UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍTICA
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Paulo Henrique Michels

COREOGRAFIA DE SERVIÇOS WEB
(Uma abordagem para a integração de serviços Web)

Florianópolis – SC
JUNHO/2007

Paulo Henrique Michels

COREOGRAFIA DE SERVIÇOS WEB
(Uma abordagem para a integração de serviços Web)

Trabalho de conclusão de curso submetido à Universidade Federal de Santa Catarina como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de informação.

Orientador: Prof. Renato Fileto, Dr.
Universidade Federal de Santa Catarina
fileto@inf.ufsc.br

Banca examinadora

Prof. Frank Augusto Siqueira, Dr.
Universidade Federal de Santa Catarina
frank@inf.ufsc.br

Prof. Mário Antônio Ribeiro Dantas, Dr.
Universidade Federal de Santa Catarina
mario@inf.ufsc.br

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

O número crescente de componentes de software sendo disponibilizados na forma de serviços Web (*Web Services*) e conseqüentemente a popularização da arquitetura orientada a serviços, faz surgir uma demanda por métodos e padrões para integração de serviços.

Este trabalho apresenta os conceitos de orquestração e coreografia de serviços e de que forma eles tratam a questão da integração de serviços. O trabalho faz uma análise mais detalhada da coreografia de serviços, apresentando em detalhes as linguagens WSCI e WS-CDL. Em seguida analisa a ferramenta PI4SOA, que auxilia na construção de coreografias utilizando a WS-CDL. Ao final, aplica os conceitos e as linguagens apresentadas na solução de um problema real de integração de serviços.

ABSTRACT

The increasing number of software components developed with the Web Services technology and consequently the popularization of the service oriented architecture (SOA), are creating the demand for methods and patterns to service integration.

This paper shows the service orquestration and the service choreography concepts and explains how they deal with the service integration problem. The paper focus on the service choreography concept, showing the WSCI and WS-CDL languages. It also analyzes the PI4SOA tool, which helps on describing choreographies using the WS-CDL language. At the end, it uses the presented concepts, languages and tools to solve a real service integration problem.

1. INTRODUÇÃO

Durante o processo de desenvolvimento de sistemas distribuídos surgem diversas barreiras, como a incompatibilidade de protocolos de comunicação, incompatibilidade de linguagens de programação, questões de segurança, incompatibilidade de tipos de dados e vários outros problemas que fazem com que a integração de sistemas seja um dos maiores problemas da área de tecnologia da informação.

As facilidades que os serviços Web (*Web services*) trazem para o desenvolvimento de sistemas distribuídos estão fazendo com que eles tenham larga adoção nos sistemas modernos. Isto faz emergir métodos de construção de sistemas baseados na integração de serviços Web e na arquitetura orientada a serviços (ou *SOA*. de *Service Oriented Architecture*, em inglês) [1].

Apesar dos benefícios da tecnologia de serviços Web, os requisitos impostos pelos sistemas atuais, impõem a necessidade do auxílio de elementos responsáveis pela integração de serviços Web e pela coordenação da execução de processos constituídos pela composição de tais serviços. Os conceitos de orquestração e coreografia estendem a arquitetura orientada a serviços Web, criando padrões e linguagens que atendam a estes requisitos [3].

O conceito de orquestração é mais maduro e dispõe de padrões bem definidos, como a linguagem WS-BPEL, que é suportada pelos principais fornecedores da área. Já o conceito de coreografia é menos debatido e apenas mais recentemente vem recebendo atenção. Apesar de existirem algumas iniciativas para o desenvolvimento de um padrão de coreografia, até o momento não existe nada suficientemente maduro ou largamente apoiado pelo mercado.

Com o objetivo de compreendermos os atuais esforços para o desenvolvimento de uma linguagem padrão para a descrição de coreografias, analisamos as linguagens WSCI e WS-CDL, com foco especial nesta última. Avaliamos também as primeiras iniciativas do mercado para o desenvolvimento de ferramentas para a descrição de coreografias, em especial a ferramenta PI4SOA, baseada na linguagem WS-CDL [4][7].

Linguagens e ferramentas somente podem ser analisadas de forma adequada quando aplicadas em um estudo de caso com um contexto próximo da realidade. Assim, este trabalho utiliza a ferramenta PI4SOA e a linguagem WS-CDL para o desenvolvimento de um estudo de caso do contexto de uma agência de viagens, com suas interações com clientes (viajantes) e alguns fornecedores (empresas aéreas).

O presente trabalho segue apresentando os padrões fundamentais de implementação de serviços Web e a arquitetura SOA na Seção 2. Em seguida, a Seção 3 mostra a necessidade de

uma extensão destas arquiteturas, através dos conceitos de orquestração e coreografia. A Seção 4 descreve brevemente alguns padrões para expressar orquestração de serviços. Como este trabalho dispensa maior atenção a coreografia de serviços, a Seção 6 descreve em detalhes as linguagens WSCI e WS-CDL e a Seção 6 descreve a ferramenta PI4SOA. A Seção 7 utiliza os conceitos e técnicas apresentados nas seções anteriores para a elaboração de um estudo de caso. Finalmente, a Seção 8 descreve alguns trabalhos relacionados e a Seção 9 apresenta as conclusões e trabalhos futuros.

2. SERVIÇOS WEB E A ARQUITETURA ORIENTADA A SERVIÇOS

A tecnologia de serviços Web (*Web services*) nasceu para atender a um dos principais problemas da área de tecnologia da informação, que foi e continua sendo a integração de sistemas. Esta tecnologia se desenvolveu a partir da especificação de um conjunto de padrões abertos, baseados em XML, que tornam os serviços Web independentes de protocolo de rede, independentes de plataforma e independentes de linguagem de programação. Os padrões que formam a base dos serviços Web são o WSDL, o SOAP e o UDDI. Eles definem um formato de mensagem, determinam um padrão para a especificação das interfaces para troca de mensagens, criam convenções para o mapeamento entre mensagem e implementação do serviço e definem mecanismos para publicação e pesquisa de serviços Web.

Como elemento central no conjunto de padrões que compõem os serviços Web está a WSDL *(Web Services Description Language)*, responsável por representar os tipos de dados trocados nas mensagens, especificar as operações disponibilizadas e efetuar o mapeamento entre mensagens e seus respectivos protocolos de transporte [20].

O WSDL está limitado a descrever os detalhes da interface de um serviço, porém outro requisito é a definição de um padrão para representação das chamadas a esses serviços e dos dados envolvidos nessas chamadas. Tal necessidade é atendida pelo padrão SOAP, um protocolo extensível para troca de mensagens e que suporta uma variedade de protocolos de transporte. O W3C define o padrão SOAP como "um protocolo leve cujo objetivo é prover uma infra-estrutura que permita a troca de informação em ambientes distribuídos e descentralizados". Uma mensagem no formato SOAP é composta por três partes [11]: (1) a definição do conteúdo da mensagem e como ela deve ser processada, (2) as regras para representação dos dados sendo transmitidos na mensagem e (3) uma convenção para representação de chamadas de procedimentos remotos e suas respectivas respostas.

WSDL e SOAP garantem uma das principais características dos serviços Web, que é a independência de protocolo de comunicação, já que permitem que as mensagens trafeguem sobre praticamente qualquer protocolo. Esta independência torna possível a utilização do protocolo de transporte HTTP, o protocolo base da Internet. Assim, chamadas a procedimentos em componentes remotos distribuídos podem ser feitas a partir de e para qualquer local da rede, seja ela uma intranet ou mesmo a Internet.

O protocolo SOAP baseia-se no mecanismo de requisição e resposta, no qual um componente de software faz uma requisição para outro componente de software que pode ou não fornecer a resposta. SOAP é um tipo de extensão do HTTP que suporta transferência de

mensagens XML. O envio da mensagem XML pelo protocolo SOAP pode ser feito através de uma requisição HTTP e o recebimento da resposta também [12].

Embora os protocolos WSDL e SOAP sejam suficientes para construir e utilizar os serviços Web, sistemas desenvolvidos utilizando apenas esses dois protocolos acabam se tornando arquiteturas muito rígidas, onde serviços são estaticamente referenciados por suas URLs e os sistemas são dependentes de URIs de serviços específicos para o seu funcionamento. Para trazer maior flexibilidade a essa arquitetura e fornecer aos sistemas obtidos pela composição de serviços Web o dinamismo que a Internet possibilita, criou-se o padrão UDDI *(Universal Description Discovery and Integration)*, que funciona como um catálogo de serviços Web disponíveis. Este catálogo de serviços permite que serviços não sejam apenas referenciados estaticamente, mas também que novos serviços sejam encontrados dinamicamente através de uma pesquisa sobre as interfaces e os metadados dos serviços publicados [21].

A Figura 1 ilustra a arquitetura orientada a serviços Web e a interação entre os diversos padrões e componentes envolvidos na implementação, publicação, busca e invocação de serviços Web.

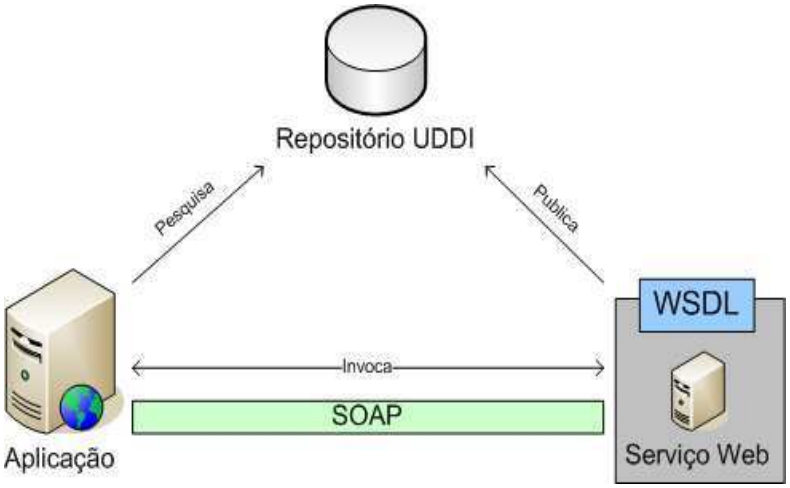


Figura 1. A arquitetura orientada a serviços Web.

Os serviços Web representam um grande avanço no sentido de facilitar a integração de sistemas distribuídos, desta forma é crescente a quantidade de sistemas que utilizam esta tecnologia e usufruem de suas vantagens no desenvolvimento dos diversos componentes de um sistema. À medida que os componentes de um sistema são implementados na forma de serviços Web, os sistemas passam a ser desenvolvidos através da integração de serviços. Tais sistemas possuem uma arquitetura baseada na reutilização de serviços, a Arquitetura

Orientada a Serviços (SOA em inglês). Como descrito em [10], "A Arquitetura Orientada a Serviços (SOA) é um paradigma para organização e utilização de entidades distribuídas que estão sob controle de diferentes domínios.". A arquitetura orientada a serviços não está diretamente relacionada e nem dependente dos serviços Web, outras tecnologias como COM, CORBA e RMI também permitem a implementação deste tipo de arquitetura, porém é com o uso de serviços Web que os benefícios da SOA são melhores alcançados.

Como podemos ver na Figura 2, o modelo SOA é baseado em entidades de software que são responsáveis pela publicação de informações sobre serviços (provedor), busca por serviços disponíveis (consumidor) e a interação entre consumidor (cliente) e provedor (serviço). A arquitetura orientada a serviços Web também baseia-se nesses três tipos de entidade e define protocolos para a implementação de cada um deles, sendo uma especialização da SOA para componentes na forma de serviços Web.

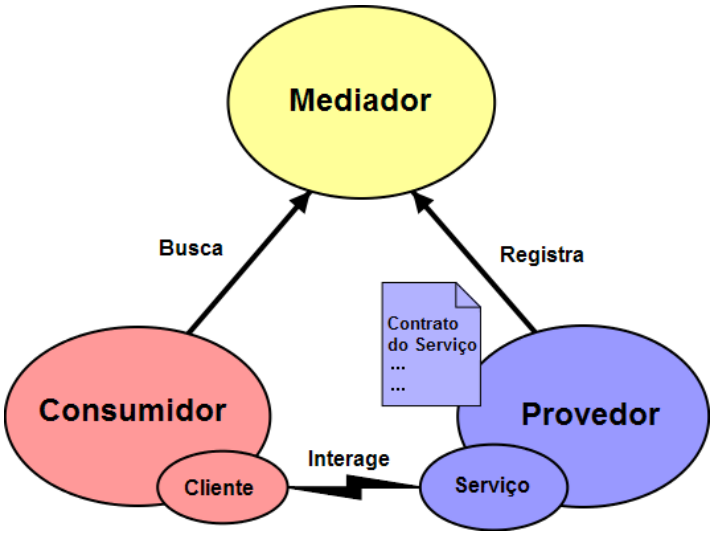


Figura 2. Arquitetura Orientada a Serviços

A arquitetura orientada a serviços traz equilíbrio a alguns dos principais requisitos presentes nos sistemas modernos [1]:

- *Distribuição:* Os sistemas de informação estão sendo utilizados em ambientes cada vez mais distribuídos, exigindo que componentes do sistema executem em máquinas fisicamente separadas.
- *Heterogeneidade:* Sistemas distribuídos tipicamente executam em ambientes heterogêneos e que muitas vezes não estão sob controle dos desenvolvedores.

- *Dinamismo:* Para atender aos ambientes de negócio cada vez mais dinâmicos, os sistemas precisam ser cada vez mais flexíveis para atender freqüentes mudanças.
- *Transparência:* Os sistemas devem ser transparentes aos detalhes da infra-estrutura de comunicação dos diferentes pontos onde executam os diversos componentes.
- *Orientação a processos:* Sistemas devem atender aos processos (*workflows*) presentes no ambiente em que são utilizados.

Uma arquitetura SOA permite que serviços sejam utilizados como componentes reutilizáveis de software, que em contraposição ao desenvolvimento de todas as partes de um sistema, é um fator que pode levar ao aumento da qualidade e da produtividade da atividade de desenvolvimento de software. Desta forma, podemos ver o uso da arquitetura SOA como algo que traz agilidade e qualidade ao desenvolvimento de sistemas distribuídos [9].

Existem diversos outros benefícios da utilização de uma arquitetura SOA, os mais notáveis são facilitar o gerenciamento do crescimento dos sistemas corporativos de larga escala e facilitar a utilização da escalabilidade da Internet pelo uso de serviços, reduzindo assim os custos das organizações para a cooperação entre organizações. O valor da SOA está em oferecer um paradigma escalável para organizar grandes sistemas em rede e que requerem interoperabilidade para atingir seus objetivos [10].

3. ORQUESTRAÇÃO E COREOGRAFIA DE SERVIÇOS

Uma arquitetura orientada a serviços, à medida que se desenvolve, traz consigo a necessidade de elementos responsáveis pela integração de serviços, intermediando a invocação dos serviços e coordenando a execução de processos. Serviços Web sem o auxílio de tais elementos são limitados a interações simples, já que o modelo de interação suportado pelo WSDL e pelo SOAP é essencialmente síncrono e sem manutenção de estado (*stateless*). O próprio protocolo HTTP, que é a escolha mais comum para se implementar serviços Web, é um protocolo orientado a requisições e respostas, cuja conexão tem o tempo de vida restrito a uma requisição e sua respectiva resposta, sem nenhuma relação com requisições anteriores ou subseqüentes.

O real potencial dos serviços Web e os benefícios da SOA somente podem ser alcançados com o auxílio de mecanismos de integração que tornem possíveis interações síncronas e assíncronas e com manutenção de estado (*stateful*) entre duas ou mais partes. Tais mecanismos são requisitos fundamentais para atender aos processos de negócio complexos presentes nos sistemas atuais [2].

Dois termos comumente utilizados na coordenação da execução de serviços são orquestração e coreografia. Embora esses termos possam parecer sinônimos à primeira vista (e muitos o tratam desta forma), na realidade são conceitos diferentes e que se complementam, sobrepondo-se apenas em alguns pontos [3].

Os termos orquestração e coreografia descrevem dois aspectos diferentes utilizados pelos padrões que estão emergindo para tratar da criação de processos de negócio a partir da integração de serviços Web. Orquestração se refere a um processo de negócio executável que pode interagir tanto com serviços internos quanto externos, sendo que essa interação é sempre representada sob a perspectiva de uma das partes, que é a que detém o controle do processo. A coreografia, por outro lado, é mais colaborativa, permitindo que cada parte envolvida descreva seu papel na interação.

A Figura 3 ilustra a diferença entre os conceitos de coreografia e orquestração. Enquanto a orquestração foca na interação entre serviços sob a perspectiva de uma das partes - o coordenador - a coreografia trata de questões como a ordenação de mensagens e a interação sob a perspectiva de todas as partes, sem a necessidade de um coordenador.

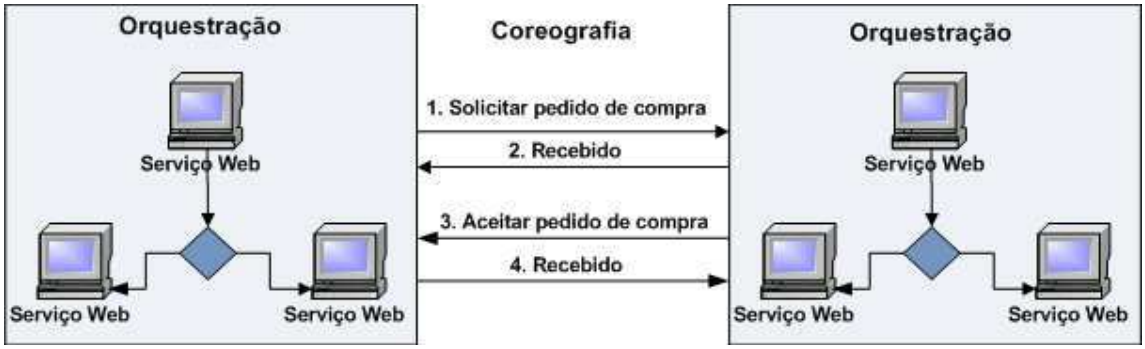


Figura 3. Diferença entre coreografia e orquestração.

A orquestração foca no comportamento sob o ponto de vista de um único participante, agindo como um controlador de um processo e conduzindo a execução desse processo seguindo sua definição. Uma linguagem de orquestração é utilizada para descrever como determinado processo deve ser executado, portanto tais linguagens são ditas executáveis.

Para exemplificar, consideremos o exemplo de um sistema de uma agência de viagens responsável pela compra de passagens aéreas, que além de fazer a aquisição de passagens, faz também uma pesquisa para encontrar a passagem com o menor preço. No exemplo da Figura 4 vemos o fluxo de execução deste sistema, que envolve a comunicação com duas empresas aéreas, a seleção da passagem com menor preço e por fim a compra da passagem, como uma orquestração coordenada pelo sistema da agência de viagens.

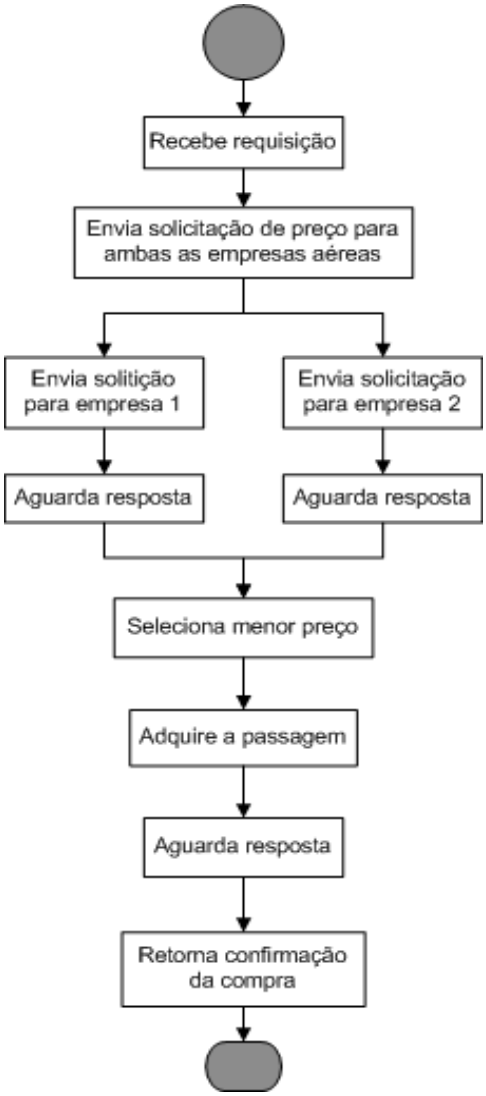


Figura 4. Exemplo de orquestração.

Como podemos ver, para se descrever a orquestração de um processo, uma linguagem de orquestração deve disponibilizar construções como controles de execução (condicionais, seqüências, paralelismo, laços de repetição), variáveis e construções para manipulá-las, tratadores de eventos, controles de tempo e controle e recuperação de exceções [13].

A coreografia descreve a colaboração dos serviços no sistema como um todo e independente de um elemento controlador atento a todas as partes envolvidas e detalhes da comunicação ponto-a-ponto entre as partes. Ao contrário das linguagens de orquestração, as linguagens de coreografia não possuem a finalidade de serem executadas, mas sim de descreverem as regras das interações entre diversos participantes. A comunicação é modelada através de conexões permanentes e com manutenção de estado.

Continuando com o exemplo da agência de viagens, a Figura 5 apresenta um exemplo de coreografia para o processo de pesquisa de preço e compra de passagens aéreas. Neste exemplo, o cliente interage com a agência de viagens e a agência de viagens com as empresas aéreas, da seguinte forma:

1. O viajante envia requisição de compra de passagem à agência de viagens.
2. A agência de viagens consulta o preço da passagem em duas diferentes empresas aéreas.
3. Cada empresa aérea responde com o preço solicitado.
4. A agência de viagem envia ao viajante a opção de vôo com o menor preço.
5. O viajante confirma a compra da passagem.
6. A agência confirma a compra da passagem com a empresa aérea vencedora da concorrência.
7. O viajante recebe o número do bilhete como resultado da confirmação da compra.

É importante notar que quando a agência de viagens recebe a confirmação da compra, ela não precisa passar novamente as informações de data, hora e trajeto para a empresa aérea, pois esta requisição é uma continuação da requisição anterior (consultarPreco). Este é um recurso muito importante da coreografia de serviços e chama-se correlação de mensagens (*correlation*). Outra observação importante é que as interações entre os participantes são numeradas, especificando a ordem de envio das mensagens.

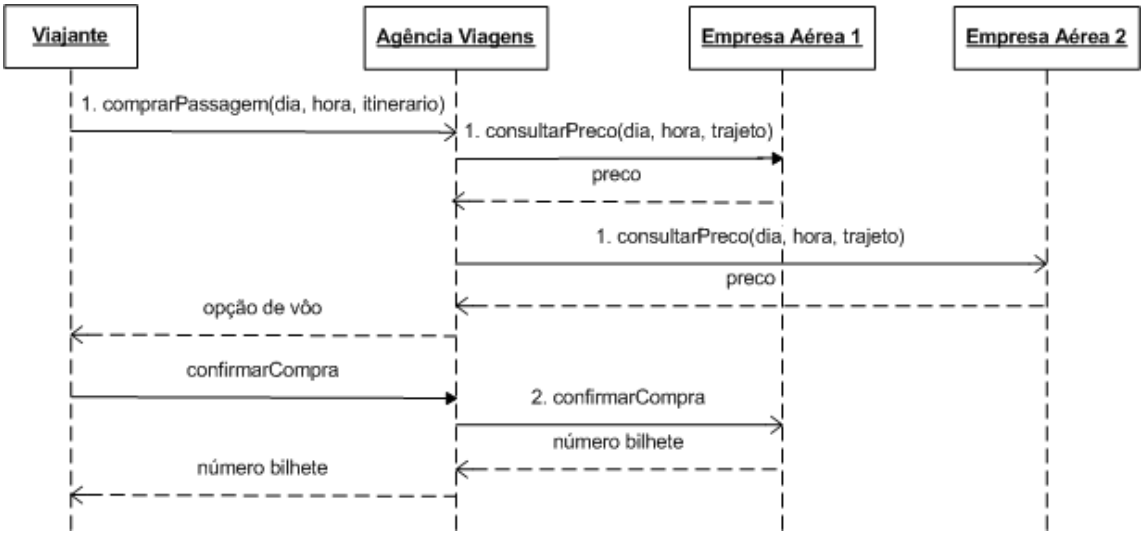


Figura 5. Exemplo de coreografia.

Podemos ver que a coreografia trata de detalhes da interação dos participantes do processo executado em colaboração, descrevendo e restringindo as mensagens que cada participante pode enviar e quais respostas são esperadas. Enquanto a descrição da interface de um serviço em WSDL define os aspectos estáticos do serviço (operações e mensagens), a coreografia define os aspectos dinâmicos da interação entre serviços.

A coreografia de serviços possui o objetivo de prover uma descrição das interações entre participantes e em nenhum momento trata das questões de implementação desta coreografia, função que deve ser desempenhada pelas linguagens de orquestração.

Uma definição de coreografia é de grande utilidade para todas as etapas do desenvolvimento de um sistema. É útil na etapa de projeto de um sistema, quando um participante pode utilizá-la para verificar se seus processos internos irão permitir que ele participe adequadamente da coreografia. Pode ser utilizada também para a geração automática das interfaces dos serviços. Para um sistema já em produção, uma definição de coreografia pode ser utilizada para se verificar se a interação entre as partes está ocorrendo como planejado. Outra grande utilidade está em permitir que determinado participante detecte exceções, como mensagens perdidas ou enviadas fora de ordem [14].

Um sistema pode ser desenvolvido com os conceitos de orquestração utilizando a estratégia de desenvolver os diversos serviços separadamente e então integrá-los, porém o comportamento final do sistema pode não ser o desejado quando os serviços não são projetados para trabalharem em conjunto desde o princípio. A coreografia auxilia neste processo criando uma visão global do sistema e das interações entre os diversos participantes, assim cada serviço pode ser desenvolvido ou adaptado consciente do papel que irá desempenhar [13].

4. PADRÕES DE ORQUESTRAÇÃO E COREOGRAFIA

4.1 PADRÕES PARA EXPRESSAR ORQUESTRAÇÃO

As primeiras iniciativas de se criar padrões que facilitassem a integração de serviços focaram na orquestração e partiram da IBM e da Microsoft, com a criação das linguagens WSFL e XLANG respectivamente. Posteriormente as duas empresas combinaram suas linguagens e com o apoio de empresas como BEA Systems, SAP e Siebel criaram a linguagem BPEL4WS, que foi submetida ao órgão de padronização OASIS e assim criou-se o padrão WS-BPEL (*Web Services Business Process Execution Language*) que atualmente está na versão 2.0.

A WS-BPEL é uma linguagem que possui muitas de suas construções semelhantes às de workflows, já que esta linguagem é focada na execução de processos seqüenciais e paralelos com múltiplos passos, tal como na teoria de workflows. O meta-modelo da WS-BPEL tem como elemento central o Processo (*Process*), onde se relacionam outros elementos como Atividade (*Activity*), Correlação (*Correlation*), Compensação (*Compensation*) e Parceiro (*Partner*). É na definição do elemento "Activity" que se define o fluxo de execução de um processo através de elementos como *Se* (*If*), *Enquanto* (*While*), *Escolha* (*Pick*), *Fluxo* (*Flow*) e *Seqüência* (*Sequence*) e o fluxo de mensagens através dos elementos *Receba* (*Receive*), *Responda* (*Reply*) e *Chame* (*Invoke*).

A WS-BPEL permite ainda a definição de dois tipos de processos, os abstratos e os executáveis. A especificação define que ambos os processos compartilham dos mesmos recursos da linguagem, porém os processos abstratos permitem que determinadas etapas sejam omitidas, assumindo assim um papel de descritor ou modelo de processo. [2]

Outra linguagem que surgiu para atender a orquestração de processos foi a BPML (*Business Process Management Language*), inicialmente especificada por um grupo que incluía empresas como Intalio e Sun. Sua última atualização foi em novembro de 2002 e então foi descontinuada em favor da WS-BPEL. As duas linguagens possuem construções semelhantes para o controle de fluxo de execução e o controle de fluxo de mensagens.

Embora as linguagens WS-BPEL e BPML atendam plenamente aos requisitos de orquestração de serviços Web, elas não tratam de forma completa e consistente a coreografia de serviços. Alguns consideram que os processos abstratos da WS-BPEL possibilitam a coreografia de serviços, porém tais processos não foram desenvolvidos com este objetivo e assim não atendem a todos os aspectos da coreografia. A própria especificação do WS-BPEL

sugere a adoção de outra linguagem para tratar da coreografia. A BPML, por sua vez, suporta a coreografia de serviços apenas quando utilizada em conjunto com outro padrão que é a WSCI.

As iniciativas de desenvolvimento de linguagens e padrões para orquestração de serviços foram as primeiras a emergir e a amadurecer, principalmente por tratarem dos aspectos mais práticos e essenciais para o desenvolvimento de sistemas com arquitetura SOA.

4.2 PADRÕES PARA EXPRESSAR COREOGRAFIA

Os padrões de coreografia começaram a surgir por iniciativas isoladas de algumas empresas ou consórcios, porém até hoje não existe um padrão abrangente e apoiado pela maioria do mercado. Entre as primeiras iniciativas voltadas especificamente para a coreografia de serviços Web está a WSCL (*Web Services Conversation Language*), da HP, que permite essencialmente a modelagem de conversações entre serviços Web e das interações entre as partes, onde os participantes evocam ou são evocados por outros participantes. [3] A WSCL é uma linguagem com sintaxe XML, onde as conversações são modeladas utilizando a idéia de máquinas de estado e as construções conversação (*conversation*) e interações (*interactions*) para definir uma coreografia entre serviços [15]. Esta especificação foi submetida à W3C em 2002 para que servisse como um documento de referência, porém seu excesso de simplicidade e a falta de uma solução para tratamento de exceções e de outros requisitos da coreografia fizeram com que ela fosse pouco lembrada [5].

Outra iniciativa importante partiu das empresas Sun, SAP, BEA e Intalio, que desenvolveram a WSCI (*Web Service Choreography Interface*), uma linguagem mais abrangente que a WSCL e que cobre praticamente todos os aspectos da coreografia e troca de mensagens entre serviços Web. A especificação suporta correlação de mensagens, regras de seqüência, tratamento de exceção, transações e colaboração dinâmica. A WSCI também foi submetida à W3C em 2002.

O workshop da W3C sobre serviços Web de abril de 2001 apontou a necessidade de uma interface comum e de uma linguagem de composição para auxiliar na coreografia de serviços Web. A partir daí criou-se em janeiro de 2003 o grupo de trabalho para especificar um padrão para a coreografia de serviços Web, o *Web Services Choreography Working Group*. Este grupo começou seus trabalhos a partir de outros documentos: o *Web Service Architecture* que define os componentes funcionais e as relações entre os componentes do modelo de serviços Web; a especificação da linguagem WSCI e principalmente a

especificação da linguagem WS-CDL (*Web Services Choreography Description Language*) que vinha sendo desenvolvida pela Oracle [13]. O grupo tem como objetivo a definição de uma linguagem que atenda aos requisitos de composição, associação, troca de mensagens e gerenciamento de estado. Os trabalhos do grupo continuam com o aperfeiçoamento da WS-CDL, que está caminhando para se tornar uma recomendação da W3C [6].

No intuito de compreendermos melhor os requisitos de uma linguagem de orquestração de serviços e de que forma são atendidos esses requisitos, iremos analisar com mais detalhes as duas linguagens mais completas e que são citadas pela literatura como referências, a WSCI e a WS-CDL.

4.2.1 WSCI

A primeira iniciativa de criação de uma linguagem que abrangesse os principais requisitos de coreografia de serviços Web surgiu com a WSCI (*Web Service Choreography Interface*). A WSCI descreve o comportamento observável de um serviço Web, ou seja, as funcionalidades do ponto de vista de um expectador externo ao serviço. Este comportamento é expresso em termos de dependências temporais e lógicas entre as mensagens trocadas, provendo regras de seqüência, correlação, tratamento de exceções e transações. Como dito anteriormente, a coreografia não trata da definição e implementação dos processos internos de um serviço, por ser uma linguagem especificamente para coreografia a WSCI também não trata destas questões.

A WSCI trabalha como uma extensão da WSDL e descreve como as operações de uma interface WSDL podem ser coreografadas em um contexto onde os serviços Web participam. Ela assume que todas as interações entre serviços seguem e implementam um determinado processo. A WSCI permite descrever também como a coreografia dessas operações deve tratar questões como correlação de mensagens, tratamento de exceções e controle de transações [7].

A WSCI procura resolver alguns dos principais desafios presentes nos sistemas mais complexos e tratar questões recorrentes como as seguintes.

1. “Mensagens podem ser enviadas e/ou recebidas em qualquer ordem?”
2. “Quais regras controlam a seqüência de mensagens?”
3. “Existe alguma relação entre quaisquer mensagens enviadas ou recebidas?”
4. “Uma seqüência de operações possui um início e um fim?”
5. “Pode um operação ser desfeita?”

6. “Pode-se desenhar uma visão global da troca de mensagens?”.

Para responder estas questões a WSCI possui as seguintes funcionalidades:

- **Coreografia de mensagens:** descreve a ordem que as mensagens podem ser enviadas ou recebidas em uma determinada troca de mensagens, as regras que governam esta ordenação e os limites de uma troca de mensagens (quando começa e quando termina).
- **Transações e compensações:** descreve quais operações são executadas de forma transacional e permite que sejam definidas ações a serem tomadas para se reverter operações caso uma transação seja desfeita.
- **Tratamento de exceções:** descreve como um serviço Web deve reagir quando uma condição excepcional ocorre e provê uma descrição de possíveis caminhos alternativos.
- **Gerenciamento de *thread*:** descreve quando e como um serviço Web é capaz de gerenciar múltiplas conversações com um ou vários participantes.
- **Propriedades e seletores:** propriedades são utilizadas para referenciar um valor, semelhante a uma variável nas linguagens de programação. Os seletores definem como o valor de uma propriedade é obtido a partir de uma mensagem recebida.
- **Conectores:** descrevem como as operações disponibilizadas pelos diferentes serviços Web interagindo em determinada troca de mensagens são conectadas. A WSCI permite o mapeamento entre operações, fazendo com que algumas operações quando invocadas gerem a chamada de outras operações em outros serviços Web.
- **Contexto operacional:** permite descrever diferentes comportamentos para um mesmo serviço Web quando interagindo em diferentes contextos.
- **Participação dinâmica:** descreve como determinado serviço Web será selecionado dinamicamente. A seleção é feita baseada em alguns critérios, como por exemplo, no conteúdo de mensagens recebidas.

A WSCI é uma linguagem com sintaxe XML cujo modelo possui elementos que tratam de cada um dos requisitos da coreografia de serviços. Os seus principais elementos são:

Elemento	Descrição
Interface (*Interface*)	Representa um cenário de interação de um serviço. A partir da interface são definidas as dependências lógicas e temporais entre mensagens. Um serviço pode ter mais de uma interface.
Atividade (*Activity*)	Uma atividade representa um comportamento de um serviço, podendo ser atômica ou complexa. Atividades atômicas representam ações como enviar ou receber uma mensagem ou esperar um determinado período de tempo. Atividades complexas são compostas de outras atividades, sendo que elas podem ser executadas de forma seqüencial, paralela, em laço de repetição (*loop*) ou de forma condicional.
Processo (*Process*)	Conjunto de ações que recebem um nome e que podem ser reutilizadas referenciado-se seu nome.
Propriedade (Property)	Equivalente a uma variável em uma linguagem de programação.
Contexto (*Context*)	Descreve o ambiente em que um conjunto de atividades é executado. Determina o contexto de utilização das propriedades, possíveis exceções e respectivos tratadores de exceção e as propriedades transacionais associadas às atividades.
Correlação (*Correlation*)	Suporta interações com manutenção de estado (*stateful*). Permite que chamadas a um ou mais serviços estejam inseridas em um contexto e que este contexto seja mantido.
Exceção (*Exception*)	Declaração de um comportamento excepcional. Pode ser causada pelo recebimento de uma mensagem considerada excepcional, ocorrência de uma falha ou por tempo excedido.
Transações e compensação (*Transaction and compensation activities*)	Garante que um conjunto de atividades será executado de forma atômica (tudo ou nada). No caso de uma transação ser desfeita, podem ser definidas atividades de recuperação, que desfaçam as ações executadas na transação.
Modelo global (*Global model*)	Assim como as interfaces modelam o comportamento observável de um serviço Web, o modelo global é responsável por conectar diferentes interfaces e descrever quais interações ocorrem entre os diversos serviços. Este elemento é o responsável por dar a visão do todo de uma coreografia.

Tabela 1. Elementos da linguagem WSCI

Os elementos chave da WSCI são a interface (*Interface*) e a visão global (*Global model*). É através desses dois elementos que (i) se descreve os comportamentos de diversos participantes de uma coreografia com uma visão isolada e específica de cada um desses participantes e (ii) define-se a coreografia com uma visão global, associando os diversos participantes e definindo suas regras de interação.

A descrição dos participantes é feita isoladamente através do elemento interface. Ela define uma forma como determinado serviço Web será utilizado. Seu objetivo é descrever apenas o comportamento público e observável de um participante e por si só não define uma orquestração [15].

A definição de uma orquestração é feita com o elemento visão global, através da combinação de interfaces WSCI e da definição das trocas de mensagens existentes entre dois ou mais participantes. A visão global consiste em um conjunto de conexões, cada uma conectando uma operação de um serviço Web a uma operação complementar de outro serviço Web [15].

Para exemplificar a utilização dos elementos interface e visão global em conjunto com os outros elementos da WSCI, vemos o exemplo abaixo, baseado no exemplo anterior de uma agência de viagens, onde existe a interação entre a agência e as empresas aéreas.

Como a WSCI é uma linguagem intimamente relacionada com a WSDL, para que uma interface WSCI possa ser melhor compreendida devemos também saber como a interface WSDL está declarada. Abaixo vemos a declaração da interface WSDL do serviço Web disponibilizado pela agência de viagens do nosso exemplo anterior, onde são disponibilizadas as operação para se efetuar uma compra de passagem (comprarPassagem) e de confirmação de compra (confirmarCompra).

```
1    <?xml version = "1.0" ?>
2    <definitions name = "AgenciaViagens"
3       targetNamespace = "http://ufsc.br/emp_aerea"
4          xmlns = "http://www.w3.org/2002/07/wsci10"
5          xmlns:wsdl = "http://schemas.xmlsoap.org/wsdl/"
6          xmlns:tns = "http://ufsc.br/agenciaviagens"
7          xmlns:defs = "http://ufsc.br/agenciaviagens/definicoes">
8
9       <portType name = "AgenciaViagens">
10          <operation name = "comprarPassagem">
11             <input message = "defs:dadosVoo"/>
12             <output message = "defs:opcaoVoo"/>
13          </operation>
14          <operation name = "confirmarCompra">
15             <output message = "defs:dadosBilhete"/>
16             <fault message = "defs:erro"/>
17          </operation>
18       </portType>
19    </definitions>
```

Figura 6. Interface WSDL agência de viagens.

Como podemos perceber, uma interface WSDL apresenta uma descrição estática de um serviço Web, deixando muitas questões a serem esclarecidas para que um cliente deste

serviço possa utilizá-lo da forma correta. Entre os aspectos deste serviço que precisam ser esclarecidos estão [7]:

- **Coreografia:** A interface WSDL não descreve a ordem correta de chamada das operações. Neste exemplo, não está definido claramente que a operação de confirmação de compra de uma passagem deve ser executada somente após uma requisição de compra.
- **Correlação:** Não está descrito na interface WSDL que uma mensagem de confirmação de compra está relacionada com uma requisição de compra que a procedeu.
- **Transações:** A interface WSDL não descreve se as operações são executadas de forma transacional, o que seria muito importante para o comprador de uma passagem, que saberia que a compra seria desfeita em caso de algum erro durante o processo.

Depois de compreendidas as operações disponibilizadas pelo serviço Web da agência de viagens e descritos os aspectos que a coreografia deste serviço deve tratar, podemos partir para a definição da interface WSCI. Com ela definimos a seqüência lógica de invocações e mensagens, correlações e tratamento de exceções para cada uma das operações de uma interface WSDL. Abaixo temos a definição da interface WSCI para o serviço Web da agência de viagens.

```
1.    <? xml version = "1.0" ?>
2.    <wsdl:definitions name = "AgenciaViagens"
3.        targetNamespace = "http://ufsc.br/AgenciaViagens"
4.        xmlns:wsdl = "http://schemas.xmlsoap.org/wsdl/"
5.        xmlns:xsd = "http://www.w3.org/2000/10/XMLSchema"
6.        xmlns:tns = "http://ufsc.br/AgenciaViagens"
7.        xmlns = "http://www.w3.org/2002/07/wsci10">
8.
9.    <correlation name = "correlacaoDadosVoo"
10.            property = "tns:identificadorVoo">
11.   </correlation>
12.
13.   <interface name = "AgenciaViagens">
14.   <process name = "ComprarPassagem
15.          instantiation = "message">
16.
17.       <sequence>
18.           <action name = "ComprarPassagem"
19.               role = "tns:AgenciaViagens"
20.               operation = "tns:AgenciaViagens/comprarPassagem">
21.           </action>
22.
23.           <action name = "ConfirmarCompra"
```

```
24.              role = "tns:AgenciaViagens"
25.              operation = "tns:AgenciaViagens/confirmarCompra">
26.              <correlate correlation="tns:correlacaoDadosVoo"/>
27.
28.              <call process = "tns:PesquisarPreco" />
29.           </action>
30.        </sequence>
31.    </process>
32.
33.    <process name = "PesquisarPreco" instantiation = "other">
34.        <action name = "pesquisarPreco"
35.            role = "tns:AgenciaViagens"
36.            operation = "tns:EmpresaAerea/consultarPreco">
37.        </action>
38.    </process>
39.    </interface>
40.    </wsdl:definitions>
```

Figura 7. Interface WSCI agência de viagens.

Embora não seja o objetivo fazer uma descrição detalhada da linguagem WSCI, existem algumas observações importantes da especificação desta interface:

- Na linha 9 vemos a declaração de uma correlação, descrevendo que mensagens contendo informações de vôo devem ser correlacionadas com requisições seguintes.
- Na linha 17 utiliza-se o elemento seqüência (*sequence*) para descrever que as ações devem ser executadas de forma seqüencial.
- As linhas 18 a 21 descrevem uma ação (*action*), desempenhada pela agência de viagens, e a relaciona com a operação comprarPassagem da interface WSDL. Com esta relação com a interface do serviço pode-se determinar as mensagens que são trocadas.
- A linha 26 descreve que evocações da operação confirmarCompra estão correlacionadas com outras operações, neste caso com comprarPassagem.
- Na linha 28 faz-se a chamada do processo PesquisarPreco. O uso de chamadas de processos permite o reuso de definições de processos.

As interfaces WSDL e WSCI do viajante da empresa aérea podem ser definidas de forma análoga à interface da agência de turismo descrita acima. Após definir todas as interfaces isoladamente, o próximo passo é criar a visão global do processo. É neste ponto que se cria a coreografia propriamente dita, onde se conectam os vários participantes e define-se o fluxo de mensagens. No código abaixo temos a definição da visão global, onde se integram as interfaces WSCI do viajante (linha 10), da agência de viagens (linha 12) e da

empresa aérea (linha 14). Em seguida definem-se as conexões entre as operações da interface do viajante com a interface da agência de viagens (linhas 25 e 28) e das operações da interface da agência de viagens com a interface da empresa aérea (linhas 33 e 36).

```
1.    <?xml version = "1.0" ?>
2.    <wsdl:definitions   name = "GlobalModel"
3.        targetNamespace = "http://ufsc.br/models"
4.        xmlns = "http://www.w3.org/2002/07/wsci10"
5.        xmlns:wsdl = "http://schemas.xmlsoap.org/wsdl/"
6.        xmlns:tra = "http://ufsc.br/Viajante"
7.        xmlns:air = "http://ufsc.br/AgenciaViagens"
8.        xmlns:ta = "http://ufsc.br/EmpresaAerea">
9.
10.   <wsdl:import namespace = "http:/ ufsc.br/Viajante"
11.                location = "http://ufsc.br/Viajante.wsci" />
12.   <wsdl:import namespace = "http://ufsc.br/AgenciaViagens"
13.                location = "http://ufsc.br/AgenciaViagens.wsci" />
14.   <wsdl:import namespace = "http://ufsc.br/EmpresaAerea"
15.                location = "http://ufsc.br/EmpresaAerea.wsci" />
16.
17.   <model name = "CompraDePassagem">
18.
19.       <interface ref = "air:EmpresaAerea" />
20.       <interface ref = "via:Viajante" />
21.       <interface ref = "agv:AgenciaViagens" />
22.
23.       <!-- Conexões ente Viajante e Agência de Viagens -->
24.
25.       <connect operations = "via:Viajante/ComprarPassagem
26.                              agv:AgenciaViagens/ComprarPassagem" />
27.
28.       <connect operations = "via:Viajante/ConfirmarCompra
29.                              agv:AgenciaViagens/ConfirmarCompra" />
30.
31.       <!-- Conexões ente Agência de Viagens e Empresa Aérea -->
32.
33.       <connect operations = "agv:AgenciaViagens/ComprarPassagem
34.                              air:EmpresaAerea/PesquisarPreco"/>
35.
36.        <connect operations = "agv:AgenciaViagens/ConfirmarCompra
37.                              air:EmpresaAerea/ConfirmarCompra"/>
38.
39.   </model>
40.   </wsdl:definitions>
```

Figura 8. Visão global WSCI agência de viagens.

Como podemos ver, a definição de uma coreografia com a linguagem WSCI é um tanto custosa se considerarmos a quantidade de artefatos que precisamos criar: uma interface para cada participante e mais a visão global. Certamente a WSCI seria mais acessível se permitisse mesclar a definição comportamental de uma interface e também a visão global em um mesmo artefato, semelhante a como é feito na linguagem WS-CDL [15].

A WSCI foi a primeira linguagem a suportar a coreografia de serviços de forma consistente, trazendo uma solução abrangente e padronizada. Em seu passado a WSCI foi muito importante por motivar uma discussão em torno do tema coreografia. No entanto e embora até hoje a WSCI tenha grande reconhecimento e sirva como uma referência no assunto, atualmente não se dá mais continuidade ao seu desenvolvimento e os atuais esforços para desenvolvimento de um padrão para coreografia estão focados na linguagem WS-CDL [7] [15].

4.2.2 WS-CDL

Assim como a WSCI, WS-CDL (*Web Services Choreography Description Language*) é uma linguagem com sintaxe XML e que tem por objetivo permitir a descrição das interações entre múltiplos participantes através da definição, sobre uma visão global, de seus comportamentos públicos e observáveis, onde mensagens trocadas ordenadamente resultam na execução de determinada atividade. Com a WS-CDL define-se uma contrato global contendo a definição das regras de ordenação e restrições envolvidas na troca de mensagens. Cada participante pode se basear nesse contrato para implementar e testar soluções que atendam e se integrem adequadamente aos outros participantes [16].

A WS-CDL é uma linguagem que está sendo desenvolvida pelo *Web Services Choreography Working Group*, um grupo de trabalho da W3C que conta com o apoio de empresas como Oracle, Novell e Adobe. Seu principal objetivo está em especificar uma linguagem declarativa, que defina a partir de um ponto de vista global o comportamento observável e comum a um grupo de participantes que cooperam para a execução de uma atividade. A linguagem deve especificar especialmente as trocas de informações e as regras de ordenação que devem ser satisfeitas [16]. Outros objetivos da WS-CDL são:

- **Reusabilidade:** Uma mesma definição de coreografia precisa ser reuzável por diferentes participantes operando em diferentes contextos.
- **Cooperação:** Coreografias devem definir a seqüência de troca de mensagens entre dois ou mais participantes ou processos, descrevendo como eles devem cooperar.
- **Múltipla colaboração:** Coreografias podem ser definidas envolvendo qualquer número de participantes e processos.

- **Semântica:** Coreografias devem ter descrições semânticas para todos os seus componentes.
- **Composição:** Coreografias podem ser combinadas para formarem uma nova coreografia, podendo ser reutilizadas em diferentes contextos.
- **Modularidade:** Coreografias podem ser definidas a partir da inclusão de outras coreografias.
- **Colaboração orientada a informações:** Coreografias descrevem como os participantes avançam na execução de um processo através das informações trocadas.
- **Sincronização de informações:** Coreografias permitem que seus participantes compartilhem informações.
- **Tratamento de exceções:** Coreografias podem definir como tratar casos de exceção que eventualmente ocorram durante a execução da coreografia.
- **Transações:** Permite que coreografias trabalhem de forma transacional e que se controle a execução de colaborações de longa duração.
- **Composição de padrões:** A especificação WSDL permite que se trabalhe em conjunto e de forma complementar com outras especificações como WS-Reliability, WSCAF, WS-Security, WS-BPEL, etc.

Para que participantes se comuniquem de forma colaborativa, serviços devem cumprir suas responsabilidades e seguir as regras de relacionamento definidas em comum acordo a todos os participantes. A colaboração é definida através de um conjunto de regras de restrição e de ordenação. Estas construções são definidas segundo o modelo da WS-CDL, que engloba as seguintes entidades:

Entidade	Descrição
Papel (*RoleType*)	Descreve o comportamento observável de um participante que irá interagir com outros participantes.
Relacionamento (*RelationshipType*)	A conexão de dois papéis (*RoleType*), onde um compromisso é firmado entre as partes para que possam cooperar com sucesso.
Participante (*ParticipantType*)	Um tipo específico de entidade, por exemplo uma organização, que desempenha um ou mais papéis.

Canal (*ChannelType*)	Um ponto de colaboração entre participantes, especificando onde e como a informação é trocada.
Coreografia (*Choreography*)	Consiste em variáveis, atividades, tratamento de exceções, finalizadores de transações e outras coreografias.
Atividades em seqüência (*Sequence*), paralelas (*Parallel*) e condicionais (*Choice*)	Estruturas típicas para suportar atividades executadas de forma seqüencial, paralela ou de forma condicional.
Interação (*Interaction*)	Uma atividade que representa uma troca de informações entre papéis. Uma interação pode ter um limite de duração, assim como outros requisitos que garantem que os participantes possuem uma compreensão comum do comportamento complementar resultante da interação.
Unidade de trabalho (*WorkUnit*)	Uma atividade que somente é executada quando determinada condição é verdadeira, por exemplo, uma condição pode ser configurada para aguardar que todos os dados estejam disponíveis.
Execução de atividade (*Perform*)	Uma atividade composta através da reutilização de coreografias, permitindo que uma atividade seja definida referenciando-se coreografias definidas separadamente.
Semântica (*Semantics*)	Permite a criação de descrições semânticas dos participantes.

Tabela 2. Elementos da linguagem WS-CDL.

O processo de definição de uma coreografia utilizando a linguagem WS-CDL é feito basicamente seguindo um conjunto de definições. Cada definição é um elemento de construção que recebe um nome único que o diferencia dos demais. Em seguida veremos a forma geral que assume uma definição de coreografia utilizando a WS-CDL. Um exemplo completo integrando todos os conceitos e utilizando alguns dos principais recursos da linguagem será visto mais adiante na seção de estudo de caso.

A visão conceitual de alto nível da estrutura de uma descrição de coreografia utilizando a linguagem WS-CDL pode ser vista na figura 6 [19].

Uma descrição de uma coreografia com a WS-CDL é contida em um pacote (*package*) que é essencialmente um agrupador para um conjunto de atividades que podem ser

desempenhadas por um ou mais participantes. A linguagem dispõe de três tipos de atividades, as atividades para controle do fluxo de execução, as atividades do tipo unidade de trabalho (*workunit*) e as atividades básicas. Um atividade pode incluir diversas sub-atividades.

Os tipos de atividades para controle do fluxo de execução são seqüenciais (*sequence*), paralelas (*parallel*) e as de escolha (*choice*). Uma atividade de seqüência descreve uma ou mais atividades que são executadas de forma seqüencial, enquanto uma atividade paralela descreve uma ou mais atividades que podem ser executadas independentemente da ordem ou ao mesmo tempo. A atividade de escolha descreve a execução de uma atividade escolhida entre um conjunto de opções, sendo que esta escolha pode ser feita com base em uma expressão lógica ou a partir da ocorrência de determinado evento.

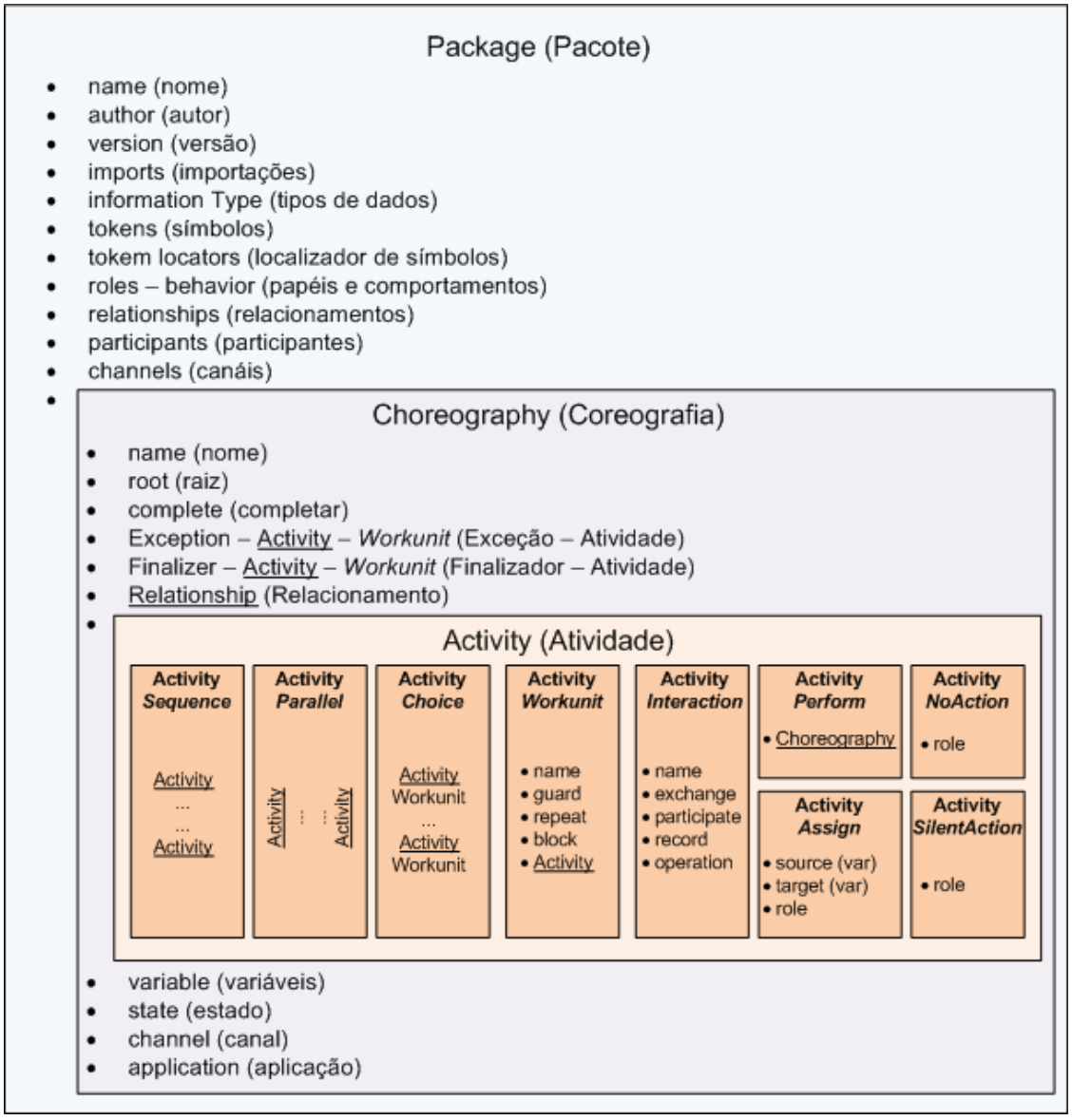


Figura 9. Modelo conceitual da WS-CDL.

O segundo tipo de atividade é chamado de unidade de trabalho (*workunit*). Uma unidade de trabalho descreve a execução condicional de uma atividade ou a execução repetida de uma atividade.

As atividades seqüências, paralelas, de escolha e as unidades de trabalho permitem a descrição básica de um controle de fluxo, semelhante ao que encontramos nas linguagens de programação. Existe semelhança também com as construções para controle de fluxo disponibilizadas pela linguagem WS-BPEL [19].

O terceiro tipo de atividade, as atividades básicas, incluem a interação (*interaction*), a atividade nula (*NoAction*), a atividade silenciosa (*SilentAction*), a atividade de atribuição (*assign*) e a atividade de execução (*perform*). As atividades nulas e silenciosas descrevem pontos de uma coreografia onde algum participante não desempenha nenhuma atividade ou desempenha alguma atividade transparente para os outros participantes, respectivamente. A atividade de atribuição é utilizada para transferir o valor de uma variável para outra. A atividade de execução é utilizada para a chamada de outra coreografia, que pode estar definida no mesmo pacote ou em um pacote importado [16].

Um dos elementos mais importantes da WS-CDL é sem dúvida a atividade de interação (*interaction*). Uma interação descreve uma troca de mensagens entre participantes, sobre a ponto de vista do receptor da mensagem. O propósito de uma interação pode ser efetuar uma requisição, fornecer uma resposta ou efetuar uma requisição, que requer uma resposta. Uma atividade de interação descreve os participantes envolvidos, a informação que será trocada e o canal utilizado para transmitir a mensagem. A informação enviada ou recebida em uma interação é descrita por uma variável (*variable*) [19].

Na linguagem WS-CDL as variáveis são utilizadas para armazenar informações relativas ao domínio da aplicação, informações de estado da coreografia, ou informações sobre os canais de interação. As variáveis possuem um tipo associado a elas (*informationType*) e são acessadas utilizando expressões XPath.

A ligação entre coreografias WS-CDL e operações descritas em interfaces WSDL é feita através de um canal (*channel*). Assim todo comportamento exposto em uma coreografia possui um canal. Um canal possui uma referência (*reference token*) que descreve como contatar o receptor das mensagens do canal e também uma identidade (*identity token*) que é utilizada para se fazer a correlação das mensagens trocadas através do canal.

Os símbolos (*tokens*) da linguagem WS-CDL são nomes utilizados para referenciar pedaços específicos do conteúdo de uma variável. Os localizadores de símbolos (*token*

locators) descrevem como os tokens podem ser encontrados utilizando expressões XPath [16].

A linguagem WS-CDL utiliza o elemento coreografia (*choreography*) para agrupar a atividade raiz e as descrições opcionais dos elementos exceção (*exception*) e finalizador (*finalizer*). O elemento exceção pode ser ativado caso alguma exceção ocorra durante a execução da coreografia, isto permite que se executem atividades que revertam as ações que foram iniciadas pela coreografia, mas foram interrompidas. O elemento finalizador pode ser ativado quando uma coreografia é finalizada com sucesso, permitindo que se confirmem as atividades executadas (*commit*), ou ainda que se desfaçam as atividades executadas devido a alguma falha durante a execução de outra coreografia (*rollback*).

Um pacote (*package*) descreve quais papéis (*RoleTypes*) estão presentes em uma coreografia e os comportamentos por ele desempenhados (*Behavior*). Descrevem-se ainda quais comportamentos serão exibidos durante a interação de dois papéis (*RelationshipType*). Finalmente, um pacote pode conter a descrição de agrupamentos lógicos de papéis (*ParticipantType*), permitindo que uma organização em particular possa desempenhar mais de um papel em uma coreografia [16].

O estudo e o desenvolvimento de padrões para coreografia de serviço é um tema relativamente novo e que ainda deixa muitas questões em aberto para discussões. A linguagem WS-CDL, por ser uma das últimas iniciativas de desenvolvimento de um padrão e apesar de herdar todo o conhecimento e estudos feitos anteriormente em linguagens como WSCI, ainda apresenta muitos pontos em aberto e outros mal resolvidos.

Alguns pontos fracos da linguagem que merecem destaque são [19]:

- **Falta de uma base formal**. Apesar de se adotar a terminologia *pi-Calculus*, a ligação com este formalismo não é muito clara. A falta de formalismo dificulta o desenvolvimento de ferramentas para validação de coreografias que garantam a interoperabilidade entre serviços Web.
- **Falta de um mapeamento claro entre a WS-CDL e a WSDL**. Ao mesmo tempo em que deixa muitos detalhes da ligação das duas em aberto, a WS-CDL restringe uma coreografia e uma interface WSDL em específico. Não se podem utilizar serviços que sejam equivalentes funcionalmente, já que a ligação é feita pelo nome da operação de uma interface WSDL. Isso impede que em tempo de execução seja feita a seleção dos participantes com base em suas capacidades funcionais, restringindo a apenas aqueles que implementam uma interface WSDL específica.

- **Somente coreografias entre dois participantes**. A WSDL não permite definir coreografias onde exista a interação simultânea entre mais de dois participantes, ou seja, apenas são permitidas interações binárias.
- **Impossibilidade de se tratar cenários com múltiplas instâncias de interações de um mesmo tipo.** Por exemplo, no contexto da agência de viagens, onde o processo de pesquisa de preços exige a comunicação simultânea com mais de uma empresa aérea.

Outro ponto desfavorável da WS-CDL é a falta de apoio por parte de grandes empresas do mercado. A linguagem foi concebida pela Oracle, que tem contribuído para o seu desenvolvimento até hoje, recebendo apoio de empresas como Sun, Adobe, Nortel e Novell. Porém, a falta de apoio de empresas como Microsoft e IBM, que são muito influentes na área, torna seu futuro um tanto duvidoso [15].

Apesar das críticas, a WS-CDL é sem dúvida o padrão mais maduro e desenvolvido atualmente para a descrição de coreografias. O fato de haver um grupo de trabalho da W3C empenhado em dar continuidade ao seu desenvolvimento com certeza fará a linguagem evoluir no sentido de resolver muitas de suas atuais falhas. Adiante vemos um exemplo de ferramenta que torna a linguagem mais acessível ao público.

5. AMBIENTE EXPERIMENTAL E ESTUDO DE CASO

5.1 ANÁLISE DA FERRAMENTA PI4SOA

Descrever coreografias utilizando a WS-CDL não é algo trivial sem o auxílio de ferramentas. Exigir que o usuário da linguagem saiba utilizar todas as suas construções, possíveis atributos e parâmetros, com certeza deixará o usuário muito suscetível a erros e será algo proibitivo em relação a produtividade. O fato da descrição da coreografia ser feita na forma de um documento XML torna a sua edição ainda mais susceptível a erros e dificulta a visão da coreografia. A visualização de forma esquemática é mais apropriada para permitir a interação e a compreensão por parte dos usuários.

Para que a WS-CDL tenha uma larga adoção e se torne viável em termos de produtividade, é essencial o uso de uma ferramenta que torne transparente ao usuário os detalhes da linguagem e que permita ao usuário focar na visão de alto nível da comunicação entre serviços, objetivo principal da coreografia de serviços. Tal ferramenta deve permitir não só a geração automática da WS-CDL, mas também propiciar uma visão gráfica de uma coreografia, tornando a linguagem acessível também aos usuários não técnicos, como analistas de negócio que contribuem para a definição dos processos.

Pensando nisso a empresa *Pi4 Technologies Foundation* criou a ferramenta Pi4SOA, cujo objetivo principal é prover uma ferramenta que auxilie no desenvolvimento de sistemas distribuídos e orientados a serviço (SOA). Seus idealizadores partem do princípio que um sistema com uma boa arquitetura pode ser desenvolvido com visão *top-down*, utilizando a linguagem WS-CDL, que pode agir como um modelo para o desenvolvimento de serviços ou para a integração de aplicações legadas. Esta metodologia de desenvolvimento garante que todos os serviços se adaptam ao comportamento esperado de uma coreografia e garante a interoperabilidade destes serviços para alcançar um objetivo [17].

A iniciativa para o desenvolvimento da ferramenta partiu de um dos líderes da especificação da WS-CDL, Steve Ross-Talbot, que acredita na WS-CDL como um grande auxilio para o desenvolvimento aplicações distribuídas e orientadas a serviços. Segundo ele, um dos problemas no desenvolvimento deste tipo de sistema é que as pessoas constroem serviços com uma visão individualista, focando apenas na funcionalidade de determinado serviço e não com uma visão global e mais abrangente do processo, onde diversos serviços interagem. Ainda segundo ele, esses problemas fazem com que sistemas desenvolvidos com

uma arquitetura orientada a serviços nem sempre atendam aos requisitos do negócio, e certamente a WS-CDL traz grandes melhorias neste aspecto [18].

Embora a ferramenta esteja em processo de desenvolvimento e o padrão WS-CDL não esteja oficialmente finalizado, já existem diversos recursos disponíveis e que permitem a definição completa de uma coreografia e a geração da interface WSDL, WS-BPEL ou Java. Os principais recursos disponíveis são:

Recurso	Descrição
Importação de coreografia	Permite que uma interface WS-CDL já definida seja importada para a ferramenta.
Exportação de coreografia	Permite que uma coreografia seja exportada para WS-CDL, relatório HTML, para *Service Description Model* ou para um diagrama UML.
Validação de coreografia	Regras padronizadas para garantir a consistência do modelo e para reforçar a semântica da WS-CDL.
Geração de interface de serviço	Geração da interface dos serviços para WSDL 1.1 ou WSDL 2.
Geração de serviço	Geração da implementação dos serviços para a linguagem Java ou para WS-BPEL.
Publicação de serviço	Publicação de serviço nos servidores Apache Axis, J2EE e processos WS-BPEL no servidor ActiveBPEL.
Modelagem de serviços	Ferramenta gráfica para modelagem de serviços.
Validação de serviços	Ferramenta para validação de serviços.
Java Runtime	Monitora a execução de coreografias para identificar mensagens trocadas fora de ordem.

Tabela 3. Recursos da ferramenta PI4SOA.

O desenvolvimento de uma coreografia utilizando a ferramenta exige que o usuário conheça os principais conceitos da linguagem WS-CDL. A própria documentação da ferramenta, que é bastante pobre, sugere que se use a especificação da WS-CDL como documentação.

As etapas para criação de uma coreografia iniciam com a criação de um novo projeto e de uma nova descrição de coreografia (*Choreography Description*), como pode ser visto na Figura 7.

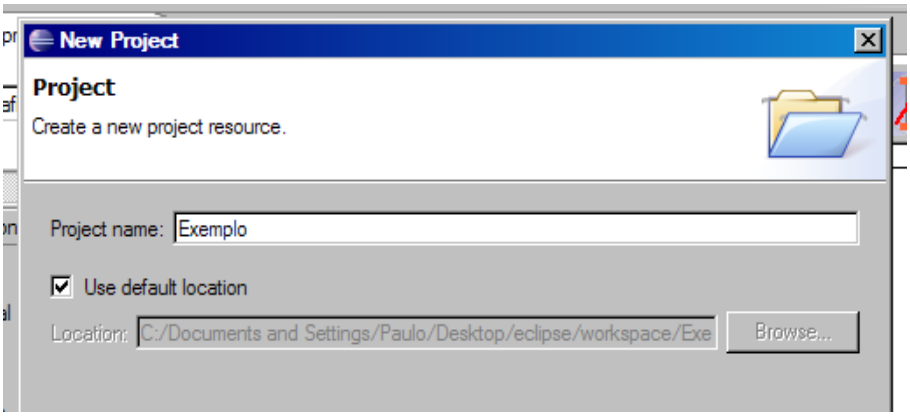


Figura 10. Criação de novo projeto.

Em seguida define-se os papéis (*RoleType*), os comportamentos (*Behavior*) e os relacionamentos (*RelationshipType*), utilizando a aba de papéis e relacionamento (**Roles and Relationships**), como pode ser visto na Figura 8. Podemos ver também na Figura 8 a janela de problemas (*Problems*) que exibe erros de validação da coreografia.

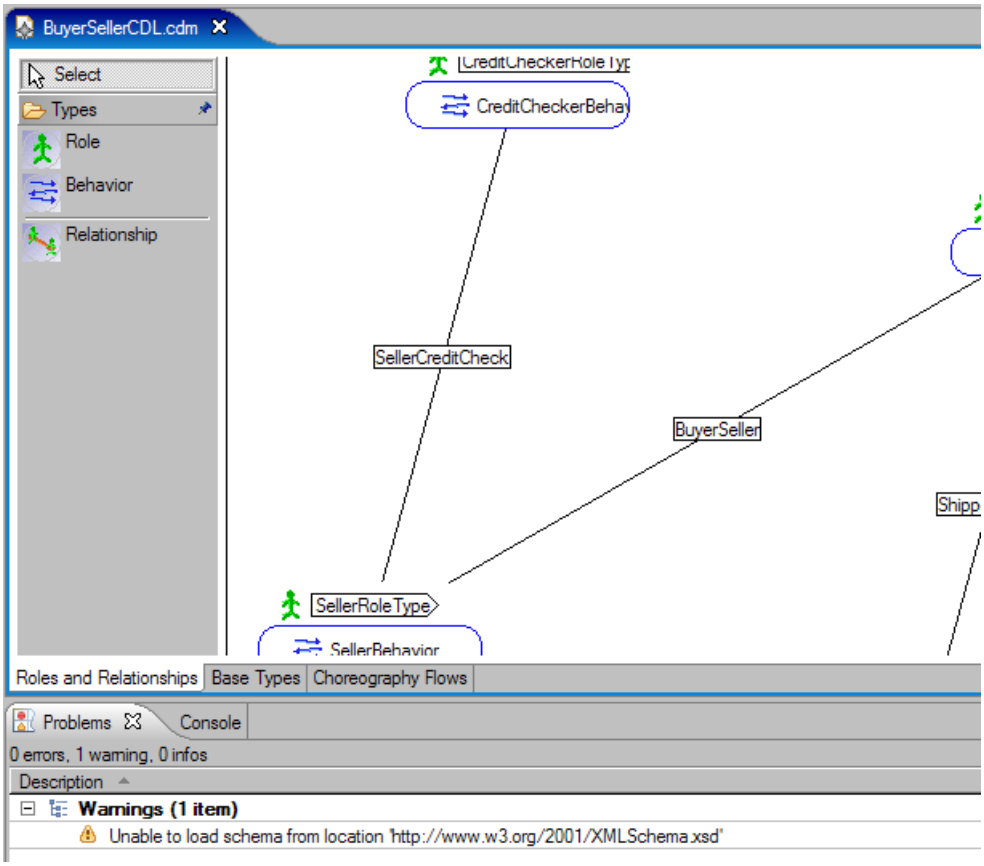


Figura 11. Papéis e Relacionamentos / Janela de problemas.

A ferramenta provê uma visão geral dos elementos base da coreografia através da aba de tipos básicos (*Base Types*), como pode ser visto na Figura 9. Deve-se utilizar esta aba para criar os elementos *Token, TokenLocator, InformationType, Participant* e *Channel*.

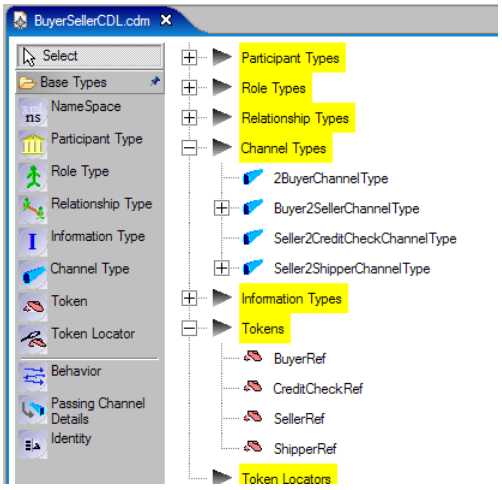


Figura 12. Tipos básicos da coreografia.

Por último, na aba fluxo da coreografia (*Choreography Flows*), inicia-se a descrição da coreografia propriamente dita, onde se definem as interações, fluxos seqüenciais e paralelos, atividades de escolha e de iteração, etc. A aba de definição das interações pode ser visto na Figura 10.

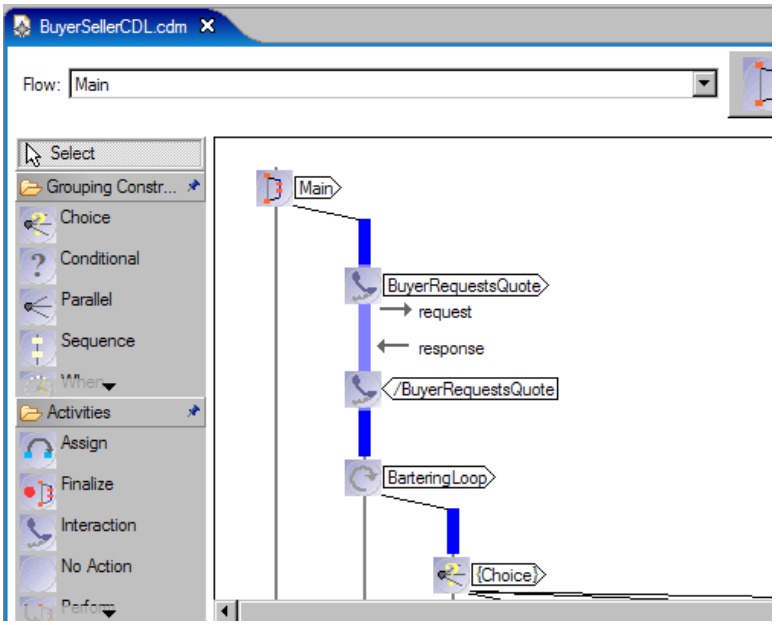


Figura 13. Aba de fluxos da coreografia.

5.2 ESTUDO DE CASO

Para compreendermos melhor o conceito de coreografia de serviços e para que façamos uma análise do atual desenvolvimento dos padrões e ferramentas para coreografia, é essencial que se apliquem os conceitos e práticas no desenvolvimento de um estudo de caso que reflita um caso real de aplicação.

Muitos são os contextos onde se podem aplicar os padrões e ferramentas para a descrição de uma coreografia, porém a escolha de um contexto para desenvolvimento de um estudo de caso deve permitir que o foco esteja na análise do que se está estudando. Um contexto com excesso de complexidade acabaria ofuscado o objetivo real do estudo de caso.

A possibilidade de desenvolvimento de um estudo de caso que permitisse não só a análise de um padrão de coreografia, mas também uma comparação com outros padrões,

motivou a escolha do contexto de uma agência de viagens e as interações existentes com um viajante interessado na compra de uma viagem e com uma empresa aérea responsável pela venda de passagens. Contexto semelhante é utilizado como estudo de caso na especificação da linguagem WSCI [7], o que permite uma comparação com o estudo de caso que será desenvolvido adiante com a linguagem WS-CDL.

Embora este contexto já venha sendo utilizado de forma simplificada desde o início do trabalho, adiante detalharemos as interações existentes entre os participantes, de forma que possamos demonstrar uma quantidade maior dos recursos da linguagem WS-CDL.

O estudo de caso serve também para analisar a ferramenta PI4SOA, descrita anteriormente. Portanto, o desenvolvimento da coreografia utilizou os recursos dessa ferramenta.

O contexto do estudo de caso inclui três participantes: um viajante, uma agência de viagens e uma empresa aérea. O objetivo da interação entre estes participantes é permitir que um viajante faça uma solicitação de compra das passagens para uma viagem (composta por um ou mais trajetos) à agência de viagens, receba uma proposta baseada nos custos repassados pela empresa aérea à agência de viagens e em seguida o viajante tem as opções de reservar, confirmar ou cancelar a viagem. A Figura 11 mostra as interações existentes entre o viajante, a agência de viagens e a empresa aérea.

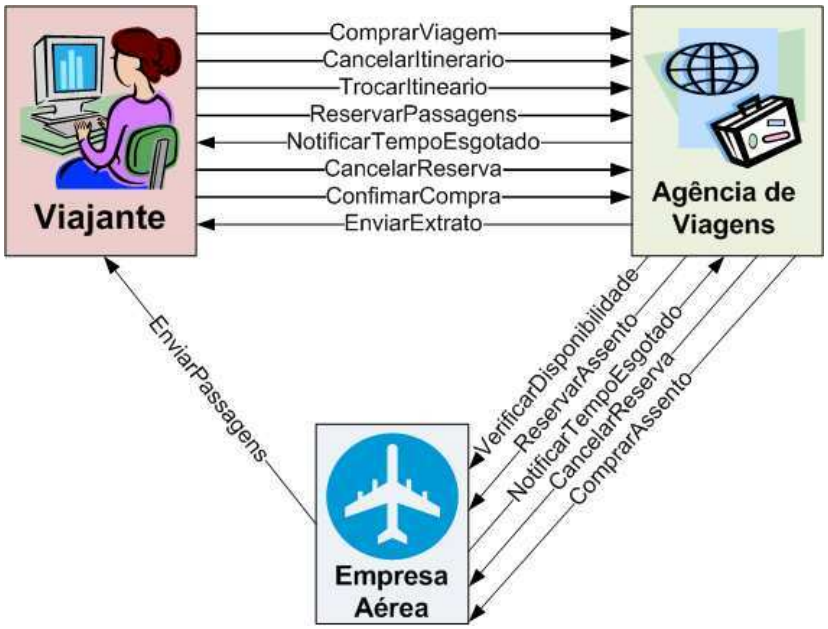


Figura 14. Interações entre os participantes do estudo de caso.

A coreografia inicia com a decisão do viajante de adquirir uma viagem, atividade totalmente independente de qualquer interação com a agência de viagens. O viajante faz então a requisição de compra de viagem à agência de viagens (ComprarViagem), passando as informações de origem, destino, data, hora e um critério de pesquisa. A agência de viagens utiliza essas informações para encontrar o melhor itinerário de vôos para viajar da origem ao destino na data e horário desejados, além do critério de pesquisa que é utilizado para encontrar as opções que proporcionem menor preço, menor tempo ou forneçam o tipo de avião desejado. A agência de viagens monta então um plano de viagem incluindo um ou mais trajetos. Para simplificar, considera-se que os trajetos da viagem são feitos com vôos de uma mesma empresa aérea.

Para cada trajeto da viagem a agência de viagens faz uma requisição à empresa aérea para verificar a disponibilidade de assentos (VerificarDisponibilidade). Quando criado o melhor itinerário de vôos e validada a disponibilidade de assentos em cada um deles, a agência de viagens retorna uma proposta de itinerário ao viajante. O viajante pode rejeitar ou aceitar uma proposta e então tomar uma das seguintes ações:

- Rejeitar a proposta invocando a operação CancelarItinerario e desistir da viagem.
- Solicitar uma nova proposta de itinerário, invocando a operação TrocarItinerário.
- Aceitar a proposta, invocando a operação ReservarPassagens e passando as informações de cartão de crédito para cobrança.

Caso o viajante decida solicitar uma nova proposta de itinerário, refaz-se o procedimento descrito acima. Caso o viajante decida aceitar a proposta e solicitar a reserva das passagens, a agência de viagens irá se comunicar com a empresa aérea para efetuar a reserva dos assentos de cada um dos trajetos (ReservarAssento). A empresa aérea faz a reserva de assentos por um período de no máximo um dia, o que significa que se a reserva não for confirmada neste tempo, uma notificação será enviada à agência de viagens (NotificarTempoEsgotado).

O viajante pode em seguida confirmar a reserva e comprar a viagem ou então cancelá-la (CancelarReserva). Caso sua decisão seja confirmar a reserva, invocando a operação ConfirmarCompra, a agência de viagens irá se comunicar com a empresa aérea e confirmar a compra dos assentos (ComprarAssento) de cada um dos trajetos. A empresa aérea processa então a compra dos assentos e envia os bilhetes diretamente ao viajante (EnviarPassagens).

No final da compra, a agência de viagens efetua a cobrança do cartão de crédito e envia um extrato ao viajante, contendo os detalhes da cobrança e uma descrição detalhada do itinerário (EnviarExtrato).

O desenvolvimento da coreografia, como já dito, foi feito utilizando a ferramenta PI4SOA. Embora a ferramenta nos permita abstrair os detalhes da linguagem WS-CDL, ao longo do texto iremos fazendo referência aos elementos WS-CDL que vão sendo utilizados, já que o objetivo é analisar a ferramenta e também a utilização da linguagem.

Iniciamos criando um novo projeto denominado AgenciaViagens e em seguida criamos uma nova descrição de coreografia denominada ReservaCompraViagem. Na Figura 12 vemos o projeto e a descrição de coreografia criados.

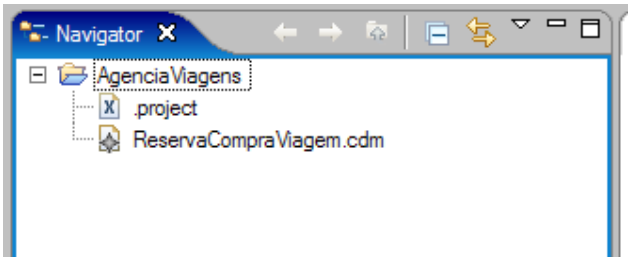


Figura 15. Projeto AgênciaViagens.

Em seguida definimos os papéis (*RoleType*), comportamentos (*Behavior*) e relacionamentos dos participantes (*RelationshipType*). Para o papel de viajante (ViajantePapel) definimos dois comportamentos, um de comprador de viagens (CompraDeViagem) que irá representar as ações de compra, reserva, cancelamento e confirmação de compra de viagens, e outro comportamento de recebedor de notificações (NotificacoesViajante), que irá representar o recebimento de passagens e o recebimento de notificações de expiração de reservas.

Para o papel de agência de viagens (AgenciaViagensPapel), definimos três comportamentos, um de recebedor de pedidos de compra de viagens (RecebimentoDeCompraDeViagem), que irá representar o recebimento de ações de compra, reserva, cancelamento e confirmação de compra de viagens, outro comportamento de compra de passagens (CompraDePassagens), que irá representar as ações de compra, reserva, cancelamento e confirmação de passagens, e outro comportamento de recebedor de notificações (NotificacoesAgencia), que irá representar o recebimento de notificações de expiração de reservas.

O terceiro papel da coreografia é o papel de empresa aérea, que possui dois comportamentos, um de recebedor de pedido de compra de assentos (RecebimentoDeCompraDeAssentos), que irá representar a pesquisa de disponibilidade, reserva, cancelamento e confirmação de compra de assentos, e outro comportamento de envio de passagens (EnvioPassagens), que irá representar o envio das passagens ao viajante.

A lógica de definição dos comportamentos de um papel deve levar em consideração o sentido do comportamento (entrada ou saída) e servir como um agrupamento lógico de operações de um serviço. Como vimos na análise da linguagem WS-CDL, um comportamento pode estar opcionalmente ligado a uma interface WSDL, portanto nem todo comportamento será implementado como um serviço Web, sendo descrito na coreografia apenas para representar as requisições que podem partir de um participante. Comportamentos que representam o recebimento de requisições deverão ter um serviço Web correspondente.

Em seguida devemos definir os relacionamentos entre os participantes e quais comportamentos estão envolvidos em cada um destes relacionamentos. O primeiro relacionamento é o de compra de viagem (CompraViagem), entre o viajante e agência de viagens. Esse relacionamento ocorre entre os comportamentos de CompraDeViagem e de RecebimentoDeCompraDeViagem.

Entre os comportamentos RecebimentoDeCompraDeViagem e RecebimentoDeCompraDeAssentos existe o relacionamento de compra de passagem (CompraPassagem). O relacionamento de compra de passagens (CompraPassagem) é definido pelo comportamento de NotificacaoAgencia e CompraDePassagens da agência de viagens com o comportamento de RecebimentoDeCompraDePassagens da empresa aérea. Por fim, o relacionamento de envio de passagens (EnvioPassagens) é definido entre os comportamentos EnvioDePassagens da empresa aérea e o comportamento NotificacoesViajante do viajante.

Na figura 13 temos o diagrama dos papéis, comportamentos e relacionamentos.

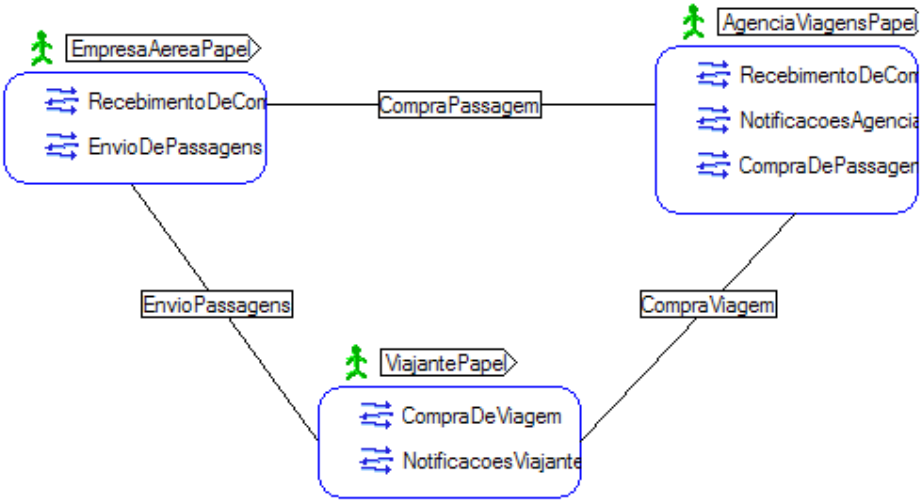


Figura 16. Papéis, comportamentos e relacionamentos.

Definidos os papéis, podemos definir os participantes (*ParticipantType*), que no nosso exemplo já estão definidos e possuem um relacionamento um-para-um para cada um dos papéis. O resultado é a criação dos participantes Viajante, AgenciaViagens e EmpresaAerea, que estão relacionados aos papéis ViajantePapel, AgenciaViagensPapel e EmpresaAereaPapel, respectivamente.

O próximo passo é a criação dos canais de comunicação (*ChannelType*), que funcionam como canais por onde passam todas as mensagens com destino a determinado participante. Os canais de comunicação definem restrições na comunicação com os participantes, como o número de vezes que um canal pode ser utilizado e o sentido das interações. Os seguintes canais devem ser criados:

Canal	Sentido interação	Restrições
ViajanteParaAgenciaViagens	Viajante↔Agência de viagens	Não possui.
AgenciaViagensParaEmpresaAerea	Agência de viagens↔Empresa aérea	Não possui.
AgenciaViagensParaViajante	Agência de viagens→viajante	Apenas requisições.
EmpresaAereaParaAgenciaViagens	Empresa aérea→Agência de viagens	Apenas requisições.
EmpresaAereaParaViajante	Empresa aérea→Viajante	Apenas requisições.

Tabela 4. Canais utilizados pela coreografia.

Após definidos os papéis, comportamentos, relacionamentos, participantes e canais, os próximos passos na definição da coreografia são criar os tipos de informação (*InformationTypes*), os símbolos (*Tokens*) e a definição do fluxo de mensagens entre os participantes.

Começando pela criação dos tipos de informação, devemos criar os tipos básicos como texto (*string*) e booleano (*boolean*), e também os tipos de informação que representam o conteúdo das mensagens trocadas entre os participantes. Esses tipos de informação funcionam como referências a tipos de dados definidos em esquemas XML ou interfaces WSDL. Os tipos de informação a serem criados são:

Tipo	Descrição
String	Representação de um texto qualquer.
Boolean	Representação dos valores lógicos "verdadeiro" e "falso".
RequisicaoViagem	Informações referentes a compra de viagem, como origem, destino, data e hora, etc.
Itinerario	Proposta de itinerário para uma viagem, obtido como resposta de uma requisição de viagem.
Assento	Informações referentes a compra de assento, como origem, destino, data, hora, vôo, poltrona, etc.
DisponibilidadeAssento	Retorno da consulta por disponibilidade de assento, com informações como data, hora, preço, etc.
Passagem	Informações de confirmação de compra de passagem.
ExtratoPassagens	Informações detalhadas de itinerário e preço das passagens.
InformacoesPagamento	Informações relativas ao cartão de crédito do viajante.

Tabela 5. Tipos de informação utilizados pela coreografia.

O próximo passo é a criação dos símbolos (*tokens*) que irão representar as referências aos participantes. Em tempo de execução estas referências representam a ligação física com um serviço Web. Desta forma, devemos criar uma referência para cada um dos participantes, resultando na criação de três símbolos:

Símbolo	Descrição
ViajanteRef	Referência a um viajante.
AgenciaViagensRef	Referência a uma agência de viagens.
EmpresaAereaRef	Referência a uma empresa aérea.

Tabela 6. Símbolos utilizados pela coreografia.

Na Figura 14 podemos ver as definições de canais, tipos de informação e símbolos.

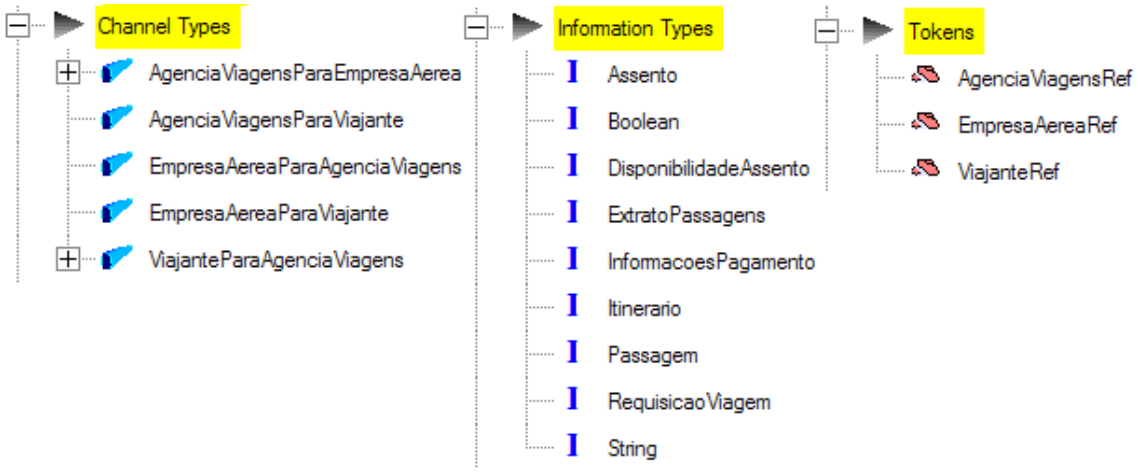


Figura 17. Canais, Tipos de Informação e Símbolos.

Até o momento definimos os papéis, os comportamentos, os relacionamentos, os participantes, os canais, os tipos de informação e os símbolos. Todavia, estes elementos servem apenas como base para a descrição da coreografia propriamente dita, que é a definição do fluxo de mensagens. Em seguida iniciamos a definição da coreografia (*choreography*). Os diagramas da coreografia como um todo podem ser vistos no Anexo 1.

Para que possamos compreender a representação gráfica da coreografia, a Figura 15 mostra a legenda dos principais elementos da coreografia.

Figura 18. Legenda coreografia.

Na parte 1 da Figura 16 podemos ver o inicio da coreografia CoreografiaCompraViagem. Seguindo a lógica de interação dos participantes, iniciamos descrevendo a requisição de compra de viagem, originada pelo viajante e submetida à agência de viagens. Na parte 1 da Figura 16 podemos ver o viajante realizando a interação ViajanteIniciaCompra, utilizando o canal ViajanteParaAgenciaViagens, com o relacionamento CompraViagem, invocando a operação comprarViagem, passando uma mensagem do tipo RequisicaoViagem e recebendo de resposta uma mensagem do tipo Itinerario.

A parte 2 da Figura 16 é continuação da parte 1 e mostra o início de uma iteração que somente será encerrada quando o usuário confirmar a escolha de um itinerário ou cancelar a requisição. Em seguida estão as interações subseqüentes ao recebimento do pedido de compra de viagem. Nela podemos ver as interações entre agência de viagens e empresa aérea no processo de criação de itinerários. Em um processo iterativo (*while*) a agência de viagens faz uma pesquisa de disponibilidade de assentos para cada trajeto da viagem. Os detalhes desta interação podem ser vistos na Figura 16.

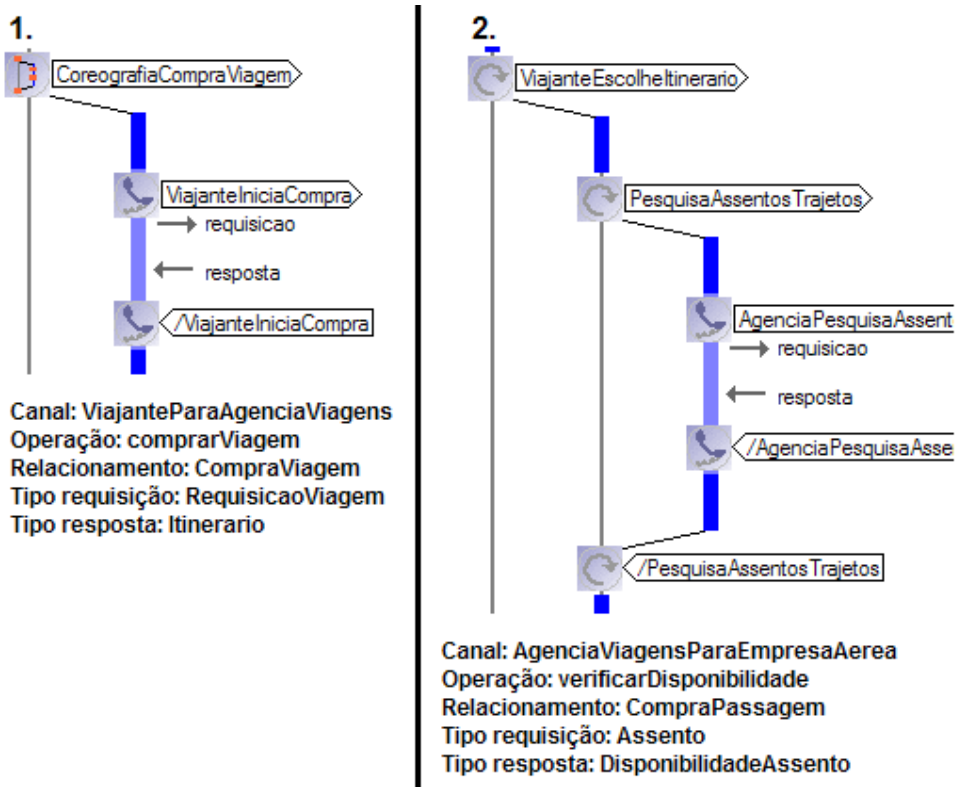


Figura 19. Início da coreografia.

Na continuação da coreografia, como próximo passo vem a opção do viajante quanto a trocar, cancelar ou confirmar a proposta de itinerário. Na figura 17 podemos ver um elemento de escolha (*choice*) sendo utilizado para descrever as três opções possíveis para o viajante. A primeira (da esquerda para direita) opção descrita é a de cancelamento do itinerário. Neste caso o viajante faz um interação com a agência de viagens invocando a operação cancelarItinerario e em seguida encerra-se a coreografia. A segunda opção descrita é a de troca de itinerário, neste caso faz-se a invocação da operação trocarItinerario e reinicia-se a iteração de escolha de itinerário (iniciada na Figura 16). Na última opção, a de confirmação do itinerário proposto, faz-se uma requisição à operação reservarPassagens e em seguida encerra-se a iteração de escolha de itinerário. Todas as interações descritas na Figura 17 utilizam o canal ViajanteParaAgenciaViagens, com o relacionamento CompraViagem.

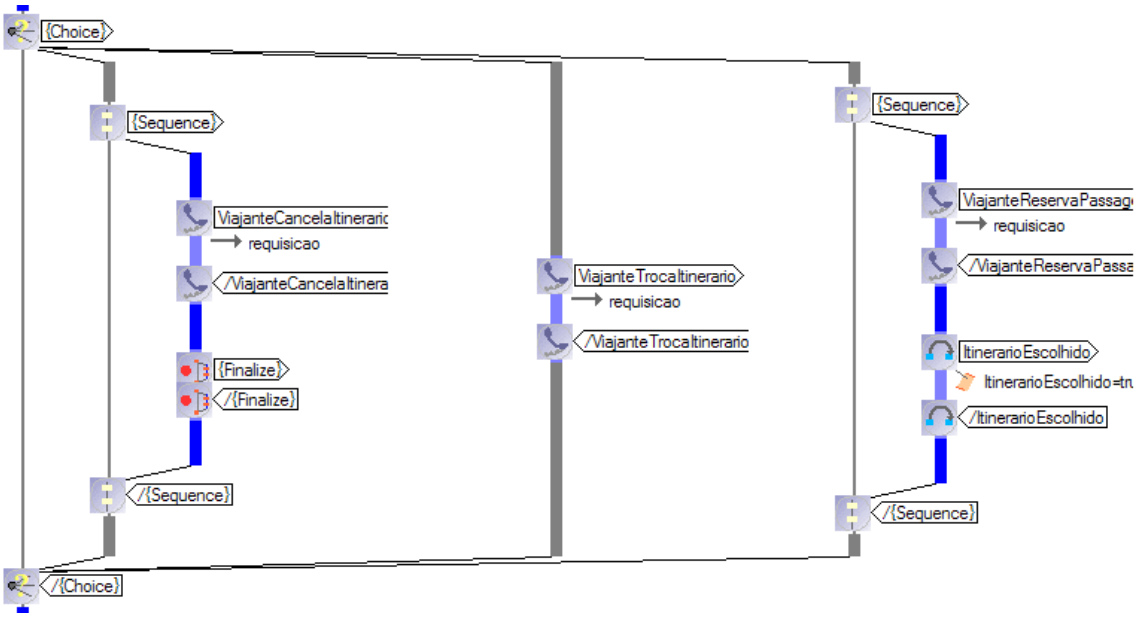


Figura 20. Troca/Cancela/Confirma proposta de itinerário.

Seguindo a lógica das interações, após o usuário confirmar a reserva das passagens, o próximo passo é reservar os assentos junto à empresa aérea. Em um iteração (*while*) a agência de viagens reserva o assento de cada trajeto do itinerário, invocando a operação reservarAssento, como podemos ver na Figura 18.

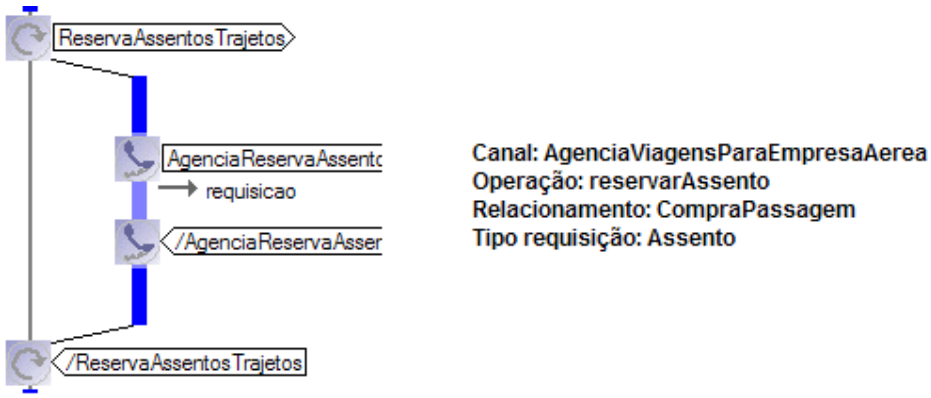


Figura 21. Reserva de assentos.

Após reservados os assentos, a coreografia entra em estado de aguardo até que uma das três opções ocorra:

1. O tempo limite da reserva seja alcançado e então a agência de viagens é notificada e por conseqüência notifica o viajante. A coreografia é finalizada.
2. O viajante cancela a reserva e então a agência de viagens cancela todos os assentos junto à empresa aérea. A coreografia é finalizada.
3. O viajante confirma a compra da viagem e então a agência de viagens confirma a compra dos assentos junto à empresa aérea.

Na Figura 19 podemos ver a coreografia das três opções.

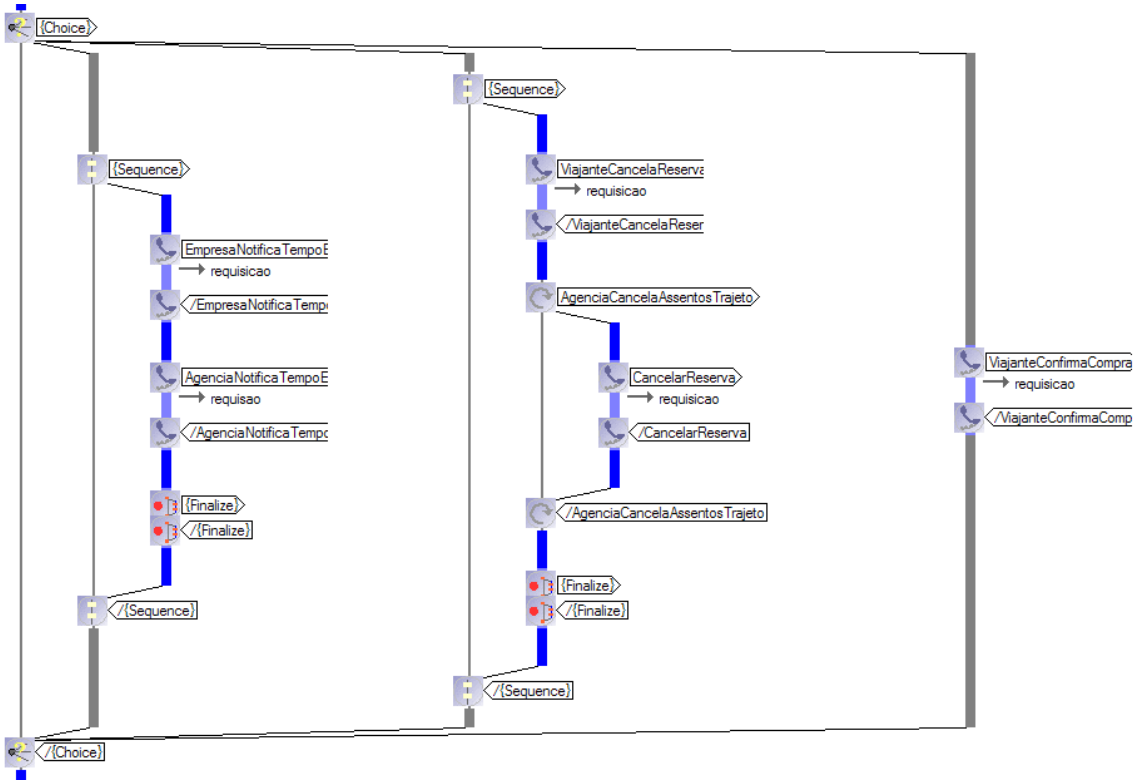


Figura 22. Tempo limite/Cancelamento/Confirmação de reserva.

Caso a opção do viajante tenha sido confirmar a compra da viagem, a coreografia segue para sua finalização, restando apenas a confirmação da compra dos assentos junto à empresa aérea, o envio das passagens ao viajante e o envio do extrato da compra.

Na parte 1 da Figura 20 podemos ver a agência de viagens se comunicando com a empresa aérea para efetuar a compra dos assentos. A invocação da operação comprar assento é repetida para o assento de cada trajeto do itinerário.

Na parte 2 da Figura 20 vemos a empresa aérea enviando as passagens diretamente ao viajante, no primeiro e único momento da coreografia onde existe interação entre esses dois participantes.

Na parte 3 da Figura 20 vemos a última interação da coreografia, onde a agência de viagens envia o extrato da compra ao viajante.

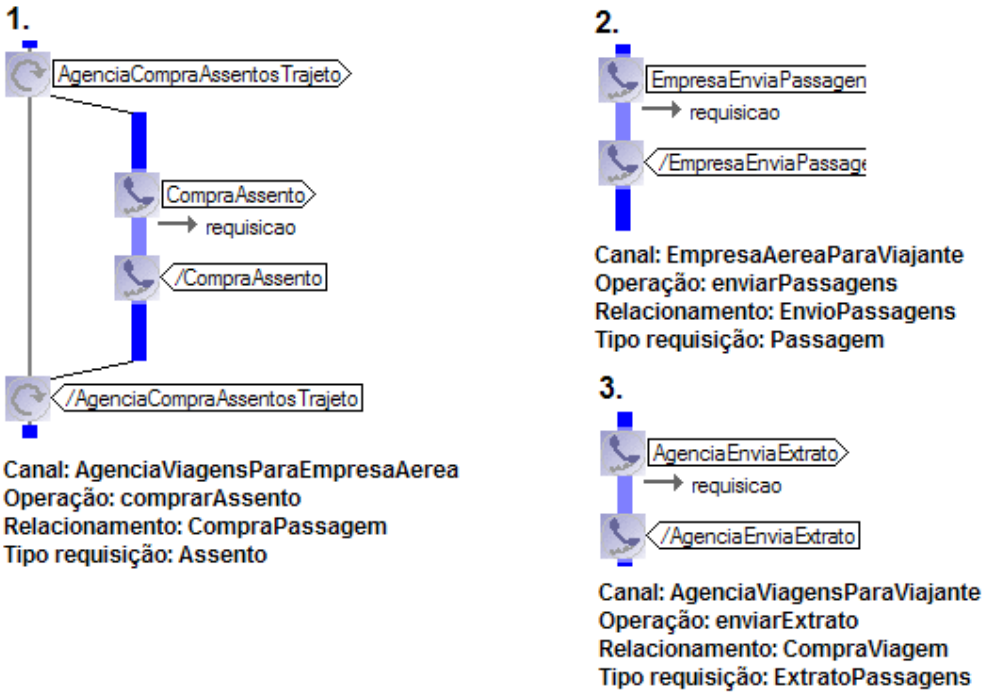


Figura 23. Coreografia da confirmação da compra da viagem.

Como podemos ver ao longo deste estudo de caso, a criação de uma coreografia não é algo trivial e exige um esforço considerável para descrever os inúmeros detalhes da interação entre participantes. Além do esforço, foi possível perceber algumas dificuldades devido à imaturidade da linguagem WS-CDL e da ferramenta PI4SOA, o que de certa forma já se esperava.

A falta de exemplos e uma documentação algumas vezes confusa fazem com que alguns elementos da linguagem sejam difíceis de usar ou induzam ao uso incorreto.

A documentação também é um dos principais pontos falhos da ferramenta PI4SOA, sendo quase nula. O aprendizado da ferramenta é baseado em alguns poucos exemplos e exige consultas freqüentes à especificação da WS-CDL.

Um problema em particular e que dificultou o desenvolvimento do estudo caso, ocorreu na utilização do recurso de correlação. Para se fazer correlação de mensagens com a linguagem WS-CDL, deve-se utilizar o elemento identidade (*identity*) em conjunto com um canal (*ChannelType*). Este recurso, apesar de estar presente na ferramenta, apresentou diversos problemas que acabaram impedindo seu uso. Caso funcionasse, poderíamos suprimir a passagem de informações em chamadas de métodos como trocarItinerario, já que este estaria correlacionado com a requisição comprarViagem anterior.

Apesar do esforço e das dificuldades encontradas, o desenvolvimento da coreografia traz consigo as vantagens descritas anteriormente no trabalho. A utilização mais espontânea e de fácil acesso está na geração das interfaces WSDL dos serviços, o que pode ser conseguido com alguns poucos cliques de mouse utilizando a ferramenta PI4SOA. O Anexo 2 apresenta as interfaces WSDL geradas.

O recurso de criação de cenários de teste baseados na coreografia, também disponível na ferramenta, não foi utilizado, o que se deixa como sugestão para trabalhos futuros.

6. TRABALHOS RELACIONADOS

Pelo fato do assunto ser relativamente novo e ainda pouco discutido, existem poucos artigos ou livros que tratam especificamente de coreografia de serviços Web. Não foram encontrados trabalhos de conclusão de curso ou dissertações de mestrado relativas ao assunto.

Peltz [3] justifica a existência dos conceitos de orquestração e coreografia e discute a necessidade dos mesmos para resolver o problema da integração de sistemas, um dos maiores problemas da área de tecnologia da informação. Peltz descreve como evoluíram os padrões de orquestração e coreografia, finalizando com uma descrição sucinta da linguagem WSCI. Mostra também uma análise interessante da colaboração entre coreografia e orquestração. Termina o artigo com um estudo de caso simples aplicando apenas os conceitos de orquestração utilizando a linguagem WS-BPEL.

Barros, Dumas e Oaks [19] fazem uma análise detalhada da estrutura do WS-CDL, discutindo os principais elementos da linguagem e descrevendo o processo de construção de coreografias com esta linguagem. O artigo termina tratando do seu objetivo principal, que é fornecer uma crítica da WS-CDL. Um dos principais pontos criticados é a falta de formalismo da linguagem, o que dificulta o desenvolvimento de ferramentas para verificação e validação de coreografias. Outra crítica interessante apresentada por [19] é o relacionamento muito restritivo que a WS-CDL possui com a WSDL. A ligação entre as linguagens atualmente exige que os serviços implementem determinada interface WSDL, impedindo o relacionamento com serviços com interfaces diferentes mas equivalentes funcionalmente. Como conclusão, o artigo sugere a criação de padrões para coreografias de serviços que sejam independentes de linguagem. Sugere também a criação de um meta-modelo de coreografia que sirva de base à WS-CDL. Tais sugestões são feitas com o propósito principal de tornar a linguagem mais formal.

7. CONCLUSÕES E TRABALHO FUTUROS

A implementação de arquiteturas orientadas a serviços (SOA) e a utilização de serviços Web para o desenvolvimento de aplicações distribuídas exigem a criação de métodos específicos para tratar a integração de serviços Web e para coordenar a execução de processos. Os conceitos de orquestração e coreografia vêm sendo muito discutidos com a crescente complexidade dos sistemas baseados nessas tecnologias. O conceito de coreografia, embora ainda não tão maduro quando o de orquestração, vem recebendo cada vez mais atenção à medida que crescem os problemas na organização das interações entre serviços.

O estudo dos conceitos e a análise das linguagens e ferramentas disponíveis permitem nos posicionar quanto ao que já é possível fazer e quanto ao que esperar da evolução deste tema no futuro próximo. Ao longo do trabalho foi possível compreender a evolução dos esforços de desenvolvimento de um padrão para coreografia de serviços. Após algumas iniciativas, a linguagem mais desenvolvida e até o momento mais promissora é a WS-CDL. Porém é um trabalho ainda em desenvolvimento e que tem alguns problemas em aberto para serem resolvidos. O mercado já vem trabalhando em ferramentas para tornar a WS-CDL mais acessível para os usuários, como a iniciativa de desenvolvimento da ferramenta PI4SOA.

Diversas outras contribuições podem ser dadas ao estudo deste tema, talvez uma das mais interessantes seja a aplicação da linguagem WS-CDL em um contexto onde trará ganhos imediatos, como o da integração em servidores OLAP (*On-Line Analytical Processing*) e servidores de mapas baseados em GIS (*Geographical Information Systems*). Este é um problema que vem recebendo grande atenção e uma solução utilizando a WS-CDL poderá trazer grandes benefícios.

Outro trabalho futuro importante é um estudo mais detalhado da integração entre WS-BPEL e WS-CDL, com uma conseqüente cooperação entre a coreografia e a orquestração de serviços. Tal contribuição pode permitir a definição de um cenário ideal que é a geração automática de processos abstratos WS-BPEL a partir de uma descrição de coreografia WS-CDL, ou vice-versa.

Outra tema para trabalho futuro é a utilização do recurso de cenários de teste disponibilizado na ferramenta PI4SOA. Nela é possível validar uma coreografia com base em um fluxo de mensagens específico.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] STAL, Michael. Using Architectural Patterns and Blueprints for Service-Oriented Architecture. IEEE Computer Society, 2006.

[2] OASIS. Web Services Business Process Execution Language, versão 2.0 draft 2, 20 de novembro de 2006.

[3] PELTZ, Chris. Web Services Orchestration and Choreography. IEEE Computer Society, outubro de 2003.

[4] BARROS, Alistair; DUMAS, Marlon; OAKS, Phillipa. A Critical Overview of the Web Services Choreography Description Language (WS-CDL). BPTrends, 2005.

[5] W3C. Web Services Conversation Language (WSCL), W3C note, versão 1.0, 14 de março 2002.

[6] W3C. Web Services Choreography Working Group Charter. Disponível em: http://www.w3.org/2005/12/wscwg-charter.html. Acesso em: Janeiro de 2007.

[7] W3C. Web Service Choreography Interface (WSCI), W3C note, versão 1.0, 8 de agosto de 2002.

[8] DURVASULA, Surekha; GUTTMANN, Martin; KUMAR, Ashok. Martin Guttmann SOA Practitioners' Guide, Part 1. Why Services-Oriented Architecture? 15 de setembro de 2006.

[9] JOHNSON, Ralph E.; FOOTE, Brian. Designing Reusable Classes, Journal of Object-Oriented Programming, vol. 1, num. 2, 1988.

[10] OASIS. Reference Model for Service Oriented Architecture. Committee Draft 1.0, 7 de Fevereiro de 2006.

[11] W3C. SOAP Version 1.2 Part 0: Primer, 24 de junho de 2006.

[12] NEWCOMER, Eric. Understanding Web Services – XML, WSDL, SOAP, and UDDI. United States: Addison-Wesley, 2002. 332p.

[13] KAVANTZAS, Nickolaos. Aggregating Web Services: Choreography and WS-CDL. Oracle Corporation. 5 de abril de 2007.

[14] Where do you apply SOA? http://www.ebpml.org/soa.htm#coord. Visitado em 14/04/2007.

[15] HAVEY, Mike. Essential Business Process Modeling. O'Reilly, 2005. 350p.

[16] W3C. Web Services Choreography Description Language (WS-CDL), W3C candidate recommendation, versão 1.0, 9 de novembro de 2005.

[17] Pi4 Technologies Foundation wiki. http://www.pi4tech.org/tiki-index.php. Visitado em: 02/06/2007.

[18] BHARTI, Nitini. Choreography tools deliver on promise of SOA. Disponível em: http://searchwebservices.techtarget.com/originalContent/0,289142,sid26_gci1095769,00.html Visitado em: 02/06/2007.

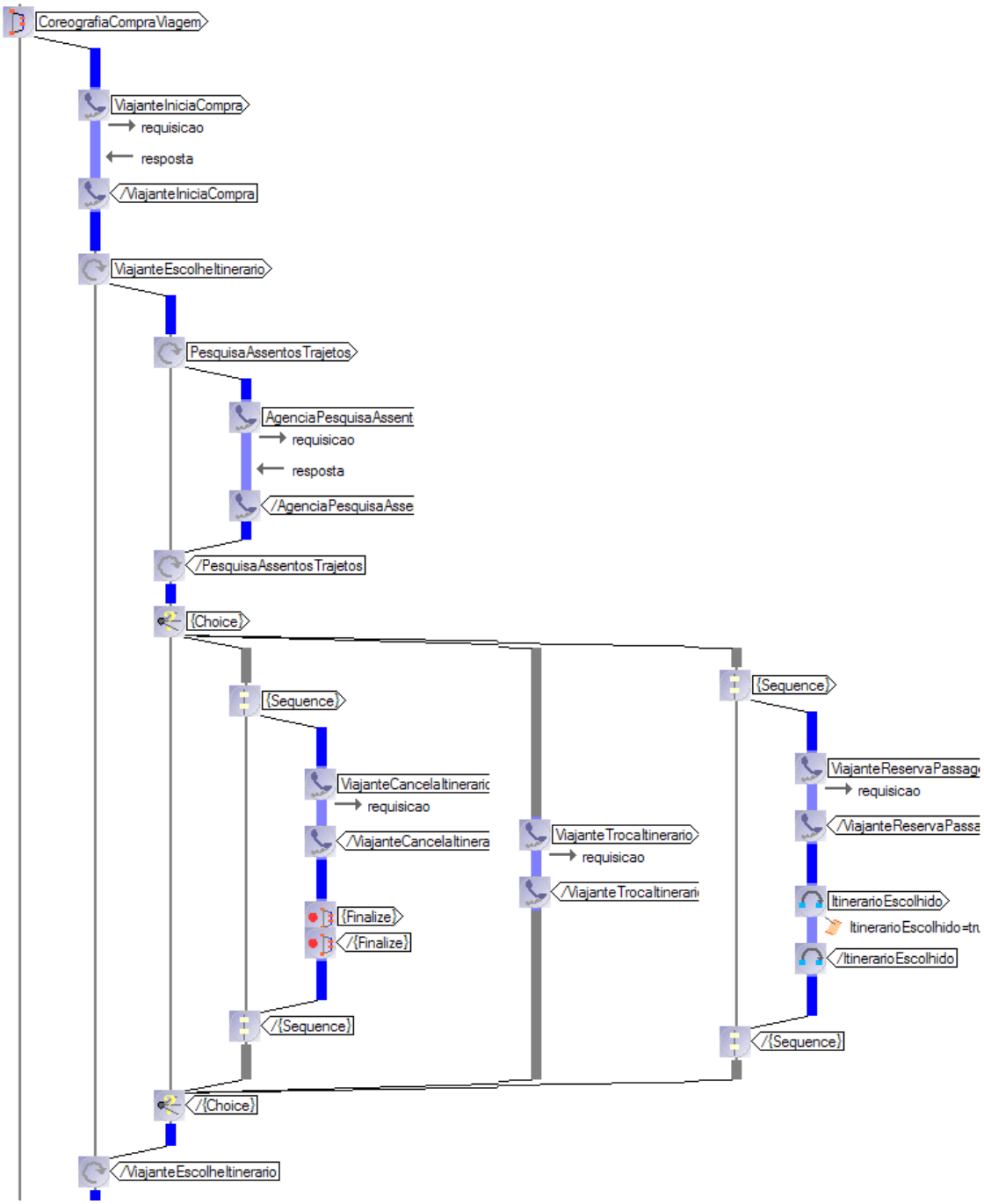
[19] BARROS, Alistair; DUMAS, Marlon; OAKS, Phillipa. A Critical Overview of the Web Service Choreography Description Language (WS-CDL). BPTrends, março de 2005.

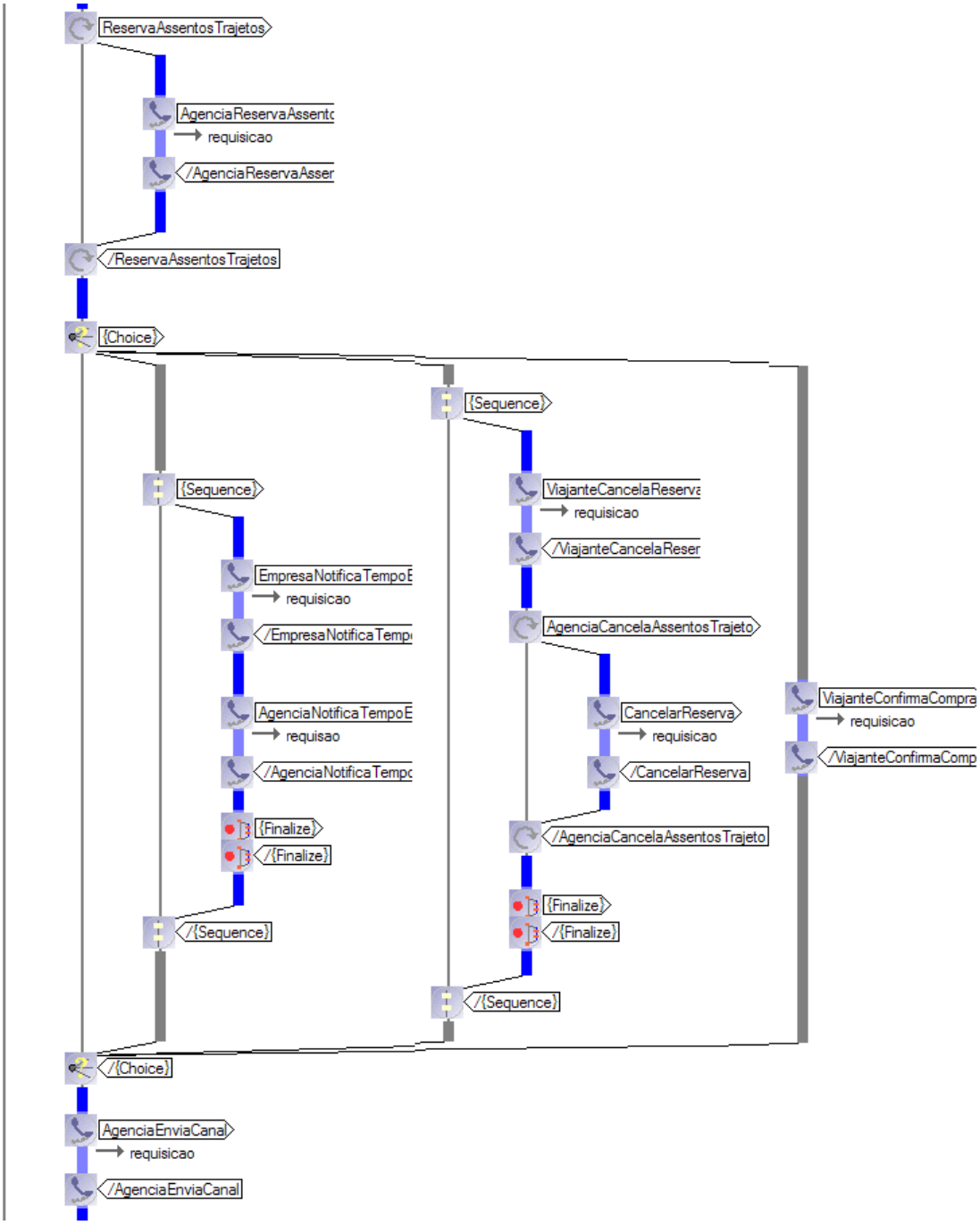
[20] W3C. Web Services Description Language (WSDL) 1.1, 15 de março 2001. Disponível em: http://www.w3.org/TR/wsdl/

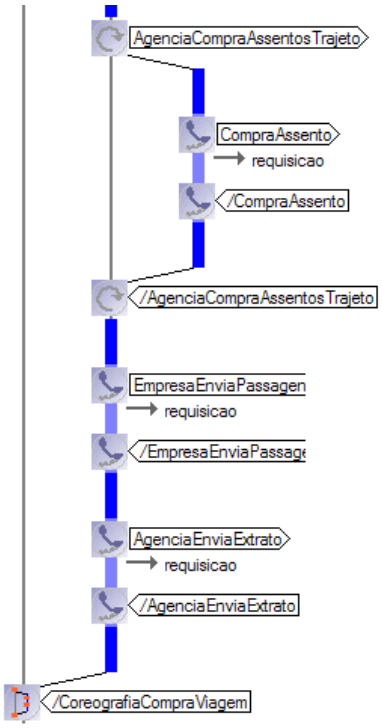
[21] OASIS. UDDI versão 3.0.2, 19 de outubro de 2004. Disponível em: http://www.oasis-open.org/committees/uddi-spec/doc/spec/v3/uddi-v3.0.2-20041019.htm

ANEXO 1

Descrição da coreografia do estudo de caso definido no capítulo 5. Define a interação entre os participantes Viajante, Agência de Viagens e Empresa Aérea.




ReservaAssentosTrajetos
AgenciaReservaAssento
requisicao
/AgenciaReservaAsser
/ReservaAssentosTrajetos
{Choice}
{Sequence}
EmpresaNotificaTempoE
requisicao
/EmpresaNotificaTempo
AgenciaNotificaTempoE
requisao
/AgenciaNotificaTempo
{Finalize}
/{Finalize}
/{Sequence}
{Sequence}
ViajanteCancelaReserva
requisicao
/ViajanteCancelaReser
AgenciaCancelaAssentosTrajeto
CancelarReserva
requisicao
/CancelarReserva
/AgenciaCancelaAssentosTrajeto
{Finalize}
/{Finalize}
/{Sequence}
ViajanteConfirmaCompra
requisicao
/ViajanteConfirmaComp
/{Choice}
AgenciaEnviaCanal
requisicao
/AgenciaEnviaCanal


AgenciaCompraAssentosTrajeto
CompraAssento
requisicao
/CompraAssento
/AgenciaCompraAssentosTrajeto
EmpresaEnviaPassagen
requisicao
/EmpresaEnviaPassage
AgenciaEnviaExtrato
requisicao
/AgenciaEnviaExtrato
/CoreografiaCompraViagem

ANEXO 2

Interfaces WSDL geradas pela ferramenta PI4SOA a partir da descrição da coreografia do estudo de caso no capítulo 5.

RecebimentoDeCompraDeViagem.wsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<description xmlns="http://www.w3.org/2005/05/wsdl"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns="http://www.ufsc.br/~pmichels/AgenciaViagens"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.ufsc.br/~pmichels/AgenciaViagens">
    <types>
        <xsd:element name="RequisicaoViagem">
            <documentation>
                pi4soa: User defined definitions
            </documentation>
        </xsd:element>
        <xsd:element name="Itinerario">
            <documentation>
                pi4soa: User defined definitions
            </documentation>
        </xsd:element>
        <xsd:element name="InformacoesPagamento">
            <documentation>
                pi4soa: User defined definitions
            </documentation>
        </xsd:element>
    </types>
    <interface name="RecebimentoDeCompraDeViagemPort">
        <operation name="comprarViagem"
pattern="http://www.w3.org/2005/05/wsdl/in-out">
            <input element="tns:RequisicaoViagem"
messageLabel="requisicao"/>
            <output element="tns:Itinerario" messageLabel="resposta"/>
        </operation>
        <operation name="cancelarItinerario"
pattern="http://www.w3.org/2005/05/wsdl/in-only">
            <input element="tns:RequisicaoViagem"
messageLabel="requisicao"/>
        </operation>
        <operation name="trocarItinerario"
pattern="http://www.w3.org/2005/05/wsdl/in-only">
            <input element="tns:RequisicaoViagem"
messageLabel="requisicao"/>
        </operation>
        <operation name="reservarPassagens"
pattern="http://www.w3.org/2005/05/wsdl/in-only">
            <input element="tns:InformacoesPagamento"
messageLabel="requisicao"/>
        </operation>
        <operation name="cancelarReserva"
pattern="http://www.w3.org/2005/05/wsdl/in-only">
            <input element="tns:Itinerario" messageLabel="requisicao"/>
        </operation>
```

```
        <operation name="confirmarCompra"
pattern="http://www.w3.org/2005/05/wsdl/in-only">
            <input element="tns:Itinerario" messageLabel="requisicao"/>
        </operation>
    </interface>
</description>
```

RecebimentoDeCompraDeAssentos.wsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<description xmlns="http://www.w3.org/2005/05/wsdl"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns="http://www.ufsc.br/~pmichels/AgenciaViagens"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.ufsc.br/~pmichels/AgenciaViagens">
    <types>
        <xsd:element name="assento">
            <documentation>
                pi4soa: User defined definitions
            </documentation>
        </xsd:element>
        <xsd:element name="disponibilidadeAssento">
            <documentation>
                pi4soa: User defined definitions
            </documentation>
        </xsd:element>
    </types>
    <interface name="RecebimentoDeCompraDeAssentosPort">
        <operation name="verificarDisponibilidade"
pattern="http://www.w3.org/2005/05/wsdl/in-out">
            <input element="tns:assento" messageLabel="requisicao"/>
            <output element="tns:disponibilidadeAssento"
messageLabel="resposta"/>
        </operation>
        <operation name="reservarAssento"
pattern="http://www.w3.org/2005/05/wsdl/in-only">
            <input element="tns:assento" messageLabel="requisicao"/>
        </operation>
        <operation name="cancelarReserva"
pattern="http://www.w3.org/2005/05/wsdl/in-only">
            <input element="tns:assento" messageLabel="requisicao"/>
        </operation>
        <operation name="enviarCanal"
pattern="http://www.w3.org/2005/05/wsdl/in-only">
            <input element="xsd:string" messageLabel="requisicao"/>
        </operation>
        <operation name="comprarAssento"
pattern="http://www.w3.org/2005/05/wsdl/in-only">
            <input element="tns:assento" messageLabel="requisicao"/>
        </operation>
    </interface>
</description>
```

NotificacoesAgenciaViagens.wsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<description xmlns="http://www.w3.org/2005/05/wsdl"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns="http://www.ufsc.br/~pmichels/AgenciaViagens"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.ufsc.br/~pmichels/AgenciaViagens">
    <types>
        <xsd:element name="disponibilidadeAssento">
            <documentation>
                pi4soa: User defined definitions
            </documentation>
        </xsd:element>
    </types>
    <interface name="NotificacoesAgenciaViagensPort">
        <operation name="notificarTempoEsgotado"
pattern="http://www.w3.org/2005/05/wsdl/in-only">
            <input element="tns:disponibilidadeAssento"
messageLabel="requisicao"/>
        </operation>
    </interface>
</description>
```

NotificacoesViajante.wsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<description xmlns="http://www.w3.org/2005/05/wsdl"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns="http://www.ufsc.br/~pmichels/AgenciaViagens"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.ufsc.br/~pmichels/AgenciaViagens">
    <types>
        <xsd:element name="RequisicaoViagem">
            <documentation>
                pi4soa: User defined definitions
            </documentation>
        </xsd:element>
        <xsd:element name="Passagem">
            <documentation>
                pi4soa: User defined definitions
            </documentation>
        </xsd:element>
        <xsd:element name="extratoPassagens">
            <documentation>
                pi4soa: User defined definitions
            </documentation>
        </xsd:element>
    </types>
    <interface name="NotificacoesViajantePort">
        <operation name="notificarTempoEsgotado"
pattern="http://www.w3.org/2005/05/wsdl/in-only">
            <input element="tns:RequisicaoViagem" messageLabel="requisao"/>
        </operation>
        <operation name="enviarPassagens"
pattern="http://www.w3.org/2005/05/wsdl/in-only">
            <input element="tns:Passagem" messageLabel="requisicao"/>
```

```
        </operation>
        <operation name="enviarExtrato"
pattern="http://www.w3.org/2005/05/wsdl/in-only">
            <input element="tns:extratoPassagens"
messageLabel="requisicao"/>
        </operation>
    </interface>
</description>
```

COREOGRAFIA DE SERVIÇOS WEB
(Uma abordagem para a integração de serviços Web)

Paulo Henrique Michels
Universidade Federal de Santa Catarina
Departamento de Informática e Estatística
pmichels@inf.ufsc.br

Renato Fileto
Universidade Federal de Santa Catarina
Departamento de Informática e Estatística
fileto@inf.ufsc.br

***RESUMO* - O número crescente de componentes de software sendo disponibilizados na forma de serviços Web *(Web Services)* e conseqüentemente a popularização da arquitetura orientada a serviços, faz surgir uma demanda por métodos e padrões para integração de serviços. Este trabalho apresenta os conceitos de orquestração e coreografia de serviços e de que forma eles tratam a questão da integração de serviços. O trabalho faz uma análise mais detalhada da coreografia de serviços, apresentando em detalhes as linguagens WSCI e WS-CDL. Em seguida analisa a ferramenta PI4SOA, que auxilia na construção de coreografias utilizando a WS-CDL.**

I. INTRODUÇÃO

Durante o processo de desenvolvimento de sistemas distribuídos surgem diversas barreiras, como a incompatibilidade de protocolos de comunicação, incompatibilidade de linguagens de programação, questões de segurança, incompatibilidade de tipos de dados e vários outros problemas que fazem com que a integração de sistemas seja um dos maiores problemas da área de tecnologia da informação.

As facilidades que os serviços Web (*Web services*) trazem para o desenvolvimento de sistemas distribuídos estão fazendo com que eles tenham larga adoção nos sistemas modernos. Isto faz emergir métodos de construção de sistemas baseados na integração de serviços Web e na arquitetura orientada a serviços (ou *SOA*. de *Service Oriented Architecture*, em inglês) [1].

Apesar dos benefícios da tecnologia de serviços Web, os requisitos impostos pelos sistemas atuais, impõem a necessidade do auxílio de elementos responsáveis pela integração de serviços Web e pela coordenação da execução de processos constituídos pela composição de tais serviços. Os conceitos de orquestração e coreografia estendem a arquitetura orientada a serviços Web, criando padrões e linguagens que atendam a estes requisitos [3].

O conceito de orquestração é mais maduro e dispõe de padrões bem definidos, como a linguagem WS-BPEL, que é suportada pelos principais fornecedores da área. Já o conceito de coreografia é menos debatido e apenas mais recentemente vem recebendo atenção. Apesar de existirem algumas iniciativas para o desenvolvimento de um padrão de coreografia, até o momento não existe nada suficientemente maduro ou largamente apoiado pelo mercado.

Com o objetivo de compreendermos os atuais esforços para o desenvolvimento de uma linguagem padrão para a descrição de coreografias, analisamos as linguagens WSCI e WS-CDL, com foco especial nesta última. Avaliamos também as primeiras iniciativas do mercado para o desenvolvimento de

ferramentas para a descrição de coreografias, em especial a ferramenta PI4SOA, baseada na linguagem WS-CDL [4] [7].

II. SERVIÇOS WEB E A ARQUITETURA ORIENTADA A SERVIÇOS

A tecnologia de serviços Web (*Web services*) nasceu para atender a um dos principais problemas da área de tecnologia da informação, que foi e continua sendo a integração de sistemas. Esta tecnologia se desenvolveu a partir da especificação de um conjunto de padrões abertos, baseados em XML, que tornam os serviços Web independentes de protocolo de rede, independentes de plataforma e independentes de linguagem de programação. Os padrões que formam a base dos serviços Web são o WSDL, o SOAP e o UDDI. Eles definem um formato de mensagem, determinam um padrão para a especificação das interfaces para troca de mensagens, criam convenções para o mapeamento entre mensagem e implementação do serviço e definem mecanismos para publicação e pesquisa de serviços Web.

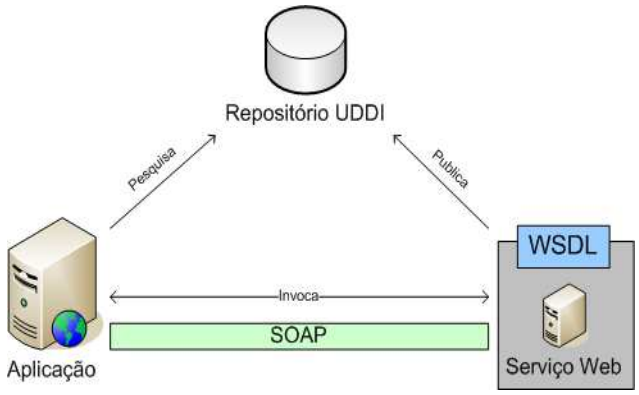


Figura 1. A arquitetura orientada a serviços Web.

A Figura 1 ilustra a arquitetura orientada a serviços Web e a interação entre os diversos padrões e componentes envolvidos na implementação, publicação, busca e invocação de serviços Web.

Os serviços Web representam um grande avanço no sentido de facilitar a integração de sistemas distribuídos, desta forma é crescente a quantidade de sistemas que utilizam esta tecnologia e usufruem de suas vantagens no desenvolvimento dos diversos componentes de um sistema. À medida que os componentes de um sistema são implementados na forma de serviços Web, os sistemas passam a ser desenvolvidos através da integração de serviços. Tais sistemas possuem uma arquitetura baseada na reutilização de serviços, a Arquitetura Orientada a Serviços (SOA em inglês). Como descrito em [10], "A Arquitetura Orientada a Serviços (SOA) é um paradigma para organização e utilização de entidades distribuídas que estão sob controle de diferentes domínios.". A arquitetura orientada a serviços não está diretamente relacionada e nem dependente dos serviços Web, outras tecnologias como COM, CORBA e RMI também permitem a implementação deste tipo de arquitetura, porém é com o uso de serviços Web que os benefícios da SOA são melhores alcançados, entre eles:

- **Distribuição:** Os sistemas de informação estão sendo utilizados em ambientes cada vez mais distribuídos, exigindo que componentes do sistema executem em máquinas fisicamente separadas.
- **Heterogeneidade:** Sistemas distribuídos tipicamente executam em ambientes heterogêneos e que muitas vezes não estão sob controle dos desenvolvedores.
- **Dinamismo:** Para atender aos ambientes de negócio cada vez mais dinâmicos, os sistemas precisam ser cada vez mais flexíveis para atender freqüentes mudanças.

▪ **Transparência:** Os sistemas devem ser transparentes aos detalhes da infra-estrutura de comunicação dos diferentes pontos onde executam os diversos componentes.
▪ **Orientação a processos:** Sistemas devem atender aos processos (*workflows*) presentes no ambiente em que são utilizados.

Uma arquitetura SOA permite que serviços sejam utilizados como componentes reutilizáveis de software, que em contraposição ao desenvolvimento de todas as partes de um sistema, é um fator que pode levar ao aumento da qualidade e da produtividade da atividade de desenvolvimento de software. Desta forma, podemos ver o uso da arquitetura SOA como algo que traz agilidade e qualidade ao desenvolvimento de sistemas distribuídos [9].

Existem diversos outros benefícios da utilização de uma arquitetura SOA, os mais notáveis são facilitar o gerenciamento do crescimento dos sistemas corporativos de larga escala e facilitar a utilização da escalabilidade da Internet pelo uso de serviços, reduzindo assim os custos das organizações para a cooperação entre organizações. O valor da SOA está em oferecer um paradigma escalável para organizar grandes sistemas em rede e que requerem interoperabilidade para atingir seus objetivos [10].

III. ORQUESTRAÇÃO E COREOGRAFIA DE SERVIÇOS

Uma arquitetura orientada a serviços, à medida que se desenvolve, traz consigo a necessidade de elementos responsáveis pela integração de serviços, intermediando a invocação dos serviços e coordenando a execução de processos. Serviços Web sem o auxílio de tais elementos são limitados a interações simples, já que o modelo de interação suportado pelo WSDL e pelo SOAP é essencialmente síncrono e sem manutenção de estado (*stateless*). O próprio protocolo HTTP, que é a escolha mais comum para se implementar serviços Web, é um protocolo orientado a requisições e respostas, cuja conexão tem o tempo de vida restrito a uma requisição e sua respectiva resposta, sem nenhuma relação com requisições anteriores ou subseqüentes.

O real potencial dos serviços Web e os benefícios da SOA somente podem ser alcançados com o auxílio de mecanismos de integração que tornem possíveis interações síncronas e assíncronas e com manutenção de estado (*stateful*) entre duas ou mais partes. Tais mecanismos são requisitos fundamentais para atender aos processos de negócio complexos presentes nos sistemas atuais [2].

Dois termos comumente utilizados na coordenação da execução de serviços são orquestração e coreografia. Embora esses termos possam parecer sinônimos à primeira vista (e muitos o tratam desta forma), na realidade são conceitos diferentes e que se complementam, sobrepondo-se apenas em alguns pontos [3].

Os termos orquestração e coreografia descrevem dois aspectos diferentes utilizados pelos padrões que estão emergindo para tratar da criação de processos de negócio a partir da integração de serviços Web. Orquestração se refere a um processo de negócio executável que pode interagir tanto com serviços internos quanto externos, sendo que essa interação é sempre representada sob a perspectiva de uma das partes, que é a que detém o controle do processo. A coreografia, por outro lado, é mais

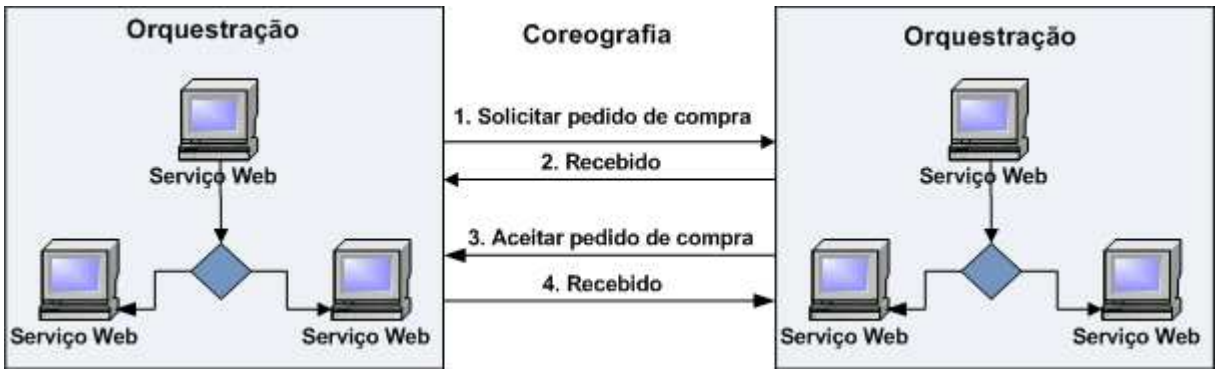


Figura 2. Diferença entre coreografia e orquestração.

colaborativa, permitindo que cada parte envolvida descreva seu papel na interação.

A Figura 3 ilustra a diferença entre os conceitos de coreografia e orquestração. Enquanto a orquestração foca na interação entre serviços sob a perspectiva de uma das partes - o coordenador - a coreografia trata de questões como a ordenação de mensagens e a interação sob a perspectiva de todas as partes, sem a necessidade de um coordenador.

A orquestração foca no comportamento sob o ponto de vista de um único participante, agindo como um controlador de um processo e conduzindo a execução desse processo seguindo sua definição. Uma linguagem de orquestração é utilizada para descrever como determinado processo deve ser executado, portanto tais linguagens são ditas executáveis. Para se descrever a orquestração de um processo, uma linguagem de orquestração deve disponibilizar construções como controles de execução (condicionais, seqüências, paralelismo, laços de repetição), variáveis e construções para manipulá-las, tratadores de eventos, controles de tempo e controle e recuperação de exceções [13].

A coreografia descreve a colaboração dos serviços no sistema como um todo e independente de um elemento controlador atento a todas as partes envolvidas e detalhes da comunicação ponto-a-ponto entre as partes. Ao contrário das linguagens de orquestração, as linguagens de coreografia não possuem a finalidade de serem executadas, mas sim de descreverem as regras das interações entre diversos participantes. A comunicação é modelada através de conexões permanentes e com manutenção de estado. Trata dos detalhes da interação dos participantes do processo executado em colaboração, descrevendo e restringindo as mensagens que cada participante pode enviar e quais respostas são esperadas. Enquanto a descrição da interface de um serviço em WSDL define os aspectos estáticos do serviço (operações e mensagens), a coreografia define os aspectos dinâmicos da interação entre serviços.

A coreografia de serviços possui o objetivo de prover uma descrição das interações entre participantes e em nenhum momento trata das questões de implementação desta coreografia, função que deve ser desempenhada pelas linguagens de orquestração.

Uma definição de coreografia é de grande utilidade para todas as etapas do desenvolvimento de um sistema. É útil na etapa de projeto de um sistema, quando um participante pode utilizá-la para verificar se seus processos internos irão permitir que ele participe adequadamente da coreografia. Pode ser utilizada também para a geração automática das interfaces dos serviços. Para um sistema já em produção, uma definição de coreografia pode ser utilizada para se verificar se a

interação entre as partes está ocorrendo como planejado. Outra grande utilidade está em permitir que determinado participante detecte exceções, como mensagens perdidas ou enviadas fora de ordem [14].

Um sistema pode ser desenvolvido com os conceitos de orquestração utilizando a estratégia de desenvolver os diversos serviços separadamente e então integrá-los, porém o comportamento final do sistema pode não ser o desejado quando os serviços não são projetados para trabalharem em conjunto desde o princípio. A coreografia auxilia neste processo criando uma visão global do sistema e das interações entre os diversos participantes, assim cada serviço pode ser desenvolvido ou adaptado consciente do papel que irá desempenhar [13].

IV. PADRÕES PARA EXPRESSAR ORQUESTRAÇÃO

As primeiras iniciativas de se criar padrões que facilitassem a integração de serviços focaram na orquestração e partiram da IBM e da Microsoft, com a criação das linguagens WSFL e XLANG respectivamente. Posteriormente as duas empresas combinaram suas linguagens e com o apoio de empresas como BEA Systems, SAP e Siebel criaram a linguagem BPEL4WS, que foi submetida ao órgão de padronização OASIS e assim criou-se o padrão WS-BPEL (*Web Services Business Process Execution Language*) que atualmente está na versão 2.0.

A WS-BPEL é uma linguagem que possui muitas de suas construções semelhantes às de workflows, já que esta linguagem é focada na execução de processos seqüenciais e paralelos com múltiplos passos, tal como na teoria de workflows. O meta-modelo da WS-BPEL tem como elemento central o Processo (*Process*), onde se relacionam outros elementos como Atividade (*Activity*), Correlação (*Correlation*), Compensação (*Compensation*) e Parceiro (*Partner*). É na definição do elemento "Activity" que se define o fluxo de execução de um processo através de elementos como *Se* (*If*), *Enquanto* (*While*), *Escolha* (*Pick*), *Fluxo* (*Flow*) e *Seqüência* (*Sequence*) e o fluxo de mensagens através dos elementos *Receba* (*Receive*), *Responda* (*Reply*) e *Chame* (*Invoke*).

A WS-BPEL permite ainda a definição de dois tipos de processos, os abstratos e os executáveis. A especificação define que ambos os processos compartilham dos mesmos recursos da linguagem, porém os processos abstratos permitem que determinadas etapas sejam omitidas, assumindo assim um papel de descritor ou modelo de processo. [2]

Outra linguagem que surgiu para atender a orquestração de processos foi a BPML (*Business Process Management Language*), inicialmente especificada por um grupo que incluía empresas como Intalio e Sun. Sua última atualização foi em novembro de 2002 e então foi descontinuada em favor da WS-BPEL. As duas linguagens possuem construções semelhantes para o controle de fluxo de execução e o controle de fluxo de mensagens.

Embora as linguagens WS-BPEL e BPML atendam plenamente aos requisitos de orquestração de serviços Web, elas não tratam de forma completa e consistente a coreografia de serviços. Alguns consideram que os processos abstratos da WS-BPEL possibilitam a coreografia de serviços, porém tais processos não foram desenvolvidos com este objetivo e assim não atendem a todos os aspectos da coreografia. A própria especificação do WS-BPEL sugere a adoção de outra linguagem para tratar da coreografia. A BPML, por sua vez, suporta a

coreografia de serviços apenas quando utilizada em conjunto com outro padrão que é a WSCI.

As iniciativas de desenvolvimento de linguagens e padrões para orquestração de serviços foram as primeiras a emergir e a amadurecer, principalmente por tratarem dos aspectos mais práticos e essenciais para o desenvolvimento de sistemas com arquitetura SOA.

V. PADRÕES PARA EXPRESSAR COREOGRAFIA

Os padrões de coreografia começaram a surgir por iniciativas isoladas de algumas empresas ou consórcios, porém até hoje não existe um padrão abrangente e apoiado pela maioria do mercado. Entre as primeiras iniciativas voltadas especificamente para a coreografia de serviços Web está a WSCL (*Web Services Conversation Language*), da HP, que permite essencialmente a modelagem de conversações entre serviços Web e das interações entre as partes, onde os participantes evocam ou são evocados por outros participantes [3]. A WSCL é uma linguagem com sintaxe XML, onde as conversações são modeladas utilizando a idéia de máquinas de estado e as construções conversação (*conversation*) e interações (*interactions*) para definir uma coreografia entre serviços [15]. Esta especificação foi submetida à W3C em 2002 para que servisse como um documento de referência, porém seu excesso de simplicidade e a falta de uma solução para tratamento de exceções e de outros requisitos da coreografia fizeram com que ela fosse pouco lembrada [5].

Outra iniciativa importante partiu das empresas Sun, SAP, BEA e Intalio, que desenvolveram a WSCI (*Web Service Choreography Interface*), uma linguagem mais abrangente que a WSCL e que cobre praticamente todos os aspectos da coreografia e troca de mensagens entre serviços Web. A especificação suporta correlação de mensagens, regras de seqüência, tratamento de exceção, transações e colaboração dinâmica. A WSCI também foi submetida à W3C em 2002.

O workshop da W3C sobre serviços Web de abril de 2001 apontou a necessidade de uma interface comum e de uma linguagem de composição para auxiliar na coreografia de serviços Web. A partir daí criou-se em janeiro de 2003 o grupo de trabalho para especificar um padrão para a coreografia de serviços Web, o *Web Services Choreography Working Group*. Este grupo começou seus trabalhos a partir de outros documentos: o *Web Service Architecture* que define os componentes funcionais e as relações entre os componentes do modelo de serviços Web; a especificação da linguagem WSCI e principalmente a especificação da linguagem WS-CDL (*Web Services Choreography Description Language*) que vinha sendo desenvolvida pela Oracle [13]. O grupo tem como objetivo a definição de uma linguagem que atenda aos requisitos de composição, associação, troca de mensagens e gerenciamento de estado. Os trabalhos do grupo continuam com o aperfeiçoamento da WS-CDL, que está caminhando para se tornar uma recomendação da W3C [6].

No intuito de compreendermos melhor os requisitos de uma linguagem de orquestração de serviços e de que forma são atendidos esses requisitos, iremos analisar com mais detalhes as duas linguagens mais completas e que são citadas pela literatura como referências, a WSCI e a WS-CDL.

VI. WSCI

A primeira iniciativa de criação de uma linguagem que abrangesse os principais requisitos de coreografia de serviços Web surgiu com a WSCI (*Web Service Choreography Interface*). A WSCI descreve o comportamento observável de um serviço Web, ou seja, as funcionalidades do ponto de vista de um expectador externo ao serviço. Este comportamento é expresso em termos de dependências temporais e lógicas entre as mensagens trocadas, provendo regras de seqüência, correlação, tratamento de exceções e transações. Como dito anteriormente, a coreografia não trata da definição e implementação dos processos internos de um serviço, por ser uma linguagem especificamente para coreografia a WSCI também não trata destas questões.

A WSCI trabalha como uma extensão da WSDL e descreve como as operações de uma interface WSDL podem ser coreografadas em um contexto onde os serviços Web participam. Ela assume que todas as interações entre serviços seguem e implementam um determinado processo. A WSCI permite descrever também como a coreografia dessas operações deve tratar questões como correlação de mensagens, tratamento de exceções e controle de transações [7]. A WSCI possui as seguintes funcionalidades:

- **Coreografia de mensagens:** descreve a ordem que as mensagens podem ser enviadas ou recebidas em uma determinada troca de mensagens, as regras que governam esta ordenação e os limites de uma troca de mensagens (quando começa e quando termina).
- **Transações e compensações:** descreve quais operações são executadas de forma transacional e permite que sejam definidas ações a serem tomadas para se reverter operações caso uma transação seja desfeita.
- **Tratamento de exceções:** descreve como um serviço Web deve reagir quando uma condição excepcional ocorre e provê uma descrição de possíveis caminhos alternativos.
- **Gerenciamento de thread:** descreve quando e como um serviço Web é capaz de gerenciar múltiplas conversações com um ou vários participantes.
- **Propriedades e seletores:** propriedades são utilizadas para referenciar um valor, semelhante a uma variável nas linguagens de programação. Os seletores definem como o valor de uma propriedade é obtido a partir de uma mensagem recebida.
- **Conectores:** descrevem como as operações disponibilizadas pelos diferentes serviços Web interagindo em determinada troca de mensagens são conectadas. A WSCI permite o mapeamento entre operações, fazendo com que algumas operações quando invocadas gerem a chamada de outras operações em outros serviços Web.
- **Contexto operacional:** permite descrever diferentes comportamentos para um mesmo serviço Web quando interagindo em diferentes contextos.
- **Participação dinâmica:** descreve como determinado serviço Web será selecionado dinamicamente. A seleção é feita baseada em alguns critérios, como por exemplo, no conteúdo de mensagens recebidas.

A WSDL apresenta uma descrição estática de um serviço Web, deixando muitas questões a serem esclarecidas para que um cliente deste serviço possa utilizá-lo da forma correta. Entre os aspectos deste serviço que precisam ser esclarecidos estão [7]:

- **Coreografia:** A interface WSDL não descreve a ordem correta de chamada das operações. Neste exemplo, não está definido claramente que a operação de confirmação de compra de uma passagem deve ser executada somente após uma requisição de compra.
- **Correlação:** Não está descrito na interface WSDL que uma mensagem de confirmação de compra está relacionada com uma requisição de compra que a procedeu.
- **Transações:** A interface WSDL não descreve se as operações são executadas de forma transacional.

A WSCI foi a primeira linguagem a suportar a coreografia de serviços de forma consistente, trazendo uma solução abrangente e padronizada. Em seu passado a WSCI foi muito importante por motivar uma discussão em torno do tema coreografia. No entanto e embora até hoje a WSCI tenha grande reconhecimento e sirva como uma referência no assunto, atualmente não se dá mais continuidade ao seu desenvolvimento e os atuais esforços para desenvolvimento de um padrão para coreografia estão focados na linguagem WS-CDL [7] [15].

VII. WS-CDL

Assim como a WSCI, WS-CDL (*Web Services Choreography Description Language*) é uma linguagem com sintaxe XML e que tem por objetivo permitir a descrição das interações entre múltiplos participantes através da definição, sobre uma visão global, de seus comportamentos públicos e observáveis, onde mensagens trocadas ordenadamente resultam na execução de determinada atividade. Com a WS-CDL define-se uma contrato global contendo a definição das regras de ordenação e restrições envolvidas na troca de mensagens. Cada participante pode se basear nesse contrato para implementar e testar soluções que atendam e se integrem adequadamente aos outros participantes [16].

A WS-CDL é uma linguagem que está sendo desenvolvida pelo *Web Services Choreography Working Group*, um grupo de trabalho da W3C que conta com o apoio de empresas como Oracle, Novell e Adobe. Seu principal objetivo está em especificar uma linguagem declarativa, que defina a partir de um ponto de vista global o comportamento observável e comum a um grupo de participantes que cooperam para a execução de uma atividade. A linguagem deve especificar especialmente as trocas de informações e as regras de ordenação que devem ser satisfeitas [16]. Outros objetivos da WS-CDL são:

- **Reusabilidade:** Uma mesma definição de coreografia precisa ser reuzável por diferentes participantes operando em diferentes contextos.
- **Cooperação:** Coreografias devem definir a seqüência de troca de mensagens entre dois ou mais participantes ou processos, descrevendo como eles devem cooperar.
- **Múltipla colaboração:** Coreografias podem ser definidas envolvendo qualquer número de participantes e processos.
- **Semântica:** Coreografias devem ter descrições semânticas para todos os seus componentes.
- **Composição:** Coreografias podem ser combinadas para formarem uma nova coreografia, podendo ser reutilizadas em diferentes contextos.
- **Modularidade:** Coreografias podem ser definidas a partir da inclusão de outras coreografias.
- **Colaboração orientada a informações:** Coreografias descrevem como os participantes

avançam na execução de um processo através das informações trocadas.

- **Sincronização de informações:** Coreografias permitem que seus participantes compartilhem informações.
- **Tratamento de exceções:** Coreografias podem definir como tratar casos de exceção que eventualmente ocorram durante a execução da coreografia.
- **Transações:** Permite que coreografias trabalhem de forma transacional e que se controle a execução de colaborações de longa duração.
- **Composição de padrões:** A especificação WSDL permite que se trabalhe em conjunto e de forma complementar com outras especificações como WS-Reliability, WSCAF, WS-Security, WS-BPEL, etc.

Para que participantes se comuniquem de forma colaborativa, serviços devem cumprir suas responsabilidades e seguir as regras de relacionamento definidas em comum acordo a todos os participantes. A colaboração é definida através de um conjunto de regras de restrição e de ordenação.

Uma descrição de uma coreografia com a WS-CDL é contida em um pacote (*package*) que é essencialmente um agrupador para um conjunto de atividades que podem ser desempenhadas por um ou mais participantes. A linguagem dispõe de três tipos de atividades, as atividades para controle do fluxo de execução, as atividades do tipo unidade de trabalho (*workunit*) e as atividades básicas. Um atividade pode incluir diversas sub-atividades.

Os tipos de atividades para controle do fluxo de execução são seqüenciais (*sequence*), paralelas (*parallel*) e as de escolha (*choice*). Uma atividade de seqüência descreve uma ou mais atividades que são executadas de forma seqüencial, enquanto uma atividade paralela descreve uma ou mais atividades que podem ser executadas independentemente da ordem ou ao mesmo tempo. A atividade de escolha descreve a execução de uma atividade escolhida entre um conjunto de opções, sendo que esta escolha pode ser feita com base em uma expressão lógica ou a partir da ocorrência de determinado evento.

O segundo tipo de atividade é chamado de unidade de trabalho (*workunit*). Uma unidade de trabalho descreve a execução condicional de uma atividade ou a execução repetida de uma atividade.

As atividades seqüências, paralelas, de escolha e as unidades de trabalho permitem a descrição básica de um controle de fluxo, semelhante ao que encontramos nas linguagens de programação. Existe semelhança também com as construções para controle de fluxo disponibilizadas pela linguagem WS-BPEL [19].

O terceiro tipo de atividade, as atividades básicas, incluem a interação (*interaction*), a atividade nula (*NoAction*), a atividade silenciosa (*SilentAction*), a atividade de atribuição (*assign*) e a atividade de execução (*perform*). As atividades nulas e silenciosas descrevem pontos de uma coreografia onde algum participante não desempenha nenhuma atividade ou desempenha alguma atividade transparente para os outros participantes, respectivamente. A atividade de atribuição é utilizada para transferir o valor de uma variável para outra. A atividade de execução é utilizada para a chamada de outra coreografia, que pode estar definida no mesmo pacote ou em um pacote importado [16].

Um dos elementos mais importantes da WS-CDL é sem dúvida a atividade de interação (*interaction*). Uma interação descreve uma troca de mensagens entre participantes, sobre a ponto de

vista do receptor da mensagem. O propósito de uma interação pode ser efetuar uma requisição, fornecer uma resposta ou efetuar uma requisição, que requer uma resposta. Uma atividade de interação descreve os participantes envolvidos, a informação que será trocada e o canal utilizado para transmitir a mensagem. A informação enviada ou recebida em uma interação é descrita por uma variável (*variable*) [19].

Na linguagem WS-CDL as variáveis são utilizadas para armazenar informações relativas ao domínio da aplicação, informações de estado da coreografia, ou informações sobre os canais de interação. As variáveis possuem um tipo associado a elas (*informationType*) e são acessadas utilizando expressões XPath.

A ligação entre coreografias WS-CDL e operações descritas em interfaces WSDL é feita através de um canal (*channel*). Assim todo comportamento exposto em uma coreografia possui um canal. Um canal possui uma referência (*reference token*) que descreve como contatar o receptor das mensagens do canal e também uma identidade (*identity token*) que é utilizada para se fazer a correlação das mensagens trocadas através do canal.

Os símbolos (*tokens*) da linguagem WS-CDL são nomes utilizados para referenciar pedaços específicos do conteúdo de uma variável. Os localizadores de símbolos (*token locators*) descrevem como os tokens podem ser encontrados utilizando expressões XPath [16].

A linguagem WS-CDL utiliza o elemento coreografia (*choreography*) para agrupar a atividade raiz e as descrições opcionais dos elementos exceção (*exception*) e finalizador (*finalizer*). O elemento exceção pode ser ativado caso alguma exceção ocorra durante a execução da coreografia, isto permite que se executem atividades que revertam as ações que foram iniciadas pela coreografia, mas foram interrompidas. O elemento finalizador pode ser ativado quando uma coreografia é finalizada com sucesso, permitindo que se confirmem as atividades executadas (*commit*), ou ainda que se desfaçam as atividades executadas devido a alguma falha durante a execução de outra coreografia (*rollback*).

Um pacote (*package*) descreve quais papéis (*RoleTypes*) estão presentes em uma coreografia e os comportamentos por ele desempenhados (*Behavior*). Descrevem-se ainda quais comportamentos serão exibidos durante a interação de dois papéis (*RelationshipType*). Finalmente, um pacote pode conter a descrição de agrupamentos lógicos de papéis (*ParticipantType*), permitindo que uma organização em particular possa desempenhar mais de um papel em uma coreografia [16].

O estudo e o desenvolvimento de padrões para coreografia de serviço é um tema relativamente novo e que ainda deixa muitas questões em aberto para discussões. A linguagem WS-CDL, por ser uma das últimas iniciativas de desenvolvimento de um padrão e apesar de herdar todo o conhecimento e estudos feitos anteriormente em linguagens como WSCI, ainda apresenta muitos pontos em aberto e outros mal resolvidos.

Alguns pontos fracos da linguagem que merecem destaque são [19]:

- **Falta de uma base formal.** Apesar de se adotar a terminologia *pi-Calculus*, a ligação com este formalismo não é muito clara. A falta de formalismo dificulta o desenvolvimento de ferramentas para validação de coreografias que garantam a interoperabilidade entre serviços Web.

- **Falta de um mapeamento claro entre a WS-CDL e a WSDL.** Ao mesmo tempo em que deixa muitos detalhes da ligação das duas em aberto, a WS-CDL restringe uma coreografia e uma interface WSDL em específico. Não se podem utilizar serviços que sejam equivalentes funcionalmente, já que a ligação é feita pelo nome da operação de uma interface WSDL. Isso impede que em tempo de execução seja feita a seleção dos participantes com base em suas capacidades funcionais, restringindo a apenas aqueles que implementam uma interface WSDL específica.
- **Somente coreografias entre dois participantes.** A WSDL não permite definir coreografias onde exista a interação simultânea entre mais de dois participantes, ou seja, apenas são permitidas interações binárias.

Outro ponto desfavorável da WS-CDL é a falta de apoio por parte de grandes empresas do mercado. A linguagem foi concebida pela Oracle, que tem contribuído para o seu desenvolvimento até hoje, recebendo apoio de empresas como Sun, Adobe, Nortel e Novell. Porém, a falta de apoio de empresas como Microsoft e IBM, que são muito influentes na área, torna seu futuro um tanto duvidoso [15].

Apesar das críticas, a WS-CDL é sem dúvida o padrão mais maduro e desenvolvido atualmente para a descrição de coreografias. O fato de haver um grupo de trabalho da W3C empenhado em dar continuidade ao seu desenvolvimento com certeza fará a linguagem evoluir no sentido de resolver muitas de suas atuais falhas. Adiante vemos um exemplo de ferramenta que torna a linguagem mais acessível ao público.

VIII. ANÁLISE DA FERRAMENTA PI4SOA

Descrever coreografias utilizando a WS-CDL não é algo trivial sem o auxílio de ferramentas. Exigir que o usuário da linguagem saiba utilizar todas as suas construções, possíveis atributos e parâmetros, com certeza deixará o usuário muito suscetível a erros e será algo proibitivo em relação a produtividade. O fato da descrição da coreografia ser feita na forma de um documento XML torna a sua edição ainda mais susceptível a erros e dificulta a leitura da coreografia. A visualização de forma esquemática é mais apropriada para permitir a interação e a compreensão por parte dos usuários.

Para que a WS-CDL tenha uma larga adoção e se torne viável em termos de produtividade, é essencial o uso de uma ferramenta que torne transparente ao usuário os detalhes da linguagem e que permita ao usuário focar na visão de alto nível da comunicação entre serviços, objetivo principal da coreografia de serviços. Tal ferramenta deve permitir não só a geração automática da WS-CDL, mas também propiciar uma visão gráfica de uma coreografia, tornando a linguagem acessível também aos usuários não técnicos, como analistas de negócio que contribuem para a definição dos processos.

Pensando nisso a empresa *Pi4 Technologies Foundation* criou a ferramenta Pi4SOA, cujo objetivo principal é prover uma ferramenta que auxilie no desenvolvimento de sistemas distribuídos e orientados a serviço (SOA). Seus idealizadores partem do princípio que um sistema com uma boa arquitetura pode ser desenvolvido com visão *top-down*, utilizando a linguagem WS-CDL, que pode agir como um modelo para o desenvolvimento de serviços ou para a integração de aplicações legadas. Esta metodologia de

desenvolvimento garante que todos os serviços se adaptem ao comportamento esperado de uma coreografia e garante a interoperabilidade destes serviços para alcançar um objetivo [17].

A iniciativa para o desenvolvimento da ferramenta partiu de um dos líderes da especificação da WS-CDL, Steve Ross-Talbot, que acredita na WS-CDL como um grande auxilio para o desenvolvimento aplicações distribuídas e orientadas a serviços. Segundo ele, um dos problemas no desenvolvimento deste tipo de sistema é que as pessoas constroem serviços com uma visão individualista, focando apenas na funcionalidade de determinado serviço e não com uma visão global e mais abrangente do processo, onde diversos serviços interagem. Ainda segundo ele, esses problemas fazem com que sistemas desenvolvidos com uma arquitetura orientada a serviços nem sempre atendam aos requisitos do negócio, e certamente a WS-CDL traz grandes melhorias neste aspecto [18].

Embora a ferramenta esteja em processo de desenvolvimento e o padrão WS-CDL não esteja oficialmente finalizado, já existem diversos recursos disponíveis e que permitem a definição completa de uma coreografia e a geração da interface WSDL, WS-BPEL ou Java. A tabela 3 mostra os principais recursos disponíveis.

Recurso	Descrição
Importação de coreografia	Permite que uma interface WS-CDL já definida seja importada para a ferramenta.
Exportação de coreografia	Permite que uma coreografia seja exportada para WS-CDL, relatório HTML, para *Service Description Model* ou para um diagrama UML.
Validação de coreografia	Regras padronizadas para garantir a consistência do modelo e para reforçar a semântica da WS-CDL.
Geração de interface de serviço	Geração da interface dos serviços para WSDL 1.1 ou WSDL 2.
Geração de serviço	Geração da implementação dos serviços para a linguagem Java ou para WS-BPEL.
Publicação de serviço	Publicação de serviço nos servidores Apache Axis, J2EE e processos WS-BPEL no servidor ActiveBPEL.
Modelagem de serviços	Ferramenta gráfica para modelagem de serviços.
Validação de serviços	Ferramenta para validação de serviços.
Java Runtime	Monitora a execução de coreografias para identificar mensagens trocadas fora de ordem.

Tabela 1. Recursos da ferramenta PI4SOA.

O desenvolvimento de uma coreografia utilizando a ferramenta exige que o usuário conheça os principais conceitos da linguagem WS-CDL. A própria documentação da ferramenta, que é bastante pobre, sugere que se use a especificação da WS-CDL como documentação.

IX. CONCLUSÕES

A implementação de arquiteturas orientadas a serviços (SOA) e a utilização de serviços Web para

o desenvolvimento de aplicações distribuídas exigem a criação de métodos específicos para tratar a integração de serviços Web e para coordenar a execução de processos. Os conceitos de orquestração e coreografia vêm sendo muito discutidos com a crescente complexidade dos sistemas baseados nessas tecnologias. O conceito de coreografia, embora ainda não tão maduro quando o de orquestração, vem recebendo cada vez mais atenção à medida que crescem os problemas na organização das interações entre serviços.

O estudo dos conceitos e a análise das linguagens e ferramentas disponíveis permitem nos posicionar quanto ao que já é possível fazer e quanto ao que esperar da evolução deste tema no futuro próximo. Neste artigo foi possível compreender a evolução dos esforços de desenvolvimento de um padrão para coreografia de serviços. Após algumas iniciativas, a linguagem mais desenvolvida e até o momento mais promissora é a WS-CDL. Porém é um trabalho ainda em desenvolvimento e que tem alguns problemas em aberto para serem resolvidos. O mercado já vem trabalhando em ferramentas para tornar a WS-CDL mais acessível para os usuários, como a iniciativa de desenvolvimento da ferramenta PI4SOA.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] STAL, Michael. Using Architectural Patterns and Blueprints for Service-Oriented Architecture. IEEE Computer Society, 2006.

[2] OASIS. Web Services Business Process Execution Language, versão 2.0 draft 2, 20 de novembro de 2006.

[3] PELTZ, Chris. Web Services Orchestration and Choreography. IEEE Computer Society, outubro de 2003.

[4] BARROS, Alistair; DUMAS, Marlon; OAKS, Phillipa. A Critical Overview of the Web Services Choreography Description Language (WS-CDL). BPTrends, 2005.

[5] W3C. Web Services Conversation Language (WSCL), W3C note, versão 1.0, 14 de março 2002.

[6] W3C. Web Services Choreography Working Group Charter. Disponível em: http://www.w3.org/2005/12/wscwg-charter.html. Acesso em: Janeiro de 2007.

[7] W3C. Web Service Choreography Interface (WSCI), W3C note, versão 1.0, 8 de agosto de 2002.

[8] DURVASULA, Surekha; GUTTMANN, Martin; KUMAR, Ashok. Martin Guttmann SOA Practitioners' Guide, Part 1. Why Services-Oriented Architecture? 15 de setembro de 2006.

[9] JOHNSON, Ralph E.; FOOTE, Brian. Designing Reusable Classes, Journal of Object-Oriented Programming, vol. 1, num. 2, 1988.

[10] OASIS. Reference Model for Service Oriented Architecture. Committee Draft 1.0, 7 de Fevereiro de 2006.

[11] W3C. SOAP Version 1.2 Part 0: Primer, 24 de junho de 2006.

[12] NEWCOMER, Eric. Understanding Web Services – XML, WSDL, SOAP, and UDDI. United States: Addison-Wesley, 2002. 332p.

[13] KAVANTZAS, Nickolaos. Aggregating Web Services: Choreography and WS-CDL. Oracle Corporation. 5 de abril de 2007.

[14] Where do you apply SOA? http://www.ebpml.org/soa.htm#coord. Visitado em 14/04/2007.

[15] HAVEY, Mike. Essential Business Process Modeling. O'Reilly, 2005. 350p.

[16] W3C. Web Services Choreography Description Language (WS-CDL), W3C candidate recommendation, versão 1.0, 9 de novembro de 2005.

[17] Pi4 Technologies Foundation wiki. http://www.pi4tech.org/tiki-index.php. Visitado em: 02/06/2007.

[18] BHARTI, Nitini. Choreography tools deliver on promise of SOA. Disponível em: http://searchwebservices.techtarget.com/originalContent/0,289142,sid26_gci1095769,00.html Visitado em: 02/06/2007.

[19] BARROS, Alistair; DUMAS, Marlon; OAKS, Phillipa. A Critical Overview of the Web Service Choreography Description Language (WS-CDL). BPTrends, março de 2005.

[20] W3C. Web Services Description Language (WSDL) 1.1, 15 de março 2001. Disponível em: http://www.w3.org/TR/wsdl/

[21] OASIS. UDDI versão 3.0.2, 19 de outubro de 2004. Disponível em: http://www.oasis-open.org/committees/uddi-spec/doc/spec/v3/uddi-v3.0.2-20041019.htm