto”>
19       <wsag:Location>//wsag:SDT/[@name=”TermoServicoCredito”@portType=”
creditoPT”@operation=”solicitarCredito”]</wsag:L>
20    </wsag:Variable>
21    <wsag:Variable Name=”Duracao” Metric=”DuracaoDaConsulta”>
22       <wsag:Location>//wsag:SDT/[@name=”TermoServicoCredito”@portType=”
creditoPT”@operation=”*”]</wsag:L>
23    </wsag:Variable>
24.</wsag:ServiceProperties>
25 <wsag:GuaranteeTerm wsag:name=”CreditoTempoResposta”>
26    <wsag:ServiceScope>
27       <wsag:ServiceName>CreditoService</wsag:ServiceName>
28    </wsag:ServiceScope>
29    <wsag:ServiceLevelObjective>
30       <exp:Less>
31          <exp:Variable>Duracao</exp:Variable>
32          <exp:Value>5</exp:Value>
33       </exp:Less>
34    </wsag:ServiceLevelObjective>
35    <wsag:BusinessValueList>
36       <wsag:Penalty>
37          <wsag:AssessmentInterval>
38             <wsag:TimeInterval>P60S</wsag:TimeInterval>
39             <wsag:ValueUnit>Real</wsag:ValueUnit>
40             <wsag:ValueExpr>
41                <exp:Value>100</exp:Value>
42             </wsag:ValueExpr>
43          </wsag:AssessmentInterval>
44       </wsag:Penalty>
```

```
45    </wsag:BusinessValueList>
46 </wsag:GuaranteeTerm>
47    </wsag:Terms>
48</wsag:AgreementOffer>
```

Listagem 3: Exemplo de contrato escrito em WS-Agreement.

Acordos e termos possuem estados de tempo de execução que podem ser monitorados. O objetivo do monitoramento de um estado do termo é observar o cumprimento do acordo em tempo de execução. Para interpretar o estado de uma garantia, o estado do termo de serviço deve ser conhecido. Se um serviço não está em execução, um termo de garantia não pode ser determinado. Para interpretar o estado de um termo de serviço, todo o estado do acordo deve ser conhecido. Se um acordo não foi aceito, os termos de serviço e de garantia não são determinados.

A verificação dos estados dos acordos e, principalmente, dos termos requer uma infra-estrutura significativa e é dependente do ambiente da aplicação e de seu domínio. A abordagem WS-Agreement prevê que as funções específicas de monitoramento de estados do acordo devam ser implementadas de acordo com o domínio da aplicação a que se aplica o contrato eletrônico. Os estados do acordo são definidos pela abordagem WS-Agreement como segue:

- Pendente: significa que a oferta de acordo foi feita, mas não foi aceita nem rejeitada.
- Observado: significa que a oferta de acordo foi feita e aceita. Este estado deve vir após o estado pendente.
- Rejeitado: significa que a oferta de acordo foi feita e rejeitada. Este estado deve vir após o estado pendente.
- Completo: significa que a oferta de acordo foi feita, aceita e que todas as atividades relacionadas ao acordo foram finalizadas. Este estado deve vir após o estado observado.

O estado do serviço para cada termo de descrição de serviço que abstratamente descrevem o estado de um serviço independente de seu domínio também pode ser requerido de acordo com a abordagem WS-Agreement. Cada elemento da lista é uma tupla (ID do termo, estado do termo de serviço). Os estados dos termos de serviço podem ser:

- Não pronto: o serviço ainda não pode ser usado.

- Pronto: o serviço pode iniciar a execução imediatamente pelo cliente ou pelo provedor.
- Processando: o serviço está pronto e atualmente está processando uma requisição ou está de outra forma ativo.
- Ocioso: o serviço está pronto, porém atualmente não usado.
- Completado: o serviço não pode ser usado mais e qualquer atividade do provedor foi finalizada.

A propriedade que representa o estado do preenchimento de cada termo de garantia do acordo também é disponibilizada de acordo com a abordagem WS-Agreement. Cada elemento da lista é uma tupla (ID do termo, estado do termo de garantia). Os estados das garantias podem ser:

- Preenchido: atualmente a garantia foi preenchida.
- Violado: atualmente a garantia foi violada.
- Não determinado: nenhuma atividade relativa a esta garantia ainda ocorreu ou está ocorrendo atualmente.

CREMONA (LUDWIG et al, 2004) é uma das poucas propostas de implementação do modelo WS-Agreement existente na literatura. Ela oferece aos provedores uma infra-estrutura para gerenciar os acordos, implementar interfaces, verificar disponibilidade da capacidade do serviço e expor os estados do acordo em tempo de execução. Ela também oferece aos clientes uma infra-estrutura para ler moldes, preenchê-los de modo a criar o acordo e assim monitorar seus estados e seu tempo de execução. A biblioteca Java do CREMONA implementa as interfaces do WS-Agreement de modo a oferecer funcionalidades de gerenciamento para as instâncias e moldes de acordos. Também define abstrações do sistema provedor de serviço para moldes e instâncias de acordos. As abstrações do sistema provedor de serviços podem ser implementadas em um ambiente de aplicação específico. Mesmo diante dessa proposta, nota-se algumas lacunas como falta de modularização de interesses de monitoramento que são especificados de forma espalhada no processo de negócio.

A próxima seção apresenta a especificação AO4BPEL que é uma iniciativa de inserir o paradigma orientado a aspectos em processos de negócio.

2.7. AO4BPEL

A proposta contida nesta dissertação utiliza o conceito de aspectos em processos de negócio. O processo de negócio é a aplicação principal e os aspectos são atividades de monitoramento do uso dos serviços Web pelo processo de negócio. Assim, faz-se necessário um recurso que associe a tecnologia de aspectos com processos de negócio.

Umas das iniciativas de introdução do conceito de aspectos em processos de negócios é a especificação AO4BPEL (*Aspect-Oriented extension to BPEL4WS*) (CHARFI e MEZINI, 2004). É uma proposta acadêmica de extensão de WS-BPEL com mecanismos de composição orientados a aspectos. AO4BPEL se propõe a resolver dois grandes problemas existentes nas linguagens de composição de serviços Web: (i) a modularização da especificação - uma vez que alguns interesses relativos à especificação da composição como controle de acesso, autenticação, e auditoria, não estão modularizados na descrição de processos; e (ii) a lógica da composição é tradicionalmente pré-definida e estática, porém muitos serviços Web novos são publicados e outros tantos desaparecem rotineiramente. Além disso, as organizações envolvidas podem alterar suas regras, parceiros e condições de colaboração. Isto motiva a necessidade de uma composição dinâmica dos serviços (CHARFI e MEZINI, 2004).

A especificação AO4BPEL é formada por duas partes principais: o processo de negócio WS-BPEL e os aspectos. O processo de negócio WS-BPEL não possui diferenças para o processo de negócio apresentado na seção 2.4, sendo utilizados os mesmos comandos e mesma linguagem do processo WS-BPEL descrito anteriormente. Os aspectos utilizados em WS-BPEL possuem características próprias específicas da natureza de processos de negócio, porém a idéia básica de se ter interesses transversais que entrecortam determinados pontos de um processo principal é a mesma da linguagem de aspectos descrita na seção 2.1. O processo de negócio e os aspectos definidos devem ser publicados em um servidor para execução. Ao executar o processo de negócio, o servidor identificará qual o momento que um *joinpoint* é encontrado e executará os aspectos inserindo, assim, seu *advice* no processo de negócio. As características próprias existentes nos aspectos da AO4BPEL são explicadas a seguir.

Os aspectos AO4BPEL são escritos em linguagem XML e utilizam a linguagem XPath para definir os *pointcuts*. Os *pointcuts* são maneiras de selecionar conjuntos de *joinpoints* que atuarão no processo de negócio com algum interesse entrecortante que deve ser executado. Os atributos do processo de negócio ou de uma certa atividade do processo podem ser utilizados como predicados para escolher certos *joinpoints* relevantes.

Um `advice` em AO4BPEL é uma atividade ou um conjunto de atividades especificado em WS-BPEL que deve ser executada antes, depois, em volta de, ou paralelamente à outra atividade, no caso a atividade que está no ponto onde foi identificado o `joinpoint`.

Um exemplo do aspecto denominado `Contador` e sua correspondente implementação em XML são apresentados na Figura 12 e Listagem 4 respectivamente.. Este aspecto conta o número de vezes que a operação `encontraUmVoo` do serviço Web do parceiro `Lufthansa` é invocada em qualquer processo de negócio. Este é um exemplo de um interesse entrecortante que pode ser modularizado como um aspecto. Este aspecto declara o serviço Web `ContadorWS` que provê a operação `incContador` para fazer a contagem. Esta operação recebe um número inteiro como parâmetro de entrada para ser adicionado ao contador de invocações corrente. O `pointcut Invocação Lufthansa` captura todas as invocações da operação `encontraUmVoo` de qualquer processo de negócio publicado no servidor. O `advice` do tipo depois associado ao `pointcut` incrementa o número total de chamadas um a um.

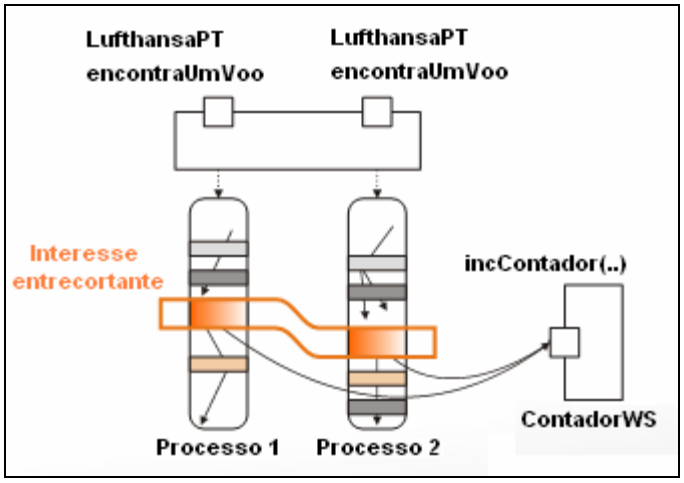

Figura 12: Aspecto Contador. Adaptado de: (CHARFI e MEZINI, 2004)

```
<aspect name="Contador">
 <partnerLinks>
  <partnerLink name="ContadorWS" partnerLinkType="ContadorPLT" myRole="caller"
partnerRole="counter"/>
 </partnerLinks>
 <variables>
  <variable name="incrementaRequest" messageType="contadorInput"/>
 </variables>
 <pointcutandadvice>
  <pointcut name="Invocação Lufthansa" contextCollection="true">
```

```
  //process//invoke[@portType=”LufthansaPT” and @operation=”encontraUmVoo”]
  </pointcut>
  <advice type=”after”>
   <sequence>
    <assign>
    <copy>
    <from expression=”1”/>
    <to variable=”incrementaRequest” part=”incrementarEm”/>
    </copy>
    </assign>
    <invoke partnerLink=”ContadorWS” portType=”ContadorPT” operation=”incContador”
inputVariable=”incrementaRequest” outputVariable=”incrementaResponse”/>
   </sequence>
  </advice>
 </pointcutandadvice>
</aspect>
```

Listagem 4: Implementação XML do Aspecto contador.

Em AO4BPEL, o processo de negócio é especificado em WS-BPEL e os aspectos são especificados de acordo com o modelo apresentado na Figura 13. Os aspectos possuem três partes: `PartnerLinks`, `Variables` e `PointcutAndAdvice`. Na parte de `PartnerLinks` são identificados os parceiros que disponibilizarão os serviços Web a serem utilizados no `Advice`. Na parte de `Variables` são identificadas quais as variáveis serão utilizadas para a realização do `Advice`. A parte de `PointcutAndAdvice` encapsula os `Pointcuts` e os `Advices`. Os `pointcuts` são escritos em linguagem Xpath (CLARK e DEROSE, 1999) e podem utilizar atributos do processo de negócio ou de uma certa atividade do processo. Os `pointcuts` identificam quais os pontos em que o aspecto deve atuar no processo de negócio. Um `pointcut` pode especificar um ou vários pontos em um ou vários processos de negócio. Os `advices` são as atividades WS-BPEL que são executadas quando um `pointcut` é identificado. No `advice` podem ser criadas quaisquer atividades WS-BPEL.

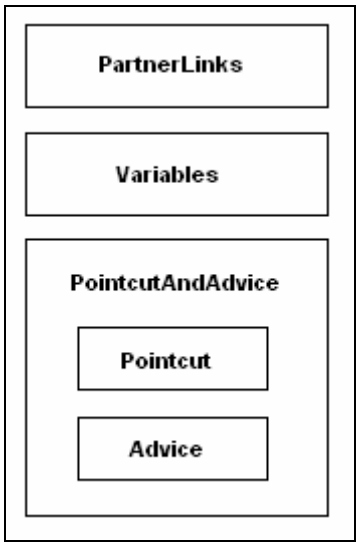

Figura 13: Modelo de aspectos para AO4BPEL. Adaptado de: (CHARFI e MEZINI, 2004)

A especificação AO4BPEL (CHARFI e MEZINI, 2004) é apenas um modelo teórico do uso de aspectos em processos de negócio. Uma proposta de implementação de middleware servidor para suporte a AO4BPEL é apresentada em (CHARFI e MEZINI, 2006). Este modelo ainda não é disponibilizado para uso o que torna a execução de processos AO4BPEL ainda não possível. O modelo de Charfi e Mezini (2006) propõe que o processo de negócio seja criado em WS-BPEL normalmente e que sejam definidos os aspectos como descrito acima. Ao publicar o processo, os aspectos também são publicados e disponibilizados no servidor. Este modelo é baseado no trabalho no servidor WS-BPEL proposto pela IBM (CURBERA el al, 2002).

Além de AO4BPEL, tem a abordagem Padus (BRAEM et al, 2006) que também aborda o uso de aspectos em processos WS-BPEL com a linguagem de *PointCut* baseada na linguagem PROLOG. Possui a vantagem de ser compatível com a infra-estrutura WS-BPEL depois de ter ocorrido a montagem dos aspectos ao processo de negócio, porém é uma abordagem que ainda não possui apoio computacional.

2.8. Considerações Finais

As organizações precisam cada vez mais disponibilizar meios para que seus processos de negócio possam ser automatizados e integrados com processos de negócio de diversas outras organizações. Os processos de negócio inter-organizacionais se beneficiam da tecnologia de serviços Web com a computação orientada a serviços. Os serviços Web podem ser publicados e utilizados por diferentes organizações de forma padronizada. A padronização também é um objetivo para as linguagens de composição de serviços Web como WS-BPEL. Esta linguagem de composição de serviços define quais são os serviços Web do processo de negócio e os

parceiros envolvidos. A utilização de serviços Web pode requerer um contrato eletrônico para regulamentar a utilização de serviços. Uma das linguagens utilizadas para definição de contratos eletrônicos é WS-Agreement que especifica atributos de qualidade de serviço a serem contratados. O monitoramento do contrato eletrônico se torna importante para o acompanhamento do que foi acordado no contrato. O último conceito apresentado neste capítulo foi o paradigma orientado a aspectos com a definição genérica de orientação a aspectos e com a apresentação da linguagem de modelagem de processos de negócio com utilização de aspectos AO4BPEL. O próximo capítulo apresenta a proposta AspectMonitor que é uma abordagem com a finalidade de realizar monitoramento de contratos eletrônicos utilizando-se do conceito de aspectos.

3. AspectMonitor: uma abordagem para monitoramento de contratos eletrônicos com aspectos

Em um ambiente COS, as organizações cliente podem contratar serviços Web de organizações provedoras para serem utilizados em seus processos de negócio. O estabelecimento de um acordo entre organizações cliente e provedoras permite que sejam definidas regras de utilização dos serviços Web. O acordo entre organizações é formalizado por meio de contratos eletrônicos. Um contrato eletrônico é criado entre as organizações para garantir a qualidade dos serviços necessários para que a organização cliente execute seu processo de negócio de forma confiável.

O monitoramento de um contrato eletrônico tem a finalidade de verificar se as propriedades de QoS estão sendo cumpridas pelas organizações. As propostas de linguagens de especificação de contratos eletrônicos como WSLA (DAN et al, 2004) e WS-Agreement (ANDRIEUX et al, 2004) definem mecanismos de monitoramento que apresentam algumas desvantagens como o excesso de instrumentação necessária no processo de negócio, excesso de novas funções que devem ser implementadas no cliente e servidor e à falta de ferramentas disponíveis para implementação efetiva do estabelecimento e monitoramento propostos.

A abordagem AspectMonitor é a proposta desta dissertação para realização de monitoramento de contratos eletrônicos descritos em WS-Agreement. Esta abordagem utiliza conceitos de aspectos aplicados a processos de negócio formados pela composição de serviços Web. AspectMonitor simplifica o monitoramento de contratos eletrônicos em WS-Agreement por meio da aplicação dos seguintes princípios:

- o contrato é pré-estabelecido entre o cliente e o provedor e definido utilizando a estrutura de oferta existente em WS-Agreement. O documento XML de oferta WS-Agreement pode ser criado por meio de diversas abordagens como a própria WS-Agreement ou a abordagem baseada em características proposta por Fantinato (2007);

- o contrato eletrônico é mapeado para aspectos de monitoramento. Os aspectos de monitoramento possuem em seus *Advices* chamadas para serviços Web monitores e variáveis utilizadas para o monitoramento relacionadas aos atributos QoS do serviço Web monitorado. Os aspectos monitores atuam no processo de negócio assim que seus *JoinPoints* são detectados na execução do processo de negócio.

Um ponto chave para a aplicação de AspectMonitor é o correto mapeamento de contratos eletrônicos em aspectos de monitoramento. Os aspectos de monitoramento devem possuir informações relevantes que auxiliem no monitoramento do serviço Web a que ele se refere. Por exemplo, para monitorar o número de vezes que um serviço é chamado, o aspecto deve possuir como variável o nome do serviço; e, para monitorar o tempo de resposta de um serviço, o aspecto deve possuir em suas variáveis o horário de início e de fim da execução do serviço Web.

Assim, a abordagem AspectMonitor possui os seguintes objetivos:

- criar formas mais elaboradas de processar os estados dos acordos por meio de serviços Web de monitoramento que são invocados por aspectos inseridos no processo de negócio;
- realizar a separação de interesses de monitoramento de contratos eletrônicos do interesse principal do processo de negócio utilizando para isto o paradigma orientado a aspectos;
- permitir a terceirização de atividades de monitoramento de contratos eletrônicos para organizações específicas;
- reduzir a complexidade do ambiente necessário para monitoramento, pelo fato do monitoramento ser realizado por serviços Web, que já é uma tecnologia em uso no domínio da aplicação não necessitando incluir diferentes tecnologias para este fim.

Este capítulo apresenta a abordagem AspectMonitor. É definida uma arquitetura para implementação da abordagem, as atividades e responsabilidades das organizações envolvidas e um conjunto de regras de mapeamento de contratos eletrônicos para aspectos de monitoramento. Um cenário exemplo é definido como forma de ilustrar os exemplos desse capítulo.

3.1. Cenário de ilustração da abordagem

A organização VENDAS LTDA utiliza um sistema de vendas de produtos. Para a realização destas vendas é utilizado um sistema informatizado com arquitetura COS. O processo de venda é implementado por meio da composição de serviços Web que realizam o pedido, a verificação de crédito do cliente e a efetivação da venda, caso o cliente tenha o crédito confirmado. O processo de negócio é formado por 3 serviços Web: serviço de pedido, serviço de consulta de crédito e serviço de efetivação da compra. Os serviços de pedido e de

efetivação da compra são próprios da organização cliente. O serviço de consulta de crédito é disponibilizado pela organização CRÉDITOS LTDA especializada em verificações de crédito. As organizações então efetivam um contrato eletrônico para definir e formalizar a utilização do serviço de consulta de crédito. O contrato estabelece que a organização cliente tem disponíveis os seguintes parâmetros de QoS: pode utilizar o serviço até 10 vezes por minuto e o tempo máximo de execução deve ser de 5 segundos por consulta.

Com a AspectMonitor é necessário definir uma organização monitora por isso também é definido que os parâmetros de QoS do serviço Web contratado será monitorado pela organização MONITOR LTDA. Assim, o serviço Web da CRÉDITOS LTDA é utilizado no processo de negócio da VENDAS LTDA e os valores referentes aos parâmetros de QoS são enviados à MONITOR LTDA que armazena estes dados para uso posterior.

3.2. Arquitetura AspectMonitor

A arquitetura de execução concebida para a AspectMonitor é diferente da considerada pelo WS-Agreement. A arquitetura WS-Agreement é mais abrangente permitindo a realização de acordos entre serviços de diferentes tipos não se restringindo a serviços Web. A arquitetura AspectMonitor se restringe a serviços Web pelo fato de ser implementada sobre um processo de negócio formado pela composição de serviços Web.

A arquitetura da AspectMonitor, conforme mostra a Figura 14, é formada por 3 organizações: cliente, provedora(s) e monitora(s). A organização cliente tem a arquitetura mais complexa incluindo uma infra-estrutura para definição de contratos eletrônicos e um servidor AO4BPEL. Um sistema COS pode ser necessário caso a organização cliente possua serviços Web próprios que são utilizados no processo de negócio e o servidor AO4BPEL permite a inserção de aspectos em processos de negócio compostos de serviços Web. A(s) organização(ões) provedora(s) possui(em) um sistema COS que armazena serviços Web de diversas naturezas e utilidades. O sistema COS da(s) organização(ões) monitora(s) possui(em) serviços Web específicos para monitoramento de atributos QoS de outros serviços Web.

O processo de negócio da organização cliente realiza chamadas aos serviços Web que podem ser próprios ou terceirizados. Caso o serviço Web requisitado possua parâmetros de QoS estabelecidos em contratos eletrônicos, um aspecto de monitoramento intercepta o serviço e realiza chamadas aos serviços Web da(s) organização(ões) monitora(s), enviando os valores relacionados aos parâmetros de QoS extraídos na execução do processo de negócio.

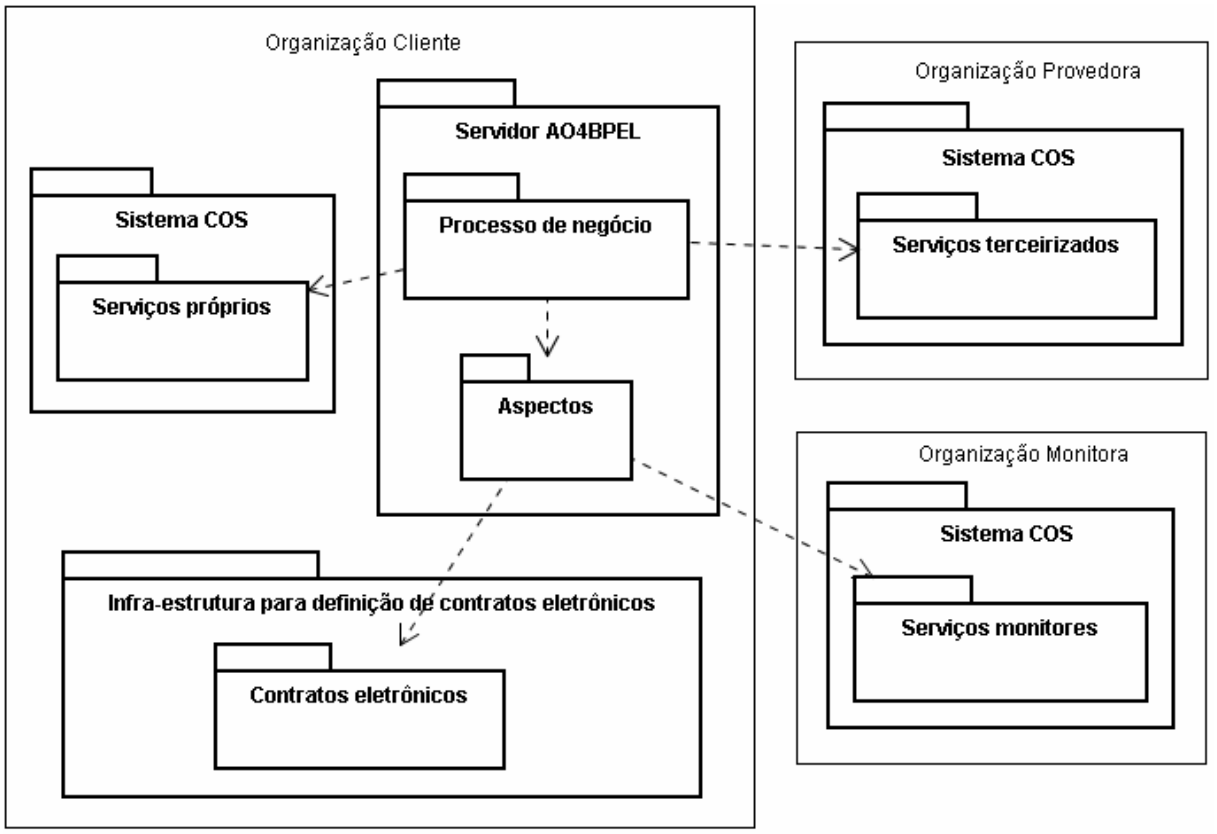

Figura 14: Arquitetura AspectMonitor

Fazendo uma análise mais detalhada da arquitetura da organização cliente percebe-se que a elaboração de contratos eletrônicos para gerenciar processos de negócio WS-BPEL exige o uso de diversas linguagens tais como WS-BPEL, WSDL, WS-Agreement. Um meta-modelo representando a interação entre essas linguagens é apresentado em Fantinato (2007) para apoiar a geração de contratos eletrônicos baseado em modelo de características. Este meta-modelo foi estendido para representar a integração de aspectos no contexto de contratos eletrônicos e processos de negócios. Os aspectos neste modelo têm a finalidade de apoiar o monitoramento de contratos eletrônicos durante a execução do processo de negócio. O meta-modelo estendido é apresentado na Figura 15. Os aspectos estão representados no meta-modelo por retângulos arredondados do lado direito da figura. Os demais elementos são integrantes do modelo original de Fantinato (2007). A abordagem de aspectos introduz 5 elementos ao modelo:

- `Aspect`: é o aspecto em si.
- `Partner Link:` representa os parceiros que atuam por meio dos aspectos;
- `Variables:` representa as variáveis necessárias à execução do `advice` do aspecto;

- `Pointcut:` representa o ponto do processo de negócio onde o aspecto vai atuar. Geralmente formado pela identificação de `portType` e `operation` que o aspecto deve atuar;
- `Advice:` é um trecho de processo de negócio a ser inserido pelo aspecto `no pointcut`. Este trecho pode conter várias atividades WS-BPEL como chamadas `invoke` a serviços Web.

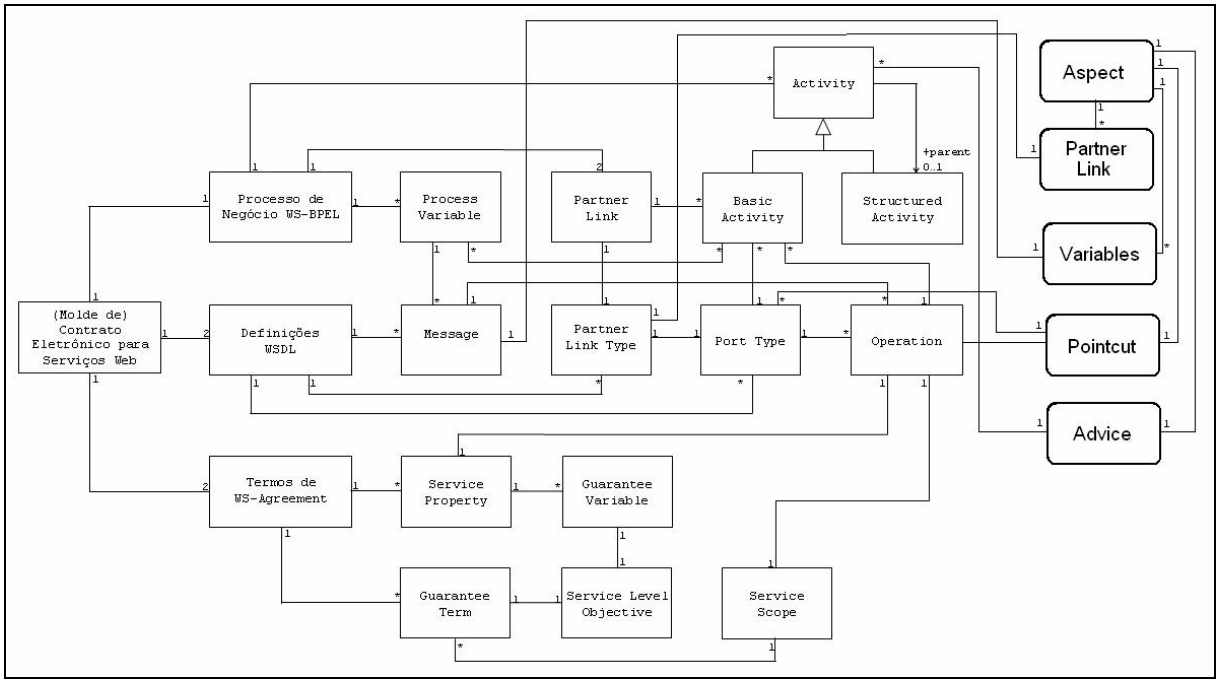

Figura 15: Meta-modelo de Fantinato (2007) extendido com aspectos

Baseado no meta-modelo da Figura 15 foi possível visualizar as regras de mapeamento de contratos eletrônicos para aspectos de monitoramento utilizadas na abordagem AspectMonitor.

3.3. Atividades e responsabilidades da AspectMonitor

As atividades e responsabilidades de cada organização estabelecida pela abordagem AspectMonitor são apresentadas nesta seção. O monitoramento envolve três organizações distintas com diferentes responsabilidades, conforme descrito na arquitetura. A Figura 16 apresenta um diagrama contendo as atividades e responsabilidades envolvidas na abordagem AspectMonitor.

Inicialmente, a organização cliente identifica a necessidade de criar um processo de negócio formado por uma composição de serviços Web. Deste modo, a organização cliente necessita definir quais serviços Web serão utilizados no processo de negócio. A organização

cliente pode desenvolver os serviços Web, mas também pode buscá-los em organizações provedoras. A forma como ocorre essa procura por serviços Web não é parte da abordagem AspectMonitor, fica à cargo da organização cliente decidir como fará a pesquisa por serviços Web. A organização provedora encontrada pela organização cliente então disponibiliza as características dos serviços Web em WSDL. Com a definição dos serviços Web, a organização cliente pode então escrever o processo de negócio utilizando AO4BPEL. Em seguida, a organização cliente define contratos eletrônicos de acordo com os níveis de QoS disponíveis na organização Provedora. A forma como ocorre essa negociação também não é definida pela abordagem AspectMonitor, esta pode ser informal ou podem ser utilizadas outras propostas de negociação de atributos de QoS como a presente em Fantinato (2007) ou a existente em WS-Agrement (ANDRIEUX et al, 2004).

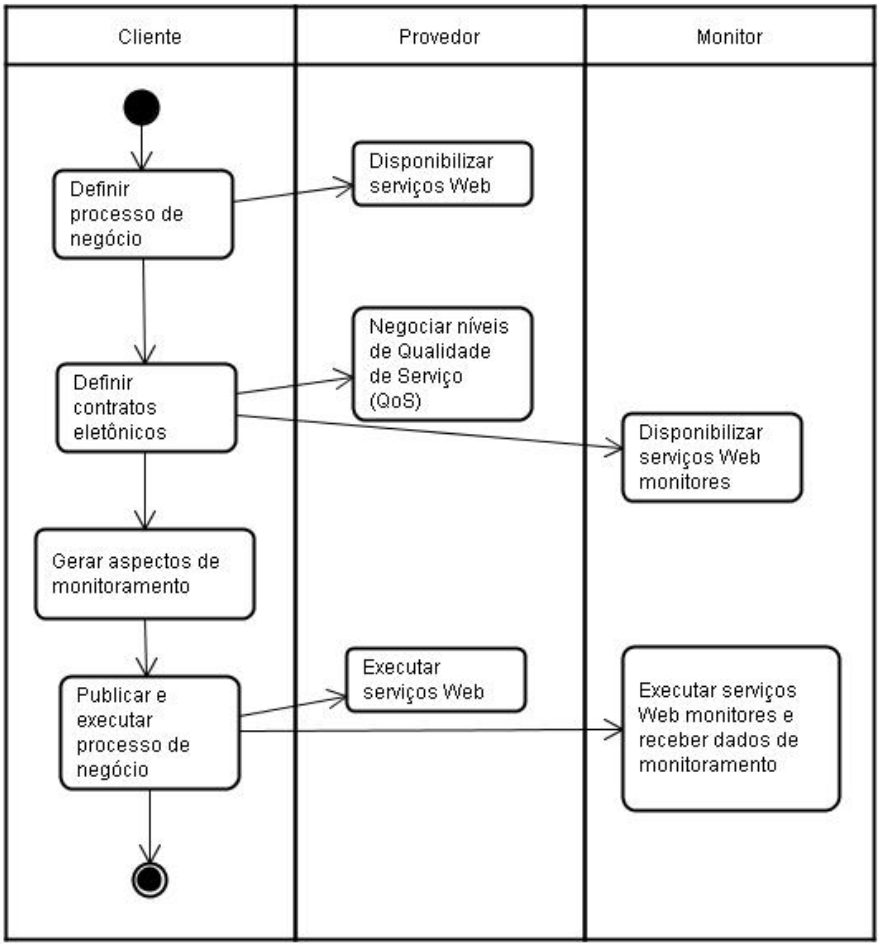

Figura 16: Responsabilidades e atividades AspectMonitor

Com os contratos definidos deve-se então fazer uma busca por organizações monitoras. As organizações monitoras disponibilizam serviços Web monitores que receberão os dados de monitoramento quando o processo de negócio for executado. A organização

cliente após realizar as negociações com organizações provedoras e monitoras terá definido um contrato eletrônico. De acordo com a abordagem AspectMonitor, este contrato eletrônico é escrito em linguagem de ofertas presente em WS-Agreement. A partir desses contratos, a organização cliente está apta a gerar os aspectos monitores. Para isto utilizará um apoio computacional específico que contém as regras de mapeamento de contratos eletrônicos WS-Agreement para aspectos AO4BPEL. Nessa etapa do processo, a organização cliente possui o processo de negócio AO4BPEL, os contratos eletrônicos WS-Agreement e os aspectos AO4BPEL. A organização cliente então publica o processo de negócio e os aspectos em um servidor AO4BPEL e pode então utilizar o processo de negócio. Ao utilizar o processo de negócio, os serviços Web da organização cliente são requisitados para realizar as atividades do processo de negócio e os serviços Web da organização monitora são requisitados para realizar as atividades de monitoramento.

3.4. Definir serviços Web a utilizar e escrever processo de negócio

A organização cliente ao criar seu processo de negócio identifica quais serviços Web devem compor este processo. Os serviços Web do processo de negócio podem ser desenvolvidos pela própria organização cliente ou desenvolvidos por organizações provedoras de serviços.

Os serviços Web selecionados são utilizados para compor o processo de negócio AO4BPEL. Os serviços Web são inseridos no processo de negócio por meio de seus arquivos WSDL. Considerando o cenário exemplo, a Listagem 5 apresenta um trecho do processo de negócio AO4BPEL que contém um `Invoke` para a operação `solicitarCredito` do serviço Web `Credito1`

```
<bpel:invoke inputVariable="CreditoRequest" name="Credito1"
operation="solicitarCredito" outputVariable="creditoResponse"
partnerLink="creditoPLT" portType="ns1:creditoPT">
```

Listagem 5: Trecho do processo de negócio AO4BPEL

Nota-se que a chamada `Invoke` da Listagem 5 necessita de várias informações referentes ao serviço Web que vai ser invocado, como variáveis de entrada e saída e operação a ser utilizada. Essas informações são encontradas nos arquivos descritores WSDL dos serviços Web. A Listagem 6 apresenta trechos do arquivo WSDL do serviço de crédito com a definição da operação `solicitarCredito` (linha 9), variável de entrada `creditoRequest` (linha 10) e saída `creditoResponse` (linha 11) e os tipos das

variáveis que são `creditoResponse` para a variável de saída (linha 2) e `creditoRequest` para a variável de entrada (linha 5).

```
01 <wsdl:definitions ...>
02   <wsdl:message name="creditoResponse">
03      <wsdl:part name="creditoLiberado" type="xsd:string"/>
04   </wsdl:message>
05   <wsdl:message name="creditoRequest">
06      <wsdl:part name="dadosCliente" type="xsd:string"/>
07   </wsdl:message>
08   <wsdl:portType name="creditoPT">
09      <wsdl:operation name="solicitarCredito">
10         <wsdl:input message="impl:creditoRequest" name="creditoRequest"/>
11         <wsdl:output message="impl:creditoResponse" name="creditoResponse"/>
12      </wsdl:operation>
13   </wsdl:portType>...
14 </wsdl:definitions>
```

Listagem 6: Arquivo WSDL do serviço Web de crédito

Todos os serviços Web utilizados na composição do processo de negócio devem ter disponíveis seus arquivos WSDL para que seja possível escrever o processo de negócio. A forma como ocorre a procura por serviços Web de organizações provedoras é indiferente para a abordagem AspectMonitor.

Após escrever o processo de negócio AO4BPEL, devem ser negociados os termos do contrato eletrônico que regulamentará a utilização de serviços Web terceirizados. A próxima seção discute a forma como pode ser realizada esta negociação e especifica como é o contrato eletrônico e seu mapeamento para aspectos de monitoramento.

3.5. Mapeamento de contratos eletrônicos para aspectos de monitoramento

Nesta atividade, as organizações cliente e provedora devem negociar os níveis de QoS que estarão disponíveis para utilização dos serviços Web terceirizados. A abordagem AspectMonitor independe da forma como ocorre a negociação e criação do contrato. Algumas formas de negociação podem ser encontradas na abordagem WS-Agreement e na abordagem de Fantinato (2007). Após a negociação, é necessário ter um contrato eletrônico que especifique essas garantias.

O contrato eletrônico utilizado pela abordagem AspectMonitor é baseado na linguagem de oferta da abordagem WS-Agreement. Esta linguagem permite que sejam

inseridos trechos escritos em outra linguagem XML específica do domínio da aplicação. Como o domínio de aplicação da abordagem AspectMonitor se restringe aos serviços Web as seguintes linguagens adicionais são utilizadas:

- WSDL (CHRISTENSEN et al, 2001): utilizada na seção *ServiceDescriptionerm* e tem a finalidade de descrever qual o serviço Web está sendo referenciado no contrato;
- WS-Addressing (BOX e CURBERA, 2007): utilizada na seção *ServiceReference*. Indica em qual endereço Web está o serviço Web em questão;
- XPAth (CLARK e DEROSE, 1999): uma expressão XPath indica com qual parte do serviço Web as variáveis do *ServiceProperties* estão relacionadas. Pode ser o serviço Web todo ou apenas determinadas operações;
- PMAC (PMAC, 2005): são as expressões utilizadas na seção *Guarantee Term*. Define a quais condições a garantia se aplica e o que é garantido;

O contrato contém três elementos principais: nome, contexto e termos. Considerando o cenário exemplo, a Listagem 7 apresenta a estrutura XML do conteúdo do contrato. O nome `ContratoParaCredito` e o ID `contrato1` são especificados nas linhas 1 e 2. O contexto contém as organizações cliente VENDAS LTDA (linha 4) e provedora CRÉDITOS LTDA (linha 5) descritas por meio de seus endereços eletrônicos.

```
01<wsag:AgreementOffer AgreementId=”contrato1”>
02    <wsag:Name>ContratoParaCredito</wsag:Name>
03    <wsag:Context>
04        <wsag:AgreementInitiator>http://www.vendasltda.com.br/</wsag:A...>
05        <wsag:AgreementResponder>http://www.creditos.com/</wsag:A...>
06    </wsag:Context>
07    <wsag:Terms>
08       ...
09    </wsag:Terms>
10</wsag:AgreementOffer>
```

Listagem 7: Estrutura XML do contrato eletrônico para a abordagem AspectMonitor

Os contratos da abordagem AspectMonitor utilizam o tipo principal de Termos Descritores de Serviço (`Service Description Terms` - SDTs) e os tipos especiais Referência de Serviço (`Service Reference`) e Propriedades de Serviço (`Service Properties`). É permitido utilizar qualquer quantidade de termos em cada contrato.

A Listagem 8 contém um exemplo de termos descritores de serviços para contratos eletrônicos. O SDT contém um nome identificador `TermoServicoCredito` e o nome do serviço a ser contratado `CreditoService` (linha 1). Para o domínio da abordagem AspectMonitor, o SDT possui uma definição WSDL do serviço Web (linha 2). Esta definição WSDL especifica as características do serviço Web como operações e variáveis.

O tipo especial de SDT Referência de Serviço é utilizado para indicar qual a localização do serviço Web (linhas 5 a 9). Propriedades de Serviço são as características do serviço que podem ser medidas como por exemplo `RequisicoesPorMinuto` e `DuracaoDaConsulta` (linhas 10 a 17) e que devem ser referenciadas posteriormente por um Termo de Garantia. Cada variável definida (linhas 11 e 14) representa uma propriedade de serviço a ser medida. O elemento `Location` (linhas 12 e 15) contém expressões que apontam para conteúdos dos SDTs e definem qual o serviço Web ou a operação de serviço Web que a variável referencia.

O elemento `Location` deve obrigatoriamente conter o `portType` e a `operation`, pois estas informações serão utilizadas para criar o `pointcut` dos aspectos de monitoramento. Um operador especial "`*`" pode ser utilizado para indicar que todas as operações devem ter a mesma propriedade de serviço. Na Listagem 8 a propriedade de serviço `DuracaoDaConsulta` (linha 14) utiliza o operador "`*`" (linha 15) para indicar que esta propriedade se aplica a todas as operações do serviço `CreditoService` (linha 10).

```
01 <wsag:SDT Name="TermoServicoCredito" ServiceName="CreditoService">
02 <wsdl:definitions ...> ...
03 </wsdl:definitions>
04 </wsag:ServiceDescriptionTerm>
05 <wsag:SR Name="ReferenciaServicoCredito" ServiceName="CreditoService">
06    <wsa:EndpointReference>
07       <wsa:Address>http://localhost/active-bpel/services/CreditoService</wsa:A>
08    </wsa:EndpointReference>
09 </wsag:ServiceReference>
10 <wsag:SP Name="PropriedadesServicoCredito" ServiceName="CreditoService">
11    <wsag:Variable Name="Requisicoes" Metric="RequisicoesPorMinuto">
12       <wsag:Location>//wsag:SDT/[@name="TermoServicoCredito"@portType="
creditoPT"@operation="solicitarCredito"]</wsag:L>
13    </wsag:Variable>
14    <wsag:Variable Name="Duracao" Metric="DuracaoDaConsulta">
15       <wsag:Location>//wsag:SDT/[@name="TermoServicoCredito"@portType="
creditoPT"@operation="*"]</wsag:L>
```

```
16    </wsag:Variable>
17.</wssag:ServiceProperties>
```

Listagem 8: Termos descritores do serviço de crédito

Os termos de garantia definem garantias mensuráveis individualmente. Os termos da garantia possuem 3 elementos:

- `Service Scope`: define qual parte do serviço a garantia se aplica. Pode ser o serviço todo ou apenas uma operação do serviço.
- `Service Level Objective`: define o que é garantido. A definição é escrita usando linguagem PMAC.
- `Business Value List`: define a valoração da garantia. Podem ser definidas multas ou recompensas.

A Listagem 9 apresenta os termos de garantia para o serviço Web `CreditoService` (linha 3). Este termo de garantia especifica o tempo médio de resposta de 5 segundos (linhas 7 e 8) para o serviço Web em questão. Também é definida uma multa para avaliações a cada 60 segundos (linha 14) de 100 reais (linhas 15 e 17) se o SLO não for cumprido.

```
01 <wsag:GuaranteeTerm wsag:name=”CreditoTempoResposta”>
02    <wsag:ServiceScope>
03       <wsag:ServiceName>CreditoService</wsag:ServiceName>
04    </wsag:ServiceScope>
05    <wsag:ServiceLevelObjective>
06       <exp:Less>
07          <exp:Variable>Duracao</exp:Variable>
08          <exp:Value>5</exp:Value>
09       </exp:Less>
10    </wsag:ServiceLevelObjective>
11    <wsag:BusinessValueList>
12       <wsag:Penalty>
13          <wsag:AssessmentInterval>
14             <wsag:TimeInterval>P60S</wsag:TimeInterval>
15             <wsag:ValueUnit>Real</wsag:ValueUnit>
16             <wsag:ValueExpr>
17                <exp:Value>100</exp:Value>
18             </wsag:ValueExpr>
19          </wsag:AssessmentInterval>
20       </wsag:Penalty>
21    </wsag:BusinessValueList>
```

```
22 </wsag:GuaranteeTerm>
```

Listagem 9: Termos de garantia do serviço de crédito

Ao gerar os contratos eletrônicos a organização cliente também já localiza e negocia com organizações monitoras a prestação de serviços de monitoramento de seu contrato eletrônico.

3.5.1 Regras para mapeamento de contratos eletrônicos para aspectos

Depois de definidos os contratos eletrônicos, os aspectos de monitoramento podem ser gerados. Para criar os aspectos de monitoramento a partir dos contratos eletrônicos é necessário o cumprimento de algumas regras de mapeamento que são apresentadas a seguir:

REGRA 1: cada propriedade de serviço descrita na seção de `ServiceProperties` é mapeada para um aspecto diferente porque cada propriedade será monitorada de forma independente;

REGRA 2: o `name` do aspecto é formado pelo nome do contrato eletrônico somado ao nome da propriedade de serviço;

REGRA 3: o `pointcut` é criado utilizando o `serviceName` do elemento `ServiceProperties` e o `portType` e `operation` contidos nas variáveis que descrevem as propriedades de serviço;

REGRA 4: as `messages e variables` são criadas de forma padronizada. Existe um tipo de dado e uma variável de entrada denominados respectivamente `monitorRequestType` e `monitorResquest` e um tipo de dados e uma variável de retorno denominados respectivamente `monitorResponseType` e `monitorReponse`;

REGRA 5: o elemento `type` deve ser preenchido levando em consideração qual o momento em que ocorrerá o monitoramento. Pode ser `after`, `before` ou `around`. Caso o tipo de propriedade de serviço que está sendo medida seja do tipo disponibilidade, acessibilidade, integridade, confiabilidade, regulamentação e segurança então será `after` pois o preenchimento da variável de entrada do serviço monitor depende da mensagem de retorno do serviço monitorado. Caso o tipo de propriedade de serviço que está sendo medida seja do tipo desempenho pode ser do tipo `around` ou `before`. O tipo `around` é utilizado quando a propriedade de serviço está medindo o tempo de resposta do serviço. Então se necessita invocar o

serviço monitor antes e depois do serviço monitorado para enviar o horário de início e de fim de forma segura. O tipo `before` é utilizado quando vai haver uma contagem do número de vezes que o serviço monitorado é executado.

REGRA 6: `inputVariable` é a variável definida na Listagem 10 `monitorRequest`;

REGRA 7: `outputVariable` é a variável definida na Listagem 10 `monitorResponse`;

REGRA 8: `portType` e `operation` do `invoke` devem ser preenchidos com dados do serviço monitor;

REGRA 9: o `serviceLinkType` deve ser preenchido com o `partnerLink` do serviço monitor.

3.5.2 Aplicação das regras no cenário exemplo

Os aspectos da abordagem AspectMonitor possuem variáveis e tipos de dados padronizados de acordo com o apresentado na Listagem 10. Estas variáveis correspondem às variáveis de entrada e saída dos serviços Web monitores, portanto um serviço Web monitor deve especificar como tipos de entrada e saída para as operações de monitoramento os tipos de dados e variáveis mostrados na Listagem 10.

O tipo de dado `monitorResquestType` (linha 2) representa a variável de entrada do serviço monitor e é formado por duas partes:

- `ServicePropertyID`: que deve ser a identificação de qual propriedade de serviço se refere este aspecto (linha 3).
- `ServicePropertyData`: é a informação principal do monitoramento. Esta informação pode variar de acordo com o tipo de requisito de QoS que a propriedade de serviço se refere (linha 4).

O tipo de dado `monitorResponseType` (linha 6) representa a variável de saída do serviço monitor e possui apenas uma parte que é a resposta do serviço monitor (linha 7). Esta mensagem de retorno do serviço monitor não influenciará na continuação da execução do processo de negócio.

```
01 <messages>
02   <message name="monitorRequestType">
```

```
03      <part name="ServicePropertyId" type="xsd:string"/>
04      <part name="ServicePropertyData" type="xsd:string"/>
05   </message>
06   <message name="monitorResponseType">
07      <part name="monitorData" type="xsd:string"/>
08   </message>
08 </messages>
10 <variables>
11   <variable name="monitorRequest" messageType="monitorRequestType"/>
12   <variable name="monitorResponse" messageType="monitorResponseType"/>
13 </variables>
```

Listagem 10: Tipos de dados das variáveis dos aspectos monitores

Considerando o cenário exemplo, a Listagem 11 apresenta um aspecto gerado pela propriedade de serviço `Requisicoes` existente no contrato eletrônico da Listagem 8. O nome do aspecto `Contrato1-Requisicoes` (linha 1) é o nome do contrato (Listagem 7, linha 1) somado ao nome da propriedade de serviço (Listagem 8, linha 11) que determina as requisições por minuto. O `PointCut CreditoInvoke` (linhas 7 a 9) é criado utilizando o `serviceName` do serviço a ser monitorado (Listagem 8, linha 10) somado ao `PortType` e o `Operation` contido nas variáveis que descrevem as propriedades de serviço que são encontradas nos elementos `Location` dos `ServiceProperties` (Listagem 8, linha 12).

Devem-se utilizar os dados referentes às organizações monitoras para definir o `ServiceLinkType` do serviço monitor que é `MonitorSLT` (linha 03) e quais são o `portType e operation` que são respectivamente `ServicoMonitor` e `monitora` (linha 22).

```
01 <aspect name="Contrato1-Requisicoes">
02    <partners>
03       <partner name="monitorPartner" serviceLinkType="MonitorSLT"/>
04    </partners>
05    <messages>...</messages>
06    <variables>...</variables>
07    <pointcut name="CreditoInvoke" contextCollection="true">
08       //process//invoke[@serviceName="CreditoService"@portType="creditoPT"
@operation="solicitarCredito"]
09    </pointcut>
10    <advice type="before">
11    <bpws:sequence>
12    <bpws:assign>
13       <bpws:copy>
```

```
14          <bpws:from expression="Contrato1-Requisicoes"/>
15          <bpws:to variable="monitorRequest" part="ServicePropertyId"/>
16       </bpws:copy>
17       <bpws:copy>
18          <bpws:from //process//invoke[@serviceStatus]/>
19          <bpws:to variable="monitorRequest" part="ServicePropertyData"/>
20       </bpws:copy>
21    </bpws:assign>
22       <bpws:invoke name="invoke" partner="monitorPartner"
portType="ServicoMonitor" operation="monitora" inputVariable="monitorRequest"
outputVariable="monitorResponse"/>
24    </advice>
25 </aspect>
```

Listagem 11: Aspecto de monitoramento gerado a partir de contratos eletrônicos

Durante a execução do processo de negócio o aspecto monitor será invocado e terá que receber os dados de monitoramento. É de grande importância para a abordagem AspectMonitor que as variáveis extraídas do processo de negócio pelos aspectos monitores (linha 18 da Listagem 11) e enviadas aos serviços monitores sejam relevantes para medir os termos descritores de serviço. As regras definidas pela abordagem AspectMonitor para preencher os dados de monitoramento em tempo de execução são explicadas a seguir:

- A variável de saída dos serviços monitores é sempre uma mensagem comum do tipo texto. Isso porque a resposta devolvida pelo serviço de monitoramento não interfere no processo de negócio.

- A variável de entrada possui duas partes. A `part ServicePropertyId` é preenchida com o nome do aspecto. De acordo com a Listagem 11 nota-se que a parte já está definida no momento de criação do aspecto. A `part ServicePropertyData` deve ser preenchida em tempo de execução e depende do tipo de requisito de QoS que está sendo medido. As informações que devem ser extraídas durante a execução do processo de negócio para preencher a parte `ServicePropertyData` são definidas abaixo. As variáveis estão diferenciadas por tipo de requisito de QoS que elas medirão:

    - disponibilidade: requisito que especifica se o serviço Web está presente para ser executado ou não. A informação a ser enviada para o serviço monitor é o *status* recebido na resposta do serviço monitorado;

- acessibilidade: requisito que especifica se o serviço Web pode ser acessado. A informação a ser enviada para o serviço monitor é o *status* recebido na resposta do serviço Web monitorado;
- integridade: requisito que especifica se a operação executada pelo serviço Web foi realizada completamente ou em caso de falha se foi totalmente desfeita. A informação a ser enviada para o serviço monitor é o *status* recebido na resposta do serviço Web monitorado;
- desempenho: requisito que especifica o número de vezes que um serviço Web pode ser requisitado em determinado tempo, ou seu tempo de resposta. As informações a serem enviadas para o serviço monitor são os tempos de início e fim da execução do serviço Web monitorado;
- confiabilidade: requisito que especifica a quantidade de erros durante a execução de um serviço Web. A informação a ser enviada para o serviço monitor é o *status* recebido na resposta do serviço Web monitorado;
- regulamentação: requisito que especifica os padrões seguidos por um serviço Web. A informação a ser enviada para o serviço monitor é o *status* recebido na resposta do serviço Web monitorado;
- segurança: requisito que especifica os artifícios de segurança utilizados com criptografia ou uso de conexão segura. A informação a ser enviada para o serviço monitor é o *status* recebido na resposta do serviço Web monitorado.

Os *status* das respostas dos serviços monitorados são utilizadas como entrada para os serviços monitores. Pode ocorrer que o serviço monitorado não esteja disponível para execução o que resultará num *status* de não disponível.

Neste ponto de desenvolvimento da abordagem AspectMonitor, a empresa cliente já possui os serviços Web necessários e seus arquivos WSDL, possui também o processo de negócio escrito em AO4BPEL, além dos contratos eletrônicos WS-Agreement e dos aspectos de monitoramento também escritos em linguagem AO4BPEL. O próximo passo é publicar o processo de negócio e os aspectos em um servidor AO4BPEL e executar o processo de negócio. A implementação da abordagem AspectMonitor é apresentada na seção a seguir.

3.6. Implementando a abordagem AspectMonitor

Baseado na arquitetura apresentada na seção 3.2, percebe-se que a implementação da abordagem AspectMonitor envolve 3 organizações distintas. As organizações provedoras e monitoras devem possuir um sistema COS para pode disponibilizar os serviços Web. Já existe apoio computacional para sistemas COS tornando possível a realização das atividades das organizações provedoras e monitoras.

A organização cliente necessita de um sistema que apóie a execução de processos de negócios em AO4BPEL para publicar o processo de negócio e os aspectos de monitoramento. Até a finalização desta dissertação, não foi encontrado um apoio computacional para AO4BPEL. A implementação da abordagem AspectMonitor necessitou sofrer algumas adaptações para ser realizada.

Para realizar alguns exemplos da abordagem proposta optou-se por utilizar WS-BPEL com apoio da ferramenta ActiveBPEL (ACTIVEBPEL, 2006). Como apresentado no capítulo 2, WS-BPEL não permite a especificação de aspectos. Assim, o processo WS-BPEL foi instrumentado introduzindo `pointcut` e adicionando o `advice` no momento definido pelo tipo do aspecto. Com isso o processo de negócio ganha uma instrumentação que simula o uso de aspectos e que é possível de ser executada com a tecnologia WS-BPEL. O processo AO4BPEL que foi criado na seção 3.4 é muito semelhante a um processo WS-BPEL então não há necessidade de mais adaptações.

Após instrumentar o processo de negócio com a simulação de aspectos de monitoramento, pode-se então publicar o processo WS-BPEL no servidor e realizar o uso do processo normalmente. Os serviços Web do provedor e do monitor serão invocados e realizarão suas atividades e suas medições.

3.7. Considerações finais

A abordagem AspectMonitor tem por objetivo realizar o monitoramento de contratos eletrônicos de forma não entrecortante ao processo de negócio principal. Outras abordagens de monitoramento exigem um grande esforço de instrumentação do processo de negócio para adicionar as funções relativas ao monitoramento. A abordagem AspectMonitor realiza o monitoramento por meio de aspectos que são executados juntamente com o processo de negócio sem a necessidade de se instrumentá-lo. Outro benefício existente na abordagem AspectMonitor é um domínio de aplicação mais simples por realizar o monitoramento por

meio de invocações de serviços Web, ao invés de implementar funções em linguagens especificas como propõe WS-Agreement.

Os papéis das organizações que participam da abordagem AspectMonitor foram esclarecidos nas seções anteriores. Apenas a organização cliente deve possuir um apoio computacional mais complexo que deve incluir suporte para tecnologia AO4BPEL. As organizações provedora e monitora necessitam apenas de apoio computacional para publicação e execução de serviços Web, o que é trivial.

Pelo fato de não existir apoio computacional para execução de processos AO4BPEL pode-se adaptar um modelo WS-BPEL por meio da inclusão dos aspectos de monitoramento no processo de negócio WS-BPEL. Como WS-BPEL e AO4BPEL utilizam os mesmos elementos de linguagem, exceto o fato que AO4BPEL faz uso de aspectos, não há necessidade de adaptações no processo de negócio AO4BPEL para transformá-lo em WS-BPEL. Com o apoio computacional existente para WS-BPEL é possível então executar o processo de negócio e observar a execução dos serviços Web das atividades normais e dos serviços Web de monitoramento.

A próxima seção apresenta um exemplo de aplicação desenvolvido por meio da abordagem AspectMonitor.

4. Um exemplo de aplicação da abordagem AspectMonitor

Neste capítulo é apresentado um exemplo aplicação da abordagem AspectMonitor com o intuito de avaliar sua aplicação em um estudo prático. O exemplo de aplicação serve de prova de conceito para avaliar seus benefícios e desvantagens. Nas próximas seções são apresentados trechos dos códigos-fonte gerados e os códigos completos estão nos apêndices ao final da dissertação.

4.1. Cenário do exemplo

A organização cliente Pacotes Turísticos LTDA utiliza um sistema de vendas de pacotes turísticos. Para a realização das vendas é utilizado um sistema informatizado com arquitetura COS. O processo de venda é implementado por meio da composição de serviços Web que realizam a compra das passagens, a reserva do hotel e a reserva de locação de automóveis. A organização cliente necessita implementar ou contratar serviços Web que realizem essas atividades.

Após análise de sua necessidade foi escolhida como melhor opção terceirizar os serviços Web de organizações provedoras pelo fato de os serviços Web terceirizados serem mais especializados cada um no seu domínio do que se fossem desenvolvidos pela própria organização cliente. O serviço de compra de passagens deve conter um banco de dados composto de várias opções de passagens em várias companhias. O serviço de reserva de hotel deve ser um especialista na catalogação de vários hotéis em diversas cidades. O serviço de reserva de automóveis deve estar interligado com empresas de locação de veículos.

O uso dos serviços Web terceirizados deverá ser regulamentado por meio de contrato eletrônico para que a organização cliente possa ter garantias de nível de QoS. A organização cliente tem um tempo máximo de resposta desejado para cada serviço Web para que não prejudique a realização de seu processo de negócio. Este tempo máximo deverá ser negociado com a organização provedora escolhida para prover o serviço Web. A organização cliente também deseja que os serviços terceirizados estejam disponíveis em determinados horários. A disponibilidade dos serviços Web também deverá ser negociada com a organização provedora.

Passagens LTDA, Hotéis LTDA e LocaAuto LTDA são as organizações escolhidas para disponibilizar os serviços de vendas de passagens, reserva de hotel e reserva de locação de veículos respectivamente. A escolha das organizações provedoras é descrita na seção 4.2.

A organização cliente juntamente com as organizações provedoras realizarão a negociação de níveis de QoS dos serviços prestados. A negociação, a definição de níveis de QoS e a criação do contrato são explicadas na seção 4.3.

Para monitorar os níveis de QoS, é necessário o monitoramento do contrato eletrônico que deverá ser realizado por uma organização monitora a ser definida também pela organização cliente.

Este cenário está de acordo com ambiente de execução da abordagem AspectMonitor. A aplicação da abordagem é descrita nas seções a seguir.

4.2. Definir serviços Web a utilizar e escrever processo de negócio

Após a organização cliente identificar a necessidade de utilização de um processo de negócio com tecnologia COS, é necessário fazer uma pesquisa para encontrar os serviços Web desejados e os provedores que possuem estes serviços.

Os nomes das organizações apresentados abaixo são apenas ilustrativos. Os serviços foram criados no ambiente de desenvolvimento Java Eclipse e todos os serviços foram publicados em um mesmo servidor Web Tomcat. A publicação dos serviços simula a existência de várias organizações. Portanto, os serviços Web existem e suas operações foram implementadas, mas a publicação ocorreu em apenas um servidor Tomcat.

O serviço de compra de passagens será disponibilizado pela organização Passagens LTDA. Este serviço contém uma operação que recebe os dados da viagem como por exemplo, dia, hora, local origem e local destino e realiza a compra das passagens. Também podem ser necessários dados adicionais para encontrar as passagens desejadas. Alguns trechos do arquivo WSDL do serviço Web `PassagensService` encontram-se na Listagem 12. No WSDL da Listagem 12 é possível observar que existe uma operação chamada `comprarPassagem` (linhas 8 a 12) e que esta operação possui o parâmetro de entrada do tipo `comprarPassagemRequest` (linhas 5 a 7) e o parâmetro de saída do tipo `comprarPassagemResponse` (linhas 2 a 4).

```
01 <wsdl:definitions ...>
02    <wsdl:message name="comprarPassagemResponse">
03       <wsdl:part name="passagemOut" type="xsd:string"/>
04    </wsdl:message>
05    <wsdl:message name="comprarPassagemRequest">
06       <wsdl:part name="dadosPassagem" type="xsd:string"/>
07    </wsdl:message>
```

```
08    <wsdl:portType name="PassagemCl">
09       <wsdl:operation name="comprarPassagem" parameterOrder="dadosPassagem">
10          <wsdl:input message="impl:comprarPassagemRequest"
name="comprarPassagemRequest"/>
11          <wsdl:output message="impl:comprarPassagemResponse"
name="comprarPassagemResponse"/>
12       </wsdl:operation>
13    </wsdl:portType>...
14 </wsdl:definitions>
```

Listagem 12: Trechos do WSDL do serviço de compra de passagens

O serviço de reserva de hotel será disponibilizado pela organização Hotéis LTDA. Este serviço possui uma operação que recebe os dados da reserva, como por exemplo local, quantidade de pessoas, data de chegada e data de saída e efetua a reserva dos quartos necessários. O serviço de reserva de automóveis será disponibilizado pela organização LocaAutos LTDA. Este serviço possui uma operação que recebe os dados da locação do automóvel como por exemplo, tipo do automóvel, quantidade de diárias e quilometragem e realiza a reserva do veículo. O WSDL desses serviços não são apresentados pois são muito semelhante ao WSDL do serviço `PassagensService.` Após definição de quais serviços Web serão utilizados é possível criar o processo de negócio AO4BPEL com a utilização das definições dos serviços Web contidos nos seus arquivos descritos WSDL.

A definição do processo de negócio da abordagem AspectMonitor deve ser realizada em linguagem AO4BPEL, porém como já explicado na seção 3.6, como não há apoio computacional para AO4BPEL a solução prática para simulação do processo de negócio é criá-lo utilizando o apoio computacional existente para WS-BPEL. A ferramenta de definição de processos de negócio WS-BPEL utilizada é a ActiveBPEL. Alguns trechos do processo de negócio são apresentados na Listagem 13. As organizações envolvidas no processo de negócio são definidas na seção de partnerLinks (linhas 3 a 8).

A operação `receive` (linha 9) é o início do processo de negócio. Nesta operação são identificados quais os parâmetros de entrada do processo de negócio que é a variável `reservarViagemRequest`. A operação `reply` (linha 11) é o final do processo que possui a variável de saída `reservarViagemResponse`. A operação `invoke` (linha 13) é o momento do processo de negócio onde o serviço Web `PassagemCl` é invocado para realizar sua operação `comprarPassagem`.

```
01 <bpel:process ...>
02 ...
```

```
03    <bpel:partnerLinks>
04       <bpel:partnerLink myRole="AgenciaTurismoRole" name="AgenciaTurismoPLT"
partnerLinkType="ns1:AgenciaTurismoPLT"/>
05       <bpel:partnerLink name="PassagemPLT" partnerLinkType="ns1:PassagemPLT"
partnerRole="PassagemRole"/>
06       <bpel:partnerLink name="HotelPLT" partnerLinkType="ns1:HotelPLT"
partnerRole="HotelRole"/>
07       <bpel:partnerLink name="CarroPLT" partnerLinkType="ns1:CarroPLT"
partnerRole="CarroRole"/>
08    </bpel:partnerLinks> ...
09    <bpel:receive createInstance="yes" name="RecebeDadosViagem"
operation="reservarViagem" partnerLink="AgenciaTurismoPLT"
portType="ns1:AgenciaTurismoPT" variable="reservarViagemRequest">
10    </bpel:receive>
11    <bpel:reply name="RetornaViagem" operation="reservarViagem"
partnerLink="AgenciaTurismoPLT" portType="ns1:AgenciaTurismoPT"
variable="reservarViagemResponse">
12    </bpel:reply> ...
13    <bpel:invoke inputVariable="comprarPassagemRequest" name="Passagem"
operation="comprarPassagem" outputVariable="comprarPassagemResponse"
partnerLink="PassagemPLT" portType="ns1:PassagemCl"> ...
14 </bpel:process>
```

Listagem 13: Trecho do processo de negócio WS-BPEL

A linguagem AO4BPEL e a linguagem WS-BPEL possuem a mesma estrutura e mesmos elementos diferenciando-se pelo fato de AO4BPEL disponibilizar meios para definir aspectos.

A próxima etapa a ser executada pela organização cliente após definir o processo de negócio e os serviços Web, é negociar junto à organização provedora quais os níveis de QoS que serão contratados e elaborar um contrato eletrônico regulamentando esses níveis. A próxima seção trata da definição dos contratos eletrônicos e de seu mapeamento para aspectos de monitoramento.

4.3. Contratos e Aspectos de Monitoramento

A organização cliente, nesta etapa, negocia os níveis de QoS com as organizações provedoras. A metodologia utilizada para esta negociação não é definida pela abordagem AspectMonitor e a negociação utilizada para este exemplo foi desenvolvida utilizando os conceitos existentes na abordagem WS-Agreement, porém sem apoio computacional, de forma manual mas seguindo os padrões propostos na abordagem WS-Agreement.

Os seguintes níveis de QoS foram negociados entre as organizações:

- A organização provedora Passagens LTDA juntamente com a organização cliente determinou que seu serviço Web deve estar disponível 24 X 7 (24 horas por dia, 7 dias por semana), ter o tempo de resposta de cada solicitação seja de no máximo 3 segundos e que o número máximo de requisições por minuto seja de 10 requisições.

- A organização provedora Hotéis LTDA juntamente com a organização cliente determinou que seu serviço Web deve estar disponível também 24 X 7, ter tempo de resposta de no máximo 4 segundos e que deve ter o máximo de 5 reaquisições por minuto.

- A organização provedora LocaAutos LTDA juntamente com a organização cliente determinou que seu serviço Web deve estar disponível também 24 X 7, ter tempo de resposta de no máximo 10 segundos e não há restrições para requisições por minutos, isto porque este serviço Web não serão tão requisitados quanto os dois anteriores devido ao fato de a locação de automóveis não ser tão comum quanto comprar passagens ou reservar hotéis.

Com essas definições pode-se escrever os contratos eletrônicos especificando os níveis de QoS desejados. A Listagem 14 apresenta um trecho de definição de contrato eletrônico referente ao acordo firmado entre a organização cliente e a organização provedora de serviços de passagens. Neste contrato, foram especificadas as três propriedades de serviço definidas em conjunto pelas organizações cliente e provedora. A propriedade de serviço que especifica a disponibilidade de 24 x 7 foi mapeada para a variável `Disponibilidade` (linha 6). A propriedade para o tempo médio de resposta foi mapeada para a variável `Resposta` (linha 9). A propriedade para o número de solicitações foi mapeada para a variável `Solicitacoes` (linha 12).

```
01 <wsag:SDT Name=”TermoServicoCredito” ServiceName=”PassagensService”>
02 <wsdl:definitions ...> ...
03 </wsdl:definitions>
04 </wsag:ServiceDescriptionTerm> ...
05 <wsag:SP Name=”PropriedadesServicoPassagens” ServiceName=”PassagensService”>
06    <wsag:Variable Name=”Disponibilidade” Metric=”eDisponivel”>
07       <wsag:Location>//wsag:SDT/[@name=”TermoServicoPassagens”@portType=”
passagemPT”@operation=”comprarPassagem”]</wsag:L>
08    </wsag:Variable>
09    <wsag:Variable Name=”Resposta” Metric=”tempoResposta”>
10       <wsag:Location>//wsag:SDT/[@name=”TermoServicoPassagens”@portType=”
passagemPT”@operation=”comprarPassagem”]</wsag:L>
```

```
11    </wsag:Variable>
12    <wsag:Variable Name=”Solicitacoes” Metric=”solicitacoesPorMinuto”>
13       <wsag:Location>//wsag:SDT/[@name=”TermoServicoPassagens”@portType=”
passagemPT”@operation=”comprarPassagem”]</wsag:L>
14    </wsag:Variable>
```

Listagem 14: Trecho do contrato eletrônico para serviços de passagens

As propriedades foram definidas e os valores de medições são definidos na seção de propriedades de serviço. A Listagem 15 apresenta a seção de termos de garantia que é a seção do contrato eletrônico que atribui valores às propriedades de serviço. A propriedade `Dispobinilidade` tem o valor `24x7` (linha 7) indicando que o serviço deve ser disponível 24 horas por dia 7 e dias por semana. As propriedades `Resposta` e `Solicitacoes` também possuem seus termos de garantia preenchidos de forma semelhante ao apresentado na Listagem 15.

```
01 <wsag:GuaranteeTerm wsag:name=”PassagensDisponivel”>
02    <wsag:ServiceScope>
03       <wsag:ServiceName>PassagensService</wsag:ServiceName>
04    </wsag:ServiceScope>
05    <wsag:ServiceLevelObjective>
06       <exp:Less>
07          <exp:Variable>Disponibilidade</exp:Variable>
08          <exp:Value>24x7</exp:Value>
09       </exp:Less>
10    </wsag:ServiceLevelObjective>...
11 </wsag:GuaranteeTerm>
```

Listagem 15: Termos de garantia para a propriedade Disponibilidade

Cada serviço Web contratado de organizações terceiras possui um contrato eletrônico como o descrito acima. Após escrever os contratos eletrônicos é possível gerar os aspectos de monitoramento. Cada propriedade de serviço descrita no contrato eletrônico gera um aspecto de monitoramento, portanto o contrato de uso do serviço de passagens gera 3 aspectos. Alguns trechos do aspecto gerado pela propriedade `Disponibilidade` são apresentados na Listagem 16. Este aspecto contém um `pointcut` (linha 7) que indica que o momento de monitoramento é ao executar a operação `comprarPassagem` do serviço `PassagensService`. O tipo do aspecto é `after` (linha 10) pois o aspecto monitor necessita da resposta do serviço monitorado para identificar se o mesmo estava disponível no momento da requisição.

```
01 <aspect name="ContratoPassagens-Disponibilidade">
```

```
02    <partners>
03       <partner name="monitorPartner" serviceLinkType="monitorSLT"/>
04    </partners> ...
07    <pointcut name="PassagensInvoke" contextCollection="true">
08       //process//invoke[@serviceName="PassagensService"@portType="passagensPT"
@operation="comprarPassagem"]
09    </pointcut>
10    <advice type="after">
11    <bpws:sequence>
12    <bpws:assign>
13       <bpws:copy>
14          <bpws:from expression="ContratoPassagens-Disponibilidade"/>
15          <bpws:to variable="monitorRequest" part="ServicePropertyId"/>
16       </bpws:copy>
17       <bpws:copy>
18          <bpws:from //process//invoke[@serviceStatus]>
19          <bpws:to variable="monitorRequest" part="ServicePropertyData"/>
20       </bpws:copy>
21    </bpws:assign>
22       <bpws:invoke name="invoke" partner="monitorPartner"
portType="ServicoMonitor" operation="monitora" inputVariable="monitorRequest"
outputVariable="monitorResponse"/>
23    </bpws:sequence>
24    </advice>
25 </aspect>
```

Listagem 16: Aspecto do serviço de passagens para disponibilidade

Os outros aspectos também devem ser gerados e seguem a mesma metodologia acima. Os artefatos gerados até agora foram o processo de negócio AO4BPEL, os contratos eletrônicos e os aspectos. Com esses elementos é possível implementar o exemplo de aplicação. A forma como ocorre a implementação é apresentada na próxima seção.

4.4. Implementando e executando

A implementação e a execução da abordagem necessita de apoio computacional para a linguagem AO4BPEL. A organização cliente deve publicar seu processo de negócio e seus aspectos num servidor AO4BPEL. Como visto na seção 3.6 não há apoio computacional para esta linguagem. A solução adotada foi inserir os aspectos no processo de negócio gerado de forma não automatizada e executar o processo como se fosse um processo WS-BPEL.

O processo AO4BPEL utiliza as mesmas operações de um processo WS-BPEL, portanto o processo AO4BPEL não precisa de alterações para ser utilizado como processo WS-BPEL. A única alteração necessária é inserir os aspectos no processo de negócio. A

Listagem 17 apresenta um trecho do processo de negócio no qual foi inserido um aspecto de monitoramento. A operação `invoke` da operação `comprarPassagem` (linha 1) é uma requisição ao serviço Web `PassagemCl` que deve ser monitorado. A operação `sequence` (linhas 2 a 14) é o `advice` do aspecto `ContratoPassagens-Disponibilidade` e foi inserido no processo WS-BPEL. A variável `monitorRequest` possui duas partes. A primeira parte recebe como conteúdo o nome do aspecto (linha 05) e a segunda parte recebe como conteúdo a variável `ComprarPassagemResponse` (linha 9) que é o retorno do serviço de passagens. A requisição ao serviço monitor envia a variável `monitorRequest` para o serviço monitor e recebe o retorno na variável `monitorResponse`.

As outras requisições a serviços monitorados também seguem a mesma regra apresentada. A requisição aos serviços monitores pode ocorrer antes, depois ou antes e depois de acordo com o tipo de aspecto.

```
...
01    <bpel:invoke inputVariable="comprarPassagemRequest" name="Passagem"
operation="comprarPassagem" outputVariable="comprarPassagemResponse"
partnerLink="PassagemPLT" portType="ns1:PassagemCl">
02    <bpel:sequence>
03    <bpel:assign>
04       <bpel:copy>
05          <bpel:from expression="ContratoPassagens-Disponibilidade"/>
06          <bpel:to variable="monitorRequest" part="ServicePropertyId"/>
07       </bpel:copy>
08       <bpel:copy>
09          <bpel:from //process//invoke[@serviceStatus]>
10          <bpel:to variable="monitorRequest" part="ServicePropertyData"/>
11       </bpel:copy>
12    </bpel:assign>
13       <bpel:invoke name="invoke" partner="monitorPartner"
portType="ServicoMonitor" operation="monitora" inputVariable="monitorRequest"
outputVariable="monitorResponse"/>
14    </bpel:sequence>
```

Listagem 17: Processo WS-BPEL com monitoramento

Após as adaptações com relação aos aspectos o processo de negócio pode ser publicado em servidor WS-BPEL. O servidor BPEL utilizado é o ActiveBPEL Engine que pode ser visto na Figura 17. A Figura 17 apresenta a tela com os processos publicados no qual aparece o processo de viagem que recebeu o nome de `ViagemProc`. Isto significa que o processo está pronto pra ser utilizado.

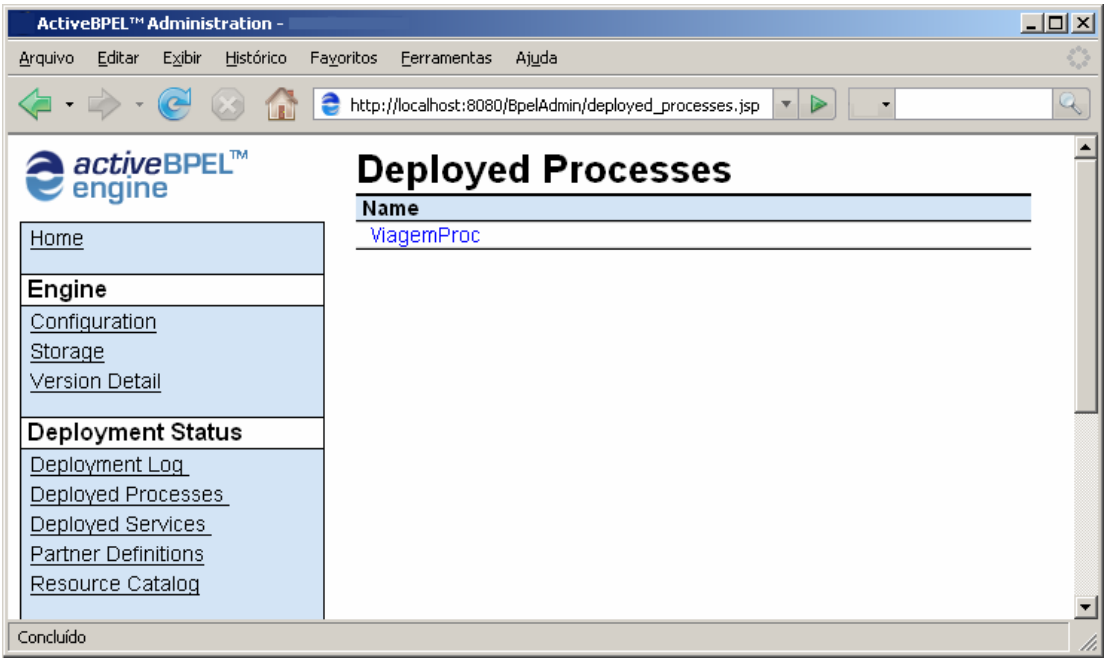

Figura 17: Servidor WS-BPEL com o processo de negócio ViagemProc publicado

O processo de negócio agora pode ser utilizado por uma aplicação de vendas de passagens da organização cliente. Outra forma de utilizar o processo é por meio do SOAP Client mostrado na seção 2.4.1. A Figura 18 apresenta a utilização do SOAP Client para testar o processo de negócio de viagem. Deve ser informada a URL do processo de negócio e um arquivo XML no formato dos dados de entrada do processo de negócio. A saída é gerada pelos serviços Web de passagens, hotéis e automóveis, juntamente com resposta dos serviços monitores. Os serviços monitores geralmente não necessitam retornar dados para o processo de negócio, neste caso apenas uma informação para fins de teste foi retornada pelos serviços monitores.

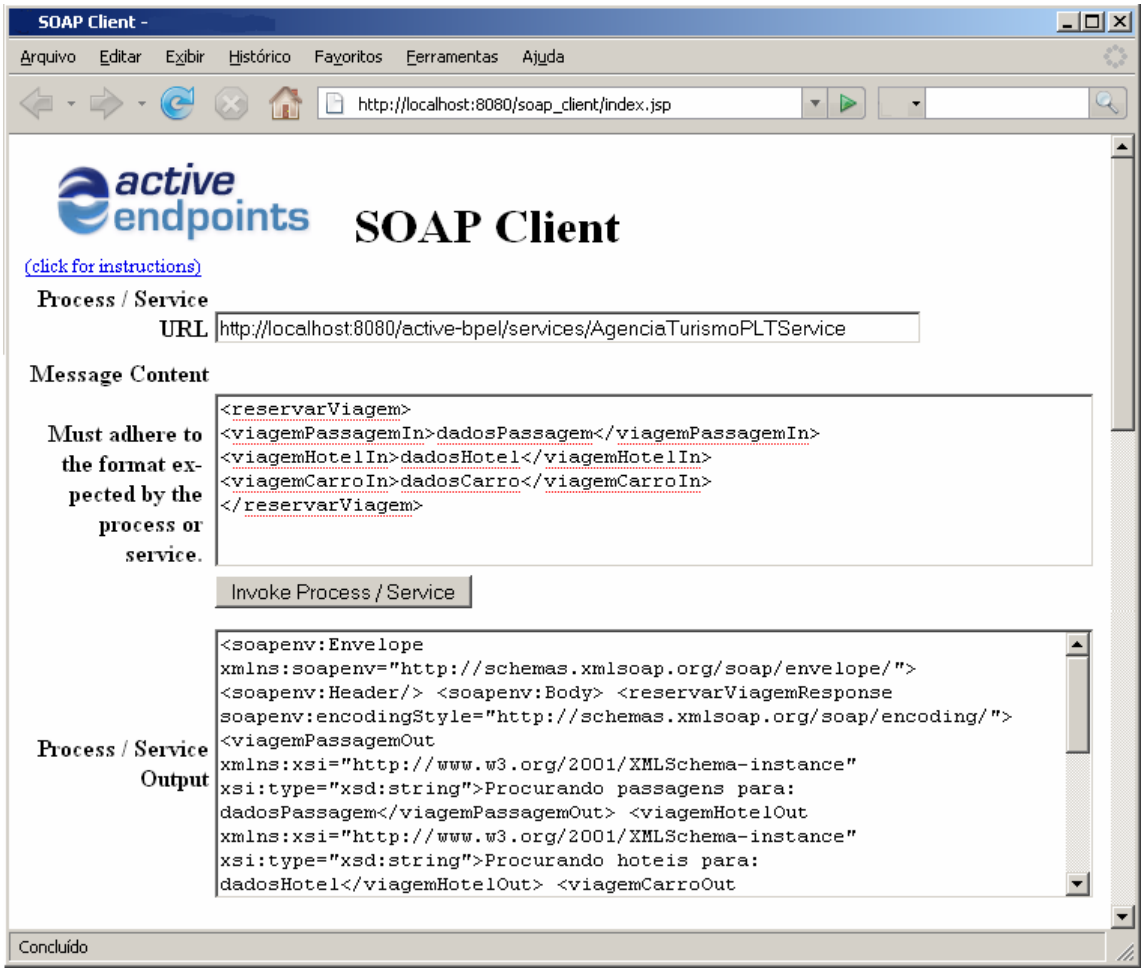

Figura 18: SOAP Client com o processo de negócio de viagem

A implementação e utilização de um processo de negócio seguindo a abordagem AspectMonitor foi demonstrada nesta seção. Os serviços Web terceirizados foram utilizados, o processo de negócio da organização cliente foi executado com sucesso e os dados de monitoramento do contrato eletrônico foram enviados para os serviços Web monitores. A forma com que os serviços Web monitores tratam os dados recebidos não está determinado pela abordagem AspectMonitor. É possível por exemplo que a organização monitora envie relatório de uso dos serviços monitorados periodicamente para as organizações cliente e provedora para que o cumprimento dos contratos eletrônicos possa ser analisado por ambas. A próxima seção apresenta as considerações finais sobre a abordagem AspectMonitor.

4.5. Considerações finais

A abordagem AspectMonitor foi utilizada por meio da implementação de uma aplicação de vendas de pacotes turísticos. A organização cliente utiliza serviços Web terceirizados e define contratos eletrônicos para especificar as obrigações e garantias das organizações cliente e provedora. A organização cliente tem a necessidade de utilizar 3 serviços Web terceirizados

para realizar seu processo de negócio. Desse modo a organização cliente contrata serviços Web de 3 organizações provedoras diferentes cada uma especialista em sua área de aplicação. Os três serviços Web contratados foram utilizados para escrever o processo de negócio de vendas de viagens. O uso de três organizações provedoras gerou a necessidade de escrever três contratos eletrônicos com várias propriedades de serviço e seus respectivos termos de garantia associados. Os termos de garantia são os níveis de QoS que devem ser disponibilizados e posteriormente monitorados. A realização do monitoramento é realizada por aspectos de monitoramento que são gerados seguindo as regras de mapeamento definidas na seção 3.5.1. Cada termo de garantia gera um aspecto de monitoramento e por isso cada contrato eletrônico pode gerar um (1) ou mais aspectos.

A execução do processo de negócio AO4BPEL e dos aspectos necessita de apoio computacional específico para esta linguagem. Pelo fato de não existir este apoio computacional, a atuação dos aspectos foi simulada inserindo-se os aspectos de monitoramento em um processo de negócio escrito em WS-BPEL de forma manual e publicando-se este processo em um servidor WS-BPEL. Isso é possível porque a estrutura e os elementos da linguagem WS-BPEL e AO4BPEL são semelhantes diferenciando-se apenas o fato de AO4BPEL permitir a definição de aspectos.

O processo de negócio foi escrito com o apoio da ferramenta ActiveBPEL Designer que utiliza WS-BPEL. O servidor WS-BPEL utilizado foi o ActiveBPEL Engine. Neste servidor, foi publicado o processo de negócio já com os aspectos de monitoramento devidamente inseridos em seus pontos corretos de chamada, que poderia ser antes, depois ou antes e depois das chamadas aos serviços a serem monitorados. Para a execução do processo de negócio foi utilizada a ferramenta SOAP Client que recebe dados de entrada executa o processo de negócio e retorna os dados gerados.

Assim, a abordagem AspectMonitor permitiu à organização cliente desenvolver seu processo de negócio sem ter necessidade de incluir instrumentação no processo de negócio relativa ao monitoramento dos contratos eletrônicos que regulamentam o uso dos serviços Web utilizados no processo. De acordo com o paradigma orientado a aspectos isto significa dizer que o interesse principal que é o processo de negócio não necessitou ser entrecortado pelo interesse secundário que é o monitoramento de contratos eletrônicos. Outros modelos de monitoramento propostos como WSLA e WS-Agreement não realizam esta separação e há a necessidade de instrumentar o processo principal com funções de monitoramento.

A abordagem AspectMonitor também simplifica o apoio computacional necessário para realizar o monitoramento se comparada aos modelos propostos em WSLA e WS-Agreement. Considerando que os contratos eletrônicos da abordagem AspectMonitor são pré-definidos e a abordagem necessita basicamente de tecnologias já existentes como serviços Web e processos de negócio AO4BPEL. As abordagens WSLA e WS-Agreement utilizam tecnologias de serviços Web, mas também necessitam que sejam implementadas diversas funções como gerenciamento de acordos, criação de templates e ofertas e negociação de propriedades de serviço para estabelecimento do contrato e funções de verificação de estados dos acordos e dos serviços Web para monitoramento do contrato.

A abordagem AspectMonitor permite uma melhor separação de responsabilidades. A organização cliente pode focar suas atividades no processo de negócio de vendas de pacotes turísticos. As organizações provedoras podem se concentrar em seus negócios de passagens, hotéis e automóveis. As organizações monitoras podem se especializar em monitoramento de níveis de QoS de contratos eletrônicos criando serviços especializados em análises de necessidades de clientes e provedores. Nas abordagens WSLA e WS-Agreement o cliente e o provedor devem também realizar atividades relacionadas ao monitoramento, fugindo do foco principal de seu processo de negócio.

Outro ponto positivo da abordagem AspectMonitor é que a mesma apresenta uma proposta de monitoramento de contratos eletrônicos aplicados no contexto de processos de negócio. As abordagens WSLA e WS-Agreement apresentam modelos de monitoramento para o contexto de cliente e provedor negociando o uso de um serviço Web apenas e não apresentam na proposta a aplicação em processos de negócio.

Uma comparação entre tempo de execução dos processos de negócio com os diferentes tipos de monitoramento não foi realizada devido à falta de apoio computacional para as abordagens WSLA, WS-Agreement e AO4BPEL.

Um ponto negativo da abordagem AspectMonitor é que pelo fato de não interferir no processo de negócio, os dados enviados aos serviços monitores devem ser retirados de forma transparente ao processo de negócio. Por esse motivo os dados enviados aos serviços monitores são na maioria das vezes a resposta do serviço monitorado. Esta resposta pode conter informações comuns caso o serviço tenha sido executado normalmente ou conter informações de erro caso o serviço não tenha sido executada de acordo com o previsto. O ideal seria o envio de informações mais relacionadas à propriedade de serviço que está sendo

medido mas isto criaria a necessidade de interferir no processo de negócio para inserir estas informações.

Futuramente serviços monitores podem ser desenvolvidos com operações mais especializadas no tratamento de níveis de QoS e dos parceiros envolvidos nos contratos eletrônicos. O objetivo desses serviços monitores seria criar um banco de dados com informações de provedores e clientes de serviços e de acordo com os contratos eletrônicos e os dados das propriedades de serviços monitorados ser capaz de fazer análises e sugestões para as organizações envolvidas. Essas análises e sugestões podem ser no intuito de alterar o contrato eletrônico ou para que as organizações busquem novos parceiros ou para que alterem as características dos serviços Web contratados. Estas análises não poderiam ser realizadas se não fosse utilizada uma abordagem que separa o interesse de monitoramento como é a AspectMonitor.

5. Conclusão

Com a finalidade de facilitar a utilização de processos de negócios interorganizacionais esta dissertação apresenta uma abordagem baseada em aspectos para monitoramento de contratos eletrônicos. As organizações buscam cada vez mais agregar valor aos seus processos de negócio. Isso motiva a integração de seus processos com os processos de outras organizações. A computação orientada a serviços e a tecnologia Web oferecem a infra-estrutura necessária para permitir essa integração ao fazer uso de serviços Web para a criação de aplicações. Os serviços Web são uma forma padronizada de se utilizar funcionalidades e recursos de aplicações de outras organizações.

A utilização de contratos eletrônicos é de grande importância para a realização de acordos, pois define os requisitos e as capacidades dos serviços Web utilizados nos processos de negócio envolvidos. O monitoramento de contratos é importante para garantir o cumprimento dos acordos. A abordagem apresentada nesta dissertação busca facilitar a inserção de elementos de monitoramento em contratos eletrônicos que regulamentam a integração de processos de negócio, de forma a separar o interesse principal do processo de negócio, do interesse de monitoramento. Assim, a abordagem contribui para melhorar a padronização e a eficiência do estabelecimento de contratos eletrônicos e o monitoramento de processos de negócio.

Vários conceitos e tecnologias relacionadas à computação orientada a serviços e a tecnologia Web foram utilizadas para a concepção da abordagem proposta nesta dissertação como: serviços Web, composição de serviços, WSDL, WS-BPEL, AO4BPEL, QoS e WS-Agreement. Esses conceitos foram somados aos conceitos do paradigma de aspectos na perspectiva de atingir uma melhor separação de interesses no contexto de processo de negócios e contratos eletrônicos, uma vez que este paradigma tem se mostrado efetivo em várias áreas de desenvolvimento de software.

A abordagem proposta consiste basicamente de uma arquitetura, de responsabilidades e atividades das organizações envolvidas e de um conjunto de regras de mapeamento de contratos eletrônicos em aspectos. A definição de contratos eletrônicos especifica atributos e níveis de QoS que determinado serviço oferece. O monitoramento desses atributos e níveis de QoS é de responsabilidade de serviços Web monitores que são relacionados aos contratos no momento de mapeamento dos contratos à aspectos. Esses aspectos são escritos na linguagem

AO4BPEL. Durante a execução do processo de negócio, os aspectos atuam nos pontos indicados nos *pointcuts* e realizam as atividades de monitoramento.

Assim, a principal contribuição da abordagem proposta é permitir um monitoramento de forma modularizada, não entrecortante ao processo de negócio. Com isso se reduz a complexidade do monitoramento, pois o próprio monitoramento pode ser realizado também por um serviço Web monitor. As ferramentas utilizadas para apoiar a abordagem foram: editor de processos WS-BPEL ActiveBpel Editor, um editor de XML para escrever os contratos eletrônicos, o ambiente de desenvolvimento Java Eclipse para criar os serviços Web, um servidor BPEL ActiveBpel Engine para publicar os serviços Web e o processo de negócio WS-BPEL.

A abordagem proposta foi avaliada por meio de um exemplo de aplicação em que se explorou as vantagens e dificuldades de implantação desta. O exemplo consistiu de um processo WS-BPEL criado no editor ActiveBpel Editor, vários serviços Web utilizados na composição do processo WS-BPEL que foram publicados no servidor ActiveBpel Engine. Neste servidor também foi publicado o processo WS-BPEL. Os contratos eletrônicos WS-Agreement foram escritos em editor de XML. Os aspectos gerados a partir dos contratos eletrônicos foram inseridos de forma manual no processo de negócio WS-BPEL. A execução do processo de negócio simulada por meio do SOAP Client disponibilizado pelo ActiveBpel.

Uma das dificuldades enfrentadas no desenvolvimento do exemplo de aplicação foi a não existência de uma máquina AO4BPEL. Para contornar foi utilizado um ambiente WS-BPEL. Neste ambiente, foi criado um processo de negócio WS-BPEL. Os aspectos de monitoramento foram gerados, e pela ausência de máquina AO4BPEL os aspectos foram inseridos no processo WS-BPEL de forma manual, assim não houve possibilidade de combinar os aspectos ao processo WS-BPEL por meio de um programa.

Existem trabalhos relacionados à padronização de contratos eletrônicos como WS-Policy (BAJAJ et al, 2006) e WS-Agreement (ANDRIEUX et al, 2004). Também podemos encontrar CREMONA (LUDWIG et al, 2004) que oferece recursos para monitoramento de contratos eletrônicos. As propostas AO4BPEL (CHARFI e MEZINI, 2004) e Padus (BRAEM et al, 2006) são propostas de linguagem de especificação de processos de negócios orientadas a aspectos. Porém, não foram encontrados trabalhos que proponham a aplicação de aspectos para separação de interesses em monitoramento, como propostos nesta dissertação.

5.1 Trabalhos futuros

O presente trabalho de mestrado concentrou-se na concepção da abordagem, embora para o seu desenvolvimento, fosse necessária a exploração de várias ferramentas de apoio a processos de negócios e contratos eletrônicos, assim existem trabalhos futuros necessários para completar a abordagem e para permitir a sua automatização, como apresentados a seguir:

- desenvolvimento de ferramentas de apoio computacional, principalmente para a ferramenta para geração dos aspectos a partir de contratos eletrônicos. Também é necessário apoio computacional para implementar processos em AO4BPEL, ou equivalente;
- avaliação empírica da abordagem proposta com base em estudos experimentais;
- comparação entre os tempos de execução das principais propostas relacionada ao monitoramento de contratos eletrônico;
- definir serviços monitores especializados em análises de níveis de QoS para sugerir melhoras nos contratos firmados ou nas relações entre as organizações cliente e provedora;
- projeto de operadores adicionais de composição para utilização nos aspectos AO4BPEL de monitoramento de acordo com as especificidades de processos de negócio;
- desenvolver um estudo de caso para avaliar a abordagem proposta baseado nos princípios de engenharia de software experimental (KITCHENHAM, 1996) (BASILI et al, 1986).

6. Referências Bibliográficas

ACTIVEBPEL, LLC. The Open Source BPEL Engine. Disponível em http://www.activebpel .org. Acessado em 12/10/2006.

ALONSO, G.; CASATI, F.; KUNO, H. e MACHIRAJU, V. Web Services: Concepts, Architectures and Applications. Springer-Verlag, Berlim, 2004.

ANDREWS, T.; CURBERA, F.; DHOLAKIA, H.; GOLAND, Y.; KLEIN, J.; LEYMANN, F.; LIU, K.; ROLLER, D.; SMITH, D.; THATTE, S.; TRICKOVIC, I. e WEERAWARANA, S. Business Process Execution Language for Web Services (Version 1.1). Maio, 2003.

ANDRIEUX, A.; CZAJKOWSKI, K.; DAN, A.; KEAHEY, K.; LUDWIG, H.; PRUYNE, J.; ROFRANO, J.; TUECKE, S. e XU, M. Web Service Agreement Specification (WS-Agreement). Global Grid Forum, Maio, 2004.

ARKIN, A.; ASKARY, S.; BLOCH, B.; CURBERA, F.; GOLAND, Y.; KARTHA, N.; LIU, C.; THATTE, S.; YENDLURI, P. e YIU, A.. Web Services Business Process Execution Language Version 2.0. Working Draft. WS-BPEL TC OASIS, Maio, 2005.

BAJAJ, S.; BOX, D.; CHAPPELL, D.; CURBERA, F.; DANIELS, G.; HALLAM-BAKER, P. e HONDO, M. Web Services Policy 1.2 Framework (WS-Policy). W3C Member Submission, Abril, 2006.

BALDAN, R.; VALLE, R.; PEREIRA, H.; HILST, S.; ABREU, M. e SOBRAL, V. Gerenciamento de Processos de Negócios BPM – Business Process Management, Editora Érica, São Paulo, 2008, ISBN: 978-85-3650-1758.

BASILI, V. R.; SELBY, R. W. e HUTCHENS, D. H. Experimentation in software engineering. IEEE Transactions on Software Engineering, Piscataway, v. 12, n. 7, p. 733-743, 1986.

BELLWOOD, T.; CLEMENT, L.; EHNEBUSKE, D.; HATELY, A.; HONDO, M.; HUSBAND, Y.; JANUSZEWSKI, K.; LEE, S.; MCKEE, B.; MUNTER, J. e RIEGEN, C. UDDI version 3.0 (Universal Description, Discovery, and Integration), Julho, 2002.

BEN-NATEN, R. CORBA: A Guide to the Common Object Request Broker Architecture. McGraw-Hill, Nova Iorque, 1995.

BOX, D.; EHNEBUSKE, D.; KAKIVAYA, G.; LAYMAN, A., MENDELSOHN, N., NIELSEN, H.; THATTE, S. e WINER, D. Simple Object Access Protocol (SOAP) 1.1. http://www.w3.org/TR/SOAP/, Maio, 2000.

BOX, D. e CURBERA, F. *Web Services Addressing (WS-Addressing)*, 2004.

BRAEM, M.; VERLAENEN, K.; JONCHEERE, N.; VANDERPERREN, W.; VAN DER STRAETEN, R.; TRUYEN, E.; JOOSEN, W. e JONCKERS, V. *Isolating process-level concerns using Padus*. In Proceedings of the 4th International Conference on Business Process Management (BPM 2006), Vienna, Austria, Springer-Verlag. LNCS 4102, Páginas 113-128, Setembro, 2006.

CASATI, F. A Conversation on Web Services: what's new, what's true, what's hot. And what's not. Proceedings of the ECAI workshop on Knowledge Transformation and the Semantic Web. Lyon, França, Julho, 2002.

CHARFI, A. e MEZINI, M. Aspect-oriented Web service composition with AO4BPEL. In Proceedings of the European Conference on Web Services (ECOWS), 2004.

CHARFI, A. e MEZINI, M. Middleware Support for BPEL Workflows in the AO4BPEL Engine. 4th International Conference on Business Process Management (BPM 2006) , Vienna, Austria, Springer-Verlag. LNCS 4102, Setembro, 2006.

CHAVEZ, C.; GARCIA, A. e KULESZA, U. Programação Orientada a Aspectos. Anais do XVII Simpósio Brasileiro de Engenharia de Software, Manaus: Universidade Federal do Amazonas, 2003.

CHRISTENSEN, E; CURBERA, F.; MEREDITH, G. e WEERAWARANA, S. Web Services Description Language (WSDL) 1.1, W3C, 2001.

CLARK, J. e DEROSE, S. XML Path Language (XPATH) Version 1.0. Disponível em http://www.w3.org/TR/xpath, Novembro, 1999.

CURBERA, F.; DUFTLER, M.; KHALAF, R.; MUKHI, N.; NAGY, W. e WEERAWARANA, S. The BPWS4J Java Runtim, Disponível em http://www.alphaworks. ibm.com/tech/bpws4j, 2002.

DAN, A.; DAVIS, D.; KEARNEY, R.; KELLER, A.; KING, R.; KUEBLER, D.; LUDWIG, H.; POLAN, M.; SPREITZER, M.; e YOUSSEF, A. Web Services on demand: WSLA-driven Automated Management, IBM Systems Journal, Special Issue on Utility Computing, Volume 43, Number 1, páginas 136-158, IBM Corporation, Março, 2004.

DAYAL, U.; HSU, M. e LADIN, R. Business Process Coordination: State of the Art, Trends, and Open Issues. VLDB 2001, Proceedings of 27th International Conference on Very Large Data Bases, Roma, Itália, ISBN 1-55860-804-4, Páginas 3-13, 2001.

FANTINATO, M., TOLEDO, M.B.F., e GIMENES, I.M.S., Contratos Eletrônicos para Sistemas de Gerenciamento de Processos de Negócio, Relatório Técnico IC-05-12, IC/UNICAMP, Brasil, 2005.

FANTINATO, M. Uma Abordagem Baseada em Características para o Estabelecimentos de Contratos Eletrônicos para Serviços Web, Tese de Doutorado, Instituto de Computação, UNICAMP, 2007.

GRIFFEL, F.; BOGER, M.; WEINREICH, H.; LAMERSDORF, W. e MERZ, M. Electronic Contracting with COSMOS - How to Establish, Negotiate and Execute Electronic Contracts on the Internet. Enterprise Distributed Object Computing Workshop, 1998, EDOC'98, Proceedings, Second International, Novembro, 1998.

HORSTMANN, M. e KIRTLAND, M. DCOM architecture, Microsoft Corporation, Julho, 1997.

HSU, M. From Marketplaces to Web Services. WISE Proceedings of the 3rd International Conference on Web Information Systems Engineering, 2002.

KARAGIANNIS, D. BPMS: Business process management systems. ACM SIGOIS Bulletin, 1995.

KELLER, A. e LUDWIG, H. The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services. Journal of Network and Systems Management, Special Issue on E-Business Management, Volume 11, Number 1, Plenum Publishing Corporation, Março, 2003.

KERSTEN, M. AspectJ: The Language and Development Tools. Workshop on Tools for Aspect-Oriented Software Development (OOPSLA 2002), Novembro, 2002.

KICZALES, G.; LAMPING, G.; MENDHEKAR, J.; MAEDA, A.; LOPES, C.; LOINGTIER, C. e IRWING, J. Aspect-Oriented Programming. In Proceedings of the 11st European Conference Object-Oriented Programming (ECOOP'97), volume 1241 of LNCS, Páginas 220–242, Springer Verlag, 1997.

KICZALES, G.; HILADALE, E.; HUGUNIN, J.; KERSTEN, M.; PALM, J. e GRISWOLD, W. Getting Started with AspectJ. Communications of the ACM, 44 (10): Páginas 59–65, Outubro, 2001.

KITCHENHAM, B. DESMET: a method for evaluating software engineering methods and tools.Technical Report TR96-09, Keele, United Kingdom, 1996. 49 p.

LEYMANN, F. Web Services Flow Language (WSFL) Version 1.0. Technical Report, International Business Machines Corporation (IBM), Maio, 2001.

LEYMANN, F.; ROLLER, D. e SCHMIDT, M. Web services and business process management. IBM Systems Journal, vol. 41, no.2, 198-211, 2002.

LUDWIG, H.; DAN, A. e KEARNEY, B. Cremona: An Architecture and Library for Creation and Monitoring of WS-Agreements. In Proceedings of the 2nd International Conference on Service Oriented Computing (ICSOC'04), Nova Iorque, Páginas 65-74, 2004.

MANI, A., e NAGARAJAN, A. Understanding Quality of Service for Web Services, 2002.

MEDEIROS, A. K.; GIMENES, I. M. S. e TOLEDO, M. B. Sistemas de Gestão de Processos de Negócios: Desafios e Oportunidades. Post-proceedings Workshop Empresarial ENEGEP, Foz de Iguaçu, Paraná, 2007.

ORACLE. Oracle BPEL Process Manager Developer's Guide 10g Release 2 (10.1.2), Disponível em http://download-east.oracle.com/docs/cd/B14099_19/integrate.1012/b14448/, Acessado em 2006.

ORACLE. BPEL Process Manager. http://www.oracle.com/technology/products/ias/bpel/, Acessado em 2007.

PAPAZOGLOU M. e GEORGAKOPOULOS, D. Service-oriented computing, Communications of the ACM: Service-Oriented Computing 46, Páginas 25–28, 2003.

PMAC. IBM Corporation: PMAC Expression Language Users Guide. Alphaworks PMAC distribution, www.alphaworks.ibm.com, 2005.

SADTLER, C. e KOVARI, P. WebSphere Business Integration Server Foundation Architecture and Overview, RedBooks, 2004.

SMITH, H. e FINGAR, P. Business Process Fusion Is Inevitable. BPTRENDS, Páginas 1-15, Março, 2004.

THATTE, S. XLANG: Web Services for Business Process Design. Microsoft Corporation, Disponível em http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm, 2001.

WALDRICH, O. From WS-Agreement to SLA negotiation. Global Grid Forum, Março, 2006.

WESKE, M., Business Process Management - Concepts, Languages, Architectures, Spinger-Verlag, 2007.

WFMC – Workflow Management Coalition. Workflow Reference Model. Bruxelas, Janeiro, 1995.

Apêndice A - Arquivos-fonte utilizados/gerados pelo exemplo de aplicação

Descritor WSDL do processo de negócio

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://localhost:8080/wiagem"
xmlns:plnk="http://docs.oasis-open.org/wsbpel/2.0/plnktype"
xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="http://localhost:8080/wiagem"
xmlns:intf="http://localhost:8080/wiagem"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<!--WSDL created by Apache Axis version: 1.4
Built on Dec 20, 2006 (01:34:32 EST)-->

   <wsdl:message name="reservarViagemResponse">
      <wsdl:part name="viagemPassagemOut" type="xsd:string"/>
      <wsdl:part name="viagemHotelOut" type="xsd:string"/>
      <wsdl:part name="viagemCarroOut" type="xsd:string"/>
      <wsdl:part name="viagemLogMonitor" type="xsd:string"/>
   </wsdl:message>

   <wsdl:message name="reservarViagemRequest">
      <wsdl:part name="viagemPassagemIn" type="xsd:string"/>
      <wsdl:part name="viagemHotelIn" type="xsd:string"/>
      <wsdl:part name="viagemCarroIn" type="xsd:string"/>
   </wsdl:message>

   <wsdl:portType name="AgenciaTurismoPT">
      <wsdl:operation name="reservarViagem">
         <wsdl:input message="impl:reservarViagemRequest"
name="reservarViagemRequest"/>
         <wsdl:output message="impl:reservarViagemResponse"
name="reservarViagemResponse"/>
      </wsdl:operation>
   </wsdl:portType>

<plnk:partnerLinkType name="AgenciaTurismoPLT">
   <plnk:role name="AgenciaTurismoRole" portType="impl:AgenciaTurismoPT" />
</plnk:partnerLinkType>

</wsdl:definitions>
```

Descritor WSDL do serviço Web de locação de automóveis

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://localhost:8080/wiagem"
xmlns:plnk="http://docs.oasis-open.org/wsbpel/2.0/plnktype"
xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="http://localhost:8080/wiagem"
xmlns:intf="http://localhost:8080/wiagem"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
```

```
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<!--WSDL created by Apache Axis version: 1.4
Built on Dec 20, 2006 (01:34:32 EST)-->

   <wsdl:message name="reservaCarroRequest">
      <wsdl:part name="dadosCarro" type="xsd:string"/>
   </wsdl:message>

   <wsdl:message name="reservaCarroResponse">
      <wsdl:part name="carroOut" type="xsd:string"/>
   </wsdl:message>

   <wsdl:portType name="CarroCl">
      <wsdl:operation name="reservaCarro" parameterOrder="dadosCarro">
         <wsdl:input message="impl:reservaCarroRequest"
name="reservaCarroRequest"/>
         <wsdl:output message="impl:reservaCarroResponse"
name="reservaCarroResponse"/>
      </wsdl:operation>
   </wsdl:portType>

   <wsdl:binding name="CarroWsSoapBinding" type="impl:CarroCl">
      <wsdlsoap:binding style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="reservaCarro">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="reservaCarroRequest">
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://localhost:8080/wiagem" use="encoded"/>
         </wsdl:input>
         <wsdl:output name="reservaCarroResponse">
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://localhost:8080/wiagem/wsdl/" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>

   <wsdl:service name="CarroClService">
      <wsdl:port binding="impl:CarroWsSoapBinding" name="CarroWs">
         <wsdlsoap:address location="http://localhost:8080/active-
bpel/services/CarroWs"/>
      </wsdl:port>
   </wsdl:service>
<plnk:partnerLinkType name="CarroPLT">
   <plnk:role name="CarroRole" portType="impl:CarroCl" />
</plnk:partnerLinkType>

</wsdl:definitions>
```

Descritor WSDL do serviço Web de reserva de hotéis

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://localhost:8080/wiagem"
xmlns:plnk="http://docs.oasis-open.org/wsbpel/2.0/plnktype"
xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="http://localhost:8080/wiagem"
xmlns:intf="http://localhost:8080/wiagem"
```

```
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<!--WSDL created by Apache Axis version: 1.4
Built on Dec 20, 2006 (01:34:32 EST)-->

   <wsdl:message name="reservaHotelResponse">
      <wsdl:part name="hotelOut" type="xsd:string"/>
   </wsdl:message>

   <wsdl:message name="reservaHotelRequest">
      <wsdl:part name="dadosHotel" type="xsd:string"/>
   </wsdl:message>

   <wsdl:portType name="HotelCl">
      <wsdl:operation name="reservaHotel" parameterOrder="dadosHotel">
         <wsdl:input message="impl:reservaHotelRequest"
name="reservaHotelRequest"/>
         <wsdl:output message="impl:reservaHotelResponse"
name="reservaHotelResponse"/>
      </wsdl:operation>
   </wsdl:portType>

   <wsdl:binding name="HotelWsSoapBinding" type="impl:HotelCl">
      <wsdlsoap:binding style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="reservaHotel">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="reservaHotelRequest">
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://localhost:8080/wiagem" use="encoded"/>
         </wsdl:input>
         <wsdl:output name="reservaHotelResponse">
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://localhost:8080/wiagem/wsdl/" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>

   <wsdl:service name="HotelClService">
      <wsdl:port binding="impl:HotelWsSoapBinding" name="HotelWs">
         <wsdlsoap:address location="http://localhost:8080/active-
bpel/services/HotelWs"/>
      </wsdl:port>
   </wsdl:service>
<plnk:partnerLinkType name="HotelPLT">
   <plnk:role name="HotelRole" portType="impl:HotelCl" />
</plnk:partnerLinkType>

</wsdl:definitions>
```

Descritor WSDL do serviço Web de reserva de passagens

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://localhost:8080/wiagem"
xmlns:plnk="http://docs.oasis-open.org/wsbpel/2.0/plnktype"
xmlns:apachesoap="http://xml.apache.org/xml-soap"
```

```
xmlns:impl="http://localhost:8080/wiagem"
xmlns:intf="http://localhost:8080/wiagem"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<!--WSDL created by Apache Axis version: 1.4
Built on Dec 20, 2006 (01:34:32 EST)-->

   <wsdl:message name="reservaPassagemResponse">
      <wsdl:part name="passagemOut" type="xsd:string"/>
   </wsdl:message>

   <wsdl:message name="reservaPassagemRequest">
      <wsdl:part name="dadosPassagem" type="xsd:string"/>
   </wsdl:message>

   <wsdl:portType name="PassagemCl">
      <wsdl:operation name="reservaPassagem"
parameterOrder="dadosPassagem">
         <wsdl:input message="impl:reservaPassagemRequest"
name="reservaPassagemRequest"/>
         <wsdl:output message="impl:reservaPassagemResponse"
name="reservaPassagemResponse"/>
      </wsdl:operation>
   </wsdl:portType>

   <wsdl:binding name="PassagemWsSoapBinding" type="impl:PassagemCl">
      <wsdlsoap:binding style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="reservaPassagem">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="reservaPassagemRequest">
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://localhost:8080/wiagem" use="encoded"/>
         </wsdl:input>

         <wsdl:output name="reservaPassagemResponse">
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://localhost:8080/wiagem/wsdl/" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>

   <wsdl:service name="PassagemClService">
      <wsdl:port binding="impl:PassagemWsSoapBinding" name="PassagemWs">
         <wsdlsoap:address location="http://localhost:8080/active-
bpel/services/PassagemWs"/>
      </wsdl:port>
   </wsdl:service>
<plnk:partnerLinkType name="PassagemPLT">
   <plnk:role name="PassagemRole" portType="impl:PassagemCl" />
</plnk:partnerLinkType>
</wsdl:definitions>
```

Descritor WSDL do serviço Web monitor

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<wsdl:definitions targetNamespace="http://localhost:8080/wiagem"
xmlns:plnk="http://docs.oasis-open.org/wsbpel/2.0/plnktype"
xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="http://localhost:8080/wiagem"
xmlns:intf="http://localhost:8080/wiagem"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<!--WSDL created by Apache Axis version: 1.4
Built on Dec 20, 2006 (01:34:32 EST)-->

   <wsdl:message name="monitorarServicoResponse">
      <wsdl:part name="monitorOut" type="xsd:string"/>
   </wsdl:message>

   <wsdl:message name="monitorarServicoRequest">
      <wsdl:part name="dadosServico" type="xsd:string"/>
   </wsdl:message>

   <wsdl:portType name="MonitorCl">
      <wsdl:operation name="monitorarServico"
parameterOrder="dadosServico">
         <wsdl:input message="impl:monitorarServicoRequest"
name="monitorarServicoRequest"/>
         <wsdl:output message="impl:monitorarServicoResponse"
name="monitorarServicoResponse"/>
      </wsdl:operation>
   </wsdl:portType>

   <wsdl:binding name="MonitorWsSoapBinding" type="impl:MonitorCl">
      <wsdlsoap:binding style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="monitorarServico">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="monitorarServicoRequest">
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://localhost:8080/wiagem" use="encoded"/>
         </wsdl:input>

         <wsdl:output name="monitorarServicoResponse">
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://localhost:8080/wiagem/wsdl/" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>

   <wsdl:service name="MonitorClService">
      <wsdl:port binding="impl:MonitorWsSoapBinding" name="MonitorWs">
         <wsdlsoap:address location="http://localhost:8080/active-
bpel/services/MonitorWs"/>
      </wsdl:port>
   </wsdl:service>
<plnk:partnerLinkType name="MonitorPLT">
   <plnk:role name="MonitorRole" portType="impl:MonitorCl" />
</plnk:partnerLinkType>

</wsdl:definitions>
```

Processo de negócio de vendas de pacotes turísticos escrito em WS-BPEL

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
BPEL Process Definition
Edited using ActiveBPEL(tm) Designer Version 3.1.0 (http://www.active-
endpoints.com)
-->
<bpel:process xmlns:bpel="http://docs.oasis-
open.org/wsbpel/2.0/process/executable"
xmlns:ns1="http://localhost:8080/wiagem"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" name="WiagemProc"
suppressJoinFailure="yes" targetNamespace="http://WiagemProc">
   <bpel:import importType="http://schemas.xmlsoap.org/wsdl/"
location="wsdl/AgenciaTurismoWs.wsdl"
namespace="http://localhost:8080/wiagem"/>
   <bpel:import importType="http://schemas.xmlsoap.org/wsdl/"
location="wsdl/PassagemWs.wsdl" namespace="http://localhost:8080/wiagem"/>
   <bpel:import importType="http://schemas.xmlsoap.org/wsdl/"
location="wsdl/HotelWs.wsdl" namespace="http://localhost:8080/wiagem"/>
   <bpel:import importType="http://schemas.xmlsoap.org/wsdl/"
location="wsdl/CarroWs.wsdl" namespace="http://localhost:8080/wiagem"/>
   <bpel:partnerLinks>
      <bpel:partnerLink myRole="AgenciaTurismoRole"
name="AgenciaTurismoPLT" partnerLinkType="ns1:AgenciaTurismoPLT"/>
      <bpel:partnerLink name="PassagemPLT"
partnerLinkType="ns1:PassagemPLT" partnerRole="PassagemRole"/>
      <bpel:partnerLink name="HotelPLT" partnerLinkType="ns1:HotelPLT"
partnerRole="HotelRole"/>
      <bpel:partnerLink name="CarroPLT" partnerLinkType="ns1:CarroPLT"
partnerRole="CarroRole"/>
   </bpel:partnerLinks>
   <bpel:variables>
      <bpel:variable messageType="ns1:reservarViagemRequest"
name="reservarViagemRequest"/>
      <bpel:variable messageType="ns1:reservarViagemResponse"
name="reservarViagemResponse"/>
      <bpel:variable messageType="ns1:reservaPassagemRequest"
name="reservaPassagemRequest"/>
      <bpel:variable messageType="ns1:reservaPassagemResponse"
name="reservaPassagemResponse"/>
      <bpel:variable messageType="ns1:reservaHotelRequest"
name="reservaHotelRequest"/>
      <bpel:variable messageType="ns1:reservaHotelResponse"
name="reservaHotelResponse"/>
      <bpel:variable messageType="ns1:reservaCarroRequest"
name="reservaCarroRequest"/>
      <bpel:variable messageType="ns1:reservaCarroResponse"
name="reservaCarroResponse"/>
   </bpel:variables>
   <bpel:flow>
      <bpel:receive createInstance="yes" name="RecebeDadosViagem"
operation="reservarViagem" partnerLink="AgenciaTurismoPLT"
portType="ns1:AgenciaTurismoPT" variable="reservarViagemRequest">
      </bpel:receive>

      <bpel:assign name="iPassagem">
         <bpel:copy>
```

```
            <bpel:from part="viagemPassagemIn"
variable="reservarViagemRequest"/>
            <bpel:to part="dadosPassagem"
variable="reservaPassagemRequest"/>
         </bpel:copy>
      </bpel:assign>

      <bpel:invoke inputVariable="reservaPassagemRequest" name="Passagem"
operation="reservaPassagem" outputVariable="reservaPassagemResponse"
partnerLink="PassagemPLT" portType="ns1:PassagemCl">
      </bpel:invoke>

      <bpel:invoke inputVariable="reservaHotelRequest" name="Hotel"
operation="reservaHotel" outputVariable="reservaHotelResponse"
partnerLink="HotelPLT" portType="ns1:HotelCl">
      </bpel:invoke>
      <bpel:assign name="oPassagem-iHotel">
         <bpel:copy>
            <bpel:from part="passagemOut"
variable="reservaPassagemResponse"/>
            <bpel:to part="viagemPassagemOut"
variable="reservarViagemResponse"/>
         </bpel:copy>

         <bpel:copy>
            <bpel:from part="viagemHotelIn"
variable="reservarViagemRequest"/>
            <bpel:to part="dadosHotel" variable="reservaHotelRequest"/>
         </bpel:copy>
      </bpel:assign>

      <bpel:invoke inputVariable="reservaCarroRequest" name="Carro"
operation="reservaCarro" outputVariable="reservaCarroResponse"
partnerLink="CarroPLT" portType="ns1:CarroCl">
      </bpel:invoke>
      <bpel:assign name="oHotel-iCarro">
         <bpel:copy>
            <bpel:from part="hotelOut" variable="reservaHotelResponse"/>
            <bpel:to part="viagemHotelOut"
variable="reservarViagemResponse"/>
         </bpel:copy>

         <bpel:copy>
            <bpel:from part="viagemCarroIn"
variable="reservarViagemRequest"/>
            <bpel:to part="dadosCarro" variable="reservaCarroRequest"/>
         </bpel:copy>
      </bpel:assign>

      <bpel:reply name="RetornaViagem" operation="reservarViagem"
partnerLink="AgenciaTurismoPLT" portType="ns1:AgenciaTurismoPT"
variable="reservarViagemResponse">
      </bpel:reply>
   </bpel:flow>
</bpel:process>
```

Contrato eletrônico WS-Agreement para serviço de passagens

```
<wsag:AgreementOffer AgreementId=”passagensContrato”>
<wsag:Name>ContratoParaPassagens</wsag:Name>
```

```
<wsag:Context>

<wsag:AgreementInitiator>http://www.pacotesturisticos.com.br/</wsag:A...>
    <wsag:AgreementResponder>http://www.passagens.com/</wsag:A...>
</wsag:Context>
<wsag:Terms>

 <wsag:ServiceDescriptionTerm Name=”TermoServicoPassagens”
ServiceName=”PassagensService”>
 <wsdl:definitions ...>...
 </wsdl:definitions>
 </wsag:ServiceDescriptionTerm> ...
 <wsag:ServiceProperties Name=”PropriedadesServicoPassagens”
ServiceName=”PassagensService”>
    <wsag:Variable Name=”Disponibilidade” Metric=”eDisponivel”>

<wsag:Location>//wsag:SDT/[@name=”TermoServicoPassagens”@portType=”passagem
PT”@operation=”comprarPassagem”]</wsag:L>
    </wsag:Variable>
    <wsag:Variable Name=”Resposta” Metric=”tempoResposta”>

<wsag:Location>//wsag:SDT/[@name=”TermoServicoPassagens”@portType=”passagem
PT”@operation=”comprarPassagem”]</wsag:L>
    </wsag:Variable>
    <wsag:Variable Name=”Solicitacoes” Metric=”solicitacoesPorMinuto”>

<wsag:Location>//wsag:SDT/[@name=”TermoServicoPassagens”@portType=”passagem
PT”@operation=”comprarPassagem”]</wsag:L>
    </wsag:Variable>
 </wsag:ServiceProperties>

 <wsag:GuaranteeTerm wsag:name=”PassagensEDisponivel”>
    <wsag:ServiceScope>
       <wsag:ServiceName>PassagensService</wsag:ServiceName>
    </wsag:ServiceScope>
    <wsag:ServiceLevelObjective>
       <exp:Less>
          <exp:Variable>Disponibilidade</exp:Variable>
          <exp:Value>24x7</exp:Value>
       </exp:Less>
    </wsag:ServiceLevelObjective>...
 </wsag:GuaranteeTerm>

 <wsag:GuaranteeTerm wsag:name=”PassagensTempoResposta”>
    <wsag:ServiceScope>
       <wsag:ServiceName>PassagensService</wsag:ServiceName>
    </wsag:ServiceScope>
    <wsag:ServiceLevelObjective>
       <exp:Less>
          <exp:Variable>Resposta</exp:Variable>
          <exp:Value>3</exp:Value>
       </exp:Less>
    </wsag:ServiceLevelObjective>...
 </wsag:GuaranteeTerm>

 <wsag:GuaranteeTerm wsag:name=”PassagensSolicitacoesPorMinuto”>
    <wsag:ServiceScope>
       <wsag:ServiceName>PassagensService</wsag:ServiceName>
    </wsag:ServiceScope>
    <wsag:ServiceLevelObjective>
       <exp:Less>
```

```
          <exp:Variable>Solicitacoes</exp:Variable>
          <exp:Value>10</exp:Value>
       </exp:Less>
    </wsag:ServiceLevelObjective>...
 </wsag:GuaranteeTerm>

</wsag:Terms>
</wsag:AgreementOffer>
```

Contrato eletrônico WS-Agreement para serviço de hotéis

```
<wsag:AgreementOffer AgreementId=”hoteisContrato”>
<wsag:Name>ContratoParaHoteis</wsag:Name>
<wsag:Context>

<wsag:AgreementInitiator>http://www.pacotesturisticos.com.br/</wsag:A...>
    <wsag:AgreementResponder>http://www.hoteis.com/</wsag:A...>
</wsag:Context>
<wsag:Terms>

 <wsag:ServiceDescriptionTerm Name=”TermoServicoHoteis”
ServiceName=”HoteisService”>
 <wsdl:definitions ...>...
 </wsdl:definitions>
 </wsag:ServiceDescriptionTerm> ...
 <wsag:ServiceProperties Name=”PropriedadesServicoHoteis”
ServiceName=”HoteisService”>
    <wsag:Variable Name=”Disponibilidade” Metric=”eDisponivel”>

<wsag:Location>//wsag:SDT/[@name=”TermoServicoHoteis”@portType=”hotelPT”@op
eration=”reservaHotel”]</wsag:L>
    </wsag:Variable>
    <wsag:Variable Name=”Resposta” Metric=”tempoResposta”>

<wsag:Location>//wsag:SDT/[@name=”TermoServicoHoteis”@portType=”hotelPT”@op
eration=”reservaHotel”]</wsag:L>
    </wsag:Variable>
    <wsag:Variable Name=”Solicitacoes” Metric=”solicitacoesPorMinuto”>

<wsag:Location>//wsag:SDT/[@name=”TermoServicoHoteis”@portType=”hotelPT”@op
eration=”reservaHotel”]</wsag:L>
    </wsag:Variable>
 </wsag:ServiceProperties>

 <wsag:GuaranteeTerm wsag:name=”HoteisEDisponivel”>
    <wsag:ServiceScope>
       <wsag:ServiceName>HoteisService</wsag:ServiceName>
    </wsag:ServiceScope>
    <wsag:ServiceLevelObjective>
       <exp:Less>
          <exp:Variable>Disponibilidade</exp:Variable>
          <exp:Value>24x7</exp:Value>
       </exp:Less>
    </wsag:ServiceLevelObjective>...
 </wsag:GuaranteeTerm>

 <wsag:GuaranteeTerm wsag:name=”HoteisTempoResposta”>
    <wsag:ServiceScope>
       <wsag:ServiceName>HoteisService</wsag:ServiceName>
    </wsag:ServiceScope>
```

```
    <wsag:ServiceLevelObjective>
       <exp:Less>
          <exp:Variable>Resposta</exp:Variable>
          <exp:Value>4</exp:Value>
       </exp:Less>
    </wsag:ServiceLevelObjective>...
 </wsag:GuaranteeTerm>

 <wsag:GuaranteeTerm wsag:name=”HoteisSolicitacoesPorMinuto”>
    <wsag:ServiceScope>
       <wsag:ServiceName>HoteisService</wsag:ServiceName>
    </wsag:ServiceScope>
    <wsag:ServiceLevelObjective>
       <exp:Less>
          <exp:Variable>Solicitacoes</exp:Variable>
          <exp:Value>5</exp:Value>
       </exp:Less>
    </wsag:ServiceLevelObjective>...
 </wsag:GuaranteeTerm>

</wsag:Terms>
</wsag:AgreementOffer>
```

Contrato eletrônico WS-Agreement para serviço de automóveis

```
<wsag:AgreementOffer AgreementId=”automoveisContrato”>
<wsag:Name>ContratoParaAutomoveis</wsag:Name>
<wsag:Context>

<wsag:AgreementInitiator>http://www.pacotesturisticos.com.br/</wsag:A...>
    <wsag:AgreementResponder>http://www.automoveis.com/</wsag:A...>
</wsag:Context>
<wsag:Terms>

 <wsag:ServiceDescriptionTerm Name=”TermoServicoAutomoveis”
ServiceName=”AutomoveisService”>
 <wsdl:definitions ...>...
 </wsdl:definitions>
 </wsag:ServiceDescriptionTerm> ...
 <wsag:ServiceProperties Name=”PropriedadesServicoAutomoveis”
ServiceName=”AutomoveisService”>
    <wsag:Variable Name=”Disponibilidade” Metric=”eDisponivel”>

<wsag:Location>//wsag:SDT/[@name=”TermoServicoAutomoveis”@portType=”automov
elPT”@operation=”comprarAutomovel”]</wsag:L>
    </wsag:Variable>
    <wsag:Variable Name=”Resposta” Metric=”tempoResposta”>

<wsag:Location>//wsag:SDT/[@name=”TermoServicoAutomoveis”@portType=”automov
elPT”@operation=”comprarAutomovel”]</wsag:L>
    </wsag:Variable>
 </wsag:ServiceProperties>

 <wsag:GuaranteeTerm wsag:name=”AutomoveisEDisponivel”>
    <wsag:ServiceScope>
       <wsag:ServiceName>AutomoveisService</wsag:ServiceName>
    </wsag:ServiceScope>
    <wsag:ServiceLevelObjective>
       <exp:Less>
          <exp:Variable>Disponibilidade</exp:Variable>
```

```
          <exp:Value>24x7</exp:Value>
       </exp:Less>
    </wsag:ServiceLevelObjective>...
 </wsag:GuaranteeTerm>

 <wsag:GuaranteeTerm wsag:name=”AutomoveisTempoResposta”>
    <wsag:ServiceScope>
       <wsag:ServiceName>AutomoveisService</wsag:ServiceName>
    </wsag:ServiceScope>
    <wsag:ServiceLevelObjective>
       <exp:Less>
          <exp:Variable>Resposta</exp:Variable>
          <exp:Value>10</exp:Value>
       </exp:Less>
    </wsag:ServiceLevelObjective>...
 </wsag:GuaranteeTerm>

</wsag:Terms>
</wsag:AgreementOffer>
```

Aspectos de monitoramento para serviços de passagens

```
<aspect name="ContratoPassagens-Disponibilidade">
    <partners>
       <partner name="monitorPartner" serviceLinkType="monitorSLT"/>
     </partners>
   <messages>
     <message name="monitorRequestType">
        <part name="ServicePropertyId" type="xsd:string"/>
        <part name="ServicePropertyData" type="xsd:string"/>
   </message >
    <message name="monitorResponseType">
       <part name="monitorData" type="xsd:string"/>
     </message>
   </messages>
   <variables>
   <variable name="monitorRequest" messageType="monitorRequestType"/>
    <variable name="monitorResponse" messageType="monitorResponseType"/>
  </variables>

    <pointcut name="PassagensInvoke" contextCollection="true">

//process//invoke[@serviceName="PassagensService"@portType="passagensPT"
@operation="comprarPassagem"]
    </pointcut>
    <advice type="after">
    <bpws:sequence>
    <bpws:assign>
       <bpws:copy>
          <bpws:from expression="ContratoPassagens-Disponibilidade"/>
          <bpws:to variable="monitorRequest" part="ServicePropertyId"/>
       </bpws:copy>
       <bpws:copy>
          <bpws:from //process//invoke[@serviceStatus]/>
          <bpws:to variable="monitorRequest" part="ServicePropertyData"/>
       </bpws:copy>
    </bpws:assign>
       <bpws:invoke name="invoke" partner="monitorPartner"
portType="ServicoMonitor" operation="monitora"
inputVariable="monitorRequest" outputVariable="monitorResponse"/>
```

```
    </bpws:sequence>
    </advice>
 </aspect>

<aspect name="ContratoPassagens-TempoResposta">
    <partners>
       <partner name="monitorPartner" serviceLinkType="monitorSLT"/>
     </partners>
   <messages>
     <message name="monitorRequestType">
        <part name="ServicePropertyId" type="xsd:string"/>
        <part name="ServicePropertyData" type="xsd:string"/>
   </message >
    <message name="monitorResponseType">
       <part name="monitorData" type="xsd:string"/>
     </message>
   </messages>
   <variables>
   <variable name="monitorRequest" messageType="monitorRequestType"/>
    <variable name="monitorResponse" messageType="monitorResponseType"/>
  </variables>

    <pointcut name="PassagensInvoke" contextCollection="true">

//process//invoke[@serviceName="PassagensService"@portType="passagensPT"
@operation="comprarPassagem"]
    </pointcut>
    <advice type="around">
    <bpws:sequence>
    <bpws:assign>
       <bpws:copy>
          <bpws:from expression="ContratoPassagens-TempoResposta"/>
          <bpws:to variable="monitorRequest" part="ServicePropertyId"/>
       </bpws:copy>
       <bpws:copy>
          <bpws:from //process//invoke[@serviceStatus]/>
          <bpws:to variable="monitorRequest" part="ServicePropertyData"/>
       </bpws:copy>
    </bpws:assign>
       <bpws:invoke name="invoke" partner="monitorPartner"
portType="ServicoMonitor" operation="monitora"
inputVariable="monitorRequest" outputVariable="monitorResponse"/>
    </bpws:sequence>
    </advice>
 </aspect>

<aspect name="ContratoPassagens-SolicitacoesPorMinuto">
    <partners>
       <partner name="monitorPartner" serviceLinkType="monitorSLT"/>
     </partners>
   <messages>
     <message name="monitorRequestType">
        <part name="ServicePropertyId" type="xsd:string"/>
        <part name="ServicePropertyData" type="xsd:string"/>
   </message >
    <message name="monitorResponseType">
       <part name="monitorData" type="xsd:string"/>
     </message>
   </messages>
   <variables>
   <variable name="monitorRequest" messageType="monitorRequestType"/>
```

```
    <variable name="monitorResponse" messageType="monitorResponseType"/>
  </variables>

    <pointcut name="PassagensInvoke" contextCollection="true">

//process//invoke[@serviceName="PassagensService"@portType="passagensPT"
@operation="comprarPassagem"]
    </pointcut>
    <advice type="around">
    <bpws:sequence>
    <bpws:assign>
       <bpws:copy>
          <bpws:from expression="ContratoPassagens-SolicitacoesPorMinuto"/>
          <bpws:to variable="monitorRequest" part="ServicePropertyId"/>
       </bpws:copy>
       <bpws:copy>
          <bpws:from //process//invoke[@serviceStatus]/>
          <bpws:to variable="monitorRequest" part="ServicePropertyData"/>
       </bpws:copy>
    </bpws:assign>
       <bpws:invoke name="invoke" partner="monitorPartner"
portType="ServicoMonitor" operation="monitora"
inputVariable="monitorRequest" outputVariable="monitorResponse"/>
    </bpws:sequence>
    </advice>
 </aspect>
```

Aspectos de monitoramento para serviços de hotéis

```
<aspect name="ContratoHoteis-Disponibilidade">
    <partners>
       <partner name="monitorPartner" serviceLinkType="monitorSLT"/>
     </partners>
   <messages>
     <message name="monitorRequestType">
        <part name="ServicePropertyId" type="xsd:string"/>
        <part name="ServicePropertyData" type="xsd:string"/>
   </message >
    <message name="monitorResponseType">
       <part name="monitorData" type="xsd:string"/>
     </message>
   </messages>
   <variables>
   <variable name="monitorRequest" messageType="monitorRequestType"/>
    <variable name="monitorResponse" messageType="monitorResponseType"/>
  </variables>

    <pointcut name="HoteisInvoke" contextCollection="true">
       //process//invoke[@serviceName="HoteisService"@portType="hoteisPT"
@operation="reservarHotel"]
    </pointcut>
    <advice type="after">
    <bpws:sequence>
    <bpws:assign>
       <bpws:copy>
          <bpws:from expression="ContratoHoteis-Disponibilidade"/>
          <bpws:to variable="monitorRequest" part="ServicePropertyId"/>
       </bpws:copy>
       <bpws:copy>
          <bpws:from //process//invoke[@serviceStatus]/>
```

```
          <bpws:to variable="monitorRequest" part="ServicePropertyData"/>
       </bpws:copy>
    </bpws:assign>
       <bpws:invoke name="invoke" partner="monitorPartner"
portType="ServicoMonitor" operation="monitora"
inputVariable="monitorRequest" outputVariable="monitorResponse"/>
    </bpws:sequence>
    </advice>
 </aspect>


 <aspect name="ContratoHoteis-TempoResposta">
    <partners>
       <partner name="monitorPartner" serviceLinkType="monitorSLT"/>
     </partners>
   <messages>
     <message name="monitorRequestType">
        <part name="ServicePropertyId" type="xsd:string"/>
        <part name="ServicePropertyData" type="xsd:string"/>
   </message >
    <message name="monitorResponseType">
       <part name="monitorData" type="xsd:string"/>
     </message>
   </messages>
   <variables>
   <variable name="monitorRequest" messageType="monitorRequestType"/>
    <variable name="monitorResponse" messageType="monitorResponseType"/>
  </variables>

    <pointcut name="HoteisInvoke" contextCollection="true">
       //process//invoke[@serviceName="HoteisService"@portType="hoteisPT"
@operation="reservarHotel"]
    </pointcut>
    <advice type="around">
    <bpws:sequence>
    <bpws:assign>
       <bpws:copy>
          <bpws:from expression="ContratoHoteis-TempoResposta"/>
          <bpws:to variable="monitorRequest" part="ServicePropertyId"/>
       </bpws:copy>
       <bpws:copy>
          <bpws:from //process//invoke[@serviceStatus]/>
          <bpws:to variable="monitorRequest" part="ServicePropertyData"/>
       </bpws:copy>
    </bpws:assign>
       <bpws:invoke name="invoke" partner="monitorPartner"
portType="ServicoMonitor" operation="monitora"
inputVariable="monitorRequest" outputVariable="monitorResponse"/>
    </bpws:sequence>
    </advice>
 </aspect>


 <aspect name="ContratoHoteis-SolicitacoesPorMinuto">
    <partners>
       <partner name="monitorPartner" serviceLinkType="monitorSLT"/>
     </partners>
   <messages>
     <message name="monitorRequestType">
        <part name="ServicePropertyId" type="xsd:string"/>
        <part name="ServicePropertyData" type="xsd:string"/>
```

```
   </message >
    <message name="monitorResponseType">
       <part name="monitorData" type="xsd:string"/>
     </message>
   </messages>
   <variables>
   <variable name="monitorRequest" messageType="monitorRequestType"/>
    <variable name="monitorResponse" messageType="monitorResponseType"/>
  </variables>

    <pointcut name="HoteisInvoke" contextCollection="true">
       //process//invoke[@serviceName="HoteisService"@portType="hoteisPT"
@operation="reservarHotel"]
    </pointcut>
    <advice type="around">
    <bpws:sequence>
    <bpws:assign>
       <bpws:copy>
          <bpws:from expression="ContratoHoteis-SolicitacoesPorMinuto"/>
          <bpws:to variable="monitorRequest" part="ServicePropertyId"/>
       </bpws:copy>
       <bpws:copy>
          <bpws:from //process//invoke[@serviceStatus]/>
          <bpws:to variable="monitorRequest" part="ServicePropertyData"/>
       </bpws:copy>
    </bpws:assign>
       <bpws:invoke name="invoke" partner="monitorPartner"
portType="ServicoMonitor" operation="monitora"
inputVariable="monitorRequest" outputVariable="monitorResponse"/>
    </bpws:sequence>
    </advice>
 </aspect>
```

Aspectos de monitoramento para serviços de automóveis

```
<aspect name="ContratoAutomoveis-Disponibilidade">
    <partners>
       <partner name="monitorPartner" serviceLinkType="monitorSLT"/>
     </partners>
   <messages>
     <message name="monitorRequestType">
        <part name="ServicePropertyId" type="xsd:string"/>
        <part name="ServicePropertyData" type="xsd:string"/>
   </message >
    <message name="monitorResponseType">
       <part name="monitorData" type="xsd:string"/>
     </message>
   </messages>
   <variables>
   <variable name="monitorRequest" messageType="monitorRequestType"/>
    <variable name="monitorResponse" messageType="monitorResponseType"/>
  </variables>

    <pointcut name="AutomoveisInvoke" contextCollection="true">

//process//invoke[@serviceName="AutomoveisService"@portType="automoveisPT"
@operation="reservarAutomovel"]
    </pointcut>
    <advice type="after">
    <bpws:sequence>
```

```
    <bpws:assign>
       <bpws:copy>
          <bpws:from expression="ContratoAutomoveis-Disponibilidade"/>
          <bpws:to variable="monitorRequest" part="ServicePropertyId"/>
       </bpws:copy>
       <bpws:copy>
          <bpws:from //process//invoke[@serviceStatus]/>
          <bpws:to variable="monitorRequest" part="ServicePropertyData"/>
       </bpws:copy>
    </bpws:assign>
       <bpws:invoke name="invoke" partner="monitorPartner"
portType="ServicoMonitor" operation="monitora"
inputVariable="monitorRequest" outputVariable="monitorResponse"/>
    </bpws:sequence>
    </advice>
 </aspect>


 <aspect name="ContratoAutomoveis-TempoResposta">
    <partners>
       <partner name="monitorPartner" serviceLinkType="monitorSLT"/>
     </partners>
   <messages>
     <message name="monitorRequestType">
        <part name="ServicePropertyId" type="xsd:string"/>
        <part name="ServicePropertyData" type="xsd:string"/>
   </message >
    <message name="monitorResponseType">
       <part name="monitorData" type="xsd:string"/>
     </message>
   </messages>
   <variables>
   <variable name="monitorRequest" messageType="monitorRequestType"/>
    <variable name="monitorResponse" messageType="monitorResponseType"/>
  </variables>

    <pointcut name="AutomoveisInvoke" contextCollection="true">

//process//invoke[@serviceName="AutomoveisService"@portType="automoveisPT"
@operation="reservarAutomovel"]
    </pointcut>
    <advice type="around">
    <bpws:sequence>
    <bpws:assign>
       <bpws:copy>
          <bpws:from expression="ContratoAutomoveis-TempoResposta"/>
          <bpws:to variable="monitorRequest" part="ServicePropertyId"/>
       </bpws:copy>
       <bpws:copy>
          <bpws:from //process//invoke[@serviceStatus]/>
          <bpws:to variable="monitorRequest" part="ServicePropertyData"/>
       </bpws:copy>
    </bpws:assign>
       <bpws:invoke name="invoke" partner="monitorPartner"
portType="ServicoMonitor" operation="monitora"
inputVariable="monitorRequest" outputVariable="monitorResponse"/>
    </bpws:sequence>
    </advice>
 </aspect>
```

```
 <aspect name="ContratoAutomoveis-SolicitacoesPorMinuto">
    <partners>
       <partner name="monitorPartner" serviceLinkType="monitorSLT"/>
     </partners>
   <messages>
     <message name="monitorRequestType">
        <part name="ServicePropertyId" type="xsd:string"/>
        <part name="ServicePropertyData" type="xsd:string"/>
   </message >
    <message name="monitorResponseType">
       <part name="monitorData" type="xsd:string"/>
     </message>
   </messages>
   <variables>
   <variable name="monitorRequest" messageType="monitorRequestType"/>
    <variable name="monitorResponse" messageType="monitorResponseType"/>
  </variables>

    <pointcut name="AutomoveisInvoke" contextCollection="true">

//process//invoke[@serviceName="AutomoveisService"@portType="automoveisPT"
@operation="reservarAutomovel"]
    </pointcut>
    <advice type="around">
    <bpws:sequence>
    <bpws:assign>
       <bpws:copy>
          <bpws:from expression="ContratoAutomoveis-
SolicitacoesPorMinuto"/>
          <bpws:to variable="monitorRequest" part="ServicePropertyId"/>
       </bpws:copy>
       <bpws:copy>
          <bpws:from //process//invoke[@serviceStatus]/>
          <bpws:to variable="monitorRequest" part="ServicePropertyData"/>
       </bpws:copy>
    </bpws:assign>
       <bpws:invoke name="invoke" partner="monitorPartner"
portType="ServicoMonitor" operation="monitora"
inputVariable="monitorRequest" outputVariable="monitorResponse"/>
    </bpws:sequence>
    </advice>
 </aspect>
```

Processo de negócio de vendas de pacotes turísticos escrito em WS-BPEL com as aspectos de monitoramento inseridos

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
BPEL Process Definition
Edited using ActiveBPEL(tm) Designer Version 3.1.0 (http://www.active-
endpoints.com)
-->
<bpel:process xmlns:bpel="http://docs.oasis-
open.org/wsbpel/2.0/process/executable"
xmlns:ns1="http://localhost:8080/wiagem"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" name="WiagemProc"
suppressJoinFailure="yes" targetNamespace="http://WiagemProc">
```

```
   <bpel:import importType="http://schemas.xmlsoap.org/wsdl/"
location="wsdl/AgenciaTurismoWs.wsdl"
namespace="http://localhost:8080/wiagem"/>
   <bpel:import importType="http://schemas.xmlsoap.org/wsdl/"
location="wsdl/PassagemWs.wsdl" namespace="http://localhost:8080/wiagem"/>
   <bpel:import importType="http://schemas.xmlsoap.org/wsdl/"
location="wsdl/HotelWs.wsdl" namespace="http://localhost:8080/wiagem"/>
   <bpel:import importType="http://schemas.xmlsoap.org/wsdl/"
location="wsdl/CarroWs.wsdl" namespace="http://localhost:8080/wiagem"/>

   <bpel:import importType="http://schemas.xmlsoap.org/wsdl/"
location="wsdl/MonitorWs.wsdl" namespace="http://localhost:8080/wiagem"/>
   <bpel:partnerLinks>
      <bpel:partnerLink myRole="AgenciaTurismoRole"
name="AgenciaTurismoPLT" partnerLinkType="ns1:AgenciaTurismoPLT"/>
      <bpel:partnerLink name="PassagemPLT"
partnerLinkType="ns1:PassagemPLT" partnerRole="PassagemRole"/>
      <bpel:partnerLink name="HotelPLT" partnerLinkType="ns1:HotelPLT"
partnerRole="HotelRole"/>
      <bpel:partnerLink name="CarroPLT" partnerLinkType="ns1:CarroPLT"
partnerRole="CarroRole"/>

      <bpel:partnerLink name="MonitorPLT" partnerLinkType="ns1:MonitorPLT"
partnerRole="MonitorRole"/>
   </bpel:partnerLinks>
   <bpel:variables>
      <bpel:variable messageType="ns1:reservarViagemRequest"
name="reservarViagemRequest"/>
      <bpel:variable messageType="ns1:reservarViagemResponse"
name="reservarViagemResponse"/>
      <bpel:variable messageType="ns1:reservaPassagemRequest"
name="reservaPassagemRequest"/>
      <bpel:variable messageType="ns1:reservaPassagemResponse"
name="reservaPassagemResponse"/>
      <bpel:variable messageType="ns1:reservaHotelRequest"
name="reservaHotelRequest"/>
      <bpel:variable messageType="ns1:reservaHotelResponse"
name="reservaHotelResponse"/>
      <bpel:variable messageType="ns1:reservaCarroRequest"
name="reservaCarroRequest"/>
      <bpel:variable messageType="ns1:reservaCarroResponse"
name="reservaCarroResponse"/>

      <bpel:variable messageType="ns1:monitorRequest"
name="monitorRequest"/>
      <bpel:variable messageType="ns1:monitorResponse"
name="monitorResponse"/>
   </bpel:variables>
   <bpel:flow>
      <bpel:receive createInstance="yes" name="RecebeDadosViagem"
operation="reservarViagem" partnerLink="AgenciaTurismoPLT"
portType="ns1:AgenciaTurismoPT" variable="reservarViagemRequest">
      </bpel:receive>

      <bpel:assign name="iPassagem">
         <bpel:copy>
            <bpel:from part="viagemPassagemIn"
variable="reservarViagemRequest"/>
            <bpel:to part="dadosPassagem"
variable="reservaPassagemRequest"/>
         </bpel:copy>
```

```
      </bpel:assign>

      <bpws:assign>
       <bpws:copy>
          <bpws:from expression="ContratoPassagens-TempoResposta"/>
          <bpws:to variable="monitorRequest" part="ServicePropertyId"/>
       </bpws:copy>
       <bpws:copy>
          <bpws:from //process//invoke[@serviceStatus]/>
          <bpws:to variable="monitorRequest" part="ServicePropertyData"/>
       </bpws:copy>
      </bpws:assign>
      <bpel:invoke name="invoke" partner="monitorPartner"
portType="ServicoMonitor" operation="monitora"
inputVariable="monitorRequest" outputVariable="monitorResponse">
      </bpel:invoke>

      <bpel:invoke inputVariable="reservaPassagemRequest" name="Passagem"
operation="reservaPassagem" outputVariable="reservaPassagemResponse"
partnerLink="PassagemPLT" portType="ns1:PassagemCl">
      </bpel:invoke>

      <bpws:assign>
       <bpws:copy>
          <bpws:from expression="ContratoPassagens-Disponibilidade"/>
          <bpws:to variable="monitorRequest" part="ServicePropertyId"/>
       </bpws:copy>
       <bpws:copy>
          <bpws:from //process//invoke[@serviceStatus]/>
          <bpws:to variable="monitorRequest" part="ServicePropertyData"/>
       </bpws:copy>
      </bpws:assign>
      <bpel:invoke name="invoke" partner="monitorPartner"
portType="ServicoMonitor" operation="monitora"
inputVariable="monitorRequest" outputVariable="monitorResponse">
      </bpel:invoke>

      <bpws:assign>
       <bpws:copy>
          <bpws:from expression="ContratoPassagens-TempoResposta"/>
          <bpws:to variable="monitorRequest" part="ServicePropertyId"/>
       </bpws:copy>
       <bpws:copy>
          <bpws:from //process//invoke[@serviceStatus]/>
          <bpws:to variable="monitorRequest" part="ServicePropertyData"/>
       </bpws:copy>
      </bpws:assign>
      <bpel:invoke name="invoke" partner="monitorPartner"
portType="ServicoMonitor" operation="monitora"
inputVariable="monitorRequest" outputVariable="monitorResponse">
      </bpel:invoke>

      <bpws:assign>
       <bpws:copy>
          <bpws:from expression="ContratoPassagens-SolicitacoesPorMinuto"/>
          <bpws:to variable="monitorRequest" part="ServicePropertyId"/>
       </bpws:copy>
       <bpws:copy>
          <bpws:from //process//invoke[@serviceStatus]/>
          <bpws:to variable="monitorRequest" part="ServicePropertyData"/>
       </bpws:copy>
```

```
      </bpws:assign>
      <bpel:invoke name="invoke" partner="monitorPartner"
portType="ServicoMonitor" operation="monitora"
inputVariable="monitorRequest" outputVariable="monitorResponse">
      </bpel:invoke>

      <bpws:assign>
       <bpws:copy>
          <bpws:from expression="ContratoHoteis-TempoResposta"/>
          <bpws:to variable="monitorRequest" part="ServicePropertyId"/>
       </bpws:copy>
       <bpws:copy>
          <bpws:from //process//invoke[@serviceStatus]/>
          <bpws:to variable="monitorRequest" part="ServicePropertyData"/>
       </bpws:copy>
      </bpws:assign>
      <bpel:invoke name="invoke" partner="monitorPartner"
portType="ServicoMonitor" operation="monitora"
inputVariable="monitorRequest" outputVariable="monitorResponse">
      </bpel:invoke>

      <bpel:invoke inputVariable="reservaHotelRequest" name="Hotel"
operation="reservaHotel" outputVariable="reservaHotelResponse"
partnerLink="HotelPLT" portType="ns1:HotelCl">
      </bpel:invoke>

      <bpws:assign>
       <bpws:copy>
          <bpws:from expression="ContratoHoteis-Disponibilidade"/>
          <bpws:to variable="monitorRequest" part="ServicePropertyId"/>
       </bpws:copy>
       <bpws:copy>
          <bpws:from //process//invoke[@serviceStatus]/>
          <bpws:to variable="monitorRequest" part="ServicePropertyData"/>
       </bpws:copy>
      </bpws:assign>
      <bpel:invoke name="invoke" partner="monitorPartner"
portType="ServicoMonitor" operation="monitora"
inputVariable="monitorRequest" outputVariable="monitorResponse">
      </bpel:invoke>

      <bpws:assign>
       <bpws:copy>
          <bpws:from expression="ContratoHoteis-TempoResposta"/>
          <bpws:to variable="monitorRequest" part="ServicePropertyId"/>
       </bpws:copy>
       <bpws:copy>
          <bpws:from //process//invoke[@serviceStatus]/>
          <bpws:to variable="monitorRequest" part="ServicePropertyData"/>
       </bpws:copy>
      </bpws:assign>
      <bpel:invoke name="invoke" partner="monitorPartner"
portType="ServicoMonitor" operation="monitora"
inputVariable="monitorRequest" outputVariable="monitorResponse">
      </bpel:invoke>

      <bpws:assign>
       <bpws:copy>
          <bpws:from expression="ContratoHoteis-SolicitacoesPorMinuto"/>
          <bpws:to variable="monitorRequest" part="ServicePropertyId"/>
       </bpws:copy>
```

```
       <bpws:copy>
          <bpws:from //process//invoke[@serviceStatus]/>
          <bpws:to variable="monitorRequest" part="ServicePropertyData"/>
       </bpws:copy>
      </bpws:assign>
      <bpel:invoke name="invoke" partner="monitorPartner"
portType="ServicoMonitor" operation="monitora"
inputVariable="monitorRequest" outputVariable="monitorResponse">
      </bpel:invoke>


      <bpel:assign name="oPassagem-iHotel">
         <bpel:copy>
            <bpel:from part="passagemOut"
variable="reservaPassagemResponse"/>
            <bpel:to part="viagemPassagemOut"
variable="reservarViagemResponse"/>
         </bpel:copy>

         <bpel:copy>
            <bpel:from part="viagemHotelIn"
variable="reservarViagemRequest"/>
            <bpel:to part="dadosHotel" variable="reservaHotelRequest"/>
         </bpel:copy>
      </bpel:assign>

      <bpws:assign>
       <bpws:copy>
          <bpws:from expression="ContratoAutomoveis-TempoResposta"/>
          <bpws:to variable="monitorRequest" part="ServicePropertyId"/>
       </bpws:copy>
       <bpws:copy>
          <bpws:from //process//invoke[@serviceStatus]/>
          <bpws:to variable="monitorRequest" part="ServicePropertyData"/>
       </bpws:copy>
      </bpws:assign>
      <bpel:invoke name="invoke" partner="monitorPartner"
portType="ServicoMonitor" operation="monitora"
inputVariable="monitorRequest" outputVariable="monitorResponse">
      </bpel:invoke>

      <bpel:invoke inputVariable="reservaCarroRequest" name="Carro"
operation="reservaCarro" outputVariable="reservaCarroResponse"
partnerLink="CarroPLT" portType="ns1:CarroCl">
      </bpel:invoke>

      <bpws:assign>
       <bpws:copy>
          <bpws:from expression="ContratoAutomoveis-Disponibilidade"/>
          <bpws:to variable="monitorRequest" part="ServicePropertyId"/>
       </bpws:copy>
       <bpws:copy>
          <bpws:from //process//invoke[@serviceStatus]/>
          <bpws:to variable="monitorRequest" part="ServicePropertyData"/>
       </bpws:copy>
      </bpws:assign>
      <bpel:invoke name="invoke" partner="monitorPartner"
portType="ServicoMonitor" operation="monitora"
inputVariable="monitorRequest" outputVariable="monitorResponse">
      </bpel:invoke>
```

```
      <bpws:assign>
       <bpws:copy>
          <bpws:from expression="ContratoAutomoveis-TempoResposta"/>
          <bpws:to variable="monitorRequest" part="ServicePropertyId"/>
       </bpws:copy>
       <bpws:copy>
          <bpws:from //process//invoke[@serviceStatus]/>
          <bpws:to variable="monitorRequest" part="ServicePropertyData"/>
       </bpws:copy>
      </bpws:assign>
      <bpel:invoke name="invoke" partner="monitorPartner"
portType="ServicoMonitor" operation="monitora"
inputVariable="monitorRequest" outputVariable="monitorResponse">
      </bpel:invoke>

      <bpel:assign name="oHotel-iCarro">
         <bpel:copy>
            <bpel:from part="hotelOut" variable="reservaHotelResponse"/>
            <bpel:to part="viagemHotelOut"
variable="reservarViagemResponse"/>
         </bpel:copy>

         <bpel:copy>
            <bpel:from part="viagemCarroIn"
variable="reservarViagemRequest"/>
            <bpel:to part="dadosCarro" variable="reservaCarroRequest"/>
         </bpel:copy>
      </bpel:assign>

      <bpel:reply name="RetornaViagem" operation="reservarViagem"
partnerLink="AgenciaTurismoPLT" portType="ns1:AgenciaTurismoPT"
variable="reservarViagemResponse">
      </bpel:reply>
   </bpel:flow>
</bpel:process>
```

Dados para teste no Soap Client

URL:
http://localhost:8080/active-bpel/services/AgenciaTurismoPLTService

ENTRADA:

```
<reservarViagem>
<viagemPassagemIn>dadosPassagem</viagemPassagemIn>
<viagemHotelIn>dadosHotel</viagemHotelIn>
<viagemCarroIn>dadosCarro</viagemCarroIn>
</reservarViagem>
```

SAÍDA:

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Header/>
   <soapenv:Body>
      <reservarViagemResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <viagemPassagemOut xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:type="xsd:string">Procurando passagens para:
dadosPassagem</viagemPassagemOut>
```

```
         <viagemHotelOut xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:type="xsd:string">Procurando hoteis para:
dadosHotel</viagemHotelOut>
         <viagemCarroOut xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:type="xsd:string">Procurando carros para:
dadosCarro</viagemCarroOut>
         <viagemLogMonitor xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:type="xsd:string">Monitorando servico:
PassagemWs:reservaPassagem:antes ; Monitorando servico:
HotelWs:reservaHotel:antes ; Monitorando servico:
HotelWs:reservaHotel:depois ; Monitorando servico:
CarroWs:reservaCarro:depois</viagemLogMonitor>
      </reservarViagemResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo