Composição de serviços Web semânticos

Guilherme Ribeiro Morisson

JUIZ DE FORA

JULHO, 2010

Composição de serviços Web semânticos

GUILHERME RIBEIRO MORISSON

Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Bacharel em Ciência da Computação

Orientador: Jairo Francisco de Souza

JUIZ DE FORA

JULHO, 2010

COMPOSIÇÃO DE SERVIÇOS WEB SEMÂNTICOS

Guilherme Ribeiro Morisson

MONOGRAFIA SUBMETIDADA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

_____

Jairo Francisco de Souza, orientador.
MSc em Engenharia de Sistemas e Computação COPPE/UFRJ

_____

Regina Maria Maciel Braga Villela
DSc em Engenharia de Sistemas e Computação COPPE/UFRJ

_____

Fernanda Cláudia Alves Campos
DSc em Engenharia de Sistemas e Computação COPPE/UFRJ

JUIZ DE FORA, MG – BRASIL

JULHO, 2010

Resumo

Nos dias de hoje, serviços estão entre os mais importantes recursos da Web e têm sido utilizados na realização de diversas tarefas, promovendo a reutilização de recursos e auxiliando na integração entre sistemas. A composição de serviços Web vem da necessidade de se juntar a funcionalidade de mais de um serviço para criar um novo serviço mais elaborado, que realize uma tarefa mais complexa, mantendo a individualidade e reusabilidade dos serviços originais. A aplicação da semântica na composição de serviços Web, que tem sido bastante estudada nos meios acadêmicos, visa à automação da criação de composições, que atualmente são realizadas manualmente.

Abstract

Agradecimentos

Agradeço a toda minha família, em especial à minha mãe que sempre me cobrou para que eu fizesse o meu melhor e me deu todas as condições para que eu chegasse até aqui. Agradeço à minha irmã pelas boas conversas e risadas. Agradeço à minha namorada, que esteve todo o tempo ao meu lado me dando força para que eu pudesse superar os momentos mais difíceis. Agradeço aos meus amigos pelos bons momentos que passamos durante esta longa caminhada e, por fim, agradeço ao meu orientador que me ajudou a completar mais este desafio.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Reduções

API	*Application Programming Interface*
B2B	*Business-to-business*
CORBA	*Common Object Request Broker Architecture*
DCOM	*Distributed Component Object Model*
EAI	*Enterprise Application Integration*
EJB	*Enterprise JavaBeans*
ERP	*Enterprise Resource Planning*
FLOWS	*First-order Logic Ontology for Web Services*
IIF	*INFRAWEBS Integration Framework*
IRI	*Internationalised Resource Identifier*
IRS	*Internet Reasoning Service*
OASIS	*Organization for the Advancement of Structured Information Standards*
OCML	*Operational Conceptual Modelling Language*
OWL	*Web Ontology Language*
OWL-S	*Web Ontology Language for Services*
PSL	*Process Specification Language*
RDF	*Resource Description Framework*
ROWS	*Rules Ontology for Web Services*
SAWSDL	*Semantic Annotations for WSDL*
SOAP	*Simple Object Access Protocol*
SWS	*Semantic Web Service*
SWSF	*Semantic Web Services Framework*
SWSI	*Semantic Web Services Initiative*
SWSL	*Semantic Web Services Language*

SWSO	*Semantic Web Services Ontology*
UDDI	*Universal Description, Discovery and Integration*
URL	*Uniform Resource Locator*
USDL	*Unified Service Description Language*
WS-BPEL	*Web Service – Business Process Execution Language*
WS-CDL	*Web Services Choreography Description Language*
WSDL	*Web Service Description Language*
WSDL-S	*Web Service Description Language Semantics*
WSMF	*Web Service Modeling Framework*
WSML	*Web Service Modeling Language*
WSMO	*Web Service Modeling Ontology*
WSMX	*Web Service Modelling eXecution environment*
XML	*eXtensible Markup Language*
XSD	*XML Schema Definition*

1. INTRODUÇÃO

Nos dias de hoje, serviços estão entre os mais importantes recursos da Web. Por serviço, entendemos *sites* da Web que não fornecem informação meramente estática, mas permitem efetivar alguma ação ou mudança no seu estado atual, como a venda de um produto ou o controle de um dispositivo físico. Para se fazer uso de um serviço Web, um software precisa de uma descrição do serviço que seja interpretável por um computador e os meios pelos quais o serviço deve ser acessado (LARA et al, 2007). Um tema de grande importância dentro do escopo de serviços Web é a composição de serviços, que vem a existir pela necessidade de se unir um conjunto de serviços para realizar uma determinada tarefa. Neste trabalho falaremos sobre a adição de semântica às técnicas de composição utilizadas atualmente, com a finalidade de possibilitar um maior nível de automatização das fases de composição de um serviço Web.

Este capítulo tratará da motivação para o desenvolvimento deste trabalho, falando das necessidades existentes atualmente na utilização de serviços Web, falaremos também sobre o objetivo do desenvolvimento do trabalho e como ele está organizado.

1.1. MOTIVAÇÃO

Serviços Web têm sido amplamente utilizados na realização de diversas tarefas por toda a rede e, cada vez mais, são vistos como uma boa maneira de promover a reutilização de recursos, além de servirem como facilitadores de integração entre sistemas.

Este trabalho tem como motivação mostrar a importância da composição de serviços Web para a realização de tarefas mais elaboradas, como uma maneira de manter a reusabilidade e o padrão de granularidade de cada serviço. Serão estudados os tipos e padrões existentes para composição e as dificuldades de se automatizar a criação de uma composição. Para este fim, serão apresentadas algumas abordagens e um estudo mais aprofundado da abordagem semântica.

1.2. OBJETIVO

O objetivo deste trabalho é descrever e demonstrar possíveis soluções para as dificuldades e problemas encontrados atualmente na composição de serviços Web,

mostrando a importância dos padrões e conceitos da Web semântica para o futuro da Web. Isto será feito através do estudo de diversos padrões, de abordagens realizadas e de ferramentas disponíveis atualmente para este fim.

1.3. ORGANIZAÇÃO DO TRABALHO

O trabalho está dividido em cinco capítulos, sendo este o primeiro, com a finalidade de introduzir o leitor ao universo dos serviços Web; no segundo falaremos sobre composição de serviços, serão definidos alguns conceitos básicos, descreveremos padrões utilizados atualmente para orquestração e coreografia e problemas encontrados com a forma atual de se compor serviços; o terceiro capítulo introduz o paradigma dos serviços Web semânticos, com padrões de descrição semântica e apresentação de algumas abordagens práticas da utilização de descrições semânticas em serviços Web; o capítulo 4 reproduz dois exemplos que foram apresentados na conferência realizada pela AAAI (*Association for the Advancement of Artificial Intelligence)* que mostram a utilização de descrições semânticas no momento da descoberta e invocação de serviços Web, além de apresentações das ferramentas utilizadas; o trabalho é encerrado com o quinto capítulo, que apresenta uma conclusão do que foi estudado, uma avaliação da situação atual das abordagens semânticas e propostas para trabalhos futuros.

2. COMPOSIÇÃO DE SERVIÇOS

Nos dias de hoje, um número crescente de empresas e organizações implementam o seu núcleo de serviços de negócio e terceirizam grande quantidade de serviços de outras aplicações através da Internet. Desta forma, a habilidade de selecionar e integrar serviços que estão espalhados pela grande rede de uma forma efetiva e eficiente se torna um importante passo para o desenvolvimento de aplicações Web (RAO & SU, 2005).

A composição de serviços web nasce da necessidade de se criar serviços mais elaborados a partir de serviços já existentes e integrar negócios (DAMASCENO, 2009). A composição visa selecionar e interconectar serviços web independentes de acordo com um objetivo a ser alcançado, permitindo que serviços de diferentes provedores sejam colocados juntos para criar serviços mais sofisticados, promovendo uma maior flexibilidade na construção de aplicações a partir de serviços primitivos e de uma forma *plug-and-play* (LÉCUÉ & DELTEIL, 2007). Por exemplo, um sistema de vendas de uma agência de viagens pode receber uma solicitação de um cliente a partir de um site, acessar sistemas de redes de hotéis para verificar se há quartos disponíveis e fazer uma reserva, ou sistemas de empresas de linhas aéreas para verificar disponibilidade de vôos e comprar uma passagem na data e horário desejados pelo cliente. Percebe-se no exemplo a possibilidade de uso de pelo menos cinco serviços web, que devem se comportar, dependendo das mensagens de resposta recebidas, de várias possíveis maneiras. Se o serviço que verifica a disponibilidade de vôos retorna uma resposta negativa, a compra da passagem não acontece e a reserva do hotel não é mais necessária, pois o cliente não conseguirá se deslocar ao local desejado. Isso nos mostra que a composição de serviços web é muito mais do que organizar serviços em sequência e aguardar que eles sejam executados. Composições representam processos de negócio, que possuem regras de negócio, que devem ser respeitadas a fim de que se chegue ao objetivo desejado. A figura 2.1 mostra um diagrama que representa as atividades do processo de compra de pacote de viagem, descrito acima.

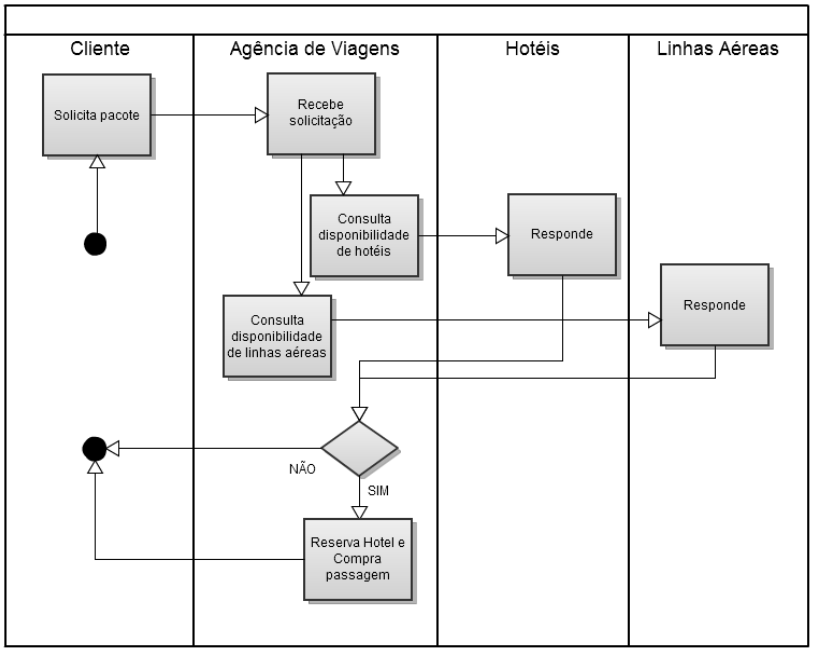

Figura 2.1: Diagrama de atividades do exemplo compra de pacote de viagem

De acordo com TSANG (2006), a composição de serviços Web tem sido uma área de interesse de pesquisas e desenvolvimento em vários domínios distintos, como, por exemplo, *frameworks* baseados em componentes, *workflows* de domínio empresarial, intercâmbio eletrônico de dados e frameworks de B2B (*business-to-business*) baseados em XML (*eXtensible Markup Language*).

Aplicações de comércio eletrônico dependem de *frameworks* de comunicação entre objetos distribuídos, como CORBA (*Common Object Request Broker Architecture*), DCOM (*Distributed Component Object Model*), EJB (*Enterprise JavaBeans*) e outras tecnologias como EAI (*Enterprise Application Integration*) e ERP (*Enterprise Resource Planning*). No domínio empresarial tem-se a necessidade de uma automatização e uma integração rápida e dinâmica de processos de negócios, para interconectar e gerenciar a comunicação entre os diversos sistemas (TSANG, 2006).

Esta seção visa discutir a composição de serviços Web, apresentando primeiramente os conceitos básicos necessários para se entender o problema da composição. Em seguida, são apresentadas as tecnologias utilizadas na descoberta de serviços, além das características de dois tipos de composição, a orquestração e a coreografia de serviços. Por fim, são apresentadas as dificuldades encontradas na utilização dos padrões atuais para composição de serviços Web.

2.1. CONCEITOS BÁSICOS

A composição de serviços Web engloba dois sub-conceitos que costumam ser confundidos entre si: orquestração e coreografia. Em uma arquitetura orientada a serviços, uma orquestração, assim como a coreografia, é utilizada para criar um novo serviço utilizando-se serviços já existentes. Em uma orquestração, um serviço central é responsável por coordenar (orquestrar) a chamada de outros serviços, segundo o que foi definido em um *workflow*, sendo que estes serviços, com exceção do orquestrador, não têm conhecimento de que fazem parte de uma composição. A orquestração pode ser comparada a um método da orientação de objetos que faz chamadas de outros métodos. Já na coreografia, não existe a figura de um serviço coordenador, que controla e coordena os demais serviços. Neste tipo de composição, cada serviço envolvido "sabe" que faz parte de uma composição e que precisa interagir com outros serviços de forma ordenada para que a composição resultante tenha sucesso (ALONSO & CASATI, 2004). A figura 2.2 mostra como funciona a coreografia de serviços Web. Percebe-se a idéia da colaboração entre os serviços, que utilizam trocas de mensagens com outros serviços (*invoke/reply*) para validar as interações e chegar ao objetivo da composição. Na figura 2.3, que representa a orquestração, o serviço Web 1 chama o coordenador, que recebe a solicitação (*receive*), faz a chamada (*invoke*) de vários serviços respeitando um fluxo de processo e depois responde ao serviço Web 1 o que foi solicitado (*reply*).

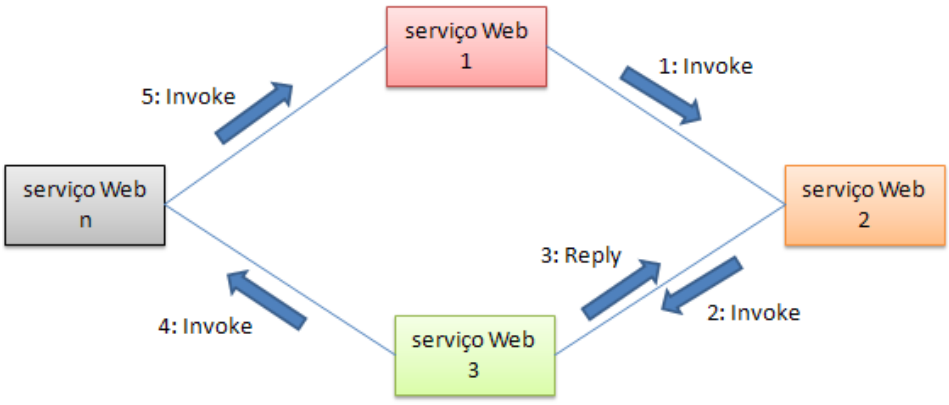

Figura 2.2: Coreografia

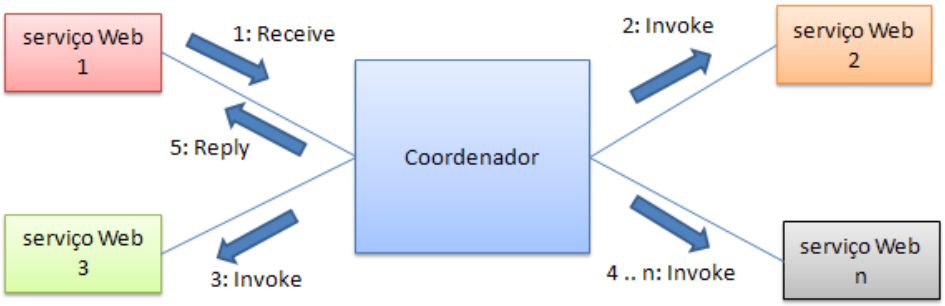

Figura 2.3: Orquestração

Em uma composição, serviços podem ser combinados de diferentes formas, sendo as principais: sequencial, paralela e condicional (DAMASCENO, 2009). Na figura 2.4(a), um serviço WS2 só começa a ser executado após o fim da execução do serviço anterior WS1, ou seja, depende do sucesso da total execução do WS1, configurando uma composição sequencial; a figura 2.4(b) mostra que, na composição paralela, a execução dos serviços WS1 e WS2 ocorre de forma simultânea; e, na figura 2.4(c) temos uma representação da composição condicional, em que é respeitada uma determinada condição para determinar qual dos serviços será executado.

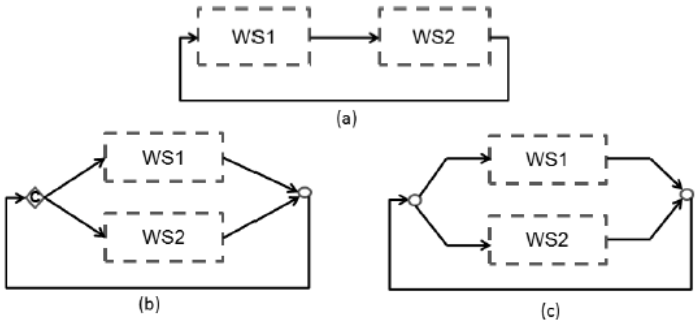

Figura 2.4: Formas básicas de Composições (DAMASCENO, 2009)

A composição de serviços pode ser estática ou dinâmica. A composição estática é declarada na forma de um *script*, onde são listados os serviços a serem executados e são especificados se os resultados de determinados serviços serão mapeados como parâmetro de entrada para outros serviços ainda a serem executados (NANDIGAM et al., 2008). A composição feita de modo estático apresenta algumas características importantes: descreve a interação dos serviços sem referenciar a nenhum modelo de implementação ou execução, permite o aninhamento de outros serviços compostos, e, mantém um alto nível de

especificação da composição, assegurando a sua exequibilidade (TSANG, 2006). A composição dinâmica visa à manutenção de uma visão atualizada dos serviços e seus estados atuais para se utilizar do serviço disponível mais adequado no momento de uma requisição. Essa preocupação com a disponibilidade se deve à grande volatilidade da Web: na arquitetura dos ambientes de rede utilizados na Internet não existe estabilidade de conexão entre dois nodos, e componentes (serviços) podem entrar e sair da rede a qualquer momento (REIS & CÉLIO, 2008). Os conceitos da composição dinâmica em geral são similares aos da composição estática, a principal diferença é que, na composição dinâmica, a listagem dos serviços e os mapeamentos de resultados ocorrem em tempo de execução.

2.2. DESCOBERTA DE SERVIÇOS WEB

Para que seja possível compor serviços Web, é desejável que exista uma forma de descobrir (e publicar) os serviços através da Web. Tal funcionalidade é oferecida pelo UDDI (*Universal Description, Discovery and Integration*). O *framework* do UDDI define um modelo de dados para registro e descoberta de informações de negócio, incluindo publicações de serviços empresariais (OASIS, 2004). A especificação do UDDI nasceu de um consórcio entre as empresas Microsoft, IBM e Ariba. A especificação objetiva a criação de um diretório logicamente centralizado e fisicamente distribuído, através de replicações. Empresas registram suas informações de contato (telefone, FAX, caixa postal, página na Web etc.), informações sobre o tipo de negócio ao qual seu serviço atende, entre outras coisas. Outras empresas podem procurar por informações registradas no UDDI para encontrar algum serviço que atenda a alguma necessidade, e descobrir informações de serviços Web específicos, através da URL (*Uniform Resource Locator*) do arquivo WSDL (*Web Service Description Language*), o qual descreve as informações técnicas do serviço, ou seja, como acessar e fazer uso daquele serviço (NEWCOMER, 2002). A figura 2.5 representa como é feita a publicação, descoberta e acesso a um serviço através do registro UDDI. O provedor do serviço cria a descrição do serviço Web e a publica no repositório UDDI. A partir daí o serviço pode ser descoberto através de uma busca por um cliente, que tem acesso ao arquivo WSDL do serviço e pode fazer a comunicação com o mesmo através de mensagens SOAP (*Simple Object Access Protocol*).

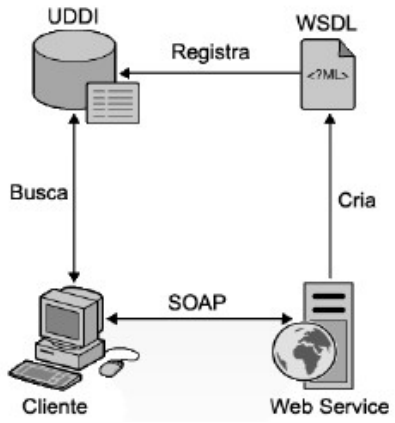

Figura 2.5: Visão geral do UDDI (DAMASCENO, 2009)

A estrutura do registro UDDI define uma hierarquia através de quatro elementos principais presentes no seu modelo de dados: *businessService, businessEntity, bindingTemplate* e *tModels*. A figura 2.6 representa um exemplo do modelo de dados do UDDI que é explicado em seguida.

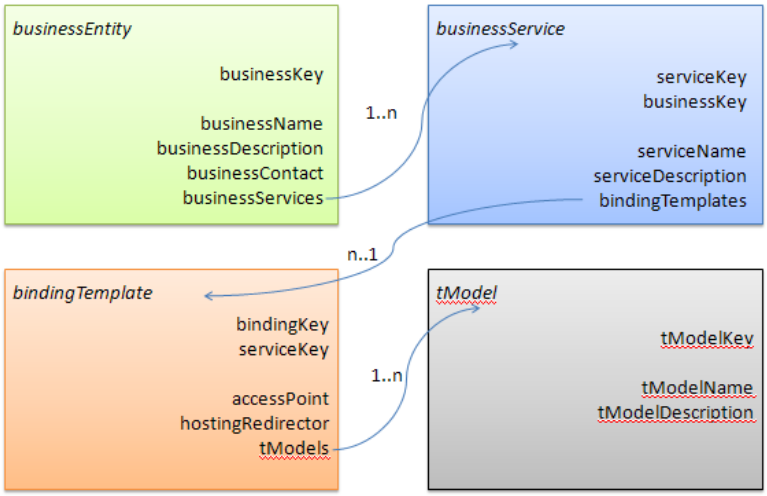

Figura 2.6: Modelo de dados do UDDI

O elemento *businessEntity* representa o provedor de um serviço Web. Nele são registrados os dados de contato, categoria de negócio, serviços oferecidos de uma determinada organização. O *businessService* é elemento filho do *businessEntity* e descreve a função de negócio de determinado serviço. Indicadores únicos indicam as categorias às quais o serviço Web pertence. Os *bindingTemplate,* por sua vez, são filhos de um *businessService* e especificam os detalhes técnicos do serviços, tais como informações de

como fazer as ligações para acessá-lo. Por fim, os *tModels* que representam registros de determinados conceitos, como taxonomia, transportes, assinaturas digitais etc. Normalmente os *tModels* contêm informações técnicas do arquivo WSDL que descreve a interface do serviço Web, mas o *tModels* é flexível o suficiente para descrever quase todo tipo de informação referente ao serviço (REIS & CÉLIO, 2008). As figuras 2.7 a 2.9 mostram o código XML de como as informações referentes a um serviços ficam armazenadas no repositório UDDI.

```
<businessEntity>
    businessKey="03754729-3D3C-48E0-854A-1C1FD576CA5B">
    <name>[nome-provedor-serviço]</name>
    <description  xml:lang="po">
        [descrição do negócio] – exemplo: Fornecedor de .....
    </description>
</businessEntity>
```

Figura 2.7: Exemplo de registro no *businessEntity* (Bosco, 2003)

```
<businessService
    servicekey="d5921160-3e16-11d5-98bf-002035229c64"
    businesskey="ba744edo-3aaf-11d5-80dc-002035229c64"
    <name>XMethods Delayed Stock Quotes</name>
    <description xml:lang="en">20-minute delayed stock quotes</description>
    <bindingTemplates>
        <bindingTemplate
            servicekey="d5921160-3e16-11d5-98bf-002035229c64"
            bindingkey="d594a970-3e16-11d5-98bf-002035229c64"
        <description xml:lang="en">
            SOAP binding for delayed stock quotes service
        </description>
        <accessPoint  URL Type="http">
            http://services.xmethods.net:80/soap
        </accessPoint>
        <tModelInstanceDetails>
            <tModelInstanceInfo
                tModelkey="uuid:0e727db0-3e14-11d5-98bf-002035229c64" />
        </tModelInstanceDetails>
        </bindingTemplate>
    </bindingTemplates>
</businessService>
```

Figura 2.8: Exemplo de registro no *businessService* (Bosco, 2003)

```
<tModel authorizedName="João Bosco" operator="[nome-site-operador]"
 tModelKey="uuid:01EBBD03-324D-4D7C-97EA—79B9C396D6EA">
   <name>Interface de Consulta de Preço</name>
   <description xml:lang="po">
      Interface SOAP para consulta de preço de produtos
   </description>
   <overviewDoc>
      <description xml:lang="po">
         inserir um arquivo WSDL
      </description>
      <overviewURL>
        http://www.[nome-provedor-serviço].[com/org/edu/gov]/services/
        [nome-arquivo-WSDL].wsdl
      </overviewURL>
   </overviewDoc>
</tModel>
```

Figura 2.9: Exemplo de registro no *tModel* (Bosco, 2003)

A descoberta de um serviço em um registro UDDI pode ser feita, por exemplo, através de busca pelo nome da organização provedora de serviços, ou por outros dados contidos no elemento *businessEntity*, pelo tipo de negócio ao qual pertence o serviço desejado, que fica armazenado no elemento *businessService*, ou pela interface desejada, como parâmetros de entrada e saída, entre outras informações, contidas nos *bindingTemplates*. A busca é realizada utilizando filtros e palavras-chaves, ou seja, é uma busca a nível sintático (OASIS, 2004).

Descobrir serviços publicados em um registro UDDI durante uma composição, como forma de tentar automatizá-la, é possível, no entanto, é bastante dificultada e pouco eficiente por causa das limitações decorrentes do fato de as composições atualmente serem focadas em um nível sintático. Da forma atual, serviços são descobertos através de palavras-chave, e não pelo seu significado e capacidade (DAMASCENO, 2009; MARTINS, 2007), ou seja, um serviço com mesma funcionalidade mas com interface distinta não seria recuperado em uma consulta no repositório UDDI.

2.3. PADRÕES PARA COMPOSIÇÃO DE SERVIÇOS

Atualmente existem alguns padrões para composição de serviços, sendo os mais conhecidos aqueles que são recomendados pela OASIS (*Organization for the Advancement of Structured Information Standards*) e pelo W3C. WS-BPEL (*Web Service – Business Process Execution Language)* e WS-Choreography são linguagens para construção de

orquestração e coreografia, respectivamente. Falaremos a seguir sobre as características e o funcionamento destes dois padrões para composição de serviços.

2.3.1. WS-BPEL

WS-BPEL é uma linguagem de orquestração para automatização de processos de negócios (OASIS, 2006). É um padrão criado pela OASIS para a execução de processos de negócio implementados como serviços Web e descreve como ocorre o relacionamento entre os serviços participantes da orquestração.

Na especificação WS-BPEL (OASIS, 2007), processos de negócio podem ser descritos de duas maneiras: como processos abstratos ou processos executáveis. De acordo com (WEERAWARANA et al., 2005), um processo abstrato é utilizado para descrever os padrões observáveis de troca de mensagem, permitindo fornecer uma visão para os parceiros de negócio sem precisar expor a lógica interna do processo de negócio. Para um parceiro saber como interagir com o processo de negócio, basta verificar o processo abstrato. Já o processo executável, especifica os detalhes exatos do processo de negócio e pode ser executado por um motor BPEL, o qual representa o serviço coordenador do processo. A figura 2.10 mostra um exemplo de processo abstrato, sobre negociação de ações on-line. Nela aparece um comprador, que solicita detalhes de ações, negocia compra e venda de ações etc. Todas as ações do processo são representadas de uma forma global, sem especificar detalhes. Um exemplo de processo executável é mostrado no fim desta seção.

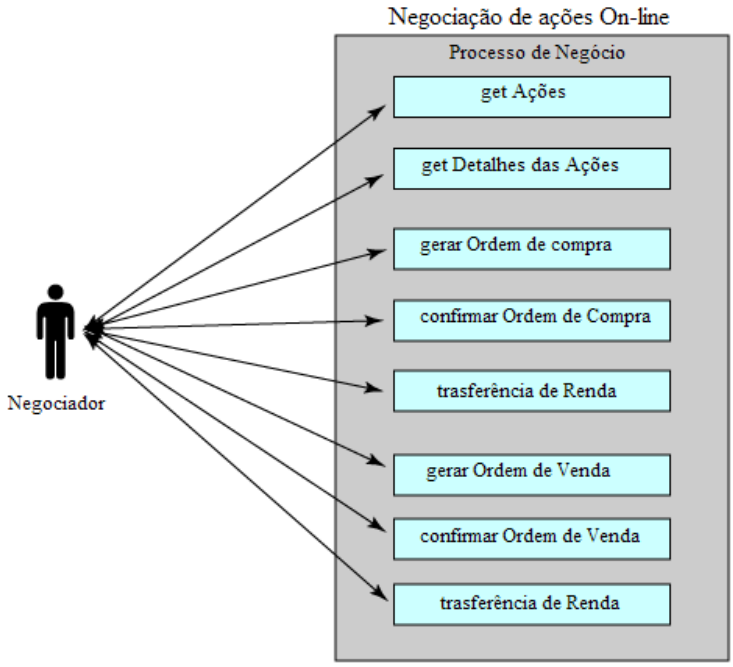

Figura 2.10: Exemplo de processo abstrato, traduzido de (IBM, 2005)

A programação com WS-BPEL pode ser comparada com a programação com outras linguagens de programação mais comuns, pois ela também oferece estruturas de repetição, condicionais, variáveis e atribuições. Isto possibilita que o processo de negócio seja visualizado como um algoritmo. No entanto, segundo DAMASCENO (2009), o uso destes recursos é simples e limitado, objetivando a facilidade de uso e aprendizagem.

Com WS-BPEL, o fato de um serviço ser ou não um serviço composto é transparente para o usuário daquele serviço. Pela especificação, a composição possui uma interface baseada no padrão WSDL, a troca de mensagens do serviço composto com o cliente que chama o serviço e do serviço coordenador com serviços que fazem parte da composição são feitas através do padrão SOAP. Como o serviço composto possui uma interface WSDL, ele pode ser cadastrado em um repositório UDDI, como se fosse um serviço atômico.

O projeto WS-BPEL consiste em dois tipos de arquivos: arquivos WSDL que especificam as interfaces dos serviços, ou seja, suas propriedades, tipo de portas, operações e tipos de dados, e, um documento XML que utiliza o vocabulário WS-BPEL para definir o processo, incluindo todas as atividades e principais variáveis envolvidas.

DAMASCENO (2009) exemplifica como uma composição especificada em WS-BPEL para um sistema de empréstimo funciona: a composição inicia sua execução quando

recebe uma chamada no *endpoint* relativo à operação de solicitação de empréstimo, pois como dito anteriormente, a composição também é um serviço Web e deve ter um WSDL definido. Após receber a invocação, um serviço Web *ClienteWS* é chamado sincronamente para validar os dados do cliente. Caso o *ClienteWS* seja válido os serviços Web *SerasaWS* e *SPCWS* são chamados em paralelo para verificar se o *ClienteWS* possui alguma restrição de débito. Se não houver nenhuma restrição, o serviço Web *EmprestimoWS* é chamado para conceder o empréstimo, e depois uma resposta é enviada ao solicitador do serviço. A figura 2.11 mostra como fica a modelagem de um processo executável WS-BPEL para o exemplo do sistema de empréstimo.

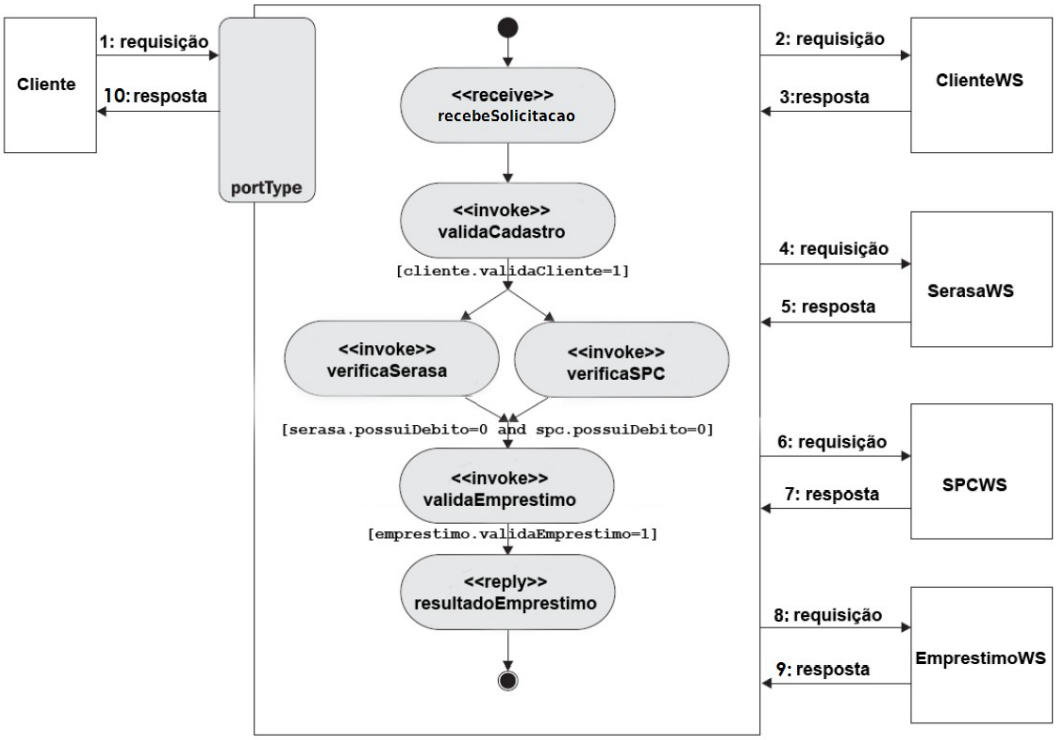

Figura 2.11: WS-BPEL para um sistema de empréstimo (DAMASCENO, 2009)

2.3.2. WS-CHOREOGRAPHY

WS-Choreography é uma especificação (W3C, 2005a) que define uma linguagem de coreografia de processos de negócio baseada em XML que descreve a cooperação entre serviços Web, definindo seus comportamentos observáveis. O principal objetivo da descrição da linguagem WS-Choreography (WS-CDL) é especificar uma linguagem declarativa que define, a partir de um ponto de vista global dos comportamentos

observáveis comuns e complementares, as trocas de informações que ocorrem e regras que precisam ser satisfeitas.

WS-Choreography nasceu da necessidade de se padronizar a representação das regras de interação entre todos os participantes de uma composição, que eram frequentemente descritas em linguagem natural. Esta padronização minimiza a possibilidade de equívocos com relação à interação e as responsabilidades de cada participante.

A especificação WS-CDL (*Web Services Choreography Description Language*) produz um contrato com a definição global das condições comuns de chamada e as restrições sob as quais as mensagens são trocadas, que é usada pelos participantes da composição para construir soluções em conformidade com ela.

Organizações são geralmente relutantes em delegar o controle de seus processos de negócio aos seus parceiros de integração, por questões de segurança, por exemplo. WS-Choreography oferece um meio pelo qual as regras de participação dentro de uma colaboração podem ser claramente definidas e aceitadas, em conjunto. Cada organização pode, então, implementar a sua parte da coreografia, conforme determinado pela definição global. É intenção da WS-CDL que cada implementação para uma visão comum seja fácil de determinar (W3C, 2005a). A figura 2.12 representa um exemplo de relação entre as empresas A e B, que desejam integrar suas implementações baseadas em serviços Web. Os analistas de negócio das empresas entram em acordo sobre os serviços envolvidos na colaboração, suas interações e regras de restrição. Eles então geram uma representação baseada em WS-CDL. Neste exemplo, uma coreografia especifica as interações entre os serviços para garantir a interoperabilidade entre as empresas, deixando as decisões reais de implementação nas mãos de cada empresa. A empresa *A* utilizou-se do padrão WS-BPEL e a *B* escolheu soluções Java para implementar sua parte da coreografia.

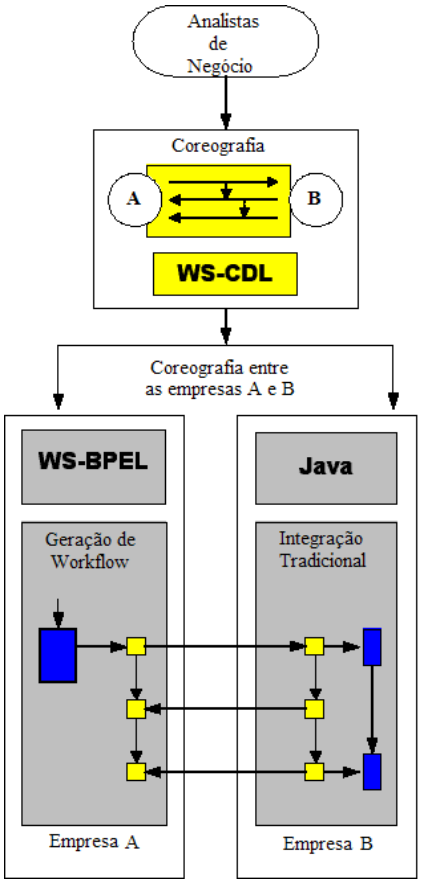

Figura 2.12: Exemplo do padrão WS-Choreography, traduzido de (W3C, 2005a)

Em W3C (2005) é explicitado que, enquanto os comportamentos observáveis dos participantes não se modificam, as lógicas e regras seguidas dentro de cada domínio de controle podem mudar e, ainda assim, a interoperabilidade é garantida.

2.4. DIFICULDADES ENCONTRADAS NA UTILIZAÇÃO DOS PADRÕES ATUAIS PARA COMPOSIÇÃO DE SERVIÇOS

Os problemas da composição de serviços Web podem ser resumidos em três pontos fundamentais: o primeiro é fazer um planejamento que descreva como os serviços Web interagem e como a funcionalidade que eles oferecem pode ser integrada para prover uma solução para o problema. O segundo ponto é a descoberta dos serviços Web que realizam as tarefas exigidas pelo planejamento. O terceiro é o gerenciamento da interação com os serviços Web (SYCARA et al., 2003). Planejamento, descoberta e conexão estão inteiramente conectados: um planejamento especifica o tipo dos serviços Web a serem descobertos, mas isso também depende dos serviços Web que estão disponíveis. Da mesma

forma, o processo de interação depende das especificações do planejamento, mas o planejamento em si depende dos requisitos da interação.

Estes problemas vêm acompanhados de um conjunto de desafios que uma infra-estrutura de composição de serviços Web precisa tratar. Para fazer a descoberta de um serviço Web, a infra-estrutura deve oferecer a representação das capacidades providas por um serviço Web e deve ser capaz de reconhecer similaridades entre os recursos oferecidos e as funcionalidades requeridas. Outro desafio é dar suporte à interação entre serviços Web. Isto deve englobar a especificação do que um serviço Web necessita e oferece o protocolo de interação que ele espera e os mecanismos de baixo nível necessários para invocá-lo.

Quando uma composição de serviços Web é criada, geralmente são estabelecidas conexões com os serviços que otimizam a execução, de acordo com algum critério pré-definido. Porém, por causa do dinamismo dos ambientes de rede, não há garantia de que os serviços escolhidos continuem sendo a melhor opção ou até que eles estejam disponíveis, e tais surpresas podem ocorrer a qualquer momento durante a execução do serviço (MARTINS, 2007). Através destes problemas é possível perceber a necessidade de se criar uma composição em que os serviços sejam escolhidos automaticamente em tempo de execução, garantindo a utilização de serviços ótimos e evitando falhas causadas por indisponibilidade.

A infra-estrutura atual de serviços Web, foca na interoperabilidade sintática. Dois padrões utilizados são SOAP e WSDL, que utilizam XSD (*XML Schema Definition*) para representar a estrutura de dados da mensagem. UDDI é um repositório de informações úteis sobre serviços Web, mas ele não implementa uma descoberta de serviços Web baseada na capacidade do serviço. WS-CDL e WS-BPEL descrevem como múltiplos serviços Web podem ser compostos juntos para prover um serviço mais complexo, no entanto, para admitir a criação de processos de alto-nível, estes precisam ser altamente detalhados, dificultando a alteração de um serviço que faz parte da composição, ficando o processo totalmente inflexível. O foco atual permanece na composição em nível sintático e, portanto, não admite uma automatização da composição de serviços Web (SYCARA et al., 2003). Desta forma, mudanças simples na interface do serviço, tais como uma alteração de um nome de uma operação ou de um tipo de dado esperado na mensagem, podem fazer com que a composição pare de funcionar.

TSANG (2006) explica que em casos onde a relação entre os serviços utilizados em um processo de negócio é permanente, ou seja, o processo não sofre grandes variações

na sua execução, a composição estática pode se caracterizar como uma alternativa suficiente, pois os componentes são conhecidos com antecedência e as ligações podem ser definidas em tempo de implementação. No entanto, no caso de processos mais complexos, em que a relação entre determinados serviços é temporária e variável, a composição estática de serviços Web pode não ser a solução ideal.

CORRALES et al. (2006) apresentam uma proposta para a descoberta de serviços através da utilização de um projeto WS-BPEL, que é a abordagem *matchmaking* de protocolo de negócios. O *matchmaking* acontece da seguinte forma: o modelo do serviço de negócio desejado é transformado em um grafo, assim como os serviços possivelmente equivalentes a ele. Em seguida é utilizado um algoritmo de comparação de grafos, levando em conta regras de comparação e decomposição de nós, gerando um resultado da similaridade, definindo quais serviços podem ser utilizados em casos de indisponibilidade etc.

Outra solução em pesquisa na comunidade acadêmica é o uso da descrição semântica para serviços Web. As abordagens referentes à composição semântica serão discutidas no próximo capítulo.

3. COMPOSIÇÃO SEMÂNTICA DE SERVIÇOS

A principal motivação para a pesquisa de novas abordagens para a composição de serviços Web é o fato de que, com as tecnologias atualmente utilizadas, as invocações a serviços Web são implementadas estaticamente, sempre referenciando o mesmo serviço Web e a mesma operação deste serviço. A baixa disponibilidade de determinado serviço influencia diretamente no desempenho da aplicação. Para evitar que isto aconteça é necessário que a aplicação consiga identificar um serviço similar que esteja com uma maior disponibilidade e então utilizá-lo (CARDOSO, 2007).

Outro problema já comentado anteriormente está ligado à descoberta de serviços, que atualmente é feita de uma forma sintática, e, caso haja alguma alteração de nome do serviço, nome de uma operação ou do tipo de dado esperado, o serviço não será mais encontrado, e consequentemente a composição não se forma da maneira correta (SRIVASTA & KOEHLER, 2003).

Neste capítulo serão apresentados conceitos, propostas, detalhes e ferramentas para uma abordagem que tem sido bastante estudada por diversos projetos de pesquisa acadêmicos pelo mundo: a abordagem semântica para a composição de serviços Web.

3.1. SERVIÇOS WEB SEMÂNTICOS

Os padrões de tecnologias existentes atualmente para o uso de serviços Web são designados para representar informações sobre a interface dos serviços, como eles são implantados ou como invocá-los, no entanto, suas habilidades para expressar quais as capacidades e requisitos do serviço são limitadas (W3C, 2005b). Diferenças superficiais na especificação de um serviço podem fazer com que um solicitante não o encontre, mesmo que ele seja o serviço desejado. Um importante passo para resolver este tipo de problema é não ser tão superficial na hora de descobrir e invocar o serviço, e identificá-lo a partir de similaridades semânticas, ou seja, pelas suas capacidades.

A abordagem semântica para o conceito de serviços Web é utilizada para descrever propriedades e capacidades dos serviços de uma forma normalizada, que não seja ambígua e que possa ser interpretada por um computador. Tais descrições são feitas na forma de construções em linguagens de marcação, que possibilitam que os serviços tenham suas descrições aumentadas com anotações semânticas, que definem o significado de seus

elementos através do mapeamento para uma ontologia, facilitando a automação do processo de invocação, descoberta, composição e monitoramento de serviços Web (CARDOSO, 2007).

O uso de ontologias é o ponto em comum entre as várias linguagens que possibilitam a adição de semântica às descrições de serviços Web, cada uma com suas particularidades. Ontologias proporcionam uma compreensão comum e compartilhada de um determinado domínio da realidade, utilizando estruturas de dados hierarquizadas com entidades e relações entre elas, além de regras de restrição (DERI, 2005). Na prática, os termos utilizados para descrever os serviços semanticamente devem estar mapeados em uma ontologia, de forma que, na hora de se buscar um determinado serviço a partir destes termos, os mesmos sejam identificados, bem como os elementos que se inter-relacionam com eles, gerando a informação da capacidade desejada para o serviço. A mesma coisa é feita com os serviços disponíveis, que no fim são comparados com a capacidade solicitada e selecionados, caso sejam equivalentes.

A semântica aplicada aos serviços Web influencia todo o desenvolvimento e implantação de uma aplicação (W3C, 2005b), como é mostrado na figura 3.1.

Durante as atividades de modelagem, o provedor do serviço pode explicar as semânticas que serão utilizadas através de anotações em partes apropriadas do serviço utilizando conceitos de um modelo semântico (ontologia). Modelos semânticos devem fornecer um acordo a respeito do significado e uso dos termos, além de definições formais e informais para evitar a ambiguidade na semântica desejada pelo provedor. Estes serviços descritos semanticamente podem então ser publicados em um registrador. Durante a descoberta, um cliente pode descrever o serviço desejado usando termos da modelagem semântica. Raciocínios técnicos podem ser usados para achar a similaridade semântica entre a descrição dos serviços registrados e o serviço solicitado. Em casos em que não há um serviço exatamente similar, um serviço novo pode ser composto para se adequar a requisição. Durante a composição, o aspecto funcional das anotações pode ser usado para agregar a funcionalidade de múltiplos serviços para criar composições utilizáveis, enquanto a semântica dos aspectos não funcionais ajuda a determinar se essas composições são legais e válidas. Durante a implantação, a semântica pode ser usada novamente para encontrar instâncias de serviços específicos para fazer a ligação entre as interfaces dos serviços. Durante a invocação, a semântica pode ser usada para representar transformações de dados. Em caso de falha de um serviço em tempo de execução, a semântica possibilita a

aplicação de se recuperar e permitir a descoberta e vinculação automática para encontrar outros serviços para substituí-lo adequadamente. Portanto, uma vez representada, a semântica pode ser aproveitada por ferramentas que automatizam a descoberta, a mediação, a composição, a execução e o monitoramento dos serviços.

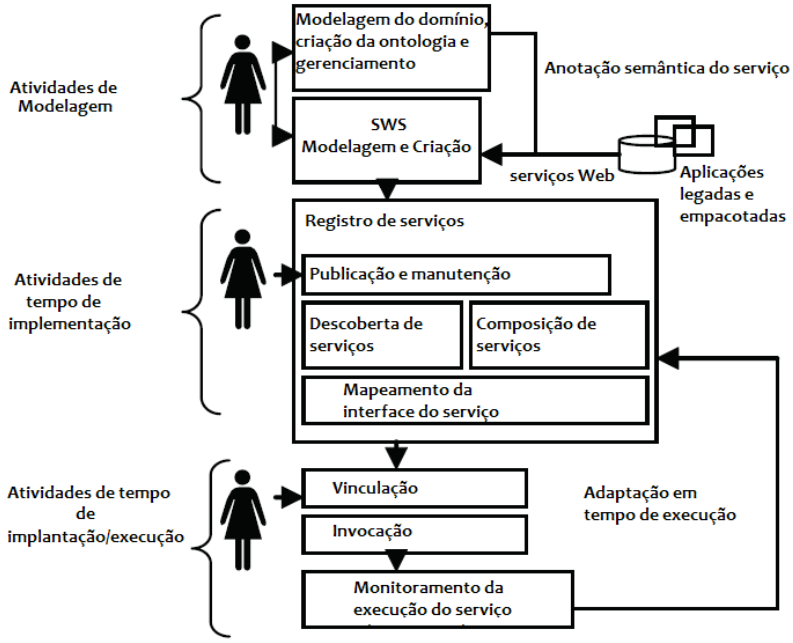

Figura 3.1: A semântica na vida de serviços Web, traduzido de (W3C, 2005b)

As pesquisas relacionadas a serviços Web semânticos visam transformar a Web, que atualmente é um repositório de informações, em um sistema mundial de computação distribuída (LARA et al., 2007). Esta evolução da Web é representada pela figura 3.2, que mostra o início da Web, primeiro como um repositório de informações puramente estáticas; depois a chegada dos serviços Web, com padrões de descrição e protocolos de troca de mensagem, dando um maior dinamismo às informações; em seguida os conceitos da Web semântica, que proporcionam uma contextualização formal da informação, com a finalidade de permitir a compreensão de significados por máquinas; e por fim, os serviços Web semânticos, que visam à utilização automática dos serviços Web, objetivando fazer uso da Web com todo o seu potencial (Luna & Santos, 2009).

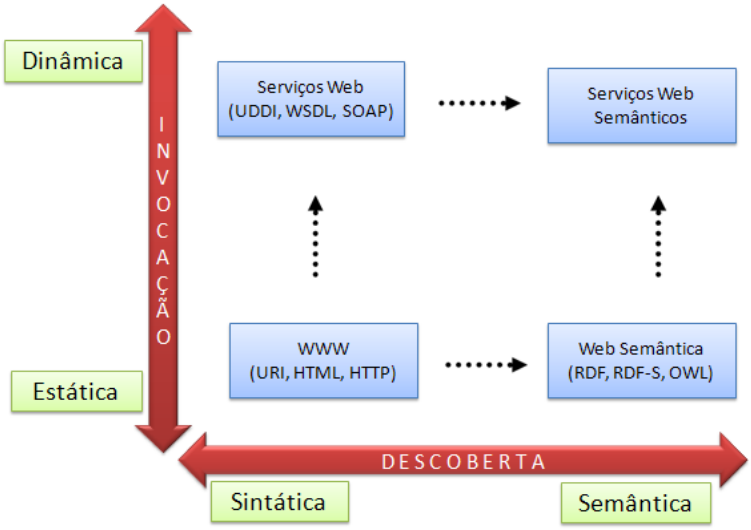

Figura 3.2: Evolução da Web

O paradigma de serviços Web semânticos possui ainda algumas tecnologias relacionadas, como padrões para modelagem de ontologias – RDF (*Resource Description Framework*) e OWL (*Web Ontology Language*) – e padrões de descrição da semântica nos serviços Web, que serão discutidos nas próximas seções.

3.2. PADRÕES PARA DESCRIÇÃO DE SEMÂNTICA EM SERVIÇOS WEB

Nesta seção, serão apresentados alguns dos mais conhecidos padrões que permitem descrever serviços Web semânticos. Entre os padrões discutidos, estão o WSDL-S (*Web Service Description Language Semantics*) e as ontologias OWL-S (*Web Ontology Language for Services*), SWSO (*Semantic Web Services Ontology*) e WSMO (*Web Service Modeling Ontology*).

3.2.1. OWL-S

OWL-S (W3C, 2004a) (antiga DAML-S - *DARPA Agent Markup Language for Services*) é uma ontologia para serviços Web, baseada no padrão OWL, que é uma linguagem para descrição de ontologias na Web. OWL é desenvolvida como uma extensão do vocabulário RDF (W3C, 2004) e é derivada da linguagem de ontologia DAML+OIL (W3C, 2004b). Estas linguagens possibilitam a criação de ontologias para qualquer domínio e a instanciação destas ontologias na descrição de um determinado site na Web.

A estrutura do OWL-S é formada por três tipos de informação sobre o serviço, que são caracterizados por perguntas, como mostra a figura 3.3.

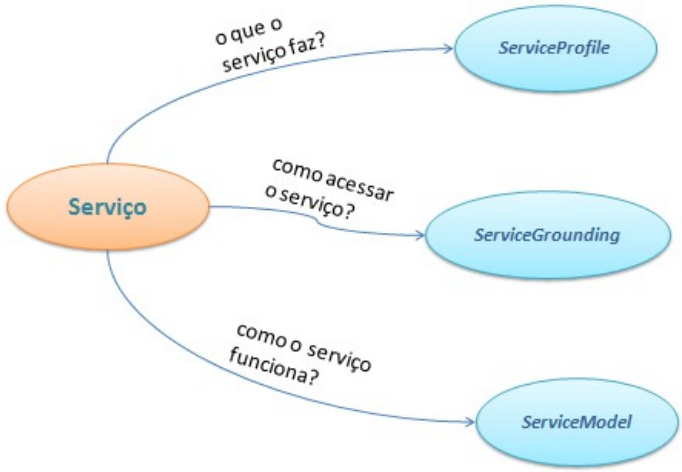

Figura 3.3: OWL-S, traduzido de (W3C, 2004)

O *ServiceProfile* diz qual a função do serviço. Cada instância da classe do serviço possui um *ServiceProfile*, que funciona como uma apresentação do mesmo. O *ServiceGrouding* é um elemento em que ficam as informações de suporte, como detalhes necessários para os protocolos de comunicação entrarem em contato com o serviço. Um *ServiceGrounding* deve estar associado com somente um serviço. Por fim, o *ServiceModel* contém a descrição do modelo de processo e uma propriedade chamada *describedBy* é utilizada pelo serviço para fazer referência ao *ServiceModel* que o descreve. Um serviço pode ser descrito por mais de um *ServiceModel.*

De acordo com a especificação do W3C (2004), utilizando a ontologia OWL-S, é possível automatizar algumas tarefas que normalmente precisam ser realizadas manualmente:

> **Descoberta de serviços Web:** automatização da localização de serviços da Web que podem ser úteis dependendo de condições definidas pelo cliente, pois fazem parte de uma determinada classe de capacidade requerida. Por exemplo, para descobrir um serviço de venda de passagens de avião entre duas cidades, que aceite um determinado cartão de crédito e ainda tenha o menor preço de mercado. Isto atualmente é feito com a ajuda de humanos, que podem usar uma busca para encontrar o serviço, e ler a página para ver se ele obedece às condições explicitadas. Com OWL-S, as informações podem ser especificadas no próprio

serviço de uma forma semântica interpretável pelo computador, sendo somente necessário um motor de busca que entenda ontologias, para que a localização seja feita;

Invocação de serviços Web: a invocação automática de um serviço Web é feita utilizando uma descrição declarativa do serviço, em vez de existir uma pré-programação para chamar esse serviço específico. A OWL-S permite a declaração de uma API (*Application Programming Interface*) interpretável pela máquina, que inclui a semântica dos argumentos a serem especificados durante a execução de uma chamada, e a semântica do que pode ser retornado quando a execução do serviço é feita com sucesso ou ocorre alguma falha;

Composição de serviços Web: seleção e composição automáticas de serviços Web a partir de uma descrição alto-nível do objetivo. Com OWL-S as informações necessárias para selecionar e compor cada serviço são especificadas semanticamente dentro do próprio serviço. Softwares devem ser implementados para manipular estas informações, juntamente com a descrição do objetivo, para que seja alcançado automaticamente.

Como já dito antes, OWL-S se baseia na expressividade da OWL para representar serviços Web semânticos. No entanto, CARDOSO (2007) nos diz que embora a linguagem OWL tenha um poder considerável na descrição de modelos de serviços Web, existem certas limitações com relação à modelagem de serviços Web complexos ou processos Web. OWL possui um bom conjunto de construções para expressar classes, mas a expressividade da linguagem apresenta limitações na descrição de propriedades, além de não prover o uso de variáveis, especialmente quando são definidas classes relacionadas com uma ontologia. Estas restrições colocam limitações na relação de entradas e saídas dos processos de composição com os de outros processos.

3.2.2. WSDL-S

WSDL-S (W3C, 2005a) é uma abordagem proposta pelo laboratório do LSDIS (*Large Scale Distributed Information Systems*) e por pesquisadores da IBM que estende as descrições de serviço WSDL com semânticas, originada do padrão SAWSDL (W3C, 2006) (*Semantic Annotations for WSDL*), recomendado pelo W3C. O padrão WSDL-S permite

fazer anotações de serviços e operações mapeando determinados elementos para um modelo ontológico. Estes serviços melhorados semanticamente devem ser publicados em um registro que implemente buscas semânticas, podendo, desta forma, serem descobertos em tempo de projeto, implantação e execução, utilizando alguns conceitos ontológicos (Li, 2005).

O padrão WSDL-S propõe cinco elementos de extensão utilizáveis na anotação de entradas, saídas e operações de serviços Web :

modelReference: faz o mapeamento (um-para-um) entre elementos da descrição e conceitos da ontologia;

SchemaMapping: atributo que pode ser adicionado a elementos XSD, para associá-los a uma ontologia (um-para-muitos);

precondition: combinação de condições em uma ontologia que devem ser garantidas antes da execução do serviço;

effects: similar ao *precondition*, se refere a condições que devem ser garantidas depois da execução do serviço;

category: estende a interface do serviço e é usada para categorizar uma informação.

Neste padrão, as anotações semânticas são associadas com vários elementos do documento WSDL, como parâmetros de entrada e saída e aspectos funcionais, como operações, pré-condições e efeitos, fazendo referência aos conceitos de um modelo semântico de domínio externo, como é mostrado na figura 3.4 (W3C, 2005b).

Segundo Miller et al. (2005), WSDL-S tem como objetivo proporcionar uma abordagem simples para criação de serviços semânticos, ao contrário de OWL-S que, em alguns casos, acaba sendo desnecessariamente complexo. A ontologia WSDL-S foi projetada para ser totalmente alinhada com o padrão WSDL 2.0 (W3C, 2005a) e mais compacta que o OWL-S. Uma das diferenças é a não separação do *ServiceProfile* do *ServiceGrounding*.

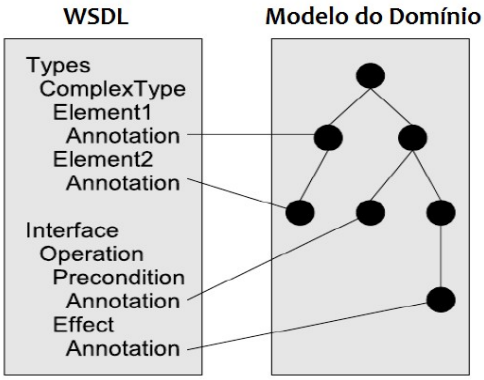

Figura 3.4: WSDL-S (W3C, 2005b)

Por conta desta visão minimalista, de acordo com CARDOSO (2007), o WSDL-S acaba não sendo muito utilizado em processos Web semânticos, uma vez que ele somente adiciona anotações semânticas em serviços Web simples, representados por WSDL's.

3.2.3. SWSO

SWSO (W3C, 2005b) é um padrão de descrição de semântica implementado a partir do *framework* SWSF (*Semantic Web Services Framework*) e baseado no OWL-S. Este *framework* foi criado pela SWSL (*Semantic Web Services Language*) *Comittee* da SWSI (*Semantic Web Services Initiative*).

SWSO é expressada de duas formas: FLOWS (*First-order Logic Ontology for Web Services*) e ROWS (*Rules Ontology for Web Services*). ROWS é produzida por uma tradução sistemática de axiomas FLOWS na linguagem SWSL-Rules. FLOWS foi especificada com base em SWSL-FOL, que é uma linguagem de lógica de primeira ordem usada como especificação da ontologia e proporciona interoperabilidade com outros modelos de processos baseados em primeira ordem e ontologias de serviços.

O modelo de processos FLOWS adiciona dois elementos fundamentais ao PSL (*Process Specification Language*) (ISO 18629), que é uma ontologia genérica para processos. Uma delas é a noção de processo atômico, assim como encontrado em OWL-S, e a outra, é uma infra-estrutura para especificar várias formas de fluxo de dados. Seguindo a organização modular do PSL, FLOWS é dividido em camadas e particionado em agrupamentos naturais. Quando especificado um domínio de aplicação, fica então fácil incorporar somente os blocos de ontologia que são necessários. Os módulos de ontologia

são explicados abaixo, e a figura 3.5 faz a representação do papel de cada um no modelo de processo.

FLOWS-Core apresenta as noções básicas de serviços como atividades compostas de atividades atômicas;

Restrições de Controle axiomatizam as construções básicas comuns em um modelo de processo em forma de *workflow*;

Restrições de ordenação permitem a especificação de atividades definidas como propriedades sequenciais de processos atômicos;

Restrições de ocorrência suportam a especificação de atividades não-determinísticas dentro de serviços;

Restrições de estado suportam a especificação de atividades que são desencadeadas por estados (de um sistema global) que satisfazem uma determinada condição;

Restrições de exceção proporcionam uma infra-estrutura básica para modelagem de exceções.

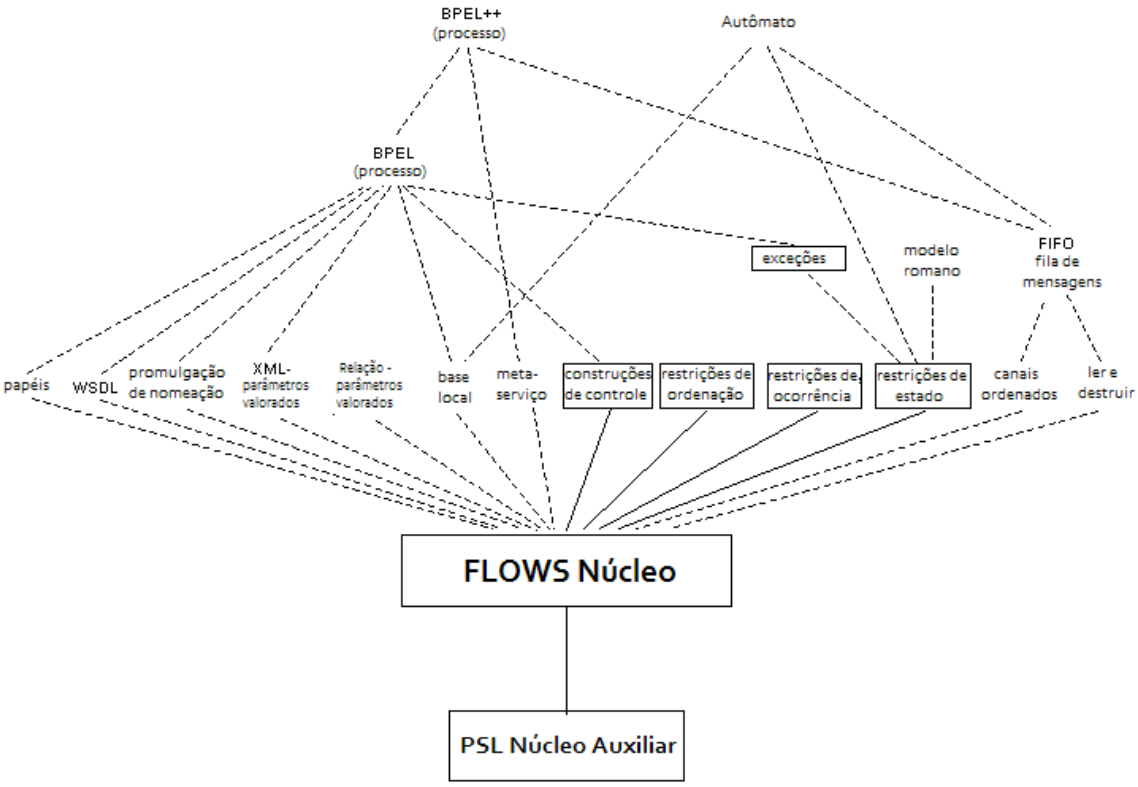

Figura 3.5: Modelo de processos do FLOWS, traduzida de (DAML, 2005)

Assim como OWL-S, SWSO é dividido em três grupos de informação: descritor do serviço, modelo do processo e *grounding*. Segundo os autores da SWSO, como OWL não é suficientemente expressiva para capturar as semânticas do modelo de processo interno ao OWL-S, eles então propuseram a utilização de lógica de primeira ordem e um particular modelo de processo construído em PSL (ISO 18629).

De acordo com Li (2005), as principais diferenças entre SWSO e OWL-S são:

1. SWSO usa lógica de primeira ordem, que permite a representação de diferentes fluxos de dados em um processo de composição, enquanto OWL-S utiliza OWL, que possui um padrão de vocabulário e construções para definição de fluxos;
2. SWSO possui um maior detalhamento das descrições do modelo de processo do que OWL-S, através dos seguintes sub-módulos: restrições de ordenação, restrições de ocorrência, restrições de estado e restrições de exceção.

3.2.4. WSMO

WSMO (*Web Service Modeling Ontology*) é uma ontologia que descreve vários aspectos relacionados a serviços Web semânticos. Ela tem como base um *framework* chamado WSMF (*Web Service Modeling Framework*), que é refinado e ampliado através de ontologia e linguagem formais. WSMF define um modelo conceitual para desenvolvimento e descrição de serviços Web, baseando-se em dois pontos principais: dissociação máxima e mediação forte. Falaremos mais de WSMF no decorrer do trabalho.

WSMO utiliza uma linguagem formal, chamada WSML (*Web Service Modeling Language*), que permite a anotação de serviços Web de acordo com o modelo conceitual. Existe também um ambiente de execução relacionado, o WSMX, que visa ao dinamismo na descoberta, seleção, mediação, invocação e interoperação de serviços Web baseados na especificação WSMO. Em (W3C, 2005c) são explicitados alguns princípios nos quais o padrão WSMO se baseia:

Conformidade com a Web: WSMO herda o conceito de identificação única para seus recursos. Além disso, WSMO adota o conceito de *namespaces* para denotação de espaços de informação consistentes, suporta XML e outras tecnologias recomendadas pelo W3C, bem como a descentralização de recursos;

Base na ontologia: ontologias são usadas como modelo de dados. O significado de todas as descrições de recursos, bem como toda a informação trocada durante o uso dos serviços, é baseado na ontologia. Este extensivo uso permite um reforçado processamento da informação, além do suporte para interoperabilidade;

Dissociação bem definida: cada recurso é especificado independentemente, sem considerar o uso ou interação com outros recursos. Isso entra em conformidade com a natureza aberta e distribuída da Web;

Mediação centralizada: mediação diz respeito à manipulação de heterogeneidades que naturalmente aparecem em ambientes abertos. Esta característica é considerada fundamental para o êxito da implantação de serviços Web;

Separação do papel ontológico: as requisições de usuários são formuladas independentemente da disponibilidade dos serviços Web. O conhecimento adjacente do WSMO faz a diferenciação entre os desejos dos clientes e os serviços disponíveis;

Execução semântica: a execução semântica formal de implementações de referência, como WSMX (*Web Service Modelling eXecution environment*), proporciona a realização técnica do WSMO;

Serviços X serviços Web: serviço Web é uma entidade computacional que permite (através da invocação) alcançar uma meta. Um serviço é o valor de fato provido por esta invocação. WSMO não especifica serviços, mas sim serviços Web, que são os meios para obter e pesquisar serviços.

Em WSMO, as descrições dos serviços são feitas com base em suas capacidades e interface. Capacidade define o serviço em termos de funcionalidade, enquanto que a interface descreve como obter a funcionalidade do serviço, isto é, como se comunicar com ele (coreografia) e como fazê-lo cooperar com outros serviços (orquestração).

A descrição formal da capacidade de um serviço Web dentro da ontologia WSMO é feita a partir de pré-condições, suposições, pós-condições e efeitos. Pré-condições são condições que precisam ser validades no espaço de informação antes da execução do serviço; suposições são condições que precisam ser válidas no mundo real antes da execução do serviço; pós-condições são as alterações causadas no espaço de informação

pela execução do serviço; e efeitos descrevem como fica o mundo real após a execução do serviço. Esta descrição necessita do projeto e desenvolvimento de ontologias relacionadas aos conceitos referentes ao domínio de aplicação do serviço, para especificar as funcionalidades do mesmo (W3C, 2005c).

O *framework* WSMF, no qual se baseia o padrão WSMO, consiste de quatro elementos de alto-nível para descrição de serviços Web semânticos (W3C, 2005c):

Ontologias, que representam o elemento-chave do WSMO, fornecem a terminologia utilizável por outros elementos. Ontologias definem semânticas formais da informação e fazem a ligação entre terminologias de máquina e humanas;

Descrições de serviços Web, que definem vários aspectos dos mesmos, conectando computadores e dispositivos usando protocolos padrões da Web e combinando dados de diferentes formas. Sua distribuição em toda a Web confere a vantagem de serem independentes de plataforma;

Metas, que são os objetivos que devem ser alcançadas pelos serviços Web. É a perspectiva do usuário do que o serviço deve proporcionar. A co-existência de metas e serviços Web como entidades não sobrepostas, garante a dissociação entre a requisição e o serviço Web. Isto é conhecido como abordagem *goal-driven* (direcionada a metas), derivada da abordagem AI do agente racional. Em tempo de execução, as metas solicitadas por um cliente são comparadas com os serviços disponibilizados, definindo desta forma quais serviços proporcionam a funcionalidade desejada pelo cliente;

Mediadores, que resolvem problemas de interoperabilidade entre os diferentes tipos de componentes que formam uma descrição WSMO. No nível de dados, resolvendo problemas de integração semântica, e, no nível de processos, resolvendo as heterogeneidades entre padrões de comunicação entre os serviços Web.

A figura 3.6, mostra como é feita uma tentativa de correspondência (*matching*) entre uma meta solicitada e um serviço disponibilizado, no instante em que uma composição está sendo feita. São comparadas as capacidades – pré e pós-condições, suposições e efeitos – e a interface entre os dois elementos. Caso o resultado desta comparação seja positivo, o serviço pode ser utilizado na composição.

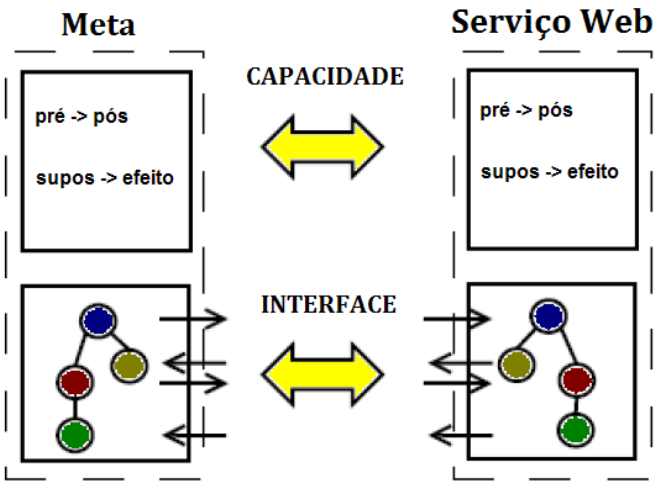

Figura 3.6: Correspondência entre meta e serviço Web, traduzida de (CASANOVA, 2009)

Após a conceituação dos padrões para descrição de semântica em serviços Web, fazemos uma comparação resumida, representada pela tabela 3.1 (Cardoso, 2007), entre os quatro padrões descritos: OWL-S, WSDL-S, SWSO e WSMO, de acordo com a linguagem e formalismo utilizados e relação com WSDL.

Tabela 3.1: Comparação entre formalismos para serviços Web semânticos, ampliada de (CARDOSO, 2007)

Padrão	**Comentário**	**Linguagem Semântica**	**Formalismo**	**Relação com WSDL**
OWL-S	Ontologia para serviços Web, baseado em OWL	OWL	Lógica de descrição	Define uma conectividade com WSDL pelo *GroundingModel* mas se sobrepõe na definição de *inputs*, *outputs* e operações
WSDL-S	Uso de elementos extensivos ao WSDL com termos de ontologias externas	Independente de uma linguagem semântica.	Independe de uma linguagem ontológica, portanto, quaisquer formalismos podem ser usados	Especifica anotações diretamente no WSDL, como elementos de extensão
SWSO	Ontologia para serviços Web, baseado em OWL	SWSO-FOL e SWSO-Rules	Lógica de primeira ordem	Define uma conectividade com WSDL através do elemento *grounding*
WSMO	Ontologia para modelagem de serviços Web, usando WSML	WSML	Lógica de descrição, lógica de primeira ordem e programação lógica	Define uma conectividade com WSDL pelo *GroundingModel* mas se sobrepõe na definição de *inputs*, *outputs* e operações

3.3. ABORDAGENS PARA COMPOSIÇÃO DE SERVIÇOS WEB SEMÂNTICOS

Nesta seção serão apresentadas algumas abordagens práticas, como forma de exemplificar as diversas áreas em que o estudo de descrições semânticas para composição de serviços Web pode ser aplicado.

Moran et al. (2007) propõem uma abordagem que realiza a composição a partir de descrições incompletas de serviços Web. Esta abordagem foi proposta para o *Semantic Web Service Challenge* (SWSC, 2007), que é um desafio internacional organizado inicialmente pelo Grupo de Lógica da Stanford University, STI Innsbruck, Friedrich-Schiller-University Jena, e a Technische Universität Dortmund. O objetivo deste desafio é desenvolver um comum entendimento de diversas tecnologias com a intenção de facilitar a automação da mediação, coreografia e descoberta de serviços Web utilizando anotações semânticas. Isto é feito a partir de comparação entre abordagens propostas.

Para exemplificar a proposta de Moran et al. (2007), toma-se como exemplo uma solicitação de um cliente para compra de um *laptop* e um *docking station*. Cada tipo de *laptop* possui determinadas restrições de compatibilidade com *docking stations*. Estas restrições de compatibilidade podem estar explícitas ou serem disponibilizadas por um provedor de serviço em tempo de execução. Com esta abordagem, uma relação genérica entre *laptops* e *docking stations* é declarada em uma ontologia de domínio e deixa-se que as ontologias de serviços individuais forneçam sua própria definição lógica, para que uma relação mais alto-nível, como por exemplo, uma ontologia particular utilizada para produtos da IBM, defina um axioma para a relação, especificando que *laptops* da marca IBM só trabalham com *docking stations* de mesma marca.

A arquitetura prevista na abordagem é orientada a metas, ou seja, em tempo de execução uma meta é solicitada e uma tentativa de encontrar um serviço adequado é feita. As ontologias utilizadas para descrever a meta e o serviço, inclusive axiomas e definições de restrições, são gravadas em uma base de conhecimento. Se um serviço correspondente não é encontrado, uma função de decomposição da meta é aplicada até que seja encontrada alguma correspondência. O padrão WSMO foi utilizado como modelo conceitual, e as definições de informação, semânticas funcionais e de comportamento foram adotadas a partir do padrão WSMO-Lite (WSMO, 2007).

A composição dos serviços utiliza a descoberta de serviços, incluindo coleta de informações em tempo de execução. São utilizadas ainda duas extensões: a decomposição iterativa de metas e manutenção de base de conhecimento utilizada nos serviços correspondentes a determinadas sub-metas, até que todas as sub-metas sejam correspondidos. A decomposição iterativa é utilizada para que, toda vez que uma meta não for correspondida a um serviço, a decomposição seja feita e uma nova tentativa de correspondência seja realizada. Esta iteração acontece até que a meta não possa mais ser decomposta.

O desafio proposto pelo *SWS Challenge* foi uma composição simples, que consiste na descoberta de serviços que juntos alcançam uma meta desejada, mas que não precisam estar em determinada ordem de invocação. Desta forma, o parâmetro de saída do algoritmo de composição proposto é uma pilha de serviços Web. Estes serviços Web são adicionados a uma pilha na medida em que são correspondidos com alguma meta (ou sub-meta).

Outro exemplo de abordagem foi feito por Agre & Marinova (2007), desenvolvido a partir de um projeto de pesquisa acadêmico chamado INFRAWEBS, que é um *framework* que consiste de unidades Web semânticas acopladas, por meio das quais são implementadas ferramentas e componentes de análise, projeto e manutenção de serviços Web semânticos baseados no padrão WSMO. O projeto é conhecido como IIF (*INFRAWEBS Integration Framework*), e sua aplicação para a composição dinâmica de serviços pode ser resumida nos seguintes passos:

1. **representação** da requisição do usuário como uma meta (elemento WSMO) baseado em um modelo de meta abstrato;
2. **decomposição** da meta do usuário em sub-metas (se necessário);
3. **descoberta** um conjunto compatível de serviços existentes – candidatos habilitados a satisfazer cada sub-meta;
4. **seleção** de um serviço concreto para cada sub-meta;
5. **execução** dos serviços selecionados.

Uma meta pode ser descrita a partir de um conjunto de ontologias importadas para sua descrição lógica, um conjunto de expressões lógicas (axiomas), ou uma interface que especifique que interações o usuário espera que o serviço tenha. Determinadas expressões lógicas podem ter diferentes papéis (pré-condição, suposição, pós-condição ou efeito) para

descrever o que o usuário deseja do serviço, bem como algum dado de entrada que o usuário está disposto a fornecer para o serviço. Na abordagem proposta as metas são representadas na linguagem WSML e são divididas em dois tipos:

Modelos de meta: representação abstrata de metas criada a partir dos serviços existentes, usados para fazer o *matching* com as metas dos clientes;

Metas do usuário: requisições específicas da meta desejada pelo cliente. Estas metas são criadas em tempo de execução como uma instância do modelo de meta correspondente.

Outra dimensão relacionada à descrição de metas é a definição se uma meta pode ser decomposta (meta composta) ou não (meta atômica). De acordo com Agre & Marinova (2007), uma abordagem de reconhecimento automático da natureza da meta é de difícil implementação, devido ao fato de a sintaxe da linguagem WSML ser bastante complexa. Por este motivo foi aplicada uma implementação mais simplista, em que um modelo de meta composta só pode ser criado a partir de modelos de metas existentes.

Um importante problema explicitado é a garantia da compatibilidade dos serviços participantes da composição, pois em uma composição dinâmica não é possível saber quais serviços entrarão na composição. Em geral, isto é feito através de uma análise de correspondência entre os requerimentos (ou restrições) impostos ao serviço por meio da linguagem de orquestração e das capacidades e interfaces destes serviços, ou seja, descrevendo o que os serviços podem fazer e como se comunicar com eles. No entanto, com a implementação simplista citada no parágrafo anterior, pode-se garantir que se um modelo de composição é formado por sub-metas compatíveis, então os serviços correspondentes a estas sub-metas também são compatíveis. A compatibilidade deve ser vista em dois níveis: (1) estática ou funcional, que se refere à descrição de serviços através de termos ontológicos, que em WSMO é resolvida por OO-Mediadores (compatibilidade entre ontologias) e (2) dinâmica ou comportamental, que tem a ver com a comunicação entre as interfaces dos serviços, que em WSMO é tratada por WW-Mediadores (compatibilidade entre serviços). No projeto INFRAWEBS, as metas e serviços seguem um conjunto de ontologias comuns, não havendo a necessidade de elementos para mediar a compatibilidade semântica. A seleção dos serviços que formarão a composição é feita manualmente pelo usuário, após a descoberta dos serviços candidatos.

KONA et al. (2008) apresentam uma proposta de *framework* que realiza planejamento, descoberta e seleção de serviços, todos de uma vez, automaticamente em um simples processo. O objetivo da pesquisa é a geração automática de *workflows* relacionados a inferências filogenéticas, que são usadas para estimar a história evolutiva de um conjunto de organismos. Para este fim, foi formalizado o problema da composição baseando-se em uma representação por grafo acíclico direcionado (figura 3.8), foi feita a apresentação de um algoritmo para resolver a composição, que leva em conta a semântica do serviço, sendo que este algoritmo descobre e seleciona automaticamente os serviços envolvidos na composição para uma dada consulta, sem a necessidade de intervenção manual, além da geração automática de descrição OWL-S para a nova composição obtida. A figura 3.7 mostra um exemplo de composição com um grafo acíclico direcionado, em que I' e CI' representam os parâmetros de entrada e as pré-condições, respectivamente, e O' e CO' representam os parâmetros de saída e as pós-condições, respectivamente. Os nós representam os serviços (S1 a S5) e as setas representam os parâmetros de entrada (quando apontam para o nó) e saída (quando saem do nó).

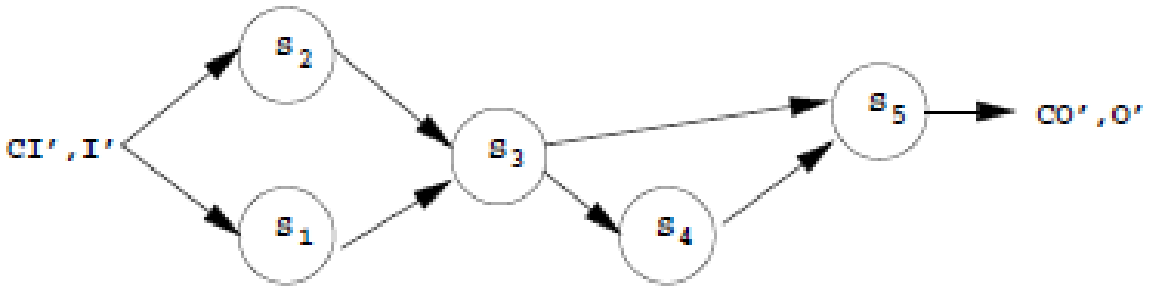

Figura 3.7: Grafo acíclico direcionado (Kona et al., 2008)

A linguagem OLW-S é usada para modelar os serviços como processos, e, ao descrever serviços compostos, mantém o estado do serviço ao longo do processo. Isto proporciona a criação de estruturas de controle, como sequências, *split-join* ou *if-then-else*, entre outras, sendo útil para descrever o tipo de composição.

A implementação do protótipo de um construtor de composição foi feita usando Prolog e foram usadas descrições semânticas dos serviços Web escritos em uma linguagem chamada USDL (*Unified Service Description Language*) (SAP, 2009). O repositório de serviços contém um documento de descrição para cada serviço publicado. O construtor da composição consiste em cinco módulos:

1. Gerador de tuplas, que converte cada serviço do repositório para um formato padrão de tuplas;
2. Leitor de consultas, que interpreta as consultas a partir das tuplas;
3. Gerador de relações semânticas, que extrai as relações semânticas e cria a lista de fatos em Prolog;
4. Processador de composição de consultas, que usa o repositório de fatos contendo todos os serviços, parâmetros de entrada e saída e relações semânticas entre os parâmetros. A saída do processador é a composição representada pelo grafo acíclico direcionado;
5. Gerador de descrição OWL-S, que gera a descrição do serviço composto usando construções dependentes do tipo de composição.

TRAVERSO & PISTORE (2009) abordam a composição de serviços semânticos explicando uma situação exemplo, de uma relação entre *Produtor*, *Entregador* e *Usuário*, formando uma composição chamada de *Encomenda e Entrega (E&E)*, como pode ser visualizado através da figura 3.8.

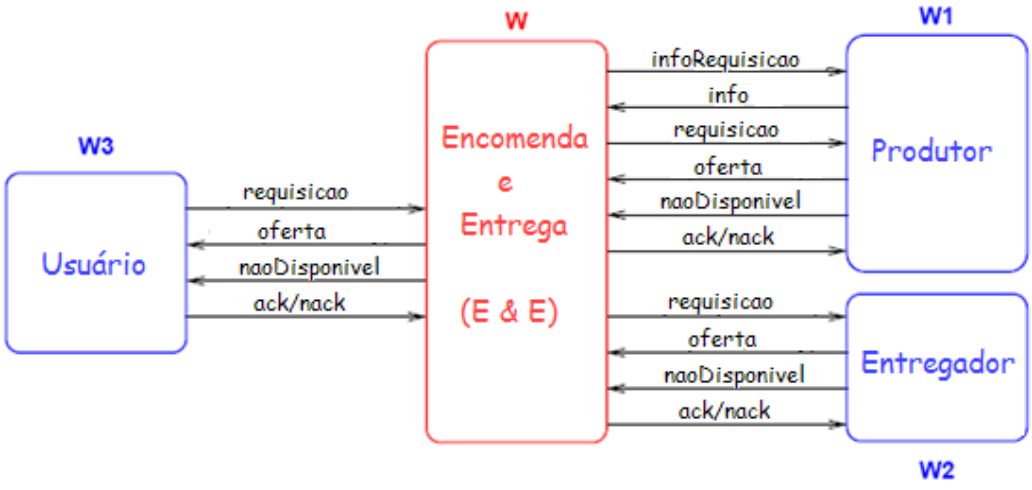

Figura 3.8: Exemplo de composição, traduzida de (TRAVERSO & PISTORE, 2009)

A necessidade de inclusão de descrições semânticas aparece quando são propostas duas metas à composição:

Meta 1: o serviço deve tentar atingir a situação ideal em que o usuário confirmou seu pedido, e o serviço confirmou os (sub-)pedidos associados aos

serviços *Produtor* e *Entregador*. Nesta situação, os dados associados aos pedidos devem ser mutuamente consistentes, por exemplo, o tempo para construir e entregar um mobiliário deve ser a soma do tempo para construir com o tempo para entregar.

Meta 2: o serviço *E & E* deve atingir uma situação de retrocedência, em que cada (sub-)pedido foi cancelado. Ou seja, deve haver uma chance de o serviço não ser confirmado.

Sabe-se que a Meta 1 corre o risco de nem sempre ser alcançada, por diversos motivos: o produto não está disponível, a entrega não é possível, o usuário pode não aceitar o custo total ou o tempo de entrega do produto etc. Em contrapartida, a Meta 2 pode ser atingida sempre que o serviço for cancelado. Analisando o caso, percebe-se que o ideal é uma **meta composta**, tendo a Meta 1 uma força maior. A meta composta ficaria assim: *Tente satisfazer a Meta 1, em caso de falha, satisfaça a Meta 2*. Metas compostas precisam ser hábeis a expressar condições sobre o comportamento de todo um serviço, condições sobre diferença de forças e preferências entre diferentes sub-metas. Nesta proposta, estas descrições e expressões lógicas seguiram o padrão OWL-S para a formalização da semântica.

4. IMPLEMENTANDO COMPOSIÇÕES SEMÂNTICAS COM WSMO STUDIO

De acordo com Dimitrov et al. (2007), as ferramentas de suporte na área de serviços Web semânticos evoluem de forma atrasada em comparação aos avanços teóricos da tecnologia, sendo que os reais progressos da mesma só poderão ser alcançados quando forem possibilitadas ferramentas de fácil utilização para tal tecnologia. Outro grande problema é que a maioria das ferramentas disponíveis não tem um foco global, dando prioridade a somente um aspecto relevante dos serviços Web semânticos, sendo que raramente os usuários utilizam uma única tarefa. Um ambiente integrado de modelagem é uma abordagem mais adequada e produtiva.

Neste capítulo, será apresentado como estudo de caso um exemplo implementado por Stollberg et al. (2006), que representa de forma simples a aplicação da linguagem WSMO na seleção e descoberta de serviços, que são os princípios básicos de uma composição, através de descrições semânticas. Este exemplo foi apresentado na *21st National Conference on Artificial Intelligence*, no ano de 2006, em Boston, EUA, que é uma conferência realizada pela *Association for the Advancement of Artificial Intelligence* (AAAI).

No exemplo, foi utilizado o ambiente de modelagem WSMO-*Studio* na versão 0.4.1 integrado ao IRS-III *Server*, que armazena descrições semânticas e realiza a comunicação com os serviços Web.

Primeiro apresentaremos breves descrições das duas ferramentas utilizadas e em seguida as motivações e os passos para a construção do exemplo de Stollberg et al. (2006).

4.1. WSMO-STUDIO

A ferramenta WSMO-*Studio* [1], que funciona como uma extensão da plataforma de desenvolvimento Eclipse, é um ambiente integrado de modelagem do domínio de serviços Web semânticos, que suporta a perspectiva do padrão WSMO. Ela inclui um editor de ontologias e elementos WSMO (mediadores, metas e serviços Web), além de um construtor de coreografia, funcionalidades de importação e exportação de elementos e possibilidade de extensão com outros ambientes.

Dimitrov et al. (2007) descrevem as principais finalidades do WSMO-*Studio:*

[1] http://www.wsmostudio.org

1. suporte e elaboração da abordagem aos serviços Web semânticos, deixando a tecnologia acessível e de fácil utilização para usuários iniciantes. Esse é um ponto considerado como crucial para possibilitar a inovação tecnológica;
2. ambiente integrado que maximize a produtividade do usuário. Os usuários necessitam de funcionalidades que cubram um conjunto de tarefas de maneira integrada;
3. ambiente extensível, em que é possível adicionar novas funcionalidades ou modificar funcionalidades existentes.

Como características relacionadas ao suporte a serviços Web, podemos citar a modelagem de ontologias, a anotação semântica de serviços Web existentes, o trabalho com repositórios semânticos, para publicação, navegação e consulta de ontologias WSMO, serviços, metas e mediadores, descoberta de serviços baseada na meta, especificação de serviços compostos (interfaces de orquestração e coreografia) e interação em tempo de execução com outros ambientes, tais como WSMX e IRS-III (*Internet Reasoning Service*). O ambiente integrado do WSMO-*Studio* possibilita ao usuário não precisar utilizar mais de uma ferramenta para concluir seus objetivos. A extensibilidade é uma característica muitas vezes ignorada, mas é essencial pelo fato de que um só ambiente muitas vezes não consegue satisfazer todas as necessidades dos clientes, e ela proporciona isto.

4.2. IRS-III SERVER

O IRS-III *Server* é um servidor para elementos WSMO. Ele pode ser usado integrando-se ao WSMO-*Studio* na forma de um *plug-in*, permitindo a importação e exportação de entidades, além de proporcionar o alcance de uma meta a partir do próprio editor, através de uma simples interface.

Os elementos WSMO são utilizados pelo servidor em dois níveis distintos: (1) a descrição do nível de conhecimento dos componentes é representada internamente em OCML (*Operational Conceptual Modelling Language*) (DOMINGUE et al., 1999); e (2) mapeamentos são feitos para conectar as descrições do nível de conhecimento a um serviço Web específico, possivelmente externo.

4.3. DESENVOLVIMENTO DOS EXEMPLOS

Os exemplos foram construídos a partir de um cenário criado por Dimitrov et al. (2007), em que é necessária a criação de serviços para um sistema de uma agência de viagens de trens pela Europa, de forma que estes serviços não devam ser escolhidos manualmente, mas sim descobertos e invocados automaticamente, através da semântica. Os serviços serão responsáveis por disponibilizar a tabela de horários e fazer a reserva de passagens para um determinado itinerário escolhido pelo cliente.

Para o desenvolvimento do projeto, são utilizadas algumas ontologias previamente criadas, formando uma base de conhecimento para:

cidades européias, onde já estão instanciadas diversas cidades;

definição de personagens, onde estão instanciados os representantes de clientes usuários dos serviços;

conceitos comuns, onde está presente o conceito de cidades, países, entre outros, e;

descrições, que será utilizada para criar as instâncias de requisições dos clientes.

Tais ontologias se encontram armazenadas no servidor IRS-III e devem ser exportadas para um projeto do WSMO-*Studio,* onde serão armazenadas na forma de arquivos *.wsml*, podendo ser visualizadas e editadas através do código fonte ou pelo próprio editor da ferramenta. O projeto WSMO deve ser criado utilizando um namespace IRI (*Internationalised Resource Identifier*), que vai possibilitar a comunicação com um servidor externo, provedor dos serviços. O namespace a ser utilizado neste caso é o seguinte: *http://irs.open.ac.uk/europeanTravelServiceDescriptions#*.

A figura 4.1 mostra o repositório de ontologias, dentro do IRS-III *Server*.

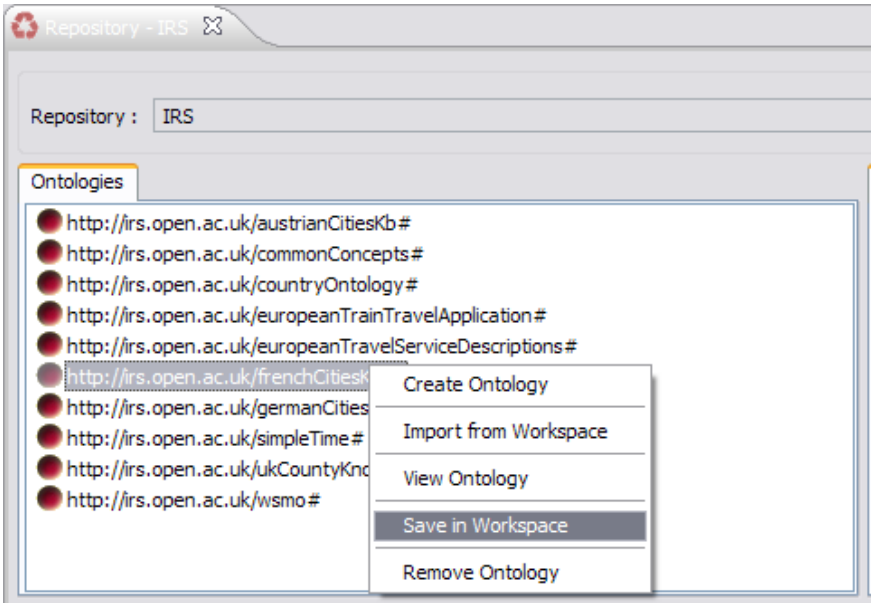

Figura 4.1: Repositório de Ontologias.

4.3.1. EXEMPLO DE REQUISIÇÃO DE TABELA DE HORÁRIOS

Este exemplo representa a requisição de uma tabela de horários de trens que fazem o transporte de passageiros entre cidades européias. O cliente deve fazer a solicitação através de um sistema, especificando a cidade de origem e destino do percurso, além da data de partida. Para o desenvolvimento será possível perceber a utilização de conceitos ontológicos na criação da meta e do serviço Web.

Dentro da ontologia de descrições, devem ser criados os conceitos para as requisições e respostas da meta a ser criada. Primeiramente são criados os termos referentes à meta de visualização da tabela de preços dos trens, como mostrado na figura 4.2.

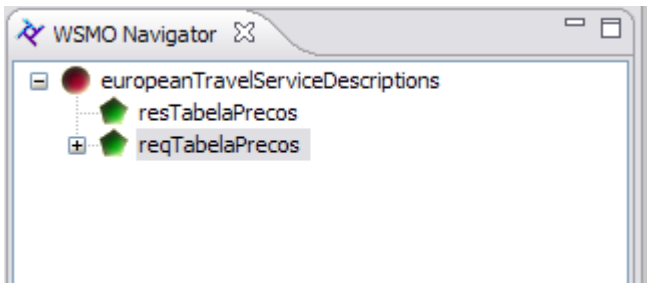

Figura 4.2: Navegador WSMO, ontologia de descrições.

Em seguida são adicionados os atributos para estes dois conceitos, como se segue: (1) o conceito de requisição (ilustrado pela figura 4.3) possui os atributos *origem, destino* e *data*, todos com cardinalidade unitária, sendo os dois primeiros do tipo *cidade* (*city*), que é um termo já existente na ontologia de conceitos comuns, e o último do tipo primário *date*; (2) o conceito de resposta (mostrado na figura 4.4) possui um atributo único chamado *tabelaDePrecos*, do tipo *string*.

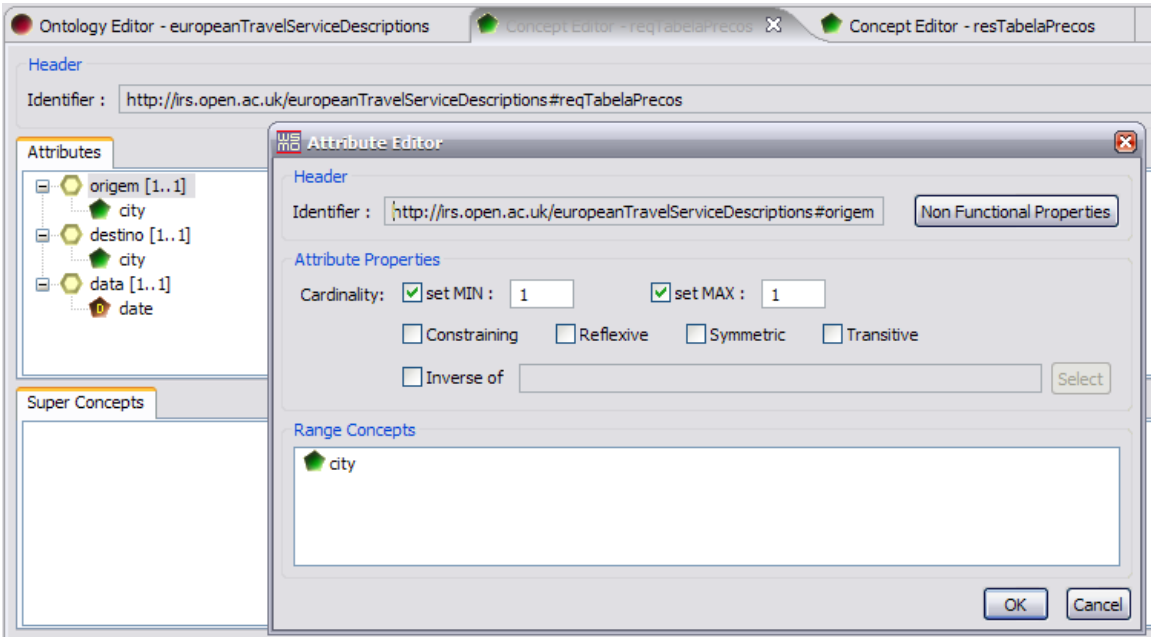

Figura 4.3: Editor de conceitos, requisição.

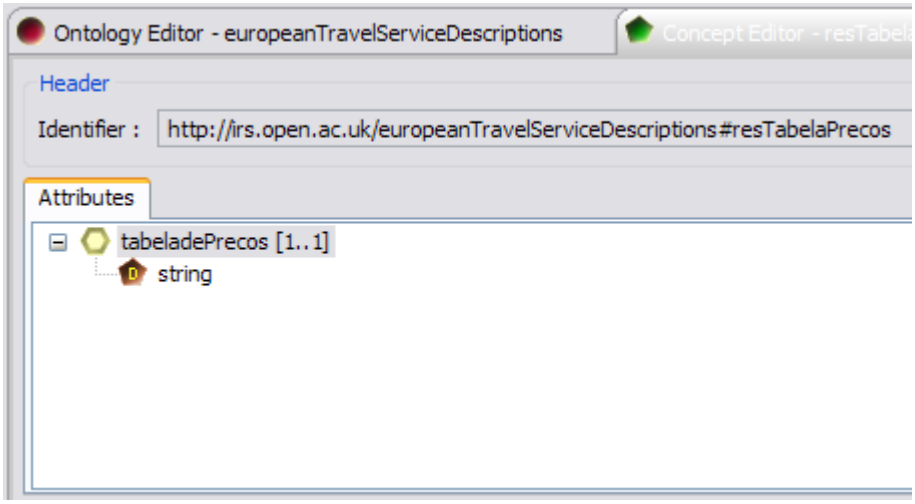

Figura 4.4: Editor de conceitos, resposta.

Após a definição dos conceitos de requisição e resposta, é criada a primeira meta a ser alcançada – a visualização da tabela de preços de determinado itinerário – selecionando a opção *New Goal* do editor. São solicitados o nome para o arquivo *wsml* e o nome identificador do elemento. Através da aba de edição da meta há a opção de importar ontologias, e, neste caso é importada a ontologia de descrições, onde estão os conceitos de descrições para a meta. Em seguida é criada a interface da meta, que permite a criação de uma coreografia e uma *state signature*, que especifica a ontologia do estado para a coreografia, onde são definidos os parâmetros de entrada e saída da meta, utilizando-se os conceitos de requisição e resposta do cliente (figura 4.5). Um importante detalhe a se frisar é que, em uma meta, a requisição do cliente faz parte da saída da meta, enquanto que a resposta refere-se à sua entrada.

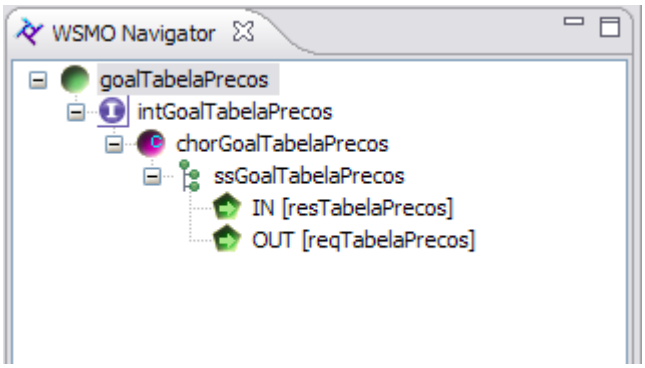

Figura 4.5: Coreografia, parâmetros de entrada e saída.

Após a implementação da meta, é criado um mediador do tipo *WebService-goal*, que é o elemento que faz a ligação entre a meta e os serviços Web que são capazes de alcançá-la. De forma similar à criação da meta, o mediador nasce da opção *New Mediator* a partir do projeto em questão, sendo necessária somente a escolha do tipo de mediador a ser criado, como mostra a figura 4.6. Dentro do editor do mediador deve ser adicionada sua meta de origem, no nosso caso, a meta *goalTabelaPrecos,* criada anteriormente.

Figura 4.6: Tipos de Mediador.

Por fim é criado o serviço Web, que deve importar a ontologia de descrições, assim como feito com a meta, além de adicionar o mediador criado acima na área de mediadores utilizados, como mostra a figura 4.7. Desta forma, está criada a ligação entre a meta, o mediador e o serviço Web.

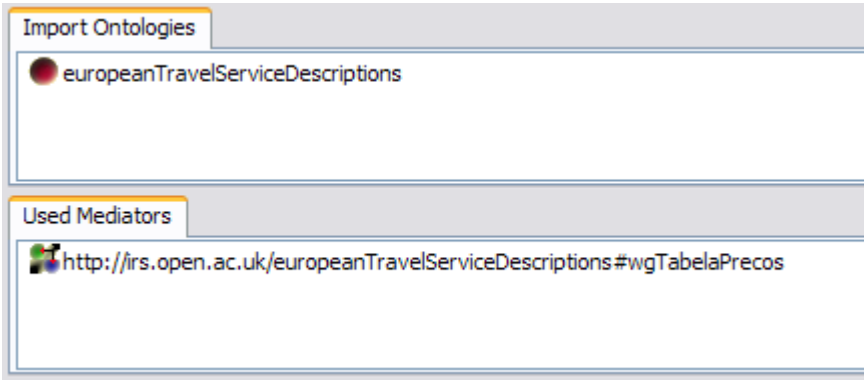

Figura 4.7: Importação das descrições e adição do mediador.

Ainda no serviço Web, cria-se uma interface e uma coreografia semelhante à meta, com a exceção de que, neste caso, o conceito de requisição diz respeito ao parâmetro de entrada do serviço, e a resposta ao parâmetro de saída. No serviço Web ainda há um detalhe a ser adicionado à requisição, chamado *grounding*, que é o elemento que vai realizar a conexão da requisição do elemento serviço Web com o serviço Web externo que vai ser invocado, como mostra a figura 4.8. Neste exemplo, os elementos de serviço Web funcionam como links para serviços Web reais, que existem em um servidor externo. O endereço da operação do serviço Web responsável por disponibilizar a tabela de preços é dado no exemplo: "*irslisp://localhost:3001/soap/europeanTravelServiceDescriptions/getTrainTimes*".

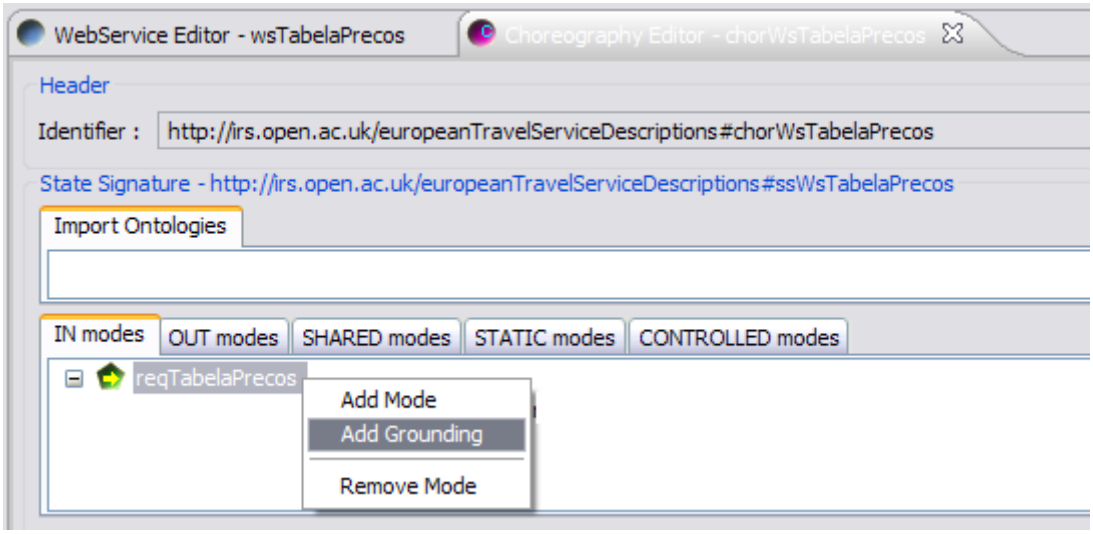

Figura 4.8: *Grounding*.

Após serem criados todos os elementos acima, é necessário exportá-los para dentro do IRS-III *Server*. Isto é feito arrastando os arquivos referentes à meta, ao mediador e ao serviço Web para dentro da visão do repositório, ilustrada pela figura 4.9.

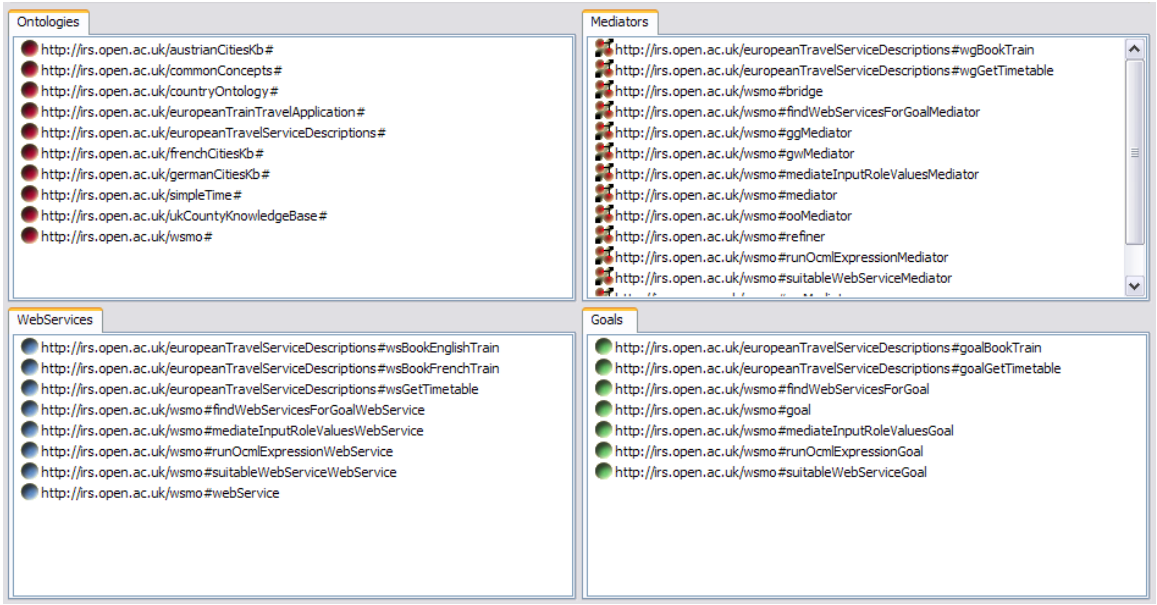

Figura 4.9: Visão dos elementos existentes no repositório do IRS-III.

Em seguida, é feita a instanciação de um objeto do conceito de requisição, que representará os dados da solicitação do cliente. Através do editor de conceitos é possível criar esta nova instância, chamada aqui de *testeTabelaPrecos* (figura 4.10) e atribuir os valores referentes aos atributos criados para o termo de requisição (figura 4.11).

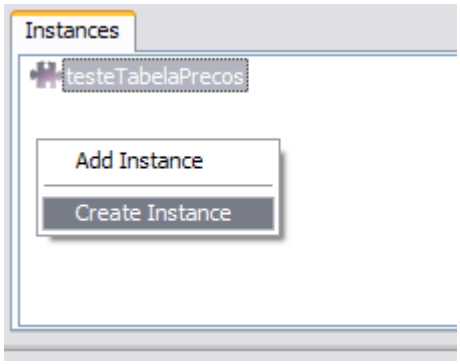

Figura 4.10: Criação da instância de um conceito.

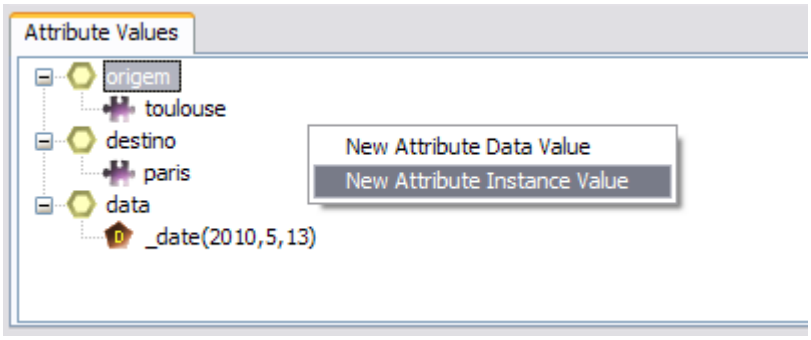

Figura 4.11: Atribuição de valores à instância.

Para invocar a meta desejada, basta utilizar a opção *Achieve Goal* do objeto de conexão com o repositório IRS-III, selecionar a meta e a instância de requisição desejada e clicar em *Execute*, como na figura 4.12. O resultado é apresentado em uma nova janela, como mostrado na figura 4.13.

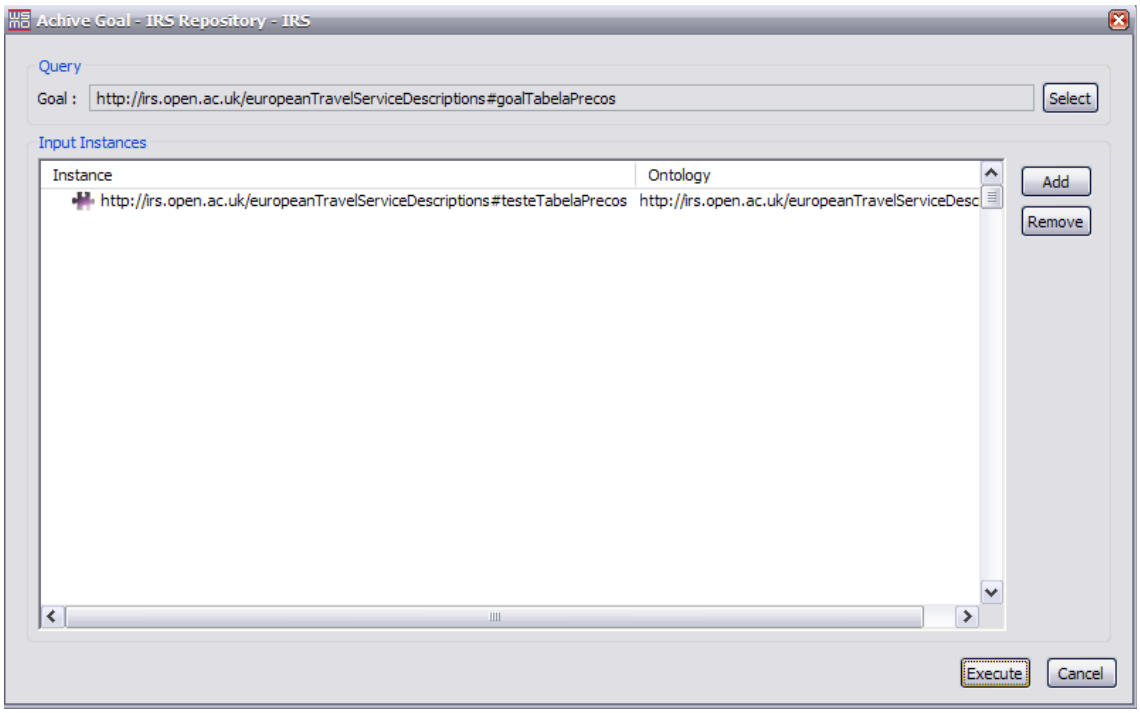

Figura 4.12: Invocação da meta.

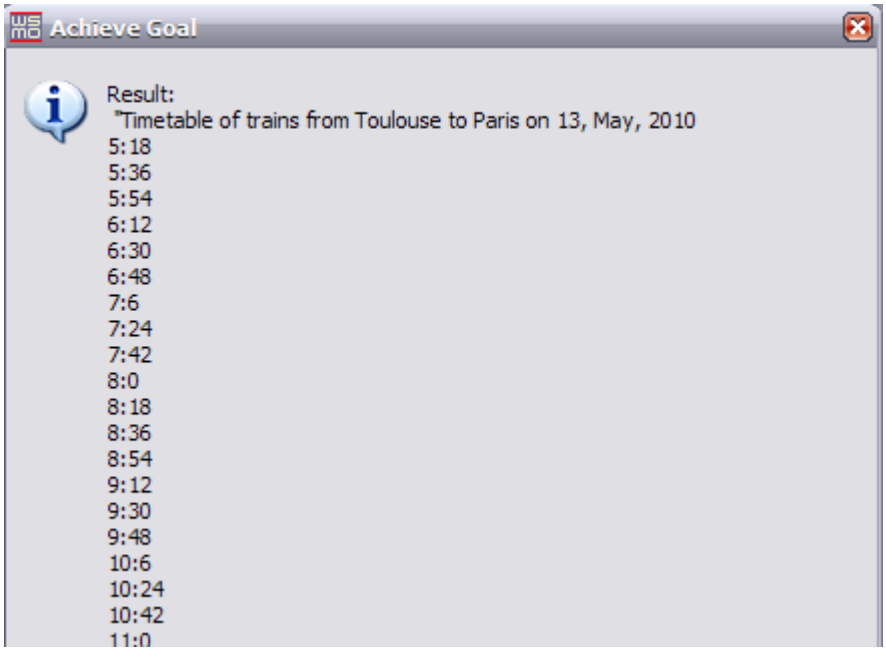

Figura 4.13: Resultado da execução do serviço.

Neste exemplo é possível perceber o uso dos conceitos ontológicos na hora de definir a requisição e resposta de um cliente, tanto dentro da meta quanto do serviço Web. Pode-se perceber também que, no momento da solicitação do serviço, não é especificado

um serviço Web a ser invocado, e isto ficará mais evidente com a apresentação do segundo exemplo.

4.3.2. EXEMPLO DE RESERVA DE PASSAGEM

Este exemplo demonstra um cenário em que um cliente deseja fazer uma reserva de passagem para uma determinada viagem entre duas cidades da Europa na data e horário desejados. Além destas informações, o cliente deve enviar o nome para o qual será feita a reserva. Serão utilizados conceitos ontológicos na criação da meta e dos serviços Web, além de uma linguagem lógica para descrever a capacidade dos serviços. A capacidade é o elemento que vai determinar qual serviço deverá ser invocado, dependendo da solicitação do cliente.

O desenvolvimento do exemplo poderá ser feito dentro do projeto criado anteriormente, pois serão utilizadas as mesmas ontologias. De forma similar ao primeiro exemplo, são criados os conceitos para requisição de resposta à solicitação, dentro da ontologia de descrições. A requisição deve possuir os atributos origem e destino – com o tipo *city*, da ontologia de conceitos comuns –, um atributo para informação de data e hora da reserva – do tipo *dateTime* –, e um atributo para o passageiro – do tipo *person*, que pode ser encontrado na ontologia de definição de personagens. A resposta deve possuir um único atributo, do tipo string, contendo a confirmação da reserva. Todos os atributos devem possuir cardinalidade unária.

Em seguida é criada a meta de reserva de passagens, importando-se a ontologia de descrições, e construindo a interface e coreografia do elemento, utilizando a mesma ideia anterior: dentro da meta, o conceito resposta é adicionado ao parâmetro de entrada (*in*) e a requisição se refere à saída (*out*).

Depois criamos um mediador da mesma maneira que foi mostrada no exemplo anterior, adicionando a meta de reserva de passagens dentro da área de origem do mediador.

Enfim, são criados os serviços Web, que são o diferencial deste exemplo. Criaremos dois serviços Web distintos: um para fazer a reserva de viagens entre cidades francesas, e outro para fazer a reserva de viagens entre cidades inglesas. Primeiramente é importada a ontologia de descrições, e adicionado o mediador criado à área de mediadores utilizados. As interfaces de ambos os serviços serão idênticas uma à outra, e similares à

interface da meta, com a diferença de que o conceito de requisição agora diz respeito ao parâmetro de entrada dos serviços, enquanto que o conceito de resposta se refere à saída do serviço. À requisição deve ser adicionado um elemento *grounding* com valores diferentes para cada um dos serviços:

serviço Web Francês: "*irslisp://localhost:3001/soap/europeanTravelServiceDescriptions/bookFrenchTrainJourney"*;

serviço Web Inglês: "*irslisp://localhost:3001/soap/europeanTravelServiceDescriptions/bookEnglishTrainJourney"*.

Assim como no primeiro exemplo, estes endereços serão responsáveis por invocarem o serviço Web de um servidor externo específico para reservas francesas e inglesas, respectivamente. Para diferenciar os dois serviços, que possuem a mesma interface, é criada a capacidade do serviço. Dentro da capacidade existem quatro subitens: pré-condições, suposições, pós-condições e efeitos. Usaremos neste exemplo o item suposições, criando um axioma para ele, que será o responsável por estabelecer "condições" para que o serviço seja chamado. Os nomes dos axiomas para o exemplo são pré-determinados, por motivos de comunicação com o serviço Web externo:

nome do axioma do serviço Web para reserva de trens da França: "*http://irs.open.ac.uk/europeanTravelServiceDescriptions#aspBookFrenchTrain"*

nome do axioma do serviço Web para reserva de trens da Inglaterra: *"http://irs.open.ac.uk/europeanTravelServiceDescriptions#aspBookEnglishTrain"*

Em seguida, dentro do axioma de cada serviço deve ser adicionada uma expressão lógica que determina que aquele serviço Web somente será invocado quando a meta tiver como origem e destino cidades francesas (figura 4.14) – no caso do serviço para reserva de trens da França – ou inglesas – no caso do serviço para reserva de trens da Inglaterra (figura 4.15).

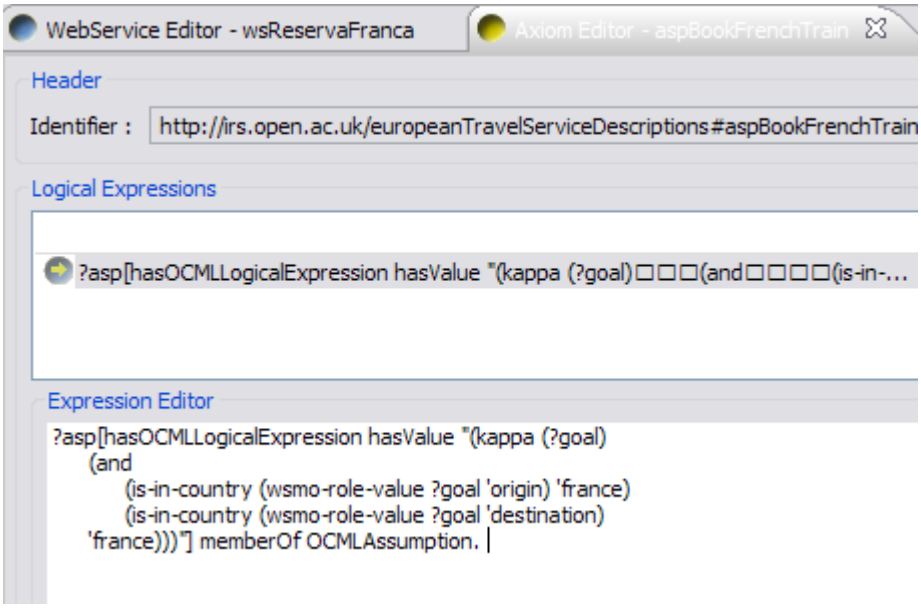

Figura 4.14: Expressão do axioma para cidades francesas.

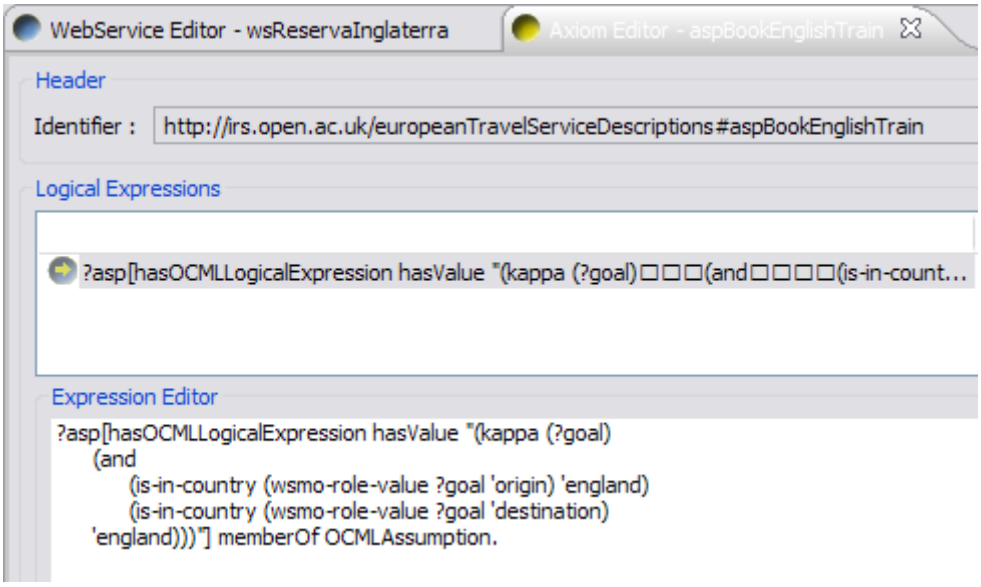

Figura 4.15: Expressão do axioma para trens ingleses.

Após a definição dos serviços, para alcançarmos a meta resta somente criar as instâncias referentes aos conceitos de requisição e resposta. Desta vez serão criadas duas instâncias (figuras 4.16 e 4.17), uma para cada serviço.

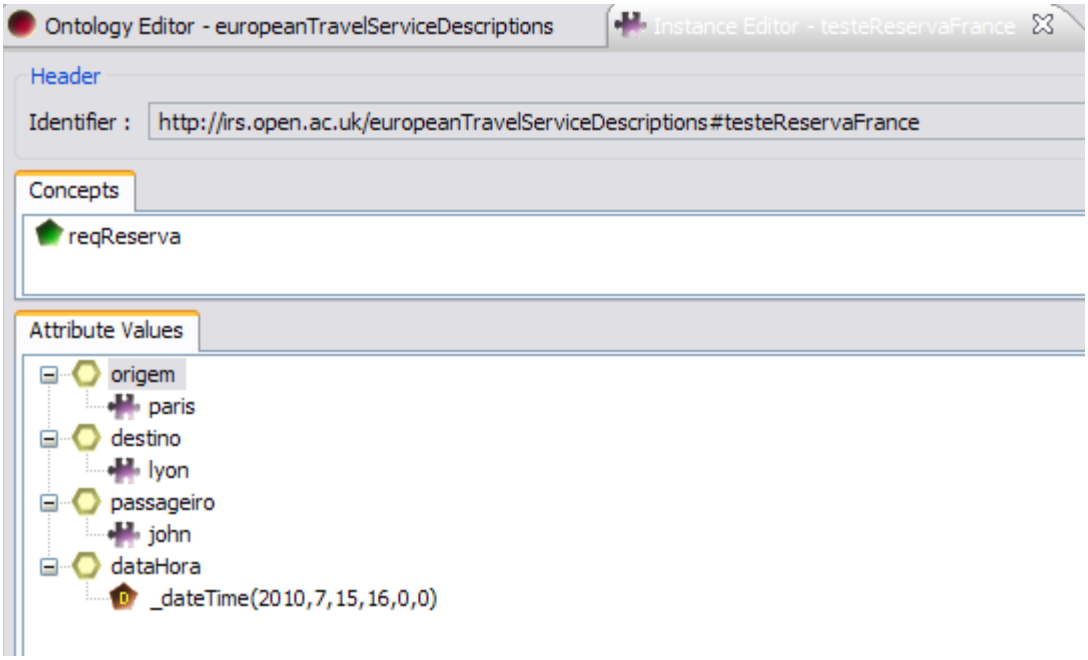

Figura 4.16: Instância de teste para reservas de trens da França

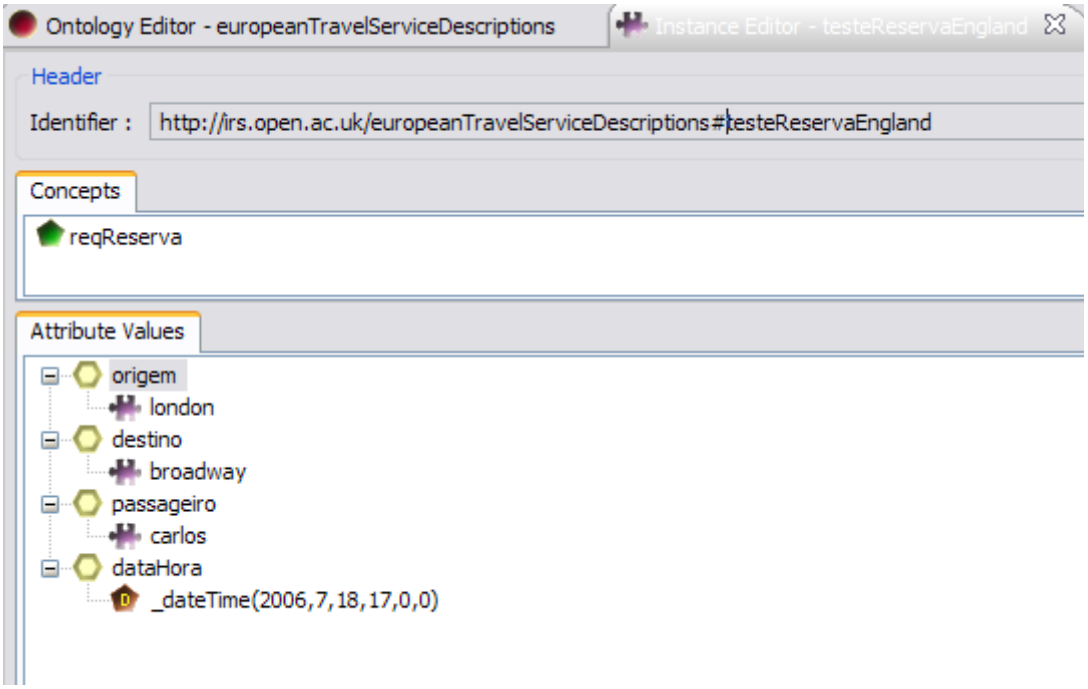

Figura 4.17: Instância de teste para reservas de trens da Inglaterra

Para invocar a meta, basta clicar em *Achieve Goal* e selecionar a meta de reserva e a instância desejada, seja para trens com origem e destino na Inglaterra ou na França. O resultado é a confirmação de reserva, que pode ser vista na figura 4.18.

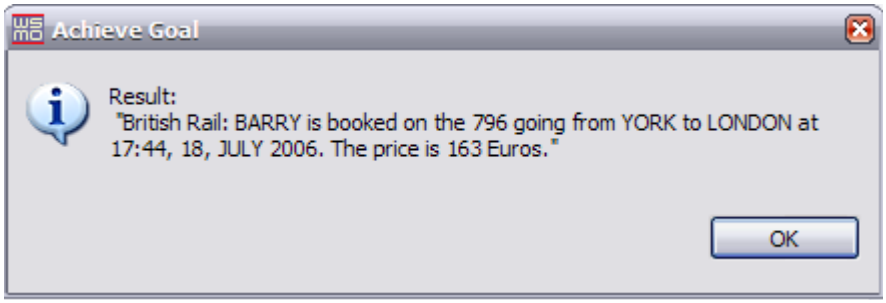

Figura 4.18: Meta de reserva de trem inglês alcançada.

Para comprovar que a capacidade do serviço Web determina se ele será invocado ou não, como teste pode-se criar uma instância de requisição em que a origem e o destino sejam cidades alemãs, por exemplo. Mesmo a instância respeitando a estrutura da interface dos serviços, o resultado será uma resposta nula, como mostra a figura 4.19, pois não foi implementado nenhum serviço Web com a capacidade de servir a itinerários de trens da Alemanha.

Figura 4.19: Meta para reserva de trens alemães não alcançada.

Este exemplo, apesar de simples, representa o funcionamento de anotações semânticas na capacidade de um serviço Web, possibilitando tomadas de decisões automáticas no momento da invocação de um serviço Web.

5. CONCLUSÃO

Durante o desenvolvimento deste trabalho foi possível concluir que o cenário da arquitetura orientada a serviços, apesar de já bastante conhecido e utilizado nos dias de hoje, ainda tem muito para evoluir, principalmente no sentido de ser menos dependente da ação humana, e se tornar cada vez mais capaz de tomar decisões automaticamente.

A aplicação da semântica no cenário dos serviços Web é um tema que tem sido bastante estudado nos meios acadêmicos, porém sua utilização em organizações empresariais ainda é pouco difundida. Foi visto que uma grande dificuldade para o crescimento de aplicações neste sentido é que as ferramentas que visam dar suporte à descrição semântica de serviços Web evoluem lentamente, em comparação aos avanços teóricos alcançados.

Neste trabalho foram reproduzidos exemplos de implementação a partir da ferramenta de modelagem de ontologias WSMO-*Studio* juntamente com um servidor IRS-III, utilizado para fazer a ligação com um servidor externo de onde os serviços foram invocados. Com a manipulação da ferramenta percebemos certa facilidade na construção de modelos ontológicos e no uso de conceitos criados para o desenvolvimento de coreografias das metas e serviços Web. Por outro lado, houve dificuldade na manipulação e criação de funções lógicas a serem utilizadas para a descrição da capacidade dos serviços Web.

Os exemplos apresentados foram criados para participação em um desafio científico, e possibilitaram uma visão prática do cenário da semântica na composição de serviços, de forma que podem servir de base para serviços de utilidade no mundo real. Apesar de serem implementações simplórias, permitem vislumbrar um futuro para a aplicação da semântica na composição de serviços Web.

REFERÊNCIAS BIBLIOGRÁFICAS

Agre, G., & Marinova, Z. (2007). An INFRAWEBS Approach to Dynamic Composition of Semantic Web Services.

Bosco, J. (2003). APIs de Busca e Publicação Em UDDI.

Casanova, M. A. (2009). Tecnologias de Banco de Dados para a Web Semântica - Módulo 10b – Composição de Serviços - Semantic Web Services.

DAML. (2005). Semantic Web Services Ontology (SWSO).

Kona, S., Bansal, A., Blake, M. B., & Gupta, G. (2008). Generalized Semantics-based Service Composition.

Moran, M., Vitvar, T., & Zaremba, M. (2007). Towards Constraint-Based Composition With Incomplete Service Descriptions.

Traverso, P., & Pistore, M. (2009). Automated Composition of Semantic Web Services into Executable Processes.

W3C. (2004a). OWL-S: Semantic Markup for Web Services.

W3C. (2004b). Resource Description Framework (RDF).

W3C. (2005a). Web Service Semantics - WSDL-S.

W3C. (2005b). Semantic Web Services Ontology (SWSO).

W3C. (2006). Semantic Annotations for WSDL Working Group.

WSMO. (2007). D11v0.2 WSMO-Lite.