UNIVERSIDADE DO RIO GRANDE DO NORTE
CENTRO DE CIÊNCIAS EXATAS E DA TERRA
DEPARTAMENTO DE INFORMÁTICA E MATEMÁTICA APLICADA
PROGRAMA DE PÓS-GRADUAÇÃO EM SISTEMAS E COMPUTAÇÃO

JOSÉ REGINALDO DE SOUSA MENDES JÚNIOR

WEBFLOWAH: UM AMBIENTE PARA ESPECIFICAÇÃO E EXECUÇÃO AD-HOC DE PROCESSOS DE NEGÓCIO BASEADOS EM SERVIÇOS WEB

Natal
2008

Livros Grátis

http://www.livrosgratis.com.br

Milhares de livros grátis para download.

JOSÉ REGINALDO DE SOUSA MENDES JÚNIOR

WEBFLOWAH: UM AMBIENTE PARA ESPECIFICAÇÃO E EXECUÇÃO AD-HOC DE PROCESSOS DE NEGÓCIO BASEADOS EM SERVIÇOS WEB

Dissertação apresentada ao Programa de Pós-Graduação em Sistemas e Computação do Departamento de Informática e Matemática Aplicada da Universidade Federal do Rio Grande do Norte como requisito parcial para a obtenção do título de Mestre em Sistemas e Computação.

Orientador: Prof. Dr. Paulo de Figueiredo Pires

Co-orientadora: Profa. Dra. Flávia Coimbra Delicato

Natal
2008

JOSÉ REGINALDO DE SOUSA MENDES JÚNIOR

WEBFLOWAH: UM AMBIENTE PARA ESPECIFICAÇÃO E EXECUÇÃO AD-HOC DE PROCESSOS DE NEGÓCIO BASEADOS EM SERVIÇOS WEB

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Sistemas e Computação e aprovada em sua forma normal pelo Programa de Pós-Graduação em Sistemas e Computação do Departamento de Informática e Matemática Aplicada da Universidade Federal do Rio Grande do Norte.

Aprovada em 29 de agosto de 2008

BANCA EXAMINADORA:

_____
Prof. Dr. Paulo de Figueiredo Pires – UFRN

_____
Profa. Dra. Flávia Coimbra Delicato – UFRN

_____
Profa. Dra. Thaís Vasconcelos Batista – UFRN

_____
Profa. Dra. Marta Lima de Queirós Mattoso – UFRJ

AGRADECIMENTOS

Aos meus orientadores Dr. Paulo de Figueiredo Pires e Dra. Flávia Coimbra Delicato, pela dedicação, paciência, confiança, compreensão e orientação.

À professora Dra. Marta Lima de Queirós Mattoso, por aceitar participar desta banca, agregando valor inestimável a este trabalho juntamente com a professora Dra. Thaís Vasconcelos Batista, que também prestou grande atenção desde as primeiras orientações para ingressar no mestrado.

Aos professores e funcionários do DIMAp, pelo apoio e serviços prestados.

Aos companheiros do PPgSC pela amizade e horas de diversão e de estudo.

Ao CNPq, pelo apoio financeiro ao desenvolvimento deste trabalho.

Aos ex-colegas professores substitutos do DIMAp pelo companheirismo.

Ao professor Dr. Adilson Lopes e à Maria da Graça pela recepção ao DIMAp e pela amizade construída ao longo do tempo.

Aos colegas de trabalho no SERPRO que passaram pela mesma experiência e me apoiaram a não desistir deste objetivo.

À minha mãe e irmãos, pelo apoio em cada etapa da minha vida.

E, finalmente, a Deus pela oportunidade e pelo privilégio que me foi dado em aprender novos conhecimentos e poder compartilhar-lhos.

RESUMO

O enfoque dado ultimamente às tecnologias de Serviços *Web* e da *Web* Semântica tem proporcionado o desenvolvimento de vários projetos de pesquisa abordando, de diferentes maneiras, o tema da composição de serviços *Web*. Entretanto, o desafio de criar um ambiente que favoreça a especificação de um processo de negócio abstrato e que seja implementado automaticamente por serviços compostos de forma dinâmica é considerado atualmente um problema em aberto. Os padrões WSDL e BPEL providos pela a indústria de *software* suportam apenas a composição manual de serviços, pois falta a eles a semântica necessária para que os serviços *Web* sejam descobertos, selecionados e combinados por agentes de *software*. As ontologias de serviço providas pela *Web* Semântica enriquecem as descrições sintáticas dos serviços *Web* de modo a facilitar a automação de tarefas, como a descoberta e a composição. Este trabalho propõe um ambiente para a especificação e execução *ad-hoc* de processos de negócio baseados em serviços *Web* chamado *WebFlowAH*. Este ambiente emprega uma ontologia de domínio comum para descrever tanto os serviços *Web* e quanto os processos de negócio. Ele permite a especificação de processos em termos de desejos ou objetivos do usuário que são expressos por conceitos da ontologia de domínio comum. Tal abordagem permite que processos de negócio sejam especificados de maneira abstrata e de alto nível, liberando o usuário de detalhes necessários para executar eficazmente o *workflow* do processo.

Palavras-chave: *Web* Semântica, Ontologia, *Workflow*, OWL-S, Descoberta e Composição de Serviços *Web*

ABSTRACT

Recently the focus given to Web Services and Semantic Web technologies has provided the development of several research projects in different ways to addressing the Web services composition issue. Meanwhile, the challenge of creating an environment that provides the specification of an abstract business process and that it is automatically implemented by a composite service in a dynamic way is considered a currently open problem. WSDL and BPEL provided by industry support only manual service composition because they lack needed semantics so that Web services are discovered, selected and combined by software agents. Services ontology provided by Semantic Web enriches the syntactic descriptions of Web services to facilitate the automation of tasks, such as discovery and composition. This work presents an environment for specifying and ad-hoc executing Web services-based business processes, named WebFlowAH. The WebFlowAH employs common domain ontology to describe both Web services and business processes. It allows processes specification in terms of users' goals or desires that are expressed based on the concepts of such common domain ontology. This approach allows processes to be specified in an abstract high level way, unburdening the user from the underline details needed to effectively run the process workflow.

Keywords: Semantic Web, Ontology, Workflow, OWL-S, Web Service Discovery and Composition.

LISTA DE FIGURAS

Figura 2.1. Modelo de Referência de *Workflow*. Fonte: WfMC (1998)..... 18
Figura 2.2. Definição de processo em XPDL para um processo de venda de produtos..... 19
Figura 2.3. Arquitetura dos serviços *Web*. Fonte: W3C (2004a) 23
Figura 2.4. Trecho da ontologia OWL do domínio de Turismo..... 27
Figura 2.5. Consulta semântica em RDQL e SPARQL..... 30
Figura 2.6. Ontologia OWL-S. Fonte: DAML (2004) 31
Figura 2.7. Trecho da ontologia OWL-S do serviço *Web* ReservarVooService 32
Figura 2.8. Trecho da ontologia WSMO do serviço *Web* ReservarVooService 33
Figura 3.1. Fluxo de mensagens do processo de reserva de pacotes turísticos 37
Figura 3.2. Protocolo de interação em BPEL do processo de reserva de pacotes turísticos 38
Figura 4.1. Conceitos básicos do ambiente WebFlowAH..... 54
Figura 4.2. Classificação de conceitos nas ontologias de domínio comum. 57
Figura 4.3. Descrição WSDL do serviço *Web* CompanhiaArea..... 59
Figura 4.4. Descrição OWL-S do serviço *Web* CompanhiaArea 60
Figura 4.5. Arquitetura conceitual do WebFlowAH. 61
Figura 4.6. Perfis de Usuário do WebFlowAH. 63
Figura 4.7. Funcionalidades do Administrador de Serviços..... 64
Figura 4.8. Funcionalidades do Especificador de Processos. 65
Figura 4.9. Tela de Especificação de Processos do WebFlowAH..... 67
Figura 5.1. Classes do componente Repositório Semântico..... 71
Figura 5.2. Ontologia Servicos.owl 73
Figura 5.3. Definição das ontologias de Objeto e Tarefa. 75
Figura 5.4. *Workflow* semântico do processo de reserva de passagens aéreas. 76
Figura 5.5. Objetivo <*Reservar, Voo*> do processo de reserva de passagens aéreas. 77
Figura 5.6. Consulta semântica que retorna os serviços para o objetivo <*Reservar, Voo*>. ... 79
Figura 5.7. Plano de execução do processo de reserva de passagens aéreas. 81
Figura 6.1. Tela Classificar Conceitos do WebFlowAH 86
Figura 6.2. Tela Assistente de *workflow* do WebFlowAH 90
Figura 6.3. Planos de execução PLAN1 e PLAN2 do caso de teste CST01 90
Figura 6.4. Planos de execução PLAN3 e PLAN4 do caso de teste CST01 91
Figura 6.5. Plano de execução PLAN5 do caso de teste CST01 92

Figura 6.6. Plano de execução PLAN6 do caso de teste CST01 93
Figura 6.7. Plano de execução PLAN7 do caso de teste CST05 94
Figura 6.8. Plano de execução PLAN8 do caso de teste CST05 95
Figura 6.9. Plano de execução PLAN9 do caso de teste CST07 96

LISTA DE TABELAS

Tabela 3.1. Comparação entre ambientes de composição 50
Tabela 5.1. Métodos da classe DocumentoOWLS 72
Tabela 5.2. Métodos da classe ServicoOWL 73
Tabela 5.3. Métodos da classe ConceitoOWL 74
Tabela 5.4. Métodos da classe WorkflowOWLS 75
Tabela 5.5. Métodos da classe CompositorSemantico 78
Tabela 5.6. Métodos da classe SeletorSemantico 80
Tabela 6.1. Funcionalidades do serviços *Web* criados no estudo de caso 84
Tabela 6.2. Análise dos resultados do estudo de caso 97

LISTA DE ABREVIATURAS E SIGLAS

BPEL – Business Process Execution Language for Web Services

BPM – Business Process Management

DL – Description Logics

HTTP – Hyper Text Transfer Protocol

IA – Inteligência Artificial

IU – Interface com Usuário

OWL – OWL Web Ontology Language

OWL-S – OWL Web Ontology Language for Web Services

RDF – Resource Description Framework

RDQL – RDF Data Query Language

SPARQL – Simple Protocol and RDF Query Language

SOAP – Simple Object Access Protocol

SWRL – Semantic Web Rule Language

UDDI – Universal Description, Discovery and Integration

UML – Unified Modeling Language

URI – Uniform Resource Identifier

URL – Uniform Resource Locator

WfMC – Workflow Management Coalition

WMS – Workflow Management System

W3C – World Wide Web Consortium

WSDL – Web Service Description Language

WSMF – Web Service Modeling Framework

WSML – Web Service Modeling Language

WSMO – Web Service Modeling Ontology

XML – EXtensible Markup Language

XPDL – XML Process Definition Language

XSD – XML Schema Definition

SUMÁRIO

1 INTRODUÇÃO 12
1.1 MOTIVAÇÃO 13
1.2 OBJETIVOS 15
1.3 ORGANIZAÇÃO DO TRABALHO 16
2 FUNDAMENTOS TEÓRICOS 17
2.1 SISTEMAS DE WORKFLOW 17
2.1.1 Conceitos Fundamentais 17
2.1.2 Benefícios para a Gerência de Processos 20
2.2 SERVIÇOS WEB 21
2.2.1 Definição 22
2.2.2 Visão da Arquitetura 22
2.2.3 Contribuições para a Automação de Processos 24
2.3 WEB SEMÂNTICA 26
2.3.1 Ontologia 27
2.3.2 Máquinas de Inferência 29
2.3.3 Serviços *Web* Semânticos 30
3 A COMPOSIÇÃO DE SERVIÇOS WEB 35
3.1 SOLUÇÕES ATUAIS DE COMPOSIÇÃO 35
3.2 CLASSIFICAÇÃO DAS TÉCNICAS DE COMPOSIÇÃO 41
3.2.1 Composição Baseada em Modelos 42
3.2.2 Composição Baseada em *Workflows* 43
3.2.3 Composição Baseada em Ontologias 45
3.2.4 Composição Baseada em Planificação IA 48
3.3 CONSIDERAÇÕES FINAIS 49
4 O AMBIENTE WEBFLOWAH 53
4.1 CONCEITOS BÁSICOS 54
4.1.1 Ontologia de Domínio Comum 55
4.1.2 Objetivo de Negócio 57
4.1.3 *Workflow* Semântico 58
4.1.4 Plano de Execução 60
4.2 ARQUITETURA CONCEITUAL 61

4.3 FUNCIONAMENTO DO AMBIENTE 63
4.3.1 Especificação do *Workflow* Semântico 66
4.3.2 Composição de Planos de Execução 68
4.3.3 Execução do *Workflow* Semântico 69
5 IMPLEMENTAÇÃO 70
5.1 REPOSITÓRIO SEMÂNTICO 70
5.2 COMPOSITOR SEMÂNTICO 77
5.3 SELETOR SEMÂNTICO 79
5.4 EXECUTOR DE WORKFLOWS 81
6 ESTUDO DE CASO 83
6.1 SERVIÇOS WEB UTILIZADOS 83
6.2 CASOS DE TESTE 88
6.2.1 Definição dos Casos de Teste 88
6.2.2 Execução dos Casos de Teste 89
6.3 ANÁLISE DOS RESULTADOS 97
7 CONCLUSÃO E TRABALHOS FUTUROS 99
7.1 CONTRIBUIÇÕES 100
7.2 TRABALHOS FUTUROS 101
REFERÊNCIAS 102

1 INTRODUÇÃO

A tecnologia de Serviços *Web* traz consideráveis benefícios à gerência de processos de negócio, uma vez que propicia a agilidade requerida pelos empreendimentos frente às necessidades atuais das rápidas mudanças no ambiente de negócios (DUSTDAR; SCHREINER, 2005; HEUVEL; YANG; PAPAZOGLOU, 2001). Ao viabilizarem um alto grau de interoperabilidade e autonomia, os serviços *Web* ampliam as fronteiras da condução de negócios através da *Internet*, pois fornecem uma infra-estrutura básica para a construção de novos processos de negócio de elevado valor agregado, a partir da composição de serviços *Web* básicos.

Uma característica marcante do ambiente provido pela tecnologia de Serviços *Web* é a possibilidade de criação de um mercado (semi) aberto de serviços, nos mesmos moldes com que os serviços comerciais são tradicionalmente oferecidos em nossa sociedade (BURBECK, 2000). Um serviço típico de venda de livros oferecido por uma livraria, por exemplo, poderia ser realizado na *Web* com a combinação dos serviços **catálogo de produtos**, **carrinho de compras** e **pagamento de compras**, caso esses serviços fossem disponibilizados como serviços *Web*. Devido ao fator escala da *Web*, uma grande variedade de serviços pode ser oferecida para a realização desse serviço de venda de livros, pois os serviços mencionados anteriormente podem ser providos por diversas organizações. Sendo assim, a combinação de serviços é mais do que simplesmente ligar as operações dos serviços disponíveis por meio dos parâmetros de entrada e saída. Essa combinação requer também que os mesmos sejam descobertos e selecionados a partir da *Web*. Além disso, pode ocorrer de um determinado serviço *Web* não existir ou ficar indisponível, necessitando que um novo serviço seja composto a partir dos serviços existentes. Portanto, para a criação de um mercado de serviços nesse ambiente é essencial a utilização de mecanismos (tais como a categorização e a busca de serviços) que permitam a descoberta, seleção e composição automáticas de serviços *Web* disponíveis.

Com relação à descoberta e à seleção de serviços, os padrões existentes de publicação e localização de serviços *Web* (UDDI, 2003; W3C, 2003) focam na definição sintática de atributos funcionais, os quais são voltados para a invocação do serviço, não considerando, portanto, outras informações importantes para a descoberta e seleção, como, por exemplo, critérios de qualidade de serviço (*Quality of Service – QoS*) ou regras de utilização do serviço (definidas por acordos de uso de serviço) (AGARWAL *et al.*, 2008

ARPINAR *et al*., 2004). Já com relação à composição de serviços, as próprias linguagens que especificam o fluxo de um processo de negócio em tempo de projeto (IBM, 2002; WfMC, 1998) necessitam que todas as atividades envolvidas no processo (implementadas por um ou mais serviços *Web*), bem como a coordenação exigida para a execução dessas atividades (a seqüência na qual os serviços *Web* serão invocados), sejam definidas manualmente pelo analista de negócio no momento da especificação do processo.

Embora os padrões mencionados anteriormente sejam amplamente utilizados na automação de processos estáticos (ou seja, que não sofrem mudanças ao longo do tempo), quando esses padrões são aplicados aos processos dinâmicos (como é o caso dos negócios eletrônicos onde as mudanças são mais comuns) a composição de serviços torna-se uma tarefa tão complexa que não pode ser tratada apenas manualmente (RAO; SU, 2004). Além de exigir um conhecimento aprofundado das descrições WSDL dos serviços *Web* para selecionar os serviços adequados ao processo de negócio, alterações na especificação atual do processo precisam ser realizadas constantemente para atender as novas necessidades do negócio. O simples fato de um serviço *Web* ficar indisponível durante a execução de um processo de negócio requer a substituição imediata desse serviço por outro idêntico (isto é, que contenha a mesma funcionalidade provida) de modo a efetivar corretamente essa execução. Em resumo, para tirar pleno proveito do potencial do ambiente *Web* no contexto da gerência de processos de negócios, os padrões e linguagens atuais ainda carecem dos mecanismos necessários para permitir a execução automatizada de serviços *Web*, isto é, a automação da descoberta, seleção, composição e invocação.

1.1 MOTIVAÇÃO

Uma das abordagens que têm sido adotadas para aumentar o grau de automação na execução de serviços *Web* é a ***Web* Semântica**. Visando fornecer descrições semanticamente enriquecidas dos serviços *Web*, a área da *Web* Semântica chamada **Serviços *Web* Semânticos** propõe o emprego de *ontologias de serviço* (DAML, 2004; WSMO, 2005) como forma de facilitar a automação de tarefas como a descoberta e a composição de serviços. De maneira simplificada, uma *ontologia* pode ser vista como uma descrição formal dos conceitos de um domínio específico e dos relacionamentos entre esses conceitos (GRUBER, 1993). Por sua vez, as ontologias de serviço representam as ontologias que descrevem formalmente as

propriedades e funcionalidades providas pelos serviços *Web* a partir de um conjunto de definições e significados, tornando esses serviços interpretáveis aos agentes de *software*.

Entretanto, o ferramental fornecido pela *Web* Semântica não é suficiente para programar um ambiente de processos de negócio baseados na composição e execução automáticas de serviços *Web*. Tais ambientes necessitam de mecanismos que sejam capazes de compreender a semântica das descrições dos serviços e, a partir dessas descrições, permitam: (i) a especificação do processo de negócio em um alto nível de abstração; (ii) a descoberta e seleção automática de serviços disponíveis que atendam as atividades especificadas no processo; (iii) a síntese (geração do plano de composição) do conjunto de serviços selecionados; e (iv) a orquestração desses serviços (a coordenação das chamadas desses serviços). Em outras palavras, esses são os requisitos necessários para que um único ambiente proveja a especificação e execução de processos de negócios baseados em serviços *Web* (MENDES JR *et al.*, 2007).

O presente trabalho propõe uma solução que permite a especificação de processos de negócio com base somente nas metas ou objetivos a serem atingidos (MILANOVIC; MALEK, 2006; ROMAN *et al.*, 2005), ao contrário de outras propostas (LAUKKANEN; HELIN, 2003; MAJITHIA; WALKER; GRAY, 2004; PATHAK *et al.*, 2006; TRAINOTTI *et al.*, 2005) baseadas na coreografia de serviços (isto é, na colaboração dos serviços envolvidos), onde um projetista deve fornecer uma especificação detalhada e completa do processo. Mesmo que essa especificação seja realizada de forma abstrata (entradas, saídas, precondições e pós-condições requeridas por uma atividade), tais propostas exigem que um projetista conheça todas as atividades realizadas pelo processo de negócio. A solução proposta neste trabalho procura simplificar a especificação definido apenas os objetivos do processo, os quais são descritos por conceitos de uma ontologia de domínio comum (CAMPOS; PIRES; CAMPOS, 2004; GUARINO, 1998). A partir desses objetivos é possível concluir, com ajuda de mecanismos de inferência (MCBRIDE, 2001; SIRIN *et al.*, 2005), quais atividades são necessárias para alcançar um objetivo, e conseqüentemente, via descoberta e composição dinâmicas de serviços *Web* (COSTA; PIRES; MATTOSO, 2004a, 2004b), construir um conjunto de serviços que, sintetizados (combinados) de determinada forma, representam uma especificação de processo que atende aos objetivos estabelecidos pelo projetista (caso esse conjunto exista).

Além de permitir a construção completa de especificações de processos de negócio a partir de objetivos de alto nível, tal solução permite também que o projetista interaja com o ambiente durante a construção da especificação de objetivos. Esse mecanismo de interação

funciona como um auxílio ao projetista, indicando caminhos possíveis para a realização da especificação, tendo como base as informações disponíveis no ambiente e solicitando informações adicionais visando à completude da especificação. O processo de síntese explora ainda a existência de serviços *Web* que forneçam funcionalidades equivalentes, de forma a gerar especificações de processos que incluam alternativas de execução, com o intuito de aumentar a robustez da especificação gerada.

1.2 OBJETIVOS

Este trabalho tem como meta projetar e implementar um ambiente, denominado **WebFlowAH**, para especificação e execução *ad-hoc* de processos de negócio orientados a objetivos e com base nas descrições semânticas de serviços *Web* (MENDES JR *et al.*, 2007). Tal ambiente permite que processos de negócio sejam especificados em um alto nível de abstração a partir das ontologias dos serviços, e implementados com a descoberta, seleção, composição e invocação automática de serviços *Web*, atendendo aos requisitos enumerados na motivação deste trabalho. Além disso, o ambiente WebFlowAH permite a simulação de processos de negócio onde, a partir da sua especificação abstrata, é implementado automaticamente um processo real contendo um conjunto de serviços *Web*, sem a necessidade de intervenção de um programador para a construção do mesmo. O termo *ad-hoc* advém do fato deste ambiente possibilitar que especificações de processos sejam elaboradas de forma não sistematizada e/ou planejada por um usuário, de modo semelhante ao que ocorre na elaboração de consultas nos sistemas de gerência de banco de dados atuais. Ou seja, um usuário não necessita saber a priori todos os detalhes necessários à implementação do processo de negócio para elaborar a sua especificação.

Para atingir essa meta estabelecida os seguintes objetivos específicos foram elaborados: (i) classificação dos conceitos contidos nas ontologias dos serviços *Web* em conceitos mais genéricos via hierarquia das ontologias, a fim de uniformizar o vocabulário de negócio definido em cada ontologia de um serviço *Web*, além de favorecer a especificação dos objetivos de um processo de negócio por meio desses conceitos; (ii) análise e integração das tecnologias (ferramentas) da *Web* Semântica e da automação de processos, mais precisamente aquelas voltadas para a descoberta e composição automática de serviços *Web*, para que o ambiente suporte tais mecanismos; (iii) automatização da implementação de

processos de negócio utilizando uma linguagem de especificação de processo já existente; (iv) produção de uma interface operacional que facilite o trabalho do usuário na especificação e execução de processos com o uso de serviços *Web*, e (v) elaboração de um estudo de caso para validar a solução proposta neste trabalho.

1.3 ORGANIZAÇÃO DO TRABALHO

O restante deste trabalho está organizado da seguinte maneira. O Capítulo 2 faz uma revisão da literatura, descrevendo os conceitos fundamentais e as tecnologias voltadas para a automação de processos de negócio aliada à descoberta e composição automáticas de serviços *Web*. O Capítulo 3 aborda o problema da composição de serviços *Web* e apresenta alguns trabalhos que estão sendo realizados nessa linha de pesquisa e que estão relacionados com a presente proposta. O Capítulo 4 mostra a arquitetura e as principais funcionalidades do ambiente WebFlowAH, enquanto que o Capítulo 5 descreve detalhes de implementação dos componentes mais importantes deste ambiente. O Capítulo 6 apresenta um estudo de caso como forma de demonstrar a utilização do ambiente proposto. O Capítulo 7 contém as conclusões deste trabalho bem como sugestões de trabalhos futuros.

2 FUNDAMENTOS TEÓRICOS

A *Business Process Management*[1] (BPM), como é usualmente chamada a gerência de processos de negócio, sempre buscou ao longo do tempo tornar os processos das empresas mais eficientes no que diz respeito à utilização dos recursos, a redução dos custos, ao aumento da produtividade e a flexibilidade frente às necessidades de mudança dos negócios (BROWN, 2005). Entretanto, esse desafio exigiu inúmeros esforços por parte da comunidade acadêmica e da indústria de *software* em desenvolver soluções que fossem incorporadas pelas empresas através da Tecnologia da Informação. Sendo assim, este capítulo faz uma breve revisão literária das principais tecnologias existentes até o presente momento, na área de BPM, e que foram concebidas a partir desses esforços realizados. Além disso, essas tecnologias serviram de embasamento teórico para a execução deste trabalho. Na Seção 2.1, é apresentada a tecnologia dos Sistemas de *Workflow*. Na Seção 2.2, é apresentada a tecnologia de Serviços *Web*. E na Seção 2.3, é apresentada a tecnologia da *Web* Semântica.

2.1 SISTEMAS DE WORKFLOW

O desenvolvimento de aplicações voltadas para o controle e a execução de processos de negócio foi suportado, a princípio, pelo uso dos Sistemas de *Workflow*. Nesta seção serão apresentados os conceitos-chave relacionados a essa tecnologia juntamente com os benefícios alcançados pela sua utilização na gerência de processos de negócio.

2.1.1 Conceitos Fundamentais

Segundo a entidade *Workflow Management Coalition*[2] (WfMC), um **processo de negócio** é um conjunto de atividades interligadas e que coletivamente realizam um objetivo ou uma meta de negócio (WfMC, 1998). Por exemplo, um processo de venda de livros pode ser descrito somente pelas seguintes atividades: 1) consultar um livro pelo título; 2) verificar

[1] http://www.bpmi.org/
[2] http://www.wfmc.org/

disponibilidade em estoque; 3) adicionar o livro desejado ao carrinho de compras; 4) efetivar o pagamento; e 5) finalizar a compra. Um ***workflow*** é justamente a automação de um processo de negócio ou parte dele, onde documentos, informações e tarefas são passadas de um participante para outro através de uma ação, segundo um conjunto de normas e procedimentos (WfMC, 1998).

A automação de um processo de negócio consiste em definir, criar, executar, monitorar e refinar *workflows* com o uso de um sistema de *software* (WfMC, 1998), mais conhecido como *Workflow Management System* (WMS) ou simplesmente **Sistema de *Workflow***. Tal sistema é capaz de interpretar uma descrição formal de um processo de negócio, expressa normalmente em uma linguagem de alto nível, e de manter os estados de execução do mesmo, delegando atividades às pessoas e aplicações envolvidas. O acompanhamento da execução do processo automatizado pode ser realizado com o uso de ferramentas de monitoração e auditoria, e a interação com aplicações externas ao processo, pode ser realizada através de mecanismos que viabilizam a comunicação entre sistemas distribuídos. Todas essas funcionalidades são identificadas no Modelo de Referência de *Workflow* (ilustrado pela Figura 2.1), que foi elaborado pela própria WfMC com o objetivo de especificar os componentes e as interfaces padrão de um WMS.

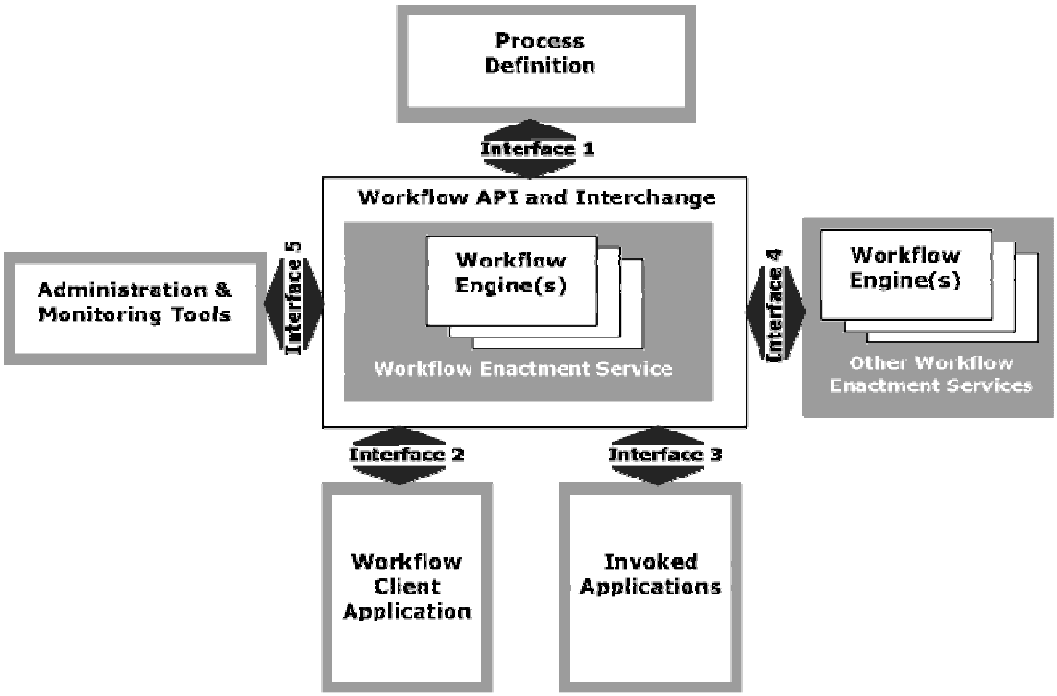
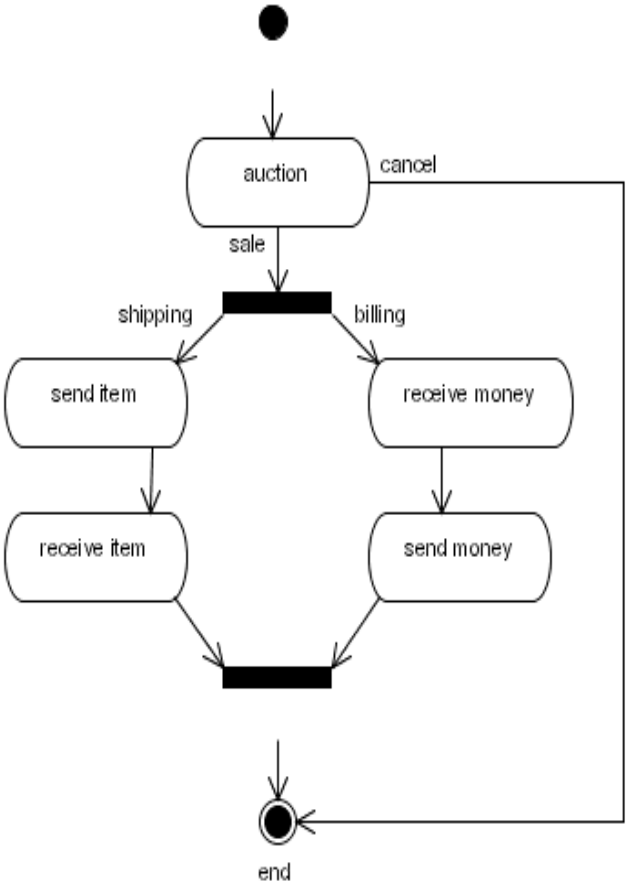


Figura 2.1. Modelo de Referência de *Workflow*. Fonte: WfMC (1998)

No WMS, a representação de um processo de negócio em um *workflow* é chamada de **definição de processo** (em inglês *process definition*), e consiste em especificar um grafo

contendo as atividades do processo e suas respectivas ligações, os critérios que indicam o início e o término do processo, e as informações a respeito das atividades individuais, como o participante e a aplicação associada (WfMC, 1998). Em outras palavras, uma definição de processo representa a especificação formal de um processo de negócio que é manipulada ou executada pelo sistema de *workflow*. O próprio WMS provê uma ferramenta que permite um usuário especificar processos de negócio através de uma linguagem baseada em *EXtensible Markup Language*[3] (XML) denominada **linguagem de definição de processo**. Dentre as linguagens de definição de processo encontradas na literatura, *Business Process Execution Language for Web Services*[4] (BPEL) e *XML Process Definition Language*[5] (XPDL) são as mais conhecidas atualmente. A primeira é uma solução que foi desenvolvida em conjunto pela IBM, BEA e *Microsoft*. Já a segunda, representa um padrão sugerido pela própria WfMC. Na Figura 2.2 é ilustrado um diagrama de atividades do processo de venda de produtos com sua respectiva definição de processo expressa em XPDL.




```
<WorkflowProcess Id="1" Name="sale process">
  <Activities>
    <Activity Id="1" Name="auction">
      <TransitionRestrictions>
        <TransitionRestriction>
          <Split Type="AND">
            <TransitionRefs>
              <TransitionRef Id="1"/>
              <TransitionRef Id="2"/>
            </TransitionRefs>
          </Split>
        </TransitionRestriction>
      </TransitionRestrictions>
    </Activity>
    <Activity Id="2" Name="send item"/>
    <Activity Id="3" Name="receive item">
      <TransitionRestrictions>
        <TransitionRestriction>
          <Join Type="AND"/>
        </TransitionRestriction>
      </TransitionRestrictions>
    </Activity>
    <Activity Id="4" Name="receive money"/>
    <Activity Id="5" Name="send money">
      <TransitionRestrictions>
        <TransitionRestriction>
          <Join Type="AND"/>
        </TransitionRestriction>
      </TransitionRestrictions>
    </Activity>
  </Activities>
  <Transitions>
    <Transition Id="1" From="1" To="2"/>
    <Transition Id="2" From="2" To="3"/>
    <Transition Id="3" From="1" To="4"/>
    <Transition Id="4" From="4" To="5"/>
  </Transitions>
</WorkflowProcess>
```

Figura 2.2. Definição de processo em XPDL para um processo de venda de produtos

[3] http://www.w3.org/XML/
[4] http://www.ibm.com/developerworks/library/ws-bpel/
[5] http://www.wfmc.org/standards/xpdl.htm

Cada **atividade** de um *workflow* representa um conjunto de tarefas que formam um passo lógico dentro de um processo de negócio (WfMC, 1998). Ela pode ser manual ou automática dependendo da sua possibilidade de automação. De acordo com a WfMC, uma atividade manual é aquela que só pode ser realizada por um recurso humano fora do escopo de um WMS (ex.: modelar um processo de negócio), ou seja, ela não é gerenciada pelo sistema de *workflow* durante a execução do processo. Já uma atividade automática, apesar do nome, pode ser executada tanto por uma pessoa quanto por um computador (ex.: retirar um item do estoque), mas sempre sob o controle do sistema de *workflow*.

Ao se executar um *workflow* no WMS, é criada uma **instância de processo** (em inglês *process instance*) para o *workflow* desejado a partir da sua própria especificação. Mais precisamente, uma instância de processo corresponde a execução individual de uma definição de processo, isto é, cada instância é construída, iniciada e encerrada pelo WMS de modo independente a qualquer outra instância preexistente (WfMC, 1998). Dessa forma, podem-se ter várias instâncias de processo para uma única definição de processo sem que uma interfira na execução da outra. O componente do WMS responsável por instanciar a definição de processo é conhecido por **máquina de *workflow***. Existem vários fabricantes de WMS na indústria de *software* e cada um implementa sua própria máquina de execução com base na linguagem de definição de processo que foi adotada.

2.1.2 Benefícios para a Gerência de Processos

Antigamente, um WMS era utilizado pelas empresas apenas para automatizar a distribuição de tarefas entre os participantes de um processo de negócio. Cada tarefa ou atividade era executada pelo seu próprio recurso (no caso uma pessoa responsável por realizar a tarefa) que sinalizava ao sistema quando a mesma era concluída. Esse tipo de processo caracterizava-se principalmente pelo controle manual das tarefas, havendo uma grande intervenção humana durante a execução do *workflow*. Com o amadurecimento dessa tecnologia, um processo de negócio passou a ser automatizado por completo (ALLEN, 2001), possibilitando que grande parte das atividades de um *workflow* seja executada de forma automática. Por exemplo, pode-se criar um conjunto de tarefas para que elas sejam realizadas em paralelo ou avaliar uma condição matemática para escolher qual a próxima tarefa a ser realizada.

Essa automatização total dos processos de negócio trouxe como benefícios para as empresas: (i) uma melhoria na eficiência do processo com a eliminação de passos desnecessários; (ii) um maior controle do processo através da padronização dos métodos de trabalho e da disponibilidade de auditoria; (iii) uma melhoria nos serviços oferecidos através de uma maior previsibilidade nos níveis de resposta ao cliente; e (iv) uma maior flexibilidade de controle de *software* sobre o processo, permitindo uma remodelagem do mesmo em caso de mudanças nos requisitos de negócio (WfMC, 1998).

Porém, com o advento da *Internet*, os Sistemas de *Workflow* encontraram novos obstáculos que tornaram o gerenciamento de processos de negócio mais complexo. Esses processos, que antes eram vistos como aplicações isoladas, passaram a se integrar criando os chamados *e-business* ou negócios eletrônicos, cujas necessidades de mudança são mais freqüentes e mais dinâmicas do que nos processos tradicionais. Nesse ponto, a falta de interoperabilidade entre as aplicações que participavam de um *workflow*, ocasionada pela heterogeneidade dos seus ambientes, e a questão da segurança na *Web*, onde filtros definidos por um sistema de segurança de rede (ou *firewall*) impediam a comunicação entre essas mesmas aplicações, foram algumas das dificuldades enfrentadas pelas empresas no âmbito da BPM. A utilização de serviços *Web* como recursos de um *workflow* foi a solução encontrada para contornar esses problemas.

2.2 SERVIÇOS WEB

A tecnologia de Serviços *Web* surgiu como uma nova forma de desenvolver aplicações distribuídas. Totalmente baseados em padrões abertos, os serviços *Web* possibilitaram a integração de aplicações heterogêneas justamente por proverem interoperabilidade entre elas via *Internet*. Outro fator importante que também favoreceu essa integração é o fato dos serviços *Web* não serem invasivos, pois eles não são barrados pelos sistemas de segurança existentes nas redes de computadores. Graças à sua infra-estrutura básica, essa tecnologia está revolucionando o mundo dos negócios eletrônicos com a construção de novos processos de negócio a partir da composição de serviços *Web* de baixo nível (ex.: reservar passagem, reservar hospedagem e alugar veículo) para oferecer serviços de elevado valor agregado (ex.: adquirir pacote turístico) (GLASS, 2000).

Nesta seção apresentaremos uma definição para serviços *Web* com base na sua arquitetura e em seguida, falaremos das contribuições providas por eles à automação de processos de negócio.

2.2.1 Definição

Um **serviço *Web*** é definido pelo *World Wide Web Consortium*[6] (W3C) como sendo um sistema de *software* identificado por uma *Uniform Resource Identifier* (URI), cujas interfaces e ligações são definidas, descritas, publicadas e descobertas via um contrato de uso, e que interage com outros sistemas usando mensagens transportadas por protocolos de *Internet* (W3C, 2004a).

Cada serviço *Web* é construído de modo a ser independente do contexto ao qual será empregado, possuindo assim um baixo acoplamento. Qualquer parte de código ou componente de um sistema pode ser transformado em um serviço *Web*, o qual pode, portanto, oferecer desde uma simples funcionalidade como a conversão de moeda ou a previsão do tempo, até funcionalidades mais complexas e que envolvem a interação com outros serviços como, por exemplo, o acompanhamento de pedidos ou o planejamento de viagens.

Em resumo, serviços *Web* são aplicações autocontidas, modulares e autodescritas que são acessíveis através da *Internet* independente da tecnologia utilizada para o acesso (SRIVASTAVA; KOEHLER, 2003).

2.2.2 Visão da Arquitetura

Para garantir a interoperabilidade, a tecnologia de Serviços *Web* apóia-se em três padrões derivados do XML, um formato amplamente aceito para troca de dados e que é a base da arquitetura dos serviços *Web*. São eles: (i) *Simple Object Access Protocol* (SOAP), um protocolo de mensagem que funciona sobre um protocolo de transporte da *Internet*, por exemplo, o *HyperText Transfer Protocol*[7] (HTTP); (ii) *Web Service Description Language* (WSDL), uma linguagem utilizada para descrever serviços *Web* como *endpoints* ou pontos

[6] http://www.w3c.org/
[7] http://www.w3.org/Protocols/

finais de comunicação capazes de trocar mensagens; e (iii) *Universal Description, Discovery and Integration* (UDDI), um diretório público que armazena as especificações WSDL dos serviços *Web* disponíveis a fim de permitir a descoberta dos mesmos (W3C, 2004a).

Juntos, esses três padrões permitem que os serviços *Web* sejam implementados e acessados de qualquer plataforma usando qualquer linguagem de programação. Tipicamente, um serviço *Web* tem a sua funcionalidade descrita em WSDL e publicada em um repositório UDDI por um agente fornecedor ou provedor de serviços. Qualquer agente consumidor que deseja executar essa funcionalidade solicita primeiramente a localização do serviço *Web* ao repositório UDDI, usando como critérios de busca o nome do agente fornecedor ou o tipo de serviço desejado. Esse repositório retorna, portanto, a descrição WSDL do serviço *Web* requisitado pelo agente consumidor. Finalmente, o agente consumidor realiza a chamada remota ao serviço *Web* com base na descrição obtida. Toda a comunicação entre o agente fornecedor, o agente consumidor e o repositório UDDI é realizada por meio de trocas de mensagens SOAP.

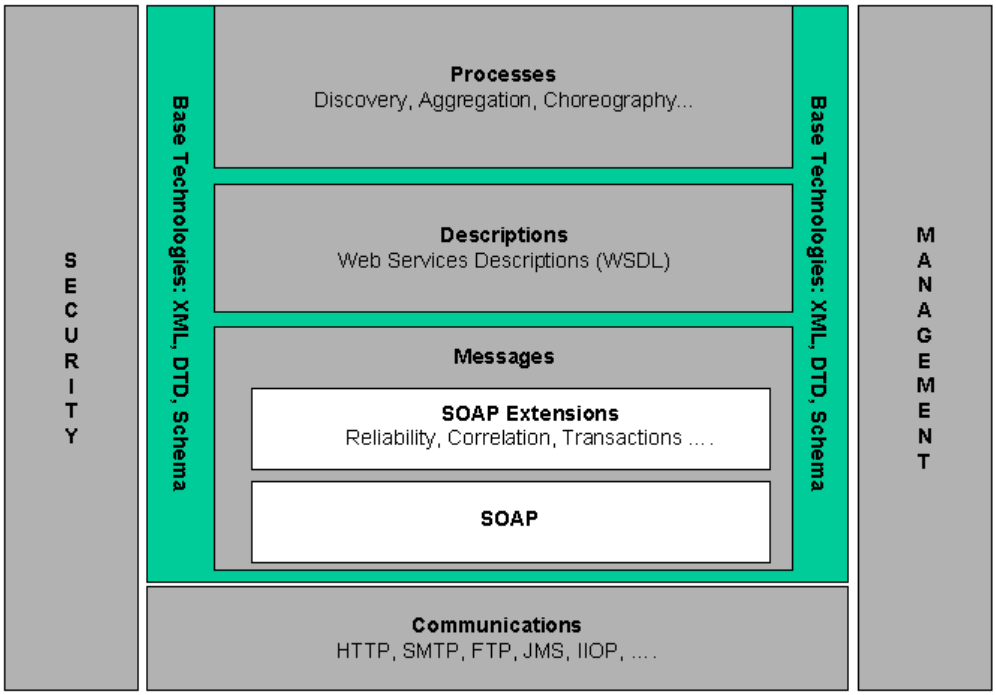


Figura 2.3. Arquitetura dos serviços *Web*. Fonte: W3C (2004a)

Por outro lado, os mecanismos providos pelos padrões SOAP, WSDL e UDDI são restritos apenas ao funcionamento básico dos serviços *Web*, ou seja, a descrição, a publicação, a localização e a invocação de serviços. Mecanismos complementares, como os de controle de transações, segurança e coordenação, são providos por outros padrões presentes na tecnologia de Serviços *Web*, nesse caso, os padrões *WS-Transaction*, *WS-Security* e *WS-Coordination*. A Figura 2.3 mostra uma visão em camadas da arquitetura dos serviços *Web*, onde é possível

identificar alguns desses padrões. Por exemplo, as camadas *Communications* e *Messages* definem a comunicação básica suportada pela tecnologia, a qual resultou no protocolo SOAP. A camada *Descriptions* trata da especificação de contratos dos serviços *Web*, que originou a linguagem WSDL. Já as camadas restantes definem aspectos estratégicos de processo de negócio que agregam valor aos serviços *Web*, dos quais resultaram vários padrões emergentes. Para este trabalho, daremos ênfase somente aos mecanismos *Discovery* e, *Aggregation* e *Choreography* definidos na camada *Processes* da arquitetura, os quais correspondem respectivamente a descoberta e a composição de serviços.

2.2.3 Contribuições para a Automação de Processos

Os serviços *Web* trouxeram importantes contribuições para a automação dos processos de negócios. A interoperabilidade é com certeza a mais visível delas, pois ajudou a resolver grande parte dos problemas confrontados pelos Sistemas de *Workflow* com a ascensão dos negócios eletrônicos. Em especial, os serviços *Web* endereçaram o fato da implementação de um *workflow* ser exposta além das fronteiras empresariais e a necessidade de se permitir um controle externo sobre o uso de aplicações de retaguarda (ou *back-end*) corporativas (PAPAZOGLOU; DUBRAY, 2004).

Como os Sistemas de *Workflow* são tipicamente orientados a atividades, eles não separam a lógica de negócio da sua implementação interna, ou seja, a própria definição de processo é usada para modelar o fluxo de controle das atividades de um processo de negócio e a maneira como elas devem ser executadas. Dessa forma, a integração desses sistemas no contexto de um negócio eletrônico torna-se inadequada uma vez que é exigido um baixo acoplamento entre eles, principalmente quando as empresas envolvidas nesse tipo de processo de negócio não compartilham suas aplicações e muito menos permitem que um WMS externo controle a execução das mesmas. A tecnologia de Serviços *Web* oferece uma solução viável para esse problema baseada na coordenação de mensagens, onde um processo de negócio é descrito por um modelo abstrato que define a comunicação exata de cada participante do processo sem necessariamente expor a sua implementação (PAPAZOGLOU; DUBRAY, 2004). O desenvolvimento de novas linguagens de definição de processo, tais como BPEL, permitiram que esse modelo fosse especificado de modo semelhante a um *workflow*, mas tendo como diferencial a capacidade de utilizar um simples serviço *Web* para efetivar uma

tarefa de negócio específica ou até mesmo compor novos serviços para automatizar todo um processo de negócio.

Diante dessa possibilidade de representar processos de negócio a partir da colaboração de serviços *Web*, foram introduzidos na literatura os termos *orquestração* e *coreografia* (W3C, 2004a). O primeiro descreve como os serviços *Web* interagem com os demais serviços sob a perspectiva e o controle de um único ponto final de comunicação (*endpoint*). Essa descrição inclui a lógica de negócio e a ordem de execução das interações dos serviços *Web*. Um exemplo simples de orquestração seria a seqüência de passos executados por um serviço de compra de livros na visão de um serviço cliente. Já o segundo termo é mais colaborativo e abrange a troca de mensagens, as regras de interação e os acordos entre vários pontos finais. Por exemplo, a seqüência de mensagens trocadas entre o serviço de compra de livros e o serviço cliente. Ou seja, a coreografia trilha a seqüência de mensagens entre as partes envolvidas no processo de negócio onde cada parte descreve seu papel exercido na interação. No contexto da automação de processos de negócio, a composição de serviços *Web* é vista como uma aplicação da orquestração (KHALAF; MUKHI; WEERAWARANA, 2003), tendo na linguagem BPEL um padrão promissor para a especificação de fluxos de mensagens dos serviços *Web*.

No entanto, para utilizar um serviço *Web* como um recurso de um *workflow* é preciso primeiramente encontrar dentre uma coleção de serviços disponíveis aqueles que satisfazem as necessidades específicas de um determinado processo de negócio. A descoberta de serviços envolve três etapas: a) buscar por serviços nos repositórios UDDI, b) localizar as descrições WSDL dos serviços, e c) selecionar os serviços mais adequados. A tecnologia de Serviços *Web* suporta esse mecanismo em cima apenas das descrições WSDL, isto é, os critérios de busca permitidos são definidos com base nas características funcionais desses serviços as quais foram descritas sintaticamente pela WSDL (ex.: tipos de dados, número de parâmetros). Depois de descobertos, os serviços *Web* são combinados de modo a representar o modelo abstrato de interações do processo de negócio. Eventualmente, pode acontecer de um serviço *Web* apropriado para implementar uma determinada atividade do processo não existir, sendo necessário compor um novo serviço a partir dos serviços *Web* existentes. Por essa razão, a composição de serviços não é simplesmente combinar dois ou mais serviços para executar uma dada atividade. Esse mecanismo envolve outros aspectos de negócio (qualidade de serviço, por exemplo) que buscam agregar valor aos serviços compostos.

Atualmente, tanto a descoberta quanto a composição de serviços são realizadas na especificação de um processo de negócio de forma manual e requerem um conhecimento

aprofundado sobre as operações dos serviços *Web* por parte do usuário. O potencial da tecnologia de Serviços *Web* como um meio de desenvolver soluções de negócios eletrônicos será alavancado se esses mecanismos tornarem-se dinâmicos. Neste trabalho entendemos por **descoberta dinâmica** a busca, localização e seleção automáticas de serviços que consideram não só as características funcionais, mas também as não-funcionais (ex.: custo, categoria) desses serviços *Web*. Já a **composição dinâmica** consiste no processo de descoberta e encadeamento dos serviços *Web* de modo a atuar como um único serviço, permitindo uma eventual substituição dos serviços indisponíveis no momento da execução do *workflow*.

Porém, para que a descoberta dinâmica dos serviços *Web* seja praticável, os serviços disponíveis na *Internet* devem estar organizados dentro de categorias hierárquicas – também chamadas de *taxonomias* – que são construídas a partir das funcionalidades providas por esses serviços e da forma como elas são invocadas (BURBECK, 2000). Do mesmo modo, a composição dinâmica de serviços só será viável se existir uma semântica bem definida que descreva as funcionalidades de um serviço *Web* em termos de negócio.

2.3 WEB SEMÂNTICA

A ***Web* Semântica** é considerada por muitos uma evolução da *Web* tradicional que busca melhorar a qualidade de acesso às informações a partir do processamento de conteúdo semântico das mesmas (W3C, 2004b). Ao adicionar um significado bem definido a qualquer recurso disponibilizado na *Internet* (uma página *Web*, por exemplo), a *Web* Semântica torna mais fácil para um agente de *software* inferir um novo tipo de informação que envolva um determinado recurso, pois ela elimina ambigüidades no tratamento desses significados.

A palavra **inferir** representa a capacidade de extrair um conhecimento que está armazenado de maneira implícita em uma base de conhecimento através de um raciocínio lógico. Para tal, a *Web* Semântica emprega o uso de ontologias como forma de conceitualizar e representar um determinado domínio, e o uso de mecanismos de inferência para a realização de consultas em cima dessas ontologias. Nesta seção será apresentada uma noção básica sobre ontologias e máquinas de inferência, e logo depois, será descrito como as ontologias agregam semântica as descrições dos serviços *Web* de modo a favorecer a descoberta e a composição automática desses mesmos serviços.

2.3.1 Ontologia

Uma **ontologia** é definida classicamente como uma especificação formal e explícita de uma conceitualização compartilhada (GRUBER, 1993). Em outras palavras, uma ontologia tenta descrever de maneira exata um modelo abstrato do mundo real a partir de conceitos, propriedades, relações, funções e regras, todos definidos explicitamente por pessoas com o propósito de tornar esse modelo interpretável por agentes de *software*. A vantagem de se utilizar ontologias é que elas fornecem um vocabulário comum para a representação de um domínio específico, cujos termos são descritos formalmente sem haver interpretações ambíguas. Ao mesmo tempo, elas permitem que o conhecimento seja compartilhado por outras ontologias como forma de reusar e aperfeiçoar o vocabulário definido.

Para poder representar um dado conhecimento através de uma ontologia, é necessário a utilização de linguagens formais baseadas em lógica capazes de expressar satisfatoriamente o conhecimento desejado e de inferir sobre o mesmo. A *OWL Web Ontology Language*[8] ou simplesmente OWL é a linguagem de marcação semântica proposta pelo W3C como padrão a ser utilizado pela *Web* Semântica para descrever ontologias. Fundamentada em *Resource Description Framework*[9] (RDF), a OWL permite que usuários escrevam o vocabulário de um dado domínio em termos de classes, instâncias, propriedades e relacionamentos. Para exemplificar o uso dessa linguagem, a Figura 2.4 mostra um trecho de uma ontologia do domínio de Turismo criada para representar as capitais brasileiras.

```
<!-- Classe -->
<owl:Class rdf:ID="Cidade">
  <rdfs:subClassOf>
    <!-- Restrição -->
    <owl:Restriction>
      <owl:onProperty rdf:resource="#municipio"/>
      <owl:cardinality
         rdf:resource="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<!-- Propriedade -->
<owl:DatatypeProperty rdf:ID="municipio">
  <rdfs:domain rdf:resource="#Cidade"/>
  <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<!-- Instância -->
<Cidade rdf:ID="NAT">
  <municipio rdf:datatype="&xsd;string">Natal</municipio>
</Cidade>
```

Figura 2.4. Trecho da ontologia OWL do domínio de Turismo

[8] http://www.w3.org/2004/OWL/
[9] http://www.w3.org/RDF/

Segundo a Figura 2.4, uma cidade é definida na ontologia em questão pela classe *rdf:ID="Cidade"* utilizando-se o construtor *owl:Class*. A propriedade *rdf:ID="municipio"*, que representa o nome do município, é definida através do construtor *owl:DatatypeProperty*. Já a relação 1:1 (lê-se: um para um) entre a classe *rdf:ID="Cidade"* e a propriedade *rdf:ID="municipio"* é realizada pelo construtor *owl:Restriction*. Por fim, tem-se a cidade de Natal definida como sendo a instância *rdf:ID="NAT"* da classe *rdf:ID="Cidade"*.

Dependendo do grau de expressividade desejado para descrever um determinado domínio, a OWL pode limitar ou não a utilização de seus construtores. Por conseqüência, sua capacidade de inferência pode se tornar mais ou menos eficiente. A rigor, quanto maior o grau de expressividade da linguagem, mais rica é a sua semântica e menos eficiente é a sua capacidade de inferência (W3C, 2004b). Com base nessa afirmação, a OWL é dividida em três sublinguagens: OWL *Lite*, OWL DL e OWL *Full*. A OWL DL é a linguagem que melhor apresenta um equilíbrio entre a semântica e o poder de raciocínio, sendo, portanto, a linguagem escolhida neste trabalho para representar ontologias. O acrônimo DL refere-se a um subconjunto da lógica conhecido por Lógicas de Descrição ou *Description Logics*[10]. As DLs são formalismos para representação de conhecimento baseados em lógica de primeira ordem que servem de base para a especificação dos construtores da linguagem OWL e para o desenvolvimento de métodos de dedução utilizados pelas máquinas de inferência durante as consultas aplicadas sobre as ontologias.

Apesar de possuir um conjunto rico de construtores, a OWL não provê formas de expressar associações envolvendo apenas as propriedades descritas em uma ontologia. Por exemplo, seja *temPai* uma propriedade que representa a relação pai e filho entre duas pessoas. Sejam **A**, **B**, **C** e **D** instâncias da classe *Pessoa*. Para representar as relações "A é pai de B" e "B é pai de C", as propriedades das instâncias filhas são definidas respectivamente como sendo *temPai(B, A)* e *temPai(C, B)* (lê-se: "B tem pai A" e "C tem pai B"). Nesse caso, como se trata de uma hierarquia, é possível inferir que "A é avô de C" devido a transitividade da propriedade *temPai* sem a necessidade de criar a propriedade *temAvo*. Todavia, se existir uma propriedade *temIrmao* onde seja definido que "D é irmão de B", não se pode inferir que "D é tio de C", pois não existe uma regra que represente a propriedade *temTio*.

Para solucionar esse problema, a linguagem *Semantic Web Rule Language*[11] (SWRL) foi desenvolvida para representar predicados de regras através da OWL. As regras aumentam a capacidade de inferência das linguagens de ontologias, permitindo expressar associações

[10] http://dl.kr.org/
[11] http://www.w3.org/Submission/SWRL/

entre propriedades e inferir informações implícitas a partir de *axiomas* (sentenças verdadeiras) embutidos nas próprias ontologias. No caso do exemplo anterior, ao invés de criar uma propriedade *temTio* para todas as instâncias da classe *Pessoa*, podemos definir a regra *temPai(?x, ?y)* e *temIrmao(?y, ?z)* implica em *temTio(?x, ?z)* para representar a propriedade desejada.

2.3.2 Máquinas de Inferência

No contexto da *Web* Semântica **máquinas de inferência** são sistemas de representação de conhecimento capazes de realizar consultas sobre as ontologias de modo a deduzir novas informações, geralmente implícitas, a respeito de um domínio preexistente. Também chamados de **raciocinadores** (do inglês *reasoner*), as máquinas de inferência provêem mecanismos que permitem verificar a consistência de ontologias e validar a classificação de conceitos (BAADER; NUTT, 2002). Esses mecanismos, denominados de **mecanismos de inferência**, são fundamentados na linguagem de descrição semântica. Atualmente, a maioria dos raciocinadores encontrados na literatura utiliza a linguagem OWL DL para processar inferências sobre as ontologias, devido, em parte, a sua capacidade de decisão (ver Seção 2.3.1). Os principais mecanismos de inferência providos por raciocinadores baseados em OWL DL, tais como *Pellet*[12], *RacerPro*[13] e *FaCT++*[14], são a *satisfatibilidade*, a *consistência*, a *classificação* e a *realização* (SIRIN *et al.*, 2005).

O mecanismo de **satisfatibilidade** determina se uma classe pode ou não ter instâncias em uma determinada ontologia. Uma classe é dita insatisfatória se qualquer instância dessa classe tornar uma ontologia inconsistente. Já o mecanismo de **consistência** verifica se uma ontologia não contém um conceito contraditório, como por exemplo, **X** é definido como uma instância das classes **A** e **B**, porém essas classes são disjuntas entre si, isto é, uma é o oposto da outra. O mecanismo de **classificação** avalia os relacionamentos hierárquicos entre as classes, permitindo identificar as subclasses de uma classe em particular (ex.: *Homem* e *Mulher* são subclasses de *Pessoa*), enquanto que o mecanismo de **realização** localiza a classe mais específica de uma dada instância.

---

[12] http://pellet.owldl.com/
[13] http://www.racer-systems.com/
[14] http://owl.man.ac.uk/factplusplus/

As inferências feitas por um raciocinador são especificadas na forma de consultas sobre as ontologias. No caso, existem dois padrões propostos pelo W3C que podem ser utilizados na especificação dessas consultas semânticas – *RDF Data Query Language*[15] (RDQL) e *Simple Protocol And RDF Query Language*[16] (SPARQL). A RDQL é uma linguagem declarativa que provê um meio de acesso às ontologias semelhante a linguagem de consulta adotada em um sistema de gerência de banco de dados. Contudo, esse acesso restringe-se exclusivamente a extração de dados a partir das ontologias e não permite a aplicação de mecanismos de inferência. A linguagem SPARQL é mais sofisticada em relação a primeira e é considerada pelo W3C como um padrão emergente a ser adotado na *Web* Semântica. Como forma de exemplificar o uso dessas linguagens, a Figura 2.5 mostra uma consulta que poderia ser realizada sobre a ontologia da Figura 2.4 para inferir os conceitos (referenciados pela variável *x*) que representam instâncias da classe *p1:Cidade*. Neste exemplo, o mecanismo de realização é utilizado para processar tal consulta.

```
-- RDQL
SELECT ?x
WHERE ( ?x rdf:type p1:Cidade )
USING p1 FOR <http://localhost/Turismo.owl#>

-- SPARQL
PREFIX p1: <http://localhost/Turismo.owl#>
SELECT ?x
WHERE ( ?x rdf:type p1:Cidade )
```

Figura 2.5. Consulta semântica em RDQL e SPARQL

2.3.3 Serviços *Web* Semânticos

A necessidade de acrescentar conteúdo semântico às descrições de serviços *Web* fez com que as tecnologias dos Serviços *Web* e da *Web* Semântica fossem combinadas resultando os chamados **Serviços *Web* Semânticos** (SWS). Um SWS é definido por uma ontologia de serviço que permite a interpretabilidade das funcionalidades providas pelo serviço *Web*, bem como a integração da descrição do serviço *Web* com o conhecimento de domínio (ou seja, a terminologia de negócio). Na literatura existem duas propostas que estão sendo utilizadas com freqüência em pesquisas na área da *Web* Semântica as quais tentam desenvolver soluções para o problema da descoberta e composição dinâmicas de serviços. São elas a *Web Ontology*

[15] http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/
[16] http://www.w3.org/TR/rdf-sparql-query/

Language for Web Services[17] (OWL-S) e a *Web Service Modeling Ontology*[18] (WSMO), descritas em maiores detalhes a seguir.

A OWL-S é uma ontologia desenvolvida pelo *DARPA Agent Markup Language Program* que descreve semanticamente as propriedades e as funcionalidades de um serviço *Web* através das linguagens OWL e SWRL, visando facilitar a automatização de tarefas na *Web* Semântica, incluindo a descoberta, execução, interoperação, composição e a monitoração da execução de serviços *Web* (DAML, 2004). Essa ontologia descreve um serviço *Web* semântico em três partes: um *Profile* (o que ele faz) representado pela classe *ServiceProfile*, um *Process Model* (como ele trabalha) representado pela classe *ServiceModel* e um *Grounding* (como ele é acessado) representado pela classe *ServiceGrounding*, como mostra a Figura 2.6.

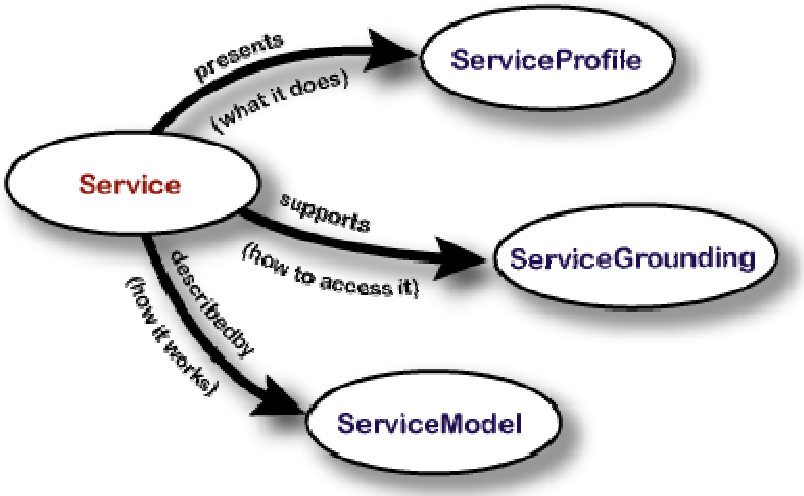


Figura 2.6. Ontologia OWL-S. Fonte: DAML (2004)

O *Profile* contém a informação necessária para um agente de busca determinar se o serviço satisfaz os requisitos de parte de uma dada especificação de processo. Essa informação compreende o fornecedor do serviço, as entradas requeridas e as saídas produzidas, as precondições necessárias e os efeitos esperados, e um conjunto de características não-funcionais (categoria, qualidade, tempo de resposta, etc.). O *Process Model* fornece uma perspectiva detalhada de como um serviço opera, fazendo uma comparação com um processo. É possível descrever o funcionamento de um serviço de três maneiras: como um *AtomicProcess*, contendo um único passo de execução; um *CompositeProcess*, composto de vários passos; ou um *SimpleProcess*, uma visão comum abstrata dos dois primeiros. Por último, o *Grounding* especifica os detalhes de como acessar o

[17] http://www.daml.org/services/owl-s/1.1/
[18] http://www.wsmo.org/

serviço (ex.: protocolos, formatos de mensagens) e une a ontologia do serviço *Web* semântico ao documento WSDL que descreve o serviço *Web*.

Um exemplo de como a OWL-S descreve semanticamente um serviço *Web* é visto na Figura 2.7 que exibe parte da ontologia de um serviço de reserva de vôos denominado *ReservarVooService*. Esse serviço é apresentado por um *ReservarVooProfile*, descrito por um *ReservarVooProcess* e suportado por um *ReservarVooGrounding*. O primeiro elemento da ontologia contém informações sobre a utilidade do serviço. Já o segundo descreve o funcionamento do serviço como um processo de um único passo, cujas entradas exigidas são *Origem*, *Destino*, *Data*, *Hora*, *Vôo* e a saída produzida é *Localizador*. O último elemento trata a forma como esse serviço é acessado via WSDL, especificando o tipo de porta (*CompanhiaAereaPortType*) e a operação (*reservarVoo*) que deverá ser invocada.

```
<!-- Service -->
<service:Service rdf:ID="ReservarVooService">
  <service:presents rdf:resource="#ReservarVooProfile"/>
  <service:describedBy rdf:resouce="#ReservarVooProcess"/>
  <service:supports rdf:resource="#ReservarVooGrounding"/>
</service:Service>
<!-- Profile -->
<profile:Profile rdf:ID="ReservarVooProfile">
  <service:presentedBy rdf:resource="#ReservarVooService"/>
  <profile:serviceName>Reserva de Vôos</profile:serviceName>
  <profile:textDescription>
    Este serviço realiza a reserva de um vôo de uma origem
    para um destino em uma data e hora específica.
  </profile:textDescription>
  <profile:hasInput rdf:resource="#Origem"/>
  <profile:hasInput rdf:resource="#Destino"/>
  <profile:hasInput rdf:resource="#Data"/>
  <profile:hasInput rdf:resource="#Hora"/>
  <profile:hasInput rdf:resource="#Voo"/>
  <profile:hasOutput rdf:resource="#Localizador"/>
</profile:Profile>
<!-- Process Model -->
<process:AtomicProcess rdf:ID="ReservarVooProcess">
  <service:describes rdf:resource="#ReservarVooService"/>
  <process:hasInput rdf:resource="#Origem"/>
  <process:hasInput rdf:resource="#Destino"/>
  <process:hasInput rdf:resource="#Data"/>
  <process:hasInput rdf:resource="#Hora"/>
  <process:hasInput rdf:resource="#Voo"/>
  <process:hasOutput rdf:resource="#Localizador"/>
</process:AtomicProcess>
<!-- Grounding -->
<grounding:WsdlGrounding rdf:ID="ReservarVooGrounding">
  <service:supportedBy rdf:resource="#ReservarVooService"/>
  <grounding:hasAtomicProcessGrounding rdf:resource="#ReservarVooWsdl"/>
</grounding:WsdlGrounding>
<grounding:WsdlAtomicProcessGrounding rdf:ID="ReservarVooWsdl">
  <grounding:owlsProcess rdf:resource="#ReservarVooProcess"/>
  <grounding:wsdlDocument>CompanhiaAerea?wsdl</grounding:wsdlDocument>
  <grounding:wsdlOperation>
    <grounding:WsdlOperationRef>
      <grounding:portType>CompanhiaAereaPortType</grounding:portType>
      <grounding:operation>reservarVoo</grounding:operation>
    </grounding:WsdlOperationRef>
  </grounding:wsdlOperation>
</grounding:WsdlAtomicProcessGrounding>
```

Figura 2.7. Trecho da ontologia OWL-S do serviço *Web* ReservarVooService

A WSMO é uma ontologia para serviços *Web* semânticos, assim como a OWL-S, que foi proposta pelo *ESSI WSMO Working Group* com o objetivo de facilitar também a automação da descoberta, combinação e invocação de serviços (ROMAN *et al*. 2005). Contudo, ela apresenta uma abordagem diferente da primeira para descrever os aspectos relevantes de um serviço *Web*.

Tendo sido definida a partir de uma extensão do arcabouço *Web Service Modeling Framework*[19] (WSMF), a WSMO provê quatro elementos de alto nível para descrever formalmente os serviços *Web* semânticos: (i) as **ontologias**, que fornecem os conceitos e os relacionamentos usados pelos demais elementos; (ii) as **metas**, que definem os objetivos dos usuários, isto é, os problemas que deveriam ser resolvidos pelos serviços *Web*; (ii) as **descrições** dos serviços, que definem vários aspectos de um serviço *Web*; e (iv) os **mediadores**, que cuidam dos problemas de interoperabilidade. Todos esses elementos são descritos utilizando uma família de linguagens formais de descrição chamada de *Web Service Modeling Language*[20] (WSML). Na Figura 2.8 é ilustrada parte de uma ontologia WSMO criada para descrever o mesmo serviço de reserva de vôos que foi exemplificado antes utilizando a OWL-S.

```
webService ReservarVooService
  nonFunctionalProperties
    dc:title hasValue "Reserva de Vôos"
    dc:description hasValue
      "Este serviço realiza a reserva de um vôo de uma origem
       para um destino em uma data e hora específica."
  endNonFunctionalProperties

  usesMediator ooMediator

  capability ReservarVooCapability
    origem ofType Origem
    destino ofType Destino
    data ofType Data
    hora ofType Hora
    voo ofType Voo
    localizador ofType Localizador
```

Figura 2.8. Trecho da ontologia WSMO do serviço *Web* ReservarVooService

Percebe-se pela Figura 2.8 que o serviço *Web* semântico *ReservarVooService* possui a funcionalidade *ReservarVooCapability* cujos parâmetros são representados pelos conceitos *origem*, *destino*, *data*, *hora*, *voo* e *numeroReserva*. Esse mesmo serviço usa o mediador *ooMediator* para fazer a conversão de dados envolvendo esses parâmetros durante a invocação do serviço.

[19] http://www1-c703.uibk.ac.at/~c70385/wese/index.html
[20] http://www.wsmo.org/wsml/

Tanto a OWL-S quanto a WSMO podem ser utilizadas para descrever as funcionalidades dos serviços *Web*. Todavia, pelo simples fato da OWL-S utilizar a linguagem OWL, um padrão estabelecido pelo W3C, optamos neste trabalho por adotar essa abordagem para a criação de ontologias dos serviços *Web* semânticos.

3 A COMPOSIÇÃO DE SERVIÇOS WEB

O enfoque dado ultimamente às tecnologias de Serviços *Web* e da *Web* Semântica tem proporcionado o desenvolvimento de vários projetos de pesquisa abordando de diferentes maneiras o tema da composição de serviços *Web*. Alguns autores (SYCARA *et al*., 2003), por exemplo, propõem que a composição de serviços seja reduzida a três problemas básicos. O primeiro deles é fazer um plano de composição que descreva como os serviços *Web* interagem entre si e como as funcionalidades providas por eles podem ser integradas de modo a fornecer um serviço composto. O segundo problema é a descoberta de serviços *Web* que realizam as tarefas requeridas pelo plano de composição. O terceiro é o gerenciamento da interação dos serviços *Web* que foram descobertos. Já outros autores (MILANOVIC; MALEK, 2006) consideram a composição de serviços um problema essencialmente de busca, o qual pode ser tratado de duas formas. Uma é iniciar com uma composição pré-definida, descrita de maneira genérica, e depois realizar *n-1* buscas para trocar cada elemento genérico de uma composição por um serviço real. A outra é descrever um conjunto de objetivos e tentar alcançá-los construindo o plano de composição a partir do zero.

Visando então estabelecer um entendimento geral sobre a composição de serviços *Web*, este capítulo traz na Seção 3.1 um resumo das atuais soluções tecnológicas disponibilizadas pela indústria de *software* e pela área acadêmica, contendo as vantagens e desvantagens de cada solução. Na Seção 3.2, será exibida uma classificação das técnicas de composição, com destaque para aquelas que buscam alcançar um processo de composição de serviços totalmente automático. Além disso, serão apresentados alguns trabalhos que estão sendo realizados a partir dessas técnicas de composição. As principais características de cada trabalho serão salientadas de modo a identificar as semelhanças e diferenças entre eles. Como esses trabalhos também estão relacionados com a solução proposta neste presente trabalho, serão tecidos comentários na Seção 3.3 comparando-os com tal solução.

3.1 SOLUÇÕES ATUAIS DE COMPOSIÇÃO

Atualmente as organizações estão buscando através da tecnologia de Serviços *Web* alavancar o gerenciamento de seus processos de negócio à medida que as mudanças exigidas em seus

empreendimentos tornam-se mais rápidas e freqüentes a cada dia (ex.: novos parceiros e novos serviços). O aumento da complexidade nesses processos faz com que os serviços *Web* existentes passem a ser combinados para atender as novas necessidades, e conseqüentemente, agreguem novos valores. Muitas vezes, essa combinação visa compor um novo serviço, pois não existe um serviço adequado para automatizar um dado processo de negócio. Essa necessidade de combinar serviços simples para gerar serviços compostos despertou, portanto, o interesse tanto da indústria de *software* quanto da área acadêmica em encontrar soluções que suportem a composição de serviços. A primeira desenvolveu uma série de padrões baseados em XML (W3C, 2004a) para formalizar a especificação, composição e execução de serviços *Web*. A última está propondo várias técnicas de composição ao incorporar semântica (DAML, 2004) às descrições dos serviços *Web*.

Os padrões WSDL (W3C, 2003) e BPEL (IBM, 2002) providos pela indústria de *software* foram amplamente aceitos na automação de processos de negócio, uma vez que permitiram a descrição e publicação de serviços *Web* e a especificação formal de processos e protocolos de interação de negócio. Mais do que isso, esses padrões proveram uma infra-estrutura básica onde processos de negócio podem ser automatizados através da composição de serviços *Web*. Primeiramente o projetista do processo de negócio especifica em BPEL um protocolo de interação para descrever o fluxo de mensagens entre os serviços que irão compor tal processo. Depois, ele seleciona os serviços adequados ao processo de negócio com base nas suas respectivas descrições WSDL, as quais foram publicadas em algum repositório de serviços (UDDI, 2003). Finalmente, os serviços *Web* selecionados são encadeados pelo projetista conforme o protocolo de interação especificado. Como resultado dessa composição, tem-se um processo executável que pode ser distribuído e instanciado em máquinas de *workflow* (WfMC, 1998) de um WMS.

Para explicar melhor como funciona a composição de serviços *Web* através dos padrões WSDL e BPEL, considere um processo de reserva de pacotes turísticos onde uma determinada agência de viagens deseja integrar sua solução com alguns provedores de serviços especializados em reservar vôos, reservar quartos de hotel e efetuar pagamentos com cartão de crédito. A troca de mensagens envolvendo os serviços participantes desse processo é representada pela Figura 3.1.

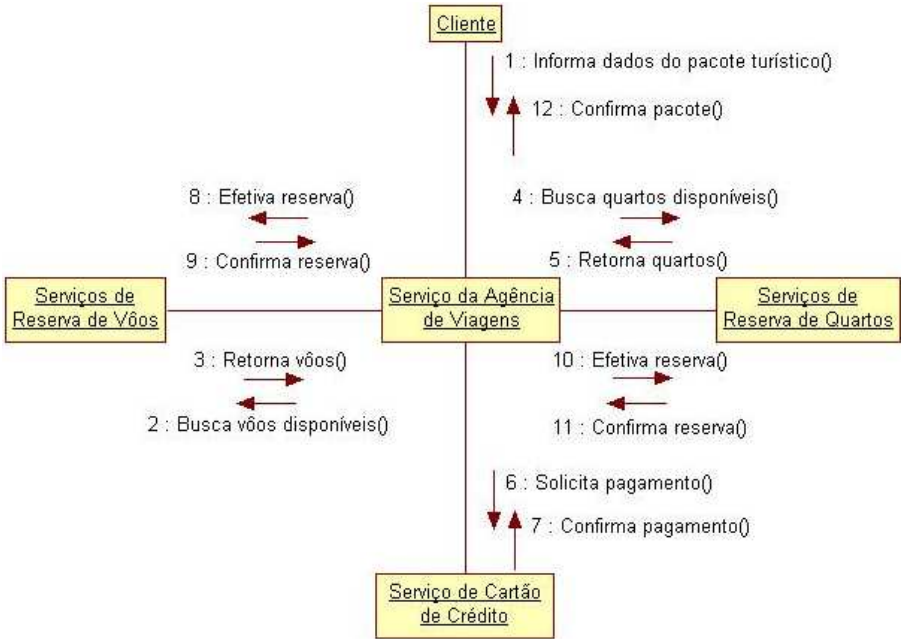


Figura 3.1. Fluxo de mensagens do processo de reserva de pacotes turísticos

Segundo a Figura 3.1, um cliente que deseja adquirir um pacote turístico deve informar ao **serviço da agência de viagens** suas preferências com relação ao hotel (tipo de quarto, café-da-manhã incluso, tem restaurante, etc.), ao destino da viagem, ao período da viagem bem como os dados referentes ao cartão de crédito (mensagem 1). Então, o serviço da agência de viagens consulta por vôos e quartos disponíveis nos devidos **serviços de reserva de vôos** e **reserva de quartos** (mensagens 2 e 4) de acordo com as informadas passadas pelo cliente. Havendo disponibilidade de ambos (mensagens 3 e 5), o serviço da agência de viagens solicita o pagamento do pacote turístico ao **serviço de cartão de crédito** (mensagem 6). Após a confirmação do pagamento pelo serviço de cartão de crédito (mensagem 7), o serviço da agência de viagens solicita a efetivação das reservas aos respectivos serviços de reserva (mensagens 8 e 10). Por fim, o serviço da agência de viagens retorna a confirmação do pacote turístico ao cliente (mensagem 12) a partir das confirmações dos serviços de reserva (mensagens 9 e 11).

Baseado nesse fluxo de mensagens, o projetista especifica um protoloco de interação para o processo de reserva de pacotes turísticos utilizando, para tal, alguma ferramenta BPEL. Essa especificação consiste de um conjunto de atividades realizadas pelo processo de negócio juntamente com as mensagens que são manipuladas em cada atividade, semelhante a uma definição de processo de um WMS (ver Figura 2.2). Após concluir a especificação desse protocolo, o projetista busca em algum repositório UDDI serviços *Web* que podem ser aproveitados na realização das atividades do processo de negócio, utilizando critérios como a categoria ou nome do provedor de serviços. A partir das descrições WSDL dos serviços

disponíveis nesse repositório, o projetista seleciona aqueles serviços que são adequados ao processo de negócio e em seguida, os adiciona à especificação do protocolo de interação.

Como exemplo, suponha que a atividade de reservar um vôo foi especificada nesse protocolo contendo as mensagens *reservaVoo* e *confirmaReserva*, conforme ilustra a Figura 3.2. O projetista busca via repositório UDDI serviços pertencentes à categoria **reserva de vôos** e depois seleciona um serviço *Web* dentre os que foram localizados no repositório. Considere agora que o respectivo serviço possui uma operação *reservar* (provida pela porta *reservarPortType*), cuja entrada é identificada pela mensagem *reservarRequest* e a saída pela mensagem *reservarResponse*. Essa mesma operação é então associada àquela atividade de reservar vôos do protoloco de interação fazendo-se um mapeamento entre as mensagens *reservaVoo* e *resevarRequest*, e outro entre as mensagens *confirmaReserva* e *reservarResponse*. Tal procedimento é realizado de modo análogo nas demais atividades especificadas no protocolo de interação até que se tenha um processo executável, o qual poderá ser instanciado por uma máquina de *workflow*.

```
<process name ="Reserva_de_Pacotes_Turisticos">
  <!-- Parceiros -->
  <partners>
    <partner name="Servico_de_Reserva_de_Quartos"/>
  </partners>

  <!-- Mapeamento de Mensagens -->
  <containers>
    <container name="reservaVoo" messageType="resevarRequest"/>
    <container name="confirmaReserva" messageType="reservarResponse"/>
  </containers>

  <!-- Protocolo de Interação -->
  <sequence>
    <flow>
      <invoke partner="Servico_de_Reserva_de_Quartos"
              portType="reservarPortType" operation="reservar"
       inputContainer="reservaVoo"/>

      <receive partner="Servico_de_Reserva_de_Quartos"
               portType="reservarPortType" operation="reservar"
              container="confirmaReserva"/>
    </flow>
  </sequence>
</process>
```

Figura 3.2. Protocolo de interação em BPEL do processo de reserva de pacotes turísticos

A principal vantagem da composição de serviços *Web* suportada pelos padrões WSDL e BPEL é a possibilidade do próprio protocolo de interação especificado em BPEL ser publicado via WSDL como um novo serviço, permitindo que outros agentes consumidores utilizem o dado processo de negócio. Contudo, ela apresenta deficiências que são descritas a seguir.

Em primeiro lugar, os fluxos de mensagens e as ligações dos serviços *Web* precisam ser conhecidos a priori em tempo de projeto, visto que o processo de negócio é executado sempre a partir da sua especificação atual. Em segundo lugar, podem ocorrer mudanças nas descrições WSDL dos serviços escolhidos para compor o protocolo de interação desse processo (implementação de uma nova funcionalidade, por exemplo) assim como falhas de execução provocadas pela indisponibilidade desses mesmos serviços, exigindo-se, dessa forma, que uma alteração na especificação seja realizada. Em terceiro lugar, a composição de serviços é guiada pelo usuário, onde o protocolo de interação precisa ser definido antes de selecionar os serviços *Web* que serão utilizados na automação do processo de negócio. Depois de selecionados, esses serviços são ordenados e combinados através de um mapeamento de dados entre a saída de um serviço com a entrada do serviço adjacente. Portanto, tal composição acaba sendo custosa, inflexível e demorada. Por último, os padrões da indústria de *software* WSDL e BPEL interpretam uma funcionalidade provida por um serviço *Web* com base apenas nas entradas, saídas e tratamentos de exceções de cada serviço e não levam em consideração outras características como o tempo de resposta, a qualidade do serviço ou as condições exigidas para a invocação do mesmo (SRIVASTAVA; KOEHLER, 2003).

Apesar de todos os esforços realizados nessa área, a composição de serviços *Web* ainda é uma tarefa altamente complexa, e está além da capacidade humana de lidar com um processo de negócio de forma totalmente manual. A complexidade, em geral, deve-se: (i) ao crescimento do número de serviços *Web* disponíveis na *Internet* nos últimos anos, criando um imenso repositório de busca; (ii) aos próprios serviços *Web* que podem ser criados e atualizados de forma dinâmica (*on the fly*), tornando necessário que o processo de composição seja capaz de detectar essa atualização em tempo de execução com base na especificação atual do processo de negócio; e (iii) ao fato dos serviços *Web* serem desenvolvidos por várias organizações, que usam modelos conceituais diferentes para descrever tais serviços, não se estabelecendo uma linguagem exclusiva para definir e avaliar os serviços *Web* de uma forma idêntica (RAO; SU, 2004).

Em parte, os problemas supracitados estão diretamente relacionados à falta de significado na sintaxe da WSDL. Sem semântica, as descrições WSDL dos serviços *Web* não podem ser interpretadas por agentes de *software*, requerendo sempre a presença de pessoas para realizar tal interpretação. Conseqüentemente, a composição não pode ser feita de forma automática, pois não é possível para uma máquina de inferência compreender o real significado de uma operação provida pelo serviço simplesmente analisando os tipos de dados descritos na interface WSDL do mesmo. A solução para esse problema está nas ontologias de

serviço OWL-S (DAML, 2004) e WSMO (WSMO, 2005) sugeridas pela *Web* Semântica, que provêem descrições semânticas para os serviços *Web*.

De fato as ontologias de serviço agregam significados às descrições WSDL tornando os serviços *Web* interpretáveis tanto para as máquinas de inferência quanto para as pessoas. Além do mais, elas oferecem aos agentes de *software* a possibilidade de compor serviços *Web* de forma automática. Tal composição pode ser dividida em quatro etapas: (i) a descoberta e combinação (ou *matching*) dos serviços; (ii) a geração do plano de composição; (iii) a execução do plano gerado; e (iv) o monitoramento da execução. Na primeira etapa os serviços *Web* são descobertos e combinados com base nas suas propriedades funcionais (entradas, saídas, precondições e pós-condições) e não-funcionais (qualidade do serviço, custo, etc.), as quais são requeridas por um dado processo de negócio. Os mecanismos de inferências (ver Seção 2.3.2) aplicados sobre as ontologias de serviço permitem a um raciocinador concluir que serviços são adequados para implementar um processo de negócio. A segunda etapa consiste em gerar o plano de composição que atenda às metas ou objetivos de um processo de negócio, isto é, sintetizar uma especificação de como coordenar os serviços *Web* descobertos inicialmente para atender tal processo de negócio. Nesse ponto, várias técnicas de composição estão sendo propostas pela área acadêmica de modo a automatizar a criação de planos de composição (ver Seção 3.2). Já na terceira etapa, o plano de composição gerado é instanciado por uma máquina de execução de onde serão invocados os serviços *Web* escolhidos para realizar o processo de negócio. Na última etapa, o plano de composição é monitorado em tempo de execução para que seja possível substituir um dado serviço *Web* que por ventura ficou indisponível ou mudou de descrição WSDL, sem ter que modificar a especificação atual desse plano.

Além de ser mais flexível em relação à solução provida pela indústria de *software* (WSDL e BPEL), a composição de serviços provida pela *Web* Semântica traz como vantagens: (i) a criação de um número ilimitado de novos serviços a partir de um conjunto limitado de serviços existentes; (ii) a eliminação de um repositório local de serviços disponíveis para a composição de serviços; (iii) o aumento no nível de abstração no sentido de que a especificação do plano de composição não é mais restrita ao conjunto original de serviços que foram previstos e definidos em tempo de projeto; (iv) a possibilidade de customizar um processo de negócio com base nas necessidades individuais de um usuário e via composição dinâmica; e (v) a atualização do processo de negócio em tempo de execução, onde novos serviços podem ser compostos enquanto os serviços antigos são utilizados pelo mesmo processo (ALAMRI; EDI; SADDIK, 2006).

3.2 CLASSIFICAÇÃO DAS TÉCNICAS DE COMPOSIÇÃO

Diante da necessidade de compor serviços *Web* para automatizar processos de negócio várias técnicas de composição têm sido propostas pela área acadêmica (CHAN; BISHOP; BARESI, 2007; COSTA; PIRES; MATTOSO, 2004a; MAJITHIA; WALKER; GRAY, 2004; MILANOVIC; MALEK, 2006; SYCARA *et al*., 2003). Em geral, essas técnicas são classificadas em manuais ou automáticas e mais comumente em estáticas ou dinâmicas (ALAMRI; EDI; SADDIK, 2006). A primeira categoria considera o nível de envolvimento do usuário na composição de serviços, seja para especificar um plano de composição ou adaptar um plano existente, por exemplo. A última, diz respeito ao momento em que os serviços *Web* são compostos, isto é, se a composição é realizada em tempo de projeto ou se em tempo de execução.

Na composição **manual**, como o próprio nome sugere, a composição é executada pelo usuário que projeta o processo de negócio, isto é, o usuário além de especificar o plano de composição é responsável por descobrir, selecionar e combinar os serviços *Web* adequados ao mesmo (como é o caso das soluções propostas pela indústria de *software* que utilizam a linguagem BPEL descritas na Seção 3.1). Na composição **automática**, essas mesmas tarefas são realizadas por agentes de *software* havendo, portanto, uma menor participação do usuário na composição de serviços. Existe ainda uma terceira possibilidade não mencionada por alguns autores (DUSTDAR; SCHREINER, 2005) que é a composição **semi-automática**, a qual mescla a descoberta e combinação automáticas de serviços *Web* com a seleção manual desses serviços (SIRIN; HENDER; PARSIA, 2003).

Na composição **estática**, o plano de composição é especificado em tempo de projeto e, portanto, os serviços *Web* que compõem tal plano precisam ser conhecidos a priori. Esse tipo de composição é muito comum em ambientes de negócio considerados estáveis, os quais não apresentam mudanças ou raramente elas ocorrem, e abrange as soluções baseadas nos padrões da indústria de software (BPEL, por exemplo). Já na composição dinâmica, o plano de composição precisa ser adaptado às mudanças em tempo de execução, com o mínimo de intervenção externa do usuário e de forma transparente, sendo, portanto, mais adequada aos ambientes altamente flexíveis, onde novos serviços são disponibilizados a todo instante e as mudanças são freqüentes.

Como a maioria das técnicas de composição propostas atualmente são automáticas e dinâmicas (ALAMRI; EDI; SADDIK, 2006), as subseções a seguir descrevem em maiores detalhes algumas das técnicas mais conhecidas.

3.2.1 Composição Baseada em Modelos

Na composição baseada em modelos, um plano de composição é especificado em duas partes. Uma contém a lógica da composição (um modelo), enquanto que a outra é a implementação do serviço composto. Tal separação tem como intuito elevar o nível de abstração da especificação do plano de composição permitindo que modelos sejam utilizados por outros agentes de *software* para compor serviços.

O trabalho realizado por Pathak *et al.* (2007) é um exemplo típico de composição baseada em modelos. Esses autores propõem um arcabouço (em inglês *framework*) chamado *MoSCoE*, que permite a composição (semi)automática de serviços *Web*, a qual é realizada em três passos básicos: a abstração, a composição e o refinamento. No primeiro passo, uma especificação de alto nível do serviço composto (isto é, a lógica da composição) é feita utilizando-se as máquinas de estados da *Unified Modeling Language*[21] (UML). No segundo passo, a especificação do serviço composto é traduzida pelo arcabouço em um autômato de estado finito, o qual representa a implementação do serviço especificado. No terceiro e último passo, esse autômato é refinado pelo usuário do arcabouço que acrescenta precondições e pós-condições utilizando anotações semânticas de um repositório de ontologias de domínio, construído a partir das descrições WSDL e OWL-S dos serviços *Web* disponíveis. O *MoSCoE* possui um módulo de gerenciamento de composição que identifica as estratégias de composição adequadas para um dado serviço composto durante o refinamento do mesmo, e um modulo de execução que lida com a invocação do serviço composto e suporta a importação e exportação de autômatos de estado finito na linguagem BPEL.

Entretanto, esse arcabouço encontra-se em uma fase inicial de desenvolvimento e precisa de mais tempo para que seja avaliado na prática seu funcionamento. Além disso, o fato de não existir fluxos de controle em máquinas de estado, tais como condição e repetição, limita a especificação do serviço composto no *MoSCoE*, não possibilitando a automação de processos de negócio mais complexos.

[21] http://www.uml.org

3.2.2 Composição Baseada em *Workflows*

Essa técnica de composição trata um serviço composto de forma similar a um *workflow* de um WMS. Nesse caso, a especificação de um serviço composto inclui um conjunto de serviços atômicos juntamente com o controle e o fluxo de dados entre esses serviços, do mesmo modo que a definição de processo em um sistema de *workflow* precisa descrever o fluxo de atividades. Pode-se dizer que a composição baseada em *workflows* é um tipo de composição baseada em modelos onde a representação da lógica do serviço composto (ou seja, o modelo) é um *workflow*.

Laukkane e Helin (2003) sugerem um agente compositor de *workflow* (*Workflow Composer Agent*) baseado nessa técnica de composição, que utiliza as ontologias dos serviços *Web* com o intuito de descobrir e compor serviços para um dado processo de negócio. Porém, o usuário desse ambiente é obrigado a especificar uma primeira versão do *workflow* onde alguns serviços *Web* são escolhidos e combinados manualmente. Essa especificação é feita utilizando-se a linguagem BPEL, e representa ao mesmo tempo a lógica e a implementação do processo de negócio. Além disso, a maneira como as descrições dos serviços *Web* são manipuladas pelo próprio agente não é bem definida nesse trabalho. Para facilitar a tarefa de especificar um *workflow*, o agente compositor permite a busca por um serviço *Web* que implemente uma funcionalidade desejada pelo processo de negócio. Esse processo de busca envolve quatro etapas: (i) identificação da funcionalidade requerida, (ii) combinação semântica de serviços *Web*; (iii) criação ou atualização do *workflow*; e (iv) execução e monitoração do *workflow*. As duas últimas etapas são necessárias para que seja possível a substituição de um serviço *Web* que se encontra indisponível no momento em que um *workflow* é executado.

Majithia, Walker e Gray (2005) apresentam um arcabouço *Automated Composition Framework* para automatizar a composição de serviços *Web* a partir do uso de tecnologias da *Web* Semântica e de Sistemas de *Workflow*. Nesse trabalho uma funcionalidade requerida por um processo de negócio pode ser representada em diferentes níveis de abstração, o que possibilita isolar a lógica de sua implementação. Esses níveis são armazenados em seus respectivos repositórios para que possam ser reutilizados por outros usuários. O primeiro nível de representação desse arcabouço consiste em definir uma funcionalidade através de um *workflow* abstrato, cujas tarefas possuem entradas e saídas referenciadas por nomes lógicos obtidos a partir das ontologias de serviço. O último nível consiste em especificar essa

funcionalidade por meio de um *workflow* concreto onde são escolhidos os serviços que combinam com as tarefas do *workflow* abstrato. Existe ainda um nível intermediário onde um modelo de processo pré-definido pode ser usado para representar uma funcionalidade caso não exista um *workflow* abstrato no repositório relativo a tal processo. No entanto, esse trabalho foca mais na visão arquitetural da solução proposta, não descrevendo claramente a ligação entre um *workflow* e as ontologias de serviços *Web*, a maneira como um modelo de processo é elaborado, e nem como os serviços são encontrados e compostos pelos componentes do arcabouço. Apenas destaca que é possível recuperar ou construir um *workflow* com base nas entradas, saídas, precondições e pós-condições da funcionalidade desejada.

Costa, Pires e Mattoso (2004b) apresentam uma ferramenta para a implementação automática de *workflows*, denominada *WebComposer*, que permite a descoberta, a seleção, a composição e a invocação de serviços *Web*. Essa implementação automática separa a lógica do *workflow* da tecnologia de implementação do mesmo, isto é, o *workflow* é especificado de forma abstrata utilizando conceitos descritos por ontologias de negócio, e durante a sua execução, os serviços são descobertos e compostos através de inferências realizadas sobre as descrições semânticas dos mesmos. No *WebComposer*, o *workflow* pode ser especificado usando-se OWL-S ou uma extensão de BPEL. Através de uma interface gráfica, o usuário especifica as entradas, precondições, saídas e efeitos desejados por ele juntamente com os respectivos valores. O usuário também especifica o fluxo de controle do *workflow*. Em seguida essa especificação é transformada visualmente em uma árvore a partir da solicitação do próprio usuário na ferramenta em questão. Depois de construída a árvore, o usuário solicita a geração dos planos de composição para o *workflow* especificado. A partir desse momento, inicia-se um processo onde se descobre os serviços disponíveis para a implementação do *workflow*, seleciona-se os serviços para cada atividade do *workflow* e por último, compõe-se as operações desses serviços selecionados para cada atividade do *workflow*. Dependendo da composição encontrada, o *WebComposer* pode criar caminhos de execução alternativos a composição principal. Esses caminhos são intercalados junto com o fluxo de controle principal da implementação do *workflow* garantindo dessa maneira uma tolerância a falhas na execução do mesmo. Porém, essa ferramenta exige por parte do usuário um conhecimento específico da linguagem escolhida para a especificação do *workflow*. Outra deficiência encontrada no *WebComposer*, é que o usuário deve conhecer de antemão os conceitos definidos pelas ontologias de negócio para que seja especificado um determinado *workflow*.

3.2.3 Composição Baseada em Ontologias

A composição baseada em ontologias é aplicada aos serviços *Web* semânticos, e baseia-se na similaridade semântica das funcionalidades dos serviços, as quais são descritas por ontologias (OWL-S, por exemplo). Para alcançar a composição, essa técnica requer uma ontologia de domínio específico (ex.: uma taxonomia) que define os conceitos e os relacionamentos entre esses conceitos, ambos utilizados na descrição semântica dos serviços *Web*.

O protótipo *Composer* desenvolvido por Sirin, Hender e Parsia (2003) é um típico ambiente de composição de serviços baseado em ontologias, no qual a geração do plano de composição é dirigida pelo usuário. Como pré-requisito, esse protótipo exige que os serviços *Web* sejam descritos através da ontologia OWL-S. A partir do *Profile* dos serviços existentes, esse ambiente constrói uma lista de serviços que é posteriormente exibida através de sua interface visual para que o usuário escolha o serviço *Web* que deseja utilizar na composição. Cada vez que o usuário seleciona um serviço, o protótipo filtra os serviços que podem ser combinados logo em seguida. Para tal, ele realiza uma combinação (*matching*) entre os parâmetros de saída do serviço selecionado inicialmente com as entradas do próximo serviço. Essa combinação leva em consideração a semântica desses parâmetros os quais são consultados por uma máquina de inferência. Ao final da composição, o usuário gera um *workflow* de serviços *Web* no ambiente *Composer* que é salvo por esse protótipo como um serviço OWL-S, o qual pode ser utilizado em outras composições. Esse ambiente provê também uma máquina de execução de serviços OWL-S que permite a invocação dos serviços *Web* compostos pelo usuário. Essa invocação é feita com base nos *Groundings* dos serviços que compõe o *workflow* do usuário. A principal característica do *Composer* é a capacidade de guiar o usuário na composição de serviços descobrindo e combinando os próximos serviços possíveis a partir de inferências sobre as ontologias dos serviços *Web*. Infelizmente, essa composição considera apenas os parâmetros de entrada e saída. As precondições e efeitos não podem ser utilizados para a composição de serviços.

O projeto IST FP6 INFRAWEBS[22] é uma plataforma de desenvolvimento aberta e extensível que foi sugerida por Nern *et al*. (2004) com o propósito de construir um *Semantic Service Engineering Framework* capaz de apoiar a criação, manutenção e execução de serviços *Web* semânticos. Essa plataforma é dividida em três camadas; *Camada de Gerenciamento do Conhecimento*, *Camada de Desenvolvimento de Serviços*, e *Camada de*

[22] http://www.infrawebs.org/

Distribuição de Serviços. A primeira camada (vista de baixo para cima) trata como os serviços estão relacionados com conhecimento de domínio, isto é, as ontologias. A segunda camada abrange a criação e a composição de serviços *Web* semânticos enquanto que a última camada, cuida da execução e monitoração desses serviços. Um produto gerado a partir desse projeto foi o arcabouço INFRAWEBS (AGRE *et al*., 2007), um conjunto de componentes de *software* que permitem o desenvolvimento de serviços *Web* semânticos baseados em WSMO, encapsulando toda a complexidade por trás da manipulação de ontologias, metas e descrições de serviços. Para tanto, esse arcabouço identifica diferentes tipos de usuários bem como as várias etapas do processo de engenharia envolvendo os serviços *Web* semânticos, além de implementar um conjunto específico de ferramentas para cada tipo de usuário.

Ao todo são quatro os tipos de usuário existentes no INFRAWEBS: (i) um **provedor de serviços**, que deseja converter seus serviços providos em serviços *Web* semânticos e publicar os serviços convertidos; (ii) um **corretor de serviços** (do inglês *broker*), que gostaria de criar e publicar um serviço cujas funcionalidades são obtidas via composição de vários serviços *Web* semânticos existentes; (iii) um **provedor de aplicações**, que gostaria de ter sua própria aplicação baseada na tecnologia de Serviços *Web* Semânticos; e (iv) um **cliente de aplicações**, que usar o arcabouço de forma transparente para encontrar e executar um serviço *Web* ou uma composição de serviços que satisfazem sua meta desejada. Já as etapas do processo de engenharia correspondem aos estágios do ciclo de vida dos serviços *Web* semânticos que são a criação, a composição, a descoberta, a seleção, a execução e a monitoração de serviços. A primeira etapa combina as atividades relacionadas à criação e publicação dos elementos WSMO. Na segunda etapa, os serviços criados podem ser combinados para prover novos serviços de valor agregado, podendo estes inclusive ser usados por outros serviços. Na terceira etapa, os serviços *Web* são descobertos a partir da combinação semântica entre as metas do usuário (meta WSMO) e as funcionalidades dos serviços (descrição de serviço WSMO). Na quarta etapa, um usuário escolhe de maneira pró-ativa quais os serviços *Web* serão executados. A seleção pode também ser feita automaticamente por um agente de *software*. Na quinta etapa, os serviços são invocados pelo usuário que fornece alguma entrada e espera pelos resultados. Por fim, a sexta etapa coleta informações sobre a execução dos serviços para que sejam usadas na fase de seleção como critérios de escolha.

Basicamente o que ocorre durante esse ciclo de vida do INFRAWEBS é a criação de serviços compostos através da conversão de um serviço *Web* existente ou da composição de vários serviços. As especificações desses serviços criados utilizam a linguagem WSML e são

armazenadas no repositório do INFRAWEBS conhecido por *Organizational Memory* (OM). Para facilitar a busca de serviços *Web*, o arcabouço suporta o uso de anotações RDF para registrar internamente as descrições WSDL e BPEL dos serviços e de consultas SPARQL para procurar essas descrições via um *Semantic Information Router* (SIR). Já para facilitar a especificação de serviços *Web* semânticos, o arcabouço disponibiliza um ambiente gráfico chamado de INFRAWEBS *Designer*, o qual é integrado com o OM, para projetar as descrições e metas baseadas nas ontologias existentes. Esse mesmo ambiente é utilizado pelos usuários provedor de aplicações e cliente de aplicações para invocar um ou mais serviços *Web* a partir da especificação de suas próprias metas.

O arcabouço IRS-III é resultado do projeto de pesquisa *Internet Reasoning Service*[23] (IRS) que foi sugerido por Domingue *et al*. (2004) com a intenção de prover uma infra-estrutura de desenvolvimento para serviços *Web* semânticos descritos em WSMO. Esse trabalho possui algumas características importantes que também estão presentes no trabalho descrito anteriormente (AGRE *et al*., 2007). A primeira delas é a capacidade de transformar automaticamente um código de programação em um serviço *Web* criando-se um envoltório (ou *wrapper*) para o código. Outra característica é a invocação de serviços dirigida por capacidade, isto é, os usuários podem invocar um ou mais serviços *Web* via metas, as quais são especificadas pelos próprios usuários em um ambiente gráfico que permite a consulta de ontologias. Tal característica oferece um nível de dinamismo para a composição, pois os serviços são descobertos em tempo de execução para atender a meta desejada. Além disso, os serviços *Web* semânticos especificados nesse arcabouço são compatíveis com os serviços *Web*, isto é, um agente consumidor pode invocar um serviço *Web* semântico do IRS-III da mesma forma com que faria com um serviço *Web* provido por algum provedor de serviços.

A composição pode ser descrita de forma estática (ligação entre os serviços) ou dinâmica (atribuída em tempo de execução pela descoberta com base nas metas do usuário). Para tal, o IRS-III suporta três modelos para representar a composição de serviços: baseado em estado, o estruturado e o fluxo de dados. No modelo baseado em estados, os **estados** representam os fluxos de controle (seqüências, laços, condicionais, etc.) da composição e as **transições** representam as atividades. A execução de uma atividade dispara a mudança de estado de modo semelhante a um evento em um diagrama de estados. Uma atividade pode ser tanto um serviço *Web* semântico (simples ou composto) como uma meta do usuário. Nesse modelo, as classes da ontologia de orquestração WSMO são utilizadas para modelar os

[23] http://kmi.open.ac.uk/projects/irs/

estados e as atividades. Essas mesmas classes são instanciadas em tempo de execução, ou seja, durante a invocação de serviços *Web* compostos. O modelo estruturado é formado por dois tipos de componentes chamados de **componente de controle** e **componente de serviço**. O primeiro provê a capacidade de definir os fluxos de controle da composição enquanto que o segundo é um envoltório que mantém as informações necessárias sobre os mediadores de ligação para uma meta de composição, isto é, ele transforma a descrição de uma meta do usuário em um serviço *Web* que é invocado pela máquina de orquestração. Já o modelo de fluxo de dados é uma especialização do modelo estruturado onde todos os serviços são definidos por um único componente de controle, portanto, ele não permite representar os fluxos de controle da composição. As ligações e os mediadores WSMO especificam os fluxos de dados. Nesse modelo, um serviço só inicia a execução quando o dado de entrada exigido por ele é provido pelo usuário.

Apesar de permitirem a descoberta, seleção, composição e invocação automáticas de serviços *Web*, tanto o INFRAWEBS quanto o IRS-III são ambientes mais voltados para o desenvolvimento de aplicações com serviços *Web* semânticos. Além do mais, eles exigem por parte do usuário um conhecimento aprimorado sobre as tecnologias da *Web* Semântica (ex.: OWL, SPARQL, WSMO, etc.).

3.2.4 Composição Baseada em Planificação IA

A maioria das técnicas de composição dinâmica propostas atualmente está relacionada com a planificação IA (RAO; SU, 2004), que tenta gerar um plano de composição a partir dos objetivos especificados pelo usuário. A hipótese sustentada por tais técnicas é que um serviço *Web* pode ser especificado através de precondições e efeitos.

No contexto da planificação IA, um serviço *Web* é mais do que um simples componente de *software* que possui parâmetros de entrada e de saída. Um serviço é também uma ação que pode alterar o estado global de um processo de negócio. Sendo assim, o estado exigido antes de um serviço ser executado é chamado de **precondição** e o estado gerado após a execução do serviço é o **efeito**. Por exemplo, considere um serviço de reserva de vôos onde os parâmetros de entrada são os dados do vôo e a saída é o número da reserva. O estado exigido para executar esse serviço é “vôo disponível” e o estado gerado com a sua execução é “reserva confirmada”. Portanto, se um objetivo do processo de reserva fosse especificado

contendo como precondição o estado “vôo disponível” e como efeito o estado “reserva confirmada”, um plano de composição poderia ser gerado automaticamente sem o conhecimento do protocolo de interação pré-definido, ou seja, nesse caso não seria necessário saber que a atividade reservar vôo é a responsável pela mudança de estado em questão.

O projeto ASTRO desenvolvido por Trainotti *et al*. (2005) utiliza essa técnica de composição para projetar e executar serviços *Web* compostos. Esse trabalho provê um ambiente, chamado ASTRO *Suite*[24], que consiste de um conjunto de ferramentas para a especificação, composição, validação e invocação de processos de negócio baseados na linguagem BPEL. A primeira ferramenta, o *WS-gen*, é responsável por gerar a composição automática de serviços a partir de uma especificação BPEL ou OWL-S do protocolo de interação que um serviço composto deve implementar, juntamente com a coreografia das conexões entre os parceiros envolvidos no processo de negócio e a meta dessa composição. A segunda ferramenta, o *WS-mon*, gera um código executável para o serviço composto com base na especificação BPEL e cria monitores para acompanhar a execução desse serviço. Esses monitores são acessados via ferramenta *WS-console* que apresenta os estados dos mesmos toda vez que um serviço composto é executado. Por último, o *WS-animator*, é uma ferramenta que permite o usuário a possibilidade de executar o serviço composto gerado para um dado processo de negócio. Apesar de suportar a composição automática de serviços, esse ambiente utiliza principalmente descrições WSDL dos serviços *Web* para a especificação dos protocolos de interação. Futuramente essas ferramentas serão estendidas para trabalhar com descrições semânticas, tais como a OWL-S e WSMO.

3.3 CONSIDERAÇÕES FINAIS

A necessidade de compor serviços *Web* básicos em serviços mais complexos está alavancando, principalmente, porque novas soluções estão sendo alcançadas. Em ambientes de composição semi-automáticos, a geração de planos de composição é dirigida por usuário, ou seja, o ambiente apenas apóia o usuário na descrição de um plano, filtrando os serviços que podem ser combinados com um determinado serviço a partir das saídas produzidas por esse serviço e das restrições definidas pelo usuário. Porém, a decisão de escolher um serviço para a composição é sempre do usuário. Nos ambientes de composição automáticos, o plano é

[24] http://www.astroproject.org

construído diretamente pelo ambiente em tempo de projeto a partir de um conjunto de serviços *Web* disponíveis, podendo ser executado no próprio ambiente ou em uma máquina de execução. Entretanto, o desafio de criar um ambiente que favoreça a especificação de um processo de negócio abstrato e que seja implementado automaticamente por um serviço composto de forma dinâmica é considerado um problema em aberto (RAO e SU, 2004).

Os trabalhos mencionados anteriormente tentam prover um ambiente de composição automática ou semi-automática baseado nas descrições sintáticas e/ou semânticas dos serviços *Web*. Entretanto, algumas considerações foram levantadas a respeito de como cada um deles buscou solucionar o problema da composição de serviços *Web* no contexto da automação de processos de negócio. A Tabela 3.1 apresenta de forma resumida essas considerações com base nos requisitos exigidos (ver Seção 1.1) para um ambiente de especificação e execução de processos de negócio baseados em serviços *Web*.

Tabela 3.1. Comparação entre ambientes de composição

	Nível de Abstração da Especificação	**Descoberta e seleção de serviços**	**Síntese e orquestração**
Workflow Composer Agent	Lógica junta com a implementação. Utiliza BPEL	Processo de busca em quatro etapas	Composição estática e manual na especificação
Automated Composition Framework	Lógica separada da implementação. Utiliza linguagens diferentes para cada nível.	Envolve nomes lógicos obtidos das ontologias de serviço	Composição automática baseada na realização do *workflow* abstrato em um *workflow* concreto
WebComposer	Lógica separada da implementação. Utiliza OWL-S e extensão de BPEL	Mecanismos de inferências sobre as descrições semânticas	Implementação automática de *workflows* com caminhos de execução alternativos
INFRAWEBS	Lógica separada da implementação. Utiliza WSML e WSMO	Combinação semântica entre metas e serviços. A escolha é manual ou automática	Conversão de um serviço *Web* existente ou composição de vários serviços
IRS-III	Lógica separada da implementação. Utiliza WSMO	Descobertos em tempo de execução	Composição estática ou dinâmica
Composer	Lógica separada da implementação. Utiliza OWL-S	Baseada nas entradas e saídas do *Profile*	Composição dirigida pelo usuário

O *Workflow Composer Agent* (LAUKKANE; HELIN, 2003) utiliza o método de composição de *workflows* para implementar um ambiente semi-automático de composição de serviços *Web*, assim como o *Automated Composition Framework* (MAJITHIA; WALKER; GRAY, 2005) e o *WebComposer* (COSTA; PIRES; MATTOSO, 2004b). A principal

diferença entre eles está no nível de abstração que ocorre tal especificação. No primeiro trabalho, o *workflow* especificado representa ao mesmo tempo a lógica e a implementação do processo de negócio, enquanto que os dois últimos separam esses modelos em pelo menos dois níveis de especificação. Já o INFRAWEBS (AGRE *et al*., 2007) e o IRS-III (DOMINGUE *et al*., 2004) implementam um ambiente automático de composição baseada em ontologias, assim como o *Composer* (SIRIN; HENDER; PARSIA, 2003). Porém, eles diferem quanto a ontologia de serviço empregada na descrição dos serviços *Web*. Enquanto dois primeiros utilizam a ontologia WSMO, o último trabalha com a ontologia OWL-S. Ademais, tanto o INFRAWEBS quanto o IRS-III são ambientes mais voltados para o desenvolvimento de aplicações com serviços *Web* semânticos do que propriamente um ambiente para especificação e execução de processos de negócio. Aliás, dentre os trabalhos que foram mencionados, apenas dois apresentaram realmente um ambiente implementado capaz de realizar a especificação de um plano de composição a partir das ontologias de serviços *Web* semânticos e de permitir a execução desse plano – o *Composer* e o *WebComposer*.

O *Composer* permite a especificação abstrata de um *workflow* utilizando as ontologias OWL-S dos serviços *Web* disponíveis. Tal especificação é guiada pelo usuário que escolhe cada serviço *Web* semântico que deve fazer parte do plano de composição, cuja especificação é representada no ambiente também como um serviço OWL-S. O *Composer* auxilia o usuário filtrando os serviços que podem ser combinados em tempo de projeto. Esse ambiente permite ainda a execução do *workflow* especificado fazendo a invocação dos serviços *Web* que são descobertos pelo *Profile*. Contudo, esse ambiente exige que especificação seja realizada por completa e não suporta fluxos de controle típicos de um WMS, tais como condição e repetição. Além do mais, tal especificação representa ao mesmo tempo lógica de negócio e a implementação de um dado processo.

Já o *WebComposer* permite a especificação incompleta de um *workflow* utilizando os conceitos das ontologias OWL-S dos serviços *Web*, onde uma simples atividade especificada pode gerar a composição de vários serviços. A própria ferramenta encarrega-se de encontrar uma composição que atenda a especificação descrita parcialmente. Dessa forma, ela separa a lógica de negócio da implementação do processo. Porém, a ferramenta apresenta algumas limitações. Em primeiro lugar, sua Interface com Usuário não é tão intuitiva e fácil de usar. Em segundo lugar, o *WebComposer* não permite navegar nas ontologias de serviço para então especificar os requisitos de um dado *workflow* (entradas, saídas, precondições e pós-

condições). Por fim, a especificação de *workflows* é realizada diretamente na linguagem de definição de processo, exigindo-se que o usuário conheça a sintaxe dessa linguagem.

Portanto, um ambiente que permita a especificação e execução de processos de negócio a partir das descrições semânticas de serviços *Web*, deve prover uma funcionalidade que auxilie o usuário na descrição do *workflow* a partir de conceitos definidos por essas ontologias e ao mesmo tempo suportar a especificação parcial ou incompleta desse *workflow*. Ademais, esse ambiente deve ser capaz de sintetizar uma composição a partir desse tipo de especificação. Essa síntese compreende a descoberta, a seleção e a combinação de um ou mais serviços que implementem um determinado objetivo do *workflow*. A solução proposta por este trabalho busca atender esses requisitos e é descrita em mais detalhes no próximo capítulo.

4 O AMBIENTE WEBFLOWAH

Conforme foi explicado no capítulo anterior, o ferramental disponível atualmente para especificação e execução de processos de negócio não oferece as funcionalidades necessárias para um usuário especificar e executar processos de negócio baseados em serviços *Web* através dos mecanismos de descoberta e composição dinâmicas. Pois, nas soluções providas atualmente pela indústria de *software*, as quais focam nas descrições sintáticas (WSDL) dos serviços *Web*, é preciso realizar a descoberta manual dos serviços que serão utilizados para a automação de um processo de negócio, bem como a composição estática de novos serviços caso não seja encontrado um serviço adequado a realizar uma determinada atividade desse processo. Já com relação às soluções procedentes da *Web* Semântica, que buscam a automação da descoberta e composição via descrições semânticas (OWL-S) dos serviços *Web*, muitas delas ainda são emergentes e provêem apenas uma infra-estrutura básica de desenvolvimento para serviços *Web* semânticos, Tais soluções não permitem de forma conjunta: (i) a especificação abstrata de um processo de negócio na forma de objetivos, os quais são descritos por uma ontologia de domínio comum; (ii) a descoberta e seleção automática dos serviços que atendem aos objetivos especificados; (iii) a combinação das funcionalidades dos serviços selecionados; e (iv) a coordenação em tempo de execução dos fluxos de mensagens desses serviços combinados.

Este trabalho, portanto, propõe um ambiente denominado **WebFlowAH** que permite a especificação e execução *ad-hoc* de processos de negócio baseados em serviços *Web*. Mais precisamente, esse ambiente procura, de forma dinâmica, descobrir, compor e invocar serviços que atendem aos objetivos de um dado processo, os quais foram obtidos a partir das descrições de serviços *Web* semânticos. Esses objetivos são definidos no WebFlowAH de forma abstrata, pois eles não possuem uma referência direta a um serviço *Web* específico, e sim a um conceito que é utilizado para descrever um ou mais serviços. Tal nível de abstração possibilita também que um processo de negócio seja especificado parcialmente ou de forma incompleta, uma vez que não é preciso detalhar todas as partes componentes do mesmo (interfaces, mensagens, protocolos de comunicação, etc.), bastando ter a representação do objetivo fim para que o processo seja sintetizado pelo ambiente, isto é, seja gerado automaticamente um plano de composição contendo os serviços *Web* apropriados ao processo.

Neste capítulo serão descritos inicialmente na Seção 4.1 os conceitos básicos relacionados ao desenvolvimento do ambiente WebFlowAH, a fim de contextualizar a abordagem proposta por este trabalho. Logo em seguida, será apresentada na Seção 4.2 uma visão arquitetural desse ambiente, destacando-se seus principais componentes. Por último, serão descritas na Seção 4.3 as funcionalidades providas pelo ambiente e que garantem a especificação e execução de processos de negócio a partir das ontologias de serviço.

4.1 CONCEITOS BÁSICOS

Os principais conceitos introduzidos neste trabalho foram: Ontologia de Domínio Comum, Objetivo de Negócio, *Workflow* Semântico e Plano de Execução. A Figura 4.1 mostra um diagrama de classes elaborado na versão 1.4 da UML contendo as relações entre esses conceitos.

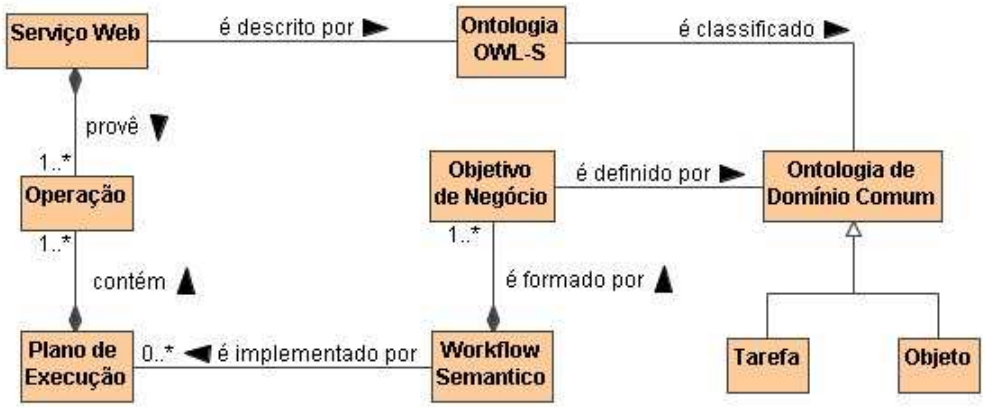


Figura 4.1. Conceitos básicos do ambiente WebFlowAH

No ambiente WebFlowAH (segundo a Figura 4.1) um serviço *Web* é descrito semanticamente por uma ontologia OWL-S. Essa ontologia é classificada neste ambiente como sendo uma Tarefa ou um Objeto da Ontologia de Domínio Comum, da qual são definidos os Objetivos de Negócio que formam um *Workflow* Semântico de um dado processo de negócio. Já um *Workflow* Semântico pode ser implementado ou não por Planos de Execução que contém operações providas por serviços *Web* descobertos e compostos dinamicamente pelo ambiente. A seguir, é apresentada uma definição mais detalhada de cada conceito.

4.1.1 Ontologia de Domínio Comum

A ontologia OWL-S descreve formalmente as propriedades e funcionalidades providas pelos serviços *Web* a partir de um conjunto de conceitos e seus respectivos relacionamentos. Entretanto, um mesmo conceito pode estar presente em diferentes ontologias, visto que cada organização cria, à sua maneira, um vocabulário próprio para representar um domínio de negócio, não existindo, desse modo, um critério comum de categorização de conceitos. Por exemplo, uma empresa **A** descreve o conceito *ComprarLivro* como sendo uma atividade do processo de compra de publicações. Uma empresa **B** pode descrever *ComprarCasa* como uma atividade para o processo de compra de imóveis. No entanto, *ComprarLivro* e *ComprarCasa* representam de forma mais específica o conceito genérico *Comprar*. Na verdade, isso é um problema em aberto da *Web* Semântica onde existem trabalhos na literatura (ALBERTONI; MARTINO, 2008; FALCONER; STOREY, 2007; HUANG; DANG; HUHNS, 2006) que tentam unificar os conceitos através da compatibilização de ontologias. Neste trabalho não estamos preocupados em resolver esse tipo de problema. Mas, para permitir a especificação de um processo de negócio com base nas ontologias de serviços *Web* é fundamental que os conceitos obedeçam a uma classificação.

Nesse aspecto, a OWL-S oferece três mecanismos para classificar as ontologias dos serviços *Web*. No primeiro, é construída uma hierarquia de subclasses a partir da classe *Profile* do serviço – *ProfileHierarchy* – semelhante a técnica provida pela programação orientada a objeto. Cada serviço tem seu *Profile* instanciado a partir de uma subclasse dessa hierarquia. No segundo, a propriedade *serviceCategory* do *Profile* é utilizada para classificar o serviço de acordo com as taxonomias oferecidas por entidades externas que padronizam essa classificação. Por último, as propriedades *serviceClassification* e *serviceProduct* da classe *Profile* são utilizadas para referenciar conceitos que descrevem respectivamente o tipo do serviço e os produtos tratados pelo mesmo. A principal finalidade desses mecanismos é a construção de um serviço de **páginas amarelas** para permitir a descoberta automática de serviços *Web* (DAML, 2004). Porém, durante a realização deste trabalho, percebemos que o terceiro mecanismo de classificação provido pela OWL-S pode ser utilizado para permitir a especificação de *workflows* semânticos a partir das ontologias dos serviços *Web*.

Por sua vez, Guarino (1998) sugere que uma ontologia seja classificada em três níveis de generalização. O primeiro, chamado de *Ontologias de Alto-Nível*, descreve os conceitos gerais como espaço, tempo, assunto, objeto, evento, ação, etc., os quais são independentes de

um problema ou domínio específico. O segundo é dividido em *Ontologias de Domínio* e *Ontologias de Tarefa*, e descrevem respectivamente o vocabulário relacionado ao domínio genérico (por exemplo, medicina, automóvel), ou uma tarefa/atividade genérica (como diagnóstico, venda), especializando os termos introduzidos no primeiro nível. O terceiro e último nível é chamado de *Ontologias de Aplicação*, e descreve conceitos especializados do segundo nível que correspondem aos papéis exercidos por entidades do domínio específico enquanto desempenham certa atividade.

Tendo como base essas propostas, foram desenvolvidas no WebFlowAH as chamadas **ontologias de domínio comum**. Essas ontologias têm como finalidade classificar os conceitos definidos pelas ontologias dos serviços *Web* em um vocabulário mais uniforme. As ontologias de domínio comum foram derivadas a partir das *Ontologias de Domínio* e *Ontologias de Tarefa* (GUARINO, 1998) e consistem nas ontologias de **Objeto** e **Tarefa**, respectivamente. A ontologia de Objeto é uma especialização das *Ontologias de Domínio* que procura descrever um objeto de domínio, por exemplo, uma **reserva** de hotel, um **pedido** de compra de livros. Já a ontologia de Tarefa descreve uma tarefa de domínio que está associada aos objetos de domínio (ex.: **reservar**, **comprar**, etc.). Apesar de ser uma especialização das *Ontologias de Tarefa*, foi mantida a mesma nomenclatura neste trabalho simplesmente porque ela também representa uma atividade de um processo de negócio.

Além de nivelar os conceitos contidos nas ontologias dos serviços *Web*, as ontologias de domínio comum possibilitam que um objetivo a ser alcançado por um processo de negócio seja representado a partir de uma associação entre os próprios conceitos. Para reforçar essa afirmação, as ontologias mencionadas no exemplo anterior foram classificadas de acordo com a hierarquia de conceitos exibida na Figura 4.2. Os conceitos classificados na ontologia de Objeto são obtidos a partir da propriedade *serviceProduct* enquanto que os conceitos classificados na ontologia de Tarefa são obtidos da propriedade *serviceClassification*, ambas definidas pelo *Profile* dos serviços *Web*.

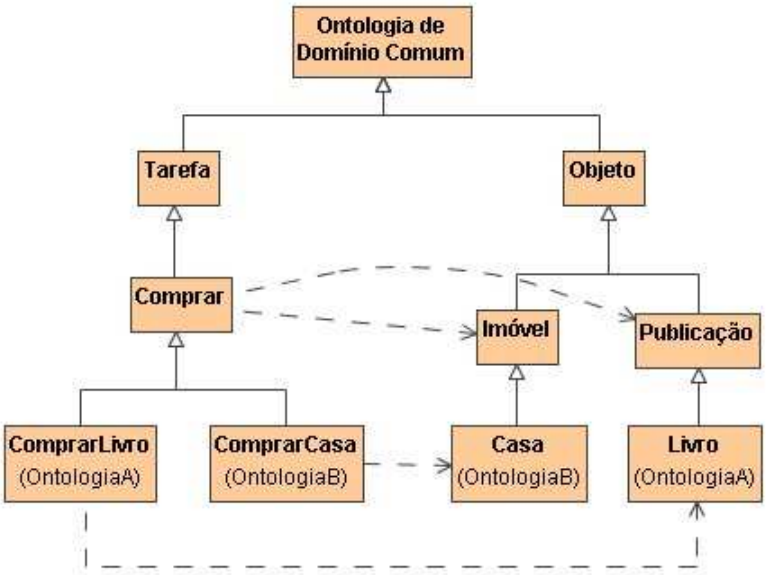


Figura 4.2. Classificação de conceitos nas ontologias de domínio comum.

A Figura 4.2 ilustra o uso das ontologias de Objeto e Tarefa no WebFlowAH, onde os conceitos *ComprarLivro* e *ComprarCasa* (ambos representando serviços *Web* semânticos) foram classificados como subclasses da tarefa *Comprar* enquanto que os conceitos *Casa* e *Livro* nas subclasses dos respectivos objetos *Imóvel* e *Publicação*. Uma associação entre os conceitos *Comprar* e *Imóvel* permite a representação de um objetivo para o processo de compra de imóveis. De maneira análoga, os conceitos *Comprar* e *Publicação* expressam um objetivo do processo de compra de publicações. Portanto, no WebFlowAH as ontologias dos serviços *Web* precisam ser classificadas em Objeto ou Tarefa para que seja possível uma especificação de alto nível de um processo de negócio.

4.1.2 Objetivo de Negócio

Um **objetivo de negócio** é a representação dos desejos do usuário com relação a um dado processo de negócio. Tal representação é baseada na planificação IA (MILANOVIC; MALEK, 2006; RAO; SU, 2004) e consiste em especificar uma tarefa ou ação juntamente com um conjunto de entradas e precondições necessárias para executar essa tarefa e um conjunto de saídas e efeitos produzidos após a execução dessa tarefa. Por exemplo, se um usuário precisa reservar uma passagem aérea, geralmente ele expressa esse desejo informando o destino da viagem, o horário do vôo, o número de escalas permitido, se o assento deve ou não ser marcado, e assim por diante. Para um processo de venda de passagens aéreas, esse

desejo poderia ser simplesmente "*reservar o vôo 9876, partindo da cidade de Fortaleza com destino a cidade de Natal, no dia 01 de julho de 2008 às 6:00 horas*". Nesse caso, o objetivo de negócio referente ao desejo em questão é formado pela tarefa *reservar vôo* e pelas entradas *cidade de origem*, *cidade de destino*, *número do vôo* e *horário do vôo*. Embora essa representação seja parecida com a especificação de uma atividade de um *workflow*, existem diferenças na definição entre esses elementos.

No contexto dos Sistemas de *Workflow*, uma atividade representa um conjunto de passos que devem ser realizados para se alcançar uma determinada meta. Esses passos são encadeados através de seus respectivos parâmetros de entrada e saída. Por exemplo, no processo de venda de passagens aéreas a atividade de reservar um vôo consiste em consultar por vôos disponíveis, selecionar um vôo dentre os vôos que foram consultados e efetivar a reserva do vôo selecionado. Já no ambiente WebFlowAH, um objetivo de negócio representa simplesmente a meta a ser alcançada, e não considera, portanto, a seqüência de passos que compõe uma atividade. O objetivo de negócio é especificado por uma tarefa e um conjunto de parâmetros necessários para a realização da mesma (entradas, saídas, precondições e efeitos).

4.1.3 *Workflow* Semântico

Um processo de negócio pode ser representado por um *workflow*, cujas atividades são realizadas de forma manual ou automática. Em se tratando de negócios eletrônicos, essas atividades são automatizadas por serviços *Web* pertencentes às organizações envolvidas no processo de negócio, os quais foram disponibilizados na *Internet* ou em uma rede corporativa. Conseqüentemente, cada atividade de um *workflow* é implementada por uma operação específica de um ou mais serviços *Web*. Por exemplo, seja *CompanhiaAerea.wsdl* (ver Figura 4.3) a descrição WSDL de um serviço de reserva e compra de passagens aéreas de uma dada companhia. Essa descrição define a sintaxe de cada operação do serviço *Web* em termos de número de parâmetros e seus respectivos tipos de dados. Se fosse para especificar um processo de venda de passagens aéreas na linguagem BPEL em um WMS, a atividade de solicitar reserva de um vôo seria certamente implementada pela operação *reservarVoo*, havendo desse modo, uma associação entre a atividade desse processo e o serviço *Web* em questão.

```
<!-- Tipos de Dados-->
<wsdl:types>
  <complexType name="voo">
    <sequence>
      <element name="origem" type="xsd:string"/>
      <element name="destino" type="xsd:string"/>
      <element name="data" type="xsd:string"/>
      <element name="hora" type="xsd:string"/>
      <element name="voo" type="xsd:string"/>
    </sequence>
  </complexType>
</wsdl:types>
<!-- Mensagens -->
<wsdl:message name="reservarVooRequest">
  <wsdl:part name="voo" type="tns:voo"/>
</wsdl:message>
<wsdl:message name="reservarVooResponse">
  <wsdl:part name="localizador" type="xsd:string"/>
</wsdl:message>
<!-- Operações -->
<wsdl:portType name="CompanhiaAereaPortType">
  <wsdl:operation name="reservarVoo">
    <wsdl:input message="reservarVooRequest"/>
    <wsdl:output message="reservarVooResponse"/>
  </wsdl:operation>
</wsdl:portType>
```

Figura 4.3. Descrição WSDL do serviço *Web* CompanhiaArea

Buscando elevar o nível de abstração de uma definição de processo, isto é, diminuir a dependência entre a especificação de um processo de negócio e os serviços *Web* disponíveis para a implementação desse processo, foi criado para o ambiente WebFlowAH um novo tipo de *workflow*, o qual foi chamado de ***workflow* semântico**. Diferentemente das abordagens estáticas as atividades de um *workflow* semântico não são associadas diretamente a operações de serviços *Web*, e sim a conceitos descritos a partir das ontologias de domínio comum. Mais especificamente, um *workflow* semântico é uma abstração de um processo de negócio descrita na forma de objetivos ou metas de negócio (ver Seção 4.1.2) a serem alcançados pelo mesmo. Por exemplo, seja *CompanhiaAerea.owl* (ilustrada na Figura 4.3) uma ontologia OWL-S para o serviço *Web* mencionado no parágrafo anterior. Ela define as propriedades desse serviço através de suas entradas, precondições, saídas e efeitos. Aquele mesmo processo de venda de passagens teria agora sua atividade de solicitar reserva especificada a partir dessa ontologia criando-se o objetivo de negócio *Reservar Vôo*, onde o conceito *Reservar* representa uma tarefa de domínio e o conceito *Vôo* um objeto de domínio, como mostra a Figura 4.3.

```
<profile:Profile rdf:ID="ReservarVooProfile">
  <service:presentedBy rdf:resource="#CompanhiaAereaService"/>
  <profile:has_process rdf:resource="#ReservarVooProcess"/>
  <profile:serviceName>Reserva de Vôos</profile:serviceName>
  <profile:textDescription>
     Este serviço realiza a reserva de um vôo de uma origem
     para um destino em uma data e hora específica.
  </profile:textDescription>
  <profile:serviceClassification rdf:datatype="&xsd;anyURI"
  >http://CompanhiaAerea.owl#Reservar</profile:serviceClassification>
  <profile:serviceProduct rdf:datatype="&xsd;anyURI"
  >http://CompanhiaAerea.owl#Voo</profile:serviceProduct>
  <profile:hasInput rdf:resource="#Origem"/>
  <profile:hasInput rdf:resource="#Destino"/>
  <profile:hasInput rdf:resource="#Data"/>
  <profile:hasInput rdf:resource="#Hora"/>
  <profile:hasInput rdf:resource="#Voo"/>
  <profile:hasPrecondition rdf:resource="#VooExiste"/>
  <profile:hasOutput rdf:resource="#Localizador"/>
  <profile:hasResult rdf:resource="#ReservaConfirmada"/>
  <profile:hasResult rdf:resource="#ReservaNaoConfirmada"/>
</profile:Profile>
```

Figura 4.4. Descrição OWL-S do serviço *Web* CompanhiaArea

4.1.4 Plano de Execução

Se um *workflow* semântico representa um processo abstrato, então para que este seja executado por uma máquina de *workflow* é necessário transformá-lo em um processo concreto. Um **plano de execução** representa o resultado de tal transformação, e consiste em um *workflow* tradicional (usando BPEL, por exemplo) contendo atividades que são automatizadas por um conjunto de serviços *Web*, os quais foram descobertos e compostos dinamicamente pelo ambiente WebFlowAH, tendo como critérios de busca e seleção os conceitos utilizados para especificar os próprios objetivos de negócio. De fato, o que ocorre durante a execução de um *workflow* semântico no WebFlowAH é que para cada objetivo de negócio é realizada a busca de um serviço *Web* que seja compatível com o mesmo, isto é, que serviço possui as mesmas entradas, precondições, saídas e efeitos do objetivo. Caso não exista um serviço que realize o objetivo de negócio desejado, o ambiente tenta compor um conjunto de serviços para atender esse objetivo. Então, os serviços *Web* descobertos e compostos pelo ambiente são combinados de modo a implementar um ou mais planos de execução. Considerando-se novamente o exemplo anterior do serviço de reserva e compra de passagens aéreas, se um processo de negócio foi especificado com o objetivo de negócio *Reservar Vôo*, é possível realizar esse processo utilizando o serviço *Web* descrito pela ontologia da Figura 4.3. Essa conclusão é obtida por uma consulta feita por uma máquina de inferência sobre a

ontologia em questão, que busca por um serviço *Web*, cujo tipo de serviço (descrito pela propriedade *serviceClassification*) é a tarefa de domínio *Reservar* e o produto endereçado pelo serviço (descrito pela propriedade *serviceProduct*) é o objeto de domínio *Vôo*.

4.2 ARQUITETURA CONCEITUAL

A arquitetura conceitual do ambiente proposto, cuja representação em um diagrama de componentes da UML 1.4 é ilustrada na Figura 4.5, é formada por sete componentes lógicos. São eles: Interface com o Usuário (IU), Gestor de *Workflows*, Gestor de Serviços, Compositor Semântico, Seletor Semântico, Executor de *Workflows* e Repositório Semântico.

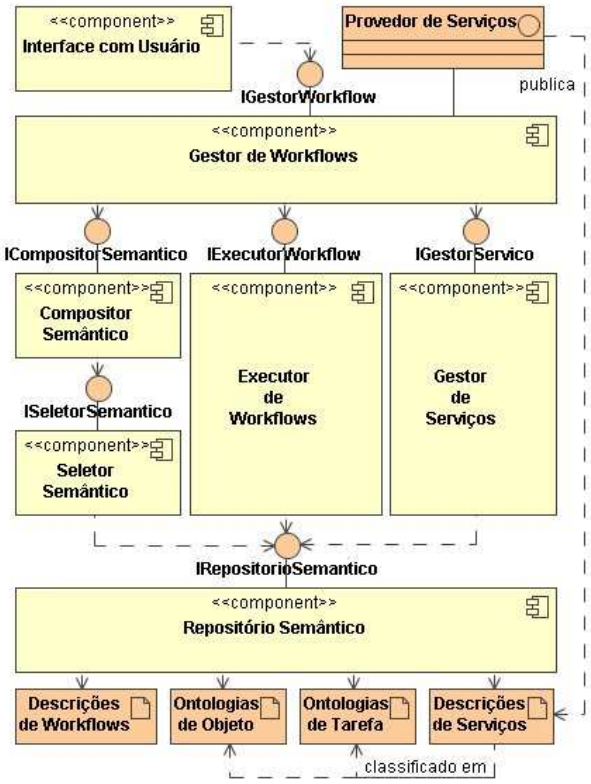


Figura 4.5. Arquitetura conceitual do WebFlowAH.

A **IU** do WebFlowAH provê serviços de uso e administração do ambiente. Para os usuários que especificam processos no WebFlowAH, a IU oferece as funcionalidades de construir, atualizar e executar *workflows* semânticos. Para os usuários que administram o WebFlowAH, a IU oferece as funcionalidades de importar, classificar e consultar ontologias de serviços *Web* fornecidas por algum Provedor de Serviços. Essa interface comunica-se com o Gestor de *Workflows* que, por sua vez, controla o Gestor de Serviços, o Executor de

Workflows e o Compositor Semântico. Esse último comunica-se ainda com o Seletor Semântico que, juntamente com os dois anteriores, interagem com o Repositório Semântico.

O **Gestor de *Workflows*** gerencia o funcionamento dos demais componentes do ambiente, suportando principalmente as funcionalidades de especificação dos *workflows* semânticos e de geração dos planos de execução. Tanto os *workflows* semânticos quanto os planos de execução são definidos através da OWL-S com o auxílio das ontologias dos serviços *Web*.

O **Gestor de Serviços** é responsável pela importação e validação das descrições OWL-S dos serviços *Web* selecionadas por um usuário administrador e que são adicionadas às ontologias de domínio comum do WebFlowAH. Todas as entradas, saídas, precondições e efeitos de um serviço *Web* semântico, importados neste ambiente pelo Gestor de Serviços, são classificados na ontologia de Objeto, enquanto que os processos desse mesmo serviço são classificados na ontologia de Tarefa. O Gestor de Serviços também auxilia o Gestor de *Workflows* na especificação de *workflows* semânticos oferecendo o serviço de consulta aos conceitos cadastrados nas ontologias Objeto e Tarefa.

A execução de um *workflow* semântico é suportada pelo **Executor de *Workflows***. Esse componente recebe os planos de execução gerados pelo Gestor de *Workflows* a um *workflow* especificado no WebFlowAH e, através de critérios de seleção (menor número de serviços, menor tempo de resposta, etc.), escolhe um plano para ser executado. Durante a execução, o Executor de *Workflows* faz chamadas remotas às operações dos serviços *Web* contidas no plano. Além disso, ele também monitora essas chamadas para verificar a ocorrência de alguma falha (indisponibilidade de um serviço, por exemplo). Se ocorrer uma falha de execução, o próprio componente encarrega-se de escolher outro plano na tentativa de terminar com sucesso a execução do *workflow* semântico. Pode-se dizer que o Executor de *Workflows* equivale a uma máquina de *workflow* de um WMS tradicional vista no Capítulo 2.

O **Compositor Semântico** é responsável por descobrir e compor os serviços *Web* para um *workflow* semântico. Primeiramente, ele tenta recuperar, dentre um conjunto de serviços disponíveis no repositório do WebFlowAH, aqueles que podem ser usados para compor o plano de execução atendendo aos objetivos do processo de negócio. Após essa etapa, este componente tenta combinar os serviços encontrados de modo a consumir todas as entradas e precondições definidas nos objetivos de negócio e a produzir todas as saídas e efeitos desejados pelo processo. Por fim, os serviços *Web* combinados são organizados pelo Compositor Semântico segundo o fluxo de mensagens entre as saídas de um serviço com as entradas do serviço subseqüente.

O **Seletor Semântico** recebe as consultas semânticas que são requisitadas pelo Compositor Semântico. Tipicamente esse componente é semelhante a um combinador (do inglês *matchmaker*) (KLUSCH; FRIES; SYCARA, 2006), sendo dividido em dois subsistemas: (i) uma máquina de inferência que obtém os resultados das consultas e (ii) uma base de conhecimento construída para a realização de consultas sobre as ontologias.

Por fim, o **Repositório Semântico** é encarregado por gerenciar a persistência das ontologias. Todas as solicitações de leitura e escrita às ontologias são atendidas por esse componente. No WebFlowAH, as ontologias são armazenados localmente em arquivos para se obter um melhor desempenho em relação ao acesso remoto aos serviços *Web* semânticos.

4.3 FUNCIONAMENTO DO AMBIENTE

No ambiente proposto neste trabalho existem dois perfis de usuário (ver Figura 4.6): (i) um especificador de processos, responsável pela construção de *workflows* semânticos; (ii) e um administrador de serviços, responsável pela importação, classificação e consulta às ontologias dos serviços *Web*.

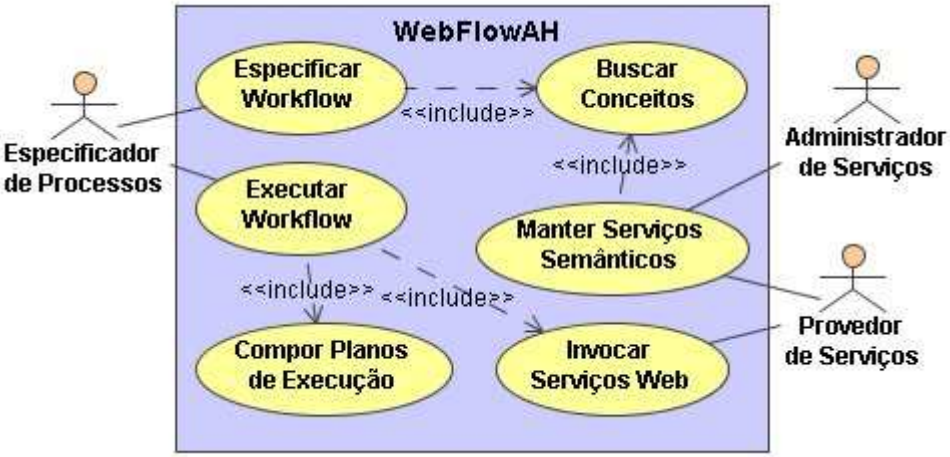


Figura 4.6. Perfis de Usuário do WebFlowAH.

O usuário **administrador de serviços** seleciona as ontologias OWL-S de serviços *Web* disponibilizadas por um Provedor de Serviços e solicita ao Gestor de *Workflows* que as mesmas sejam importadas no Repositório Semântico. O Gestor de *Workflows* encaminha essa solicitação ao Gestor de Serviços que recupera as ontologias OWL-S a partir de uma *Uniform Resource Locator* (URL) especificada pelo próprio usuário administrador. Além disso, o Gestor de Serviços também verifica a consistência de cada ontologia antes de solicitar ao

Repositório Semântico que as mesmas sejam armazenadas localmente. Finalmente, o administrador de serviços classifica os conceitos definidos nas ontologias importadas, onde esses conceitos são persistidos nas ontologias de Objeto e Tarefa pelo Repositório Semântico. A Figura 4.7 mostra um diagrama de seqüência da UML 1.4 contendo os passos realizados pelo usuário administrador de serviços no WebFlowAH.

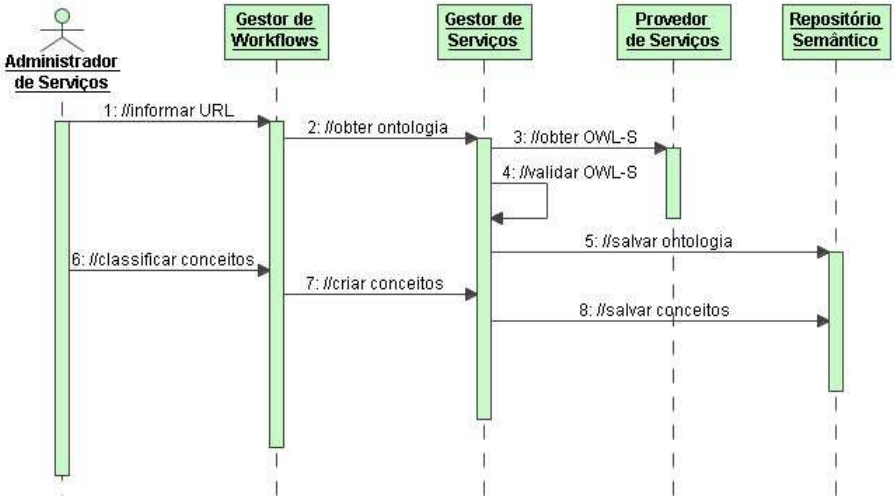


Figura 4.7. Funcionalidades do Administrador de Serviços.

Para exemplificar como funcionaria na prática a seqüência de passos realizados pelo administrador de serviços, considere a ontologia OWL-S da Figura 4.2 que descreve um serviço de reserva e compra de passagens aéreas. As entradas exigidas pelo serviço são descritas pelos conceitos *Origem*, *Destino*, *Data*, *Hora e Vôo* enquanto que a saída produzida é descrita pelo conceito *Localizador*. A propriedade *serviceClassification* está definida com o conceito *Reservar* e a propriedade *serviceProduct* pelo conceito *Vôo*. Após a importação dessa ontologia, os conceitos são extraídos pelo Gestor de Serviços e apresentados ao administrador de serviços via IU do WebFlowAH. O próprio Gestor de Serviços sugere que o conceito *Reservar* seja classificado como uma instância de Tarefa enquanto os demais conceitos como instâncias de Objeto. Caso esteja correta, o administrador de serviços confirma essa classificação que é, então, armazenada em suas respectivas ontologias pelo Repositório Semântico.

Já o usuário **especificador de processos** utiliza o ambiente WebFlowAH para definir um *workflow* semântico que represente um determinado processo de negócio. Para tal, ele deve informar os objetivos desse processo, cada um contendo suas entradas, precondições, saídas e efeitos desejados. O Gestor de *Workflows* auxilia o especificador de processos na definição dos objetivos de negócio permitindo a realização de consultas sobre as ontologias de domínio comum a partir do Gestor de Serviços. Essa consulta é encaminhada ao

Repositório Semântico que retorna os conceitos de Tarefa e Objeto disponíveis no ambiente. Com base nos conceitos retornados pelo Repositório Semântico, o usuário especificador escolhe aqueles que definem os objetivos de negócio desejados.

Após concluir a especificação do *workflow* semântico, o especificador de processos pede ao Gestor de *Workflows* a execução do *workflow* especificado. Nesse momento, o Gestor de *Workflows* solicita ao Compositor Semântico a geração de uma composição de serviços *Web* capaz de implementar o processo em questão. Então, o Compositor Semântico solicita ao Seletor Semântico que realize inferências sobre as ontologias de serviço disponíveis no Repositório Semântico e, com isso, descubra de forma dinâmica quais combinações de serviços atendem aos objetivos do processo, isto é, quais composições produzem as saídas e efeitos desejados pelo processo de negócio a partir de um conjunto de entradas e precondições exigidas. O resultado dessa composição é repassado ao Gestor de *Workflows* que gera, ao final, um ou mais planos de execução. Durante a execução, um plano, dentre os gerados, é escolhido (utilizando-se critérios tais como o número de operações ou o tempo de resposta de cada serviço) pelo Executor de *Workflows* para que sejam realizadas chamadas remotas aos serviços *Web*. O diagrama de seqüência da UML 1.4 ilustrado na Figura 4.8 descreve sucintamente esses passos.

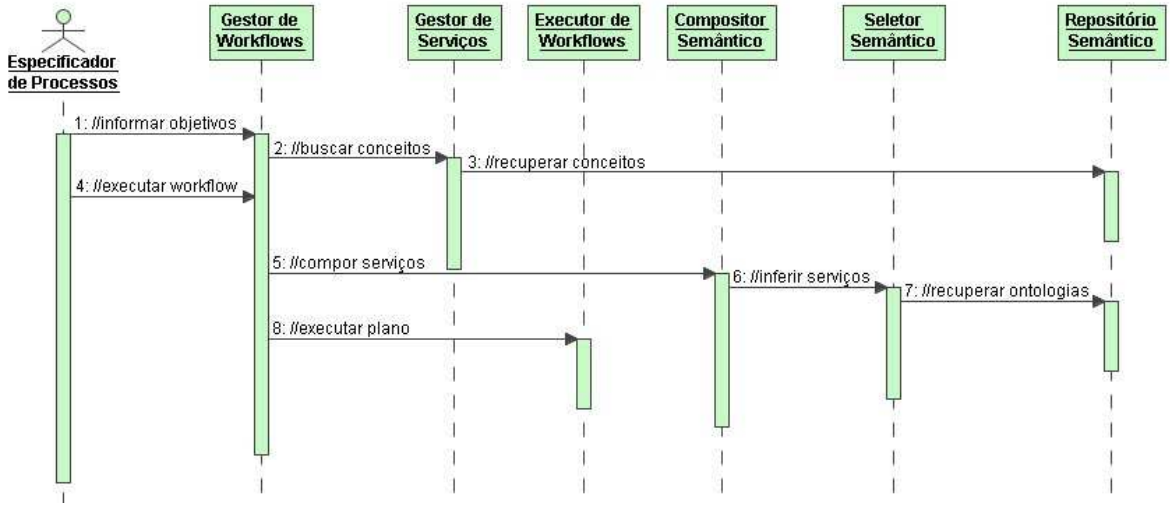


Figura 4.8. Funcionalidades do Especificador de Processos.

Nas abordagens estáticas (tais como BPEL) é preciso ter um conhecimento prévio do processo de negócio para que o mesmo seja automatizado. Diferentemente, o ambiente WebFlowAH permite que um especificador de processos seja capaz de construir um *workflow* semântico mesmo não estando familiarizado com o vocabulário do processo a ser modelado. Através de um assistente disponibilizado no ambiente proposto, o especificador de processos poderá escolher os objetivos de negócio como também identificar quais entradas e

precondições são exigidas, e quais saídas e efeitos são produzidos por um dado objetivo. Cabe ao administrador de serviços do WebFlowAH importar e classificar as ontologias dos serviços *Web* para que o especificador de processos tenha proveito de tal assistente.

As subseções a seguir descrevem o funcionamento do WebFlowAH em maiores detalhes.

4.3.1 Especificação do *Workflow* Semântico

A especificação de um *workflow* semântico no ambiente WebFlowAH consiste em descrever os objetivos de um processo de negócio através das ontologias Objeto e Tarefa. Ao contrário de outras ferramentas tradicionais de *workflows*, onde é preciso construir um *workflow* com todas as atividades realizadas por um processo de negócio, o WebFlowAH disponibiliza um assistente para auxiliar o especificador de processos a determinar quais são esses objetivos, sem a necessidade de definir cada uma das atividades. De fato, esse assistente exibe uma lista de conceitos do tipo Tarefa e outra do tipo Objeto, onde o especificador de processos poderá escolher uma tupla <*Tarefa, Objeto*> (como por exemplo <*Comprar, Casa*> ou <*Reservar, Passagem*>) para descrever um dado objetivo de negócio. O assistente também exibe listas contendo entradas, saídas, precondições e efeitos relacionados com o objetivo especificado, e que são adicionadas à definição do *workflow* semântico pelo especificador de processos. Portanto, no WebFlowAH não é preciso descrever todas as atividades de um processo de negócio, apenas os objetivos de negócio, o que caracteriza uma especificação de processo incompleta ou parcial. Com a realização de inferências sobre as ontologias Objeto e Tarefa, e a composição de planos de execução, o WebFlowAH identifica quais atividades precisam ser selecionadas para atingir um determinado objetivo.

Por outro lado, se precisarmos criar especificações de processo mais complexas, o WebFlowAH provê fluxos de controle que são os mesmos oferecidos por uma típica ferramenta de definição de processo de um WMS. Tais fluxos são imprescindíveis para aquelas situação onde não é possível representar um processo de negócio em um *workflow* contendo uma única atividade. A IU do ambiente ilustrada pela Figura 4.9 mostra os fluxos de controle disponíveis para a especificação de *workflow* semântico. São eles: *seqüência* (S), *divisão-junção* (D), *escolha* (E), *condição* (C) e *repetição* (R). O primeiro fluxo estabelece a ordem de execução das atividades de um *workflow*. O segundo fluxo possibilita a execução de

atividades em paralelo. O terceiro fluxo cria a possibilidade de escolher uma atividade a ser executada dentre um conjunto de opções. O quarto fluxo estabelece regras para que uma atividade seja executada. Os conceitos do tipo Objeto são utilizados para a definição dessas regras. O último fluxo possibilita a criação de laços de repetição para um conjunto de atividades.

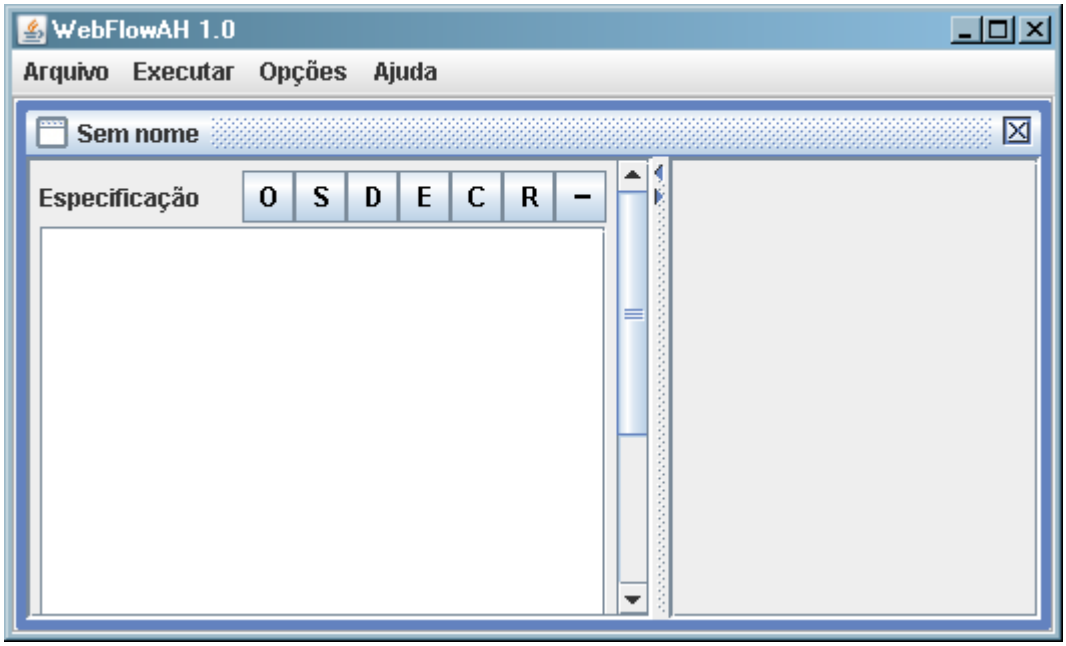


Figura 4.9. Tela de Especificação de Processos do WebFlowAH.

Internamente, a especificação de um *workflow* semântico transcorre da seguinte maneira. O especificador de processos solicita ao Gestor de *Workflows* os conceitos das ontologias de domínio comum do WebFlowAH para descrever os objetivos de negócio (via assistente). O Gestor de *Workflows* aciona então o Gestor de Serviços que consulta os conceitos disponíveis através do Repositório Semântico. Após a definição de um objetivo de negócio, o Gestor de Serviços é novamente acionado pelo Gestor de *Workflows* para recuperar os conceitos relacionados às entradas, saídas, precondições e efeitos daquele objetivo. Essa seqüência é repetida para os demais objetivos do processo. Caso seja preciso adicionar algum fluxo de controle (condição, repetição, bifurcação, etc.) na especificação do *workflow* semântico, o especificador de processo pode utilizar as opções existentes no ambiente WebFlowAH. Por fim, o especificador de processos pede ao Gestor de *Workflows* que a especificação do processo de negócio seja armazenada no repositório apropriado.

4.3.2 Composição de Planos de Execução

A composição de planos de execução é uma funcionalidade importante do WebFlowAH que está diretamente relacionada com a descoberta e a composição dinâmicas de serviços *Web*. Diversas pesquisas nessa área levaram ao desenvolvimento de novos algoritmos, componentes e ferramentas para suportar esses mecanismos (AGRE *et al.*, 2007; COSTA; PIRES; MATTOSO, 2004b; MEDJAHED; BOUGUETTAYA; ELMAGARMID, 2003; PATHAK *et al.*, 2006; PISTORE *et al.*, 2004; SIRIN; HENDER; PARSIA, 2003). Entretanto, o foco deste trabalho não é propor uma nova solução para o problema da automação da descoberta e da composição, e sim fazer uso das soluções existentes para construir um ambiente que permita a especificação e execução de processos de negócio com base nas descrições dos serviços *Web* semânticos.

Nesse contexto, o algoritmo proposto por Costa, Pires e Mattoso (2004b) sugere que os serviços *Web* sejam descobertos e compostos dinamicamente sem nenhuma intervenção humana, mesmo que ocorram mudanças na especificação do processo de negócio ou que um dos serviços venha a ficar indisponível. Outra característica particular desse trabalho é a geração de composições alternativas para uma mesma especificação, o que aumenta a garantia de que um processo será executado corretamente mesmo que falhas aconteçam durante a execução.

Medjahed, Bouguettaya, Elmagarmid (2003) propõem um modelo de composabilidade de serviços *Web* que consiste de quatro fases separadas: especificação, combinação, seleção e geração. Na primeira fase são criadas as especificações de alto nível de uma composição desejada. Na segunda fase são gerados os planos de composição que estão em conformidade com a dada especificação usando-se um algoritmo de *matchmaking*. Na terceira fase um plano de composição é escolhido com base em critérios de qualidade (por exemplo, custo) para ser usado na última fase onde se gera uma descrição detalhada do serviço composto.

Esses trabalhos serviram de alicerce para a especificação dos componentes Compositor Semântico e Seletor Semântico. O primeiro é responsável por compor os serviços *Web* a partir da especificação de um *workflow* semântico, tendo como parâmetros as entradas e as precondições exigidas e as saídas e os efeitos produzidos pelo processo de negócio que foi especificado no ambiente WebFlowAH. O outro componente tenta descobrir que serviços *Web* combinam (*matching*) com esses parâmetros do processo de negócio.

4.3.3 Execução do *Workflow* Semântico

Para executar um *workflow* semântico elaborado por um especificador de processos no ambiente WebFlowAH, o Executor de *Workflows* escolhe um dos planos de execução que foram gerados na etapa de composição, e realiza a invocação dos serviços *Web* segundo a especificação do próprio *workflow* semântico. Essa escolha, a princípio, é feita de acordo com o menor caminho do fluxo de controle, isto é, aquele que tem menos serviços *Web* a serem chamados. Um monitoramento do plano de execução é realizado pelo próprio Executor de *Workflows* para verificar a ocorrência de alguma falha, seja por um serviço não estar disponível ou devido a qualquer outro problema não identificado. No caso de ocorrer uma falha, o Executor de *Workflows* busca outro plano que atenda a especificação do mesmo *workflow* semântico de modo a garantir a execução do processo de negócio. Tal busca pode levar em consideração o número de serviços utilizados no plano de execução ou o tempo de resposta de cada serviço para a escolha do próximo plano.

5 IMPLEMENTAÇÃO

O ambiente WebFlowAH foi projetado inicialmente para integrar soluções existentes na área da *Web* Semântica que suportem a descoberta e composição dinâmicas de serviços *Web*. Porém, a baixa quantidade de ferramentas disponibilizadas pelas soluções que foram pesquisadas durante a elaboração deste trabalho, em particular aquelas relacionadas com a ontologia OWL-S, dificultaram a realização de uma análise comparativa mais detalhada que resultasse na definição de critérios significativos para a escolha adequada dessas ferramentas. Por essa razão, achamos coerente avaliar fatores técnicos, tais como a linguagem de programação adotada e a existência de documentação, para determinar quais ferramentas poderiam ser utilizadas no desenvolvimento do WebFlowAH. Tais fatores também nortearam as decisões de projeto do WebFlowAH. Por exemplo, a maioria das ferramentas providas pela *Web* Semântica são implementadas utilizando a linguagem de programação *Java*. Conseqüentemente, o ambiente proposto neste trabalho foi implementado utilizando essa mesma linguagem.

Neste capítulo daremos ênfase principalmente a maneira como os componentes do WebFlowAH **Repositório Semântico** (Seção 5.1), **Compositor Semântico** (Seção 5.2), **Seletor Semântico** (Seção 5.3) e **Executor de *Workflows*** (Seção 5.4), apresentados previamente na Seção 4.2, foram implementados a partir das ferramentas selecionadas para cada um deles. Tais componentes formam o cerne deste ambiente, e por essa razão, foram escolhidos para explicar a implementação do mesmo.

5.1 REPOSITÓRIO SEMÂNTICO

Para implementar o componente Repositório Semântico, foi utilizada uma API *Java* desenvolvida pelo *Mindswap Group* da *University of Maryland*[25] que suporta a manipulação tanto de descrições semânticas OWL-S quanto de regras semânticas expressas em SWRL, a OWL-S API (MINDSWAP, 2004). Tal API é uma extensão da API *Jena* (HP, 2008), um arcabouço *Java* desenvolvido pelo *HP Labs Semantic Web Research*[26] que suporta a

[25] http;//www.mindswap.org
[26] http://www.hpl.hp.com/semweb/

manipulação de ontologias em RDF e OWL, e a execução de consultas em RDQL e SPARQL.

No WebFlowAH, o Repositório Semântico é responsável por armazenar as ontologias que são manipuladas pelos demais componentes deste ambiente, funcionando como uma base de conhecimento, cujas informações mantidas são referentes a um determinado domínio. Para representar tal base de conhecimento foi implementada neste trabalho a classe *Java* **RepositorioSemantico** que utiliza uma interface disponibilizada pela OWL-S API chamada *OWLKnowledgeBase.* Já as funcionalidades de leitura, validação e escrita das ontologias suportadas pelo Repositório Semântico são distribuídas entre quatros classes *Java* (ilustradas na Figura 5.1), as quais são implementadas separadamente da classe anterior: (i) DocumentoOWLS; (ii) ServicoOWL; (iii) ConceitoOWL; e (iv) WorkflowOWLS. Cada uma dessas classes é responsável por uma ontologia específica do ambiente WebFlowAH e são descritas em maiores detalhes a seguir.

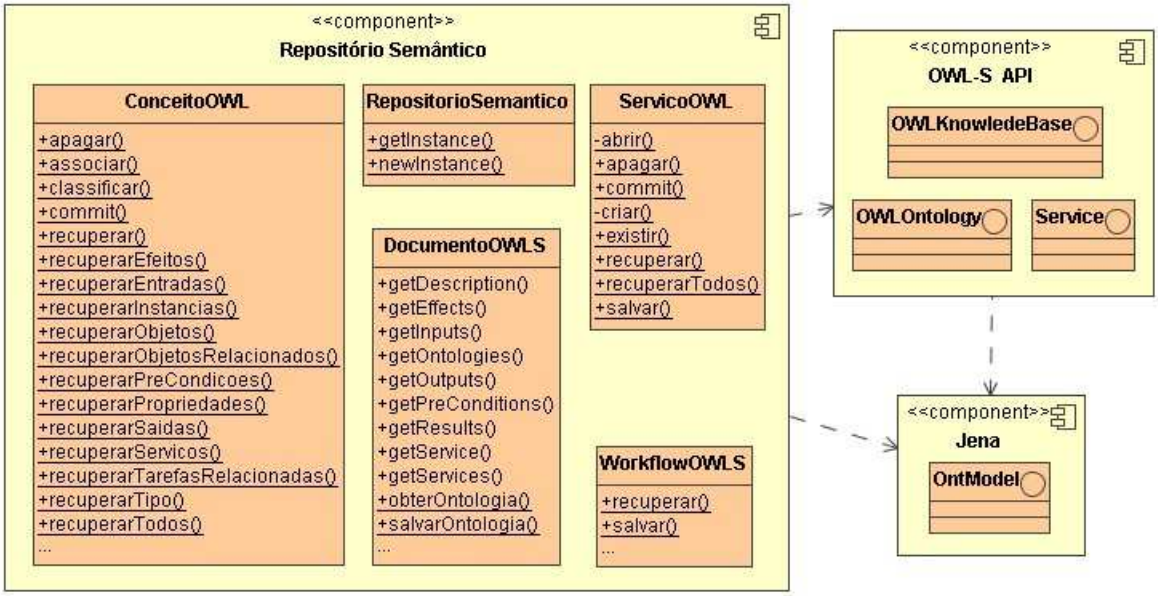


Figura 5.1. Classes do componente Repositório Semântico

A classe **DocumentoOWLS** tem como funcionalidades a importação e a validação das ontologias OWL-S dos serviços *Web* no ambiente WebFlowAH, as quais são oferecidas por algum provedor de serviços. Essa classe é precisamente um envoltório (do inglês *wrapper*) da interface *Service* disponibilizada pela OWL-S API que manipula uma ontologia de um serviço *Web*. Tal solução adotada de envolver (*wrapping*) um componente existente visa principalmente criar um baixo acoplamento entre o WebFlowAH e a OWS-L API, permitindo em um trabalho futuro a utilização de outras ferramentas voltadas à manipulação de

ontologias OWL-S. A Tabela 5.1 mostra um resumo dos métodos que são incluídos nessa classe.

Tabela 5.1. Métodos da classe DocumentoOWLS

`String`	`getDescription(Object arg)` Retorna a informação sobre o serviço descrita no *Profile* pela propriedade *textDescription*
`List`	`getEffects(Object arg)` Retorna os efeitos gerados pelo serviço descritos no *Profile* pela propriedade *hasEffect*
`List`	`getInputs(Object arg)` Retorna as entradas usadas pelo serviço descritas no *Profile* pela propriedade *hasInput*
`List`	`getOntologies()` Retorna as ontologias que compõe a ontologia OWL-S
`List`	`getOutputs(Object arg)` Retorna as saídas produzidas pelo serviço descritas no *Profile* pela propriedade *hasOutput*
`List`	`getPreConditions(Object arg)` Retorna as precondições exigidas pelo serviço descritas no *Profile* pela propriedade *hasPrecondition*
`List`	`getResults(Object arg)` Retorna os resultados obtidos pelo serviço descritos no *Profile* pela propriedade *hasResult*
`List`	`getServices()` Retorna os serviços descritos na ontologia OWL-S
`boolean`	`isConsistent()` Verifica se uma ontologia OWL-S está consistente
`Static DocumentoOWLS`	`obterOntologia(String arg, boolean local)` Obtém uma ontologia OWL-S a partir de uma localização
`static void`	`salvarOntologia(DocumentoOWLS arg)` Salva a ontologia no Repositório Semântico

A classe **ServicoOWL** é responsável por armazenar informações sobre as ontologias dos serviços *Web* que foram importadas pelo usuário administrador de serviços no WebFlowAH. Para tal, foi definida uma ontologia chamada *Servicos.owl* (ilustrada pela Figura 5.2) que é manipulada a partir dos métodos oferecidos pela classe em questão. Essa ontologia descreve um serviço *Web* a partir da classe *rdf:ID="Servico"* cujas propriedades *rdf:ID="nome"*, *rdf:ID="categoria"* e *rdf:ID="ontologia"* representam respectivamente o **nome** do serviço, a **categoria** do serviço e a **ontologia** do serviço. Consultas realizadas em cima dessa ontologia permitem ao componente Seletor Semântico (ver Seção 5.3) construir uma base de conhecimento a partir da qual serão feitas inferências sobre os serviços que atendem a especificação de um determinado *workflow* semântico. Os métodos oferecidos pela classe ServicoOWL são descritos logo em seguida na Tabela 5.2.

```
<owl:Class rdf:ID="Servico">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#nome"/>
      <owl:cardinality rdf:datatype="&xsd;int">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#categoria"/>
      <owl:cardinality rdf:datatype="&xsd;int">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#ontologia"/>
      <owl:cardinality rdf:datatype="&xsd;int">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:DatatypeProperty rdf:ID="nome">
  <rdfs:domain rdf:resource="#Servico"/>
  <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="categoria">
  <rdfs:domain rdf:resource="#Servico"/>
  <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="ontologia">
  <rdfs:domain rdf:resource="#Servico"/>
  <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
```

Figura 5.2. Ontologia Servicos.owl

Tabela 5.2. Métodos da classe ServicoOWL

`static void`	`abrir()` Carrega a ontologia *Servicos.owl*
`static void`	`apagar(ServicoSemantico servico)` Remove o serviço descrito na ontologia *Servicos.owl*
`static void`	`criar()` Cria a ontologia *Servicos.owl*
`static void`	`commit()` Grava a ontologia *Servicos.owl*
`static boolean`	`existir(String uri)` Verifica se um serviço existe na ontologia *Servicos.owl*
`static ServicoSemantico`	`Recuperar(String uri)` Recupera o serviço descrito na ontologia *Servicos.owl*
`static List`	`recuperarTodos()` Recupera todos os serviços da ontologia *Servicos.owl*
`static void`	`salvar(ServicoSemantico servico)` Adiciona um serviço na ontologia *Servicos.owl*

A classe **ConceitoOWL** oferece métodos voltados para a manutenção das ontologias de Objeto e Tarefa, e em especial para a classificação das ontologias dos serviços *Web* importadas no WebFlowAH pelo usuário administrador de serviços. Esses métodos são descritos sumariamente na Tabela 5.3 e foram implementados utilizando-se as interfaces

OWLOntology e *OntModel* disponibilizadas respectivamente pelas ferramentas OWL-S API e *Jena*.

Tabela 5.3. Métodos da classe ConceitoOWL

`static void`	`abrir()` Carrega a ontologia *Conceitos.owl*
`static void`	`apagar(Conceito conceito)` Remove o conceito descrito na ontologia *Conceitos.owl*
`static void`	`associar(Conceito conceito)` Cria uma associação entre conceitos
`static void`	`classificar(Conceito conceito)` Classifica um conceito em Objeto ou Tarefa
`static void`	`criar()` Cria a ontologia *Conceitos.owl*
`static void`	`commit()` Grava a ontologia *Conceitos.owl*
`static Conceito`	`recuperar(URI uri)` Recupera o conceito descrito na ontologia *Conceitos.owl*
`static List`	`recuperarEfeitos(Conceito tarefa, Conceito objeto)` Recupera os conceitos associados ao conceito do tipo Tarefa pela propriedade *geradoPor*
`static List`	`recuperarEntradas(Conceito tarefa, Conceito objeto)` Recupera os conceitos associados ao conceito do tipo Tarefa pela propriedade *usadoPor*
`static List`	`recuperarObjetos(Conceito tarefa)` Recupera os conceitos do tipo Objeto associados com o conceito do tipo Tarefa pela propriedade *envolve*
`static List`	`recuperarPreCondicoes(Conceito tarefa, Conceito objeto)` Recupera os conceitos associados ao conceito do tipo Tarefa pela propriedade *exigidoPor*
`static List`	`recuperarSaidas(Conceito tarefa, Conceito objeto)` Recupera os conceitos associados ao conceito do tipo Tarefa pela propriedade *criadoPor*
`static List`	`recuperarTarefas()` Recupera os conceitos do tipo Tarefa da ontologia *Conceitos.owl*
`static Conceito`	`recuperarTipo(URI uri)` Recupera a classificação de um dado conceito, ou seja, Objeto ou Tarefa
`static List`	`recuperarTodos(String uri)` Recupera todos os conceitos da ontologia *Conceitos.owl*

Vistas sob a ótica da implementação, as ontologias de domínio comum do WebFlowAH são na realidade classes expressas na linguagem OWL que estão armazenadas na ontologia *Conceitos.owl*, conforme ilustra a Figura 5.3. A classe *rdf:ID="Objeto"* representa um objeto de domínio e a classe *rdf:ID="Tarefa"* representa uma tarefa de domínio. Uma tarefa pode estar associada a um ou mais objetos por meio da sua propriedade *rdf:ID="envolve"*. Já um objeto pode estar associado de quatro maneiras a uma tarefa. A propriedade *rdf:ID="usadoPor"* indica que o objeto é uma entrada usada pela tarefa. A

propriedade *rdf:ID="exigidoPor"* indica que o objeto é uma precondição exigida pela tarefa. A propriedade *rdf:ID="criadoPor"* indica que o objeto é uma saída produzida pela tarefa. E finalmente a propriedade *rdf:ID="geradoPor"* indica que o objeto é um efeito gerado pela tarefa. Dessa maneira, é possível inferir tarefas que envolvem um objeto específico ou quais objetos são usados, exigidos, criados ou gerados por uma determinada tarefa. Essa associação garante o funcionamento do assistente disponibilizado pelo WebFlowAH para auxiliar o especificador de processos na definição dos objetivos de um *workflow* semântico.

```
<owl:Class rdf:ID="Object">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="usadoPor">
          <rdfs:domain rdf:resource="#Objeto"/>
          <rdfs:range rdf:resource="#Tarefa"/>
        </owl:ObjectProperty>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:ID="Imovel">
  <rdfs:subClassOf rdf:resource="#Objeto"/>
</owl:Class>

<obj:Imovel rdf:about="OntologiaB#Casa"/>
  <obj:usadoPor rdf:resource="OntologiaB#ComprarCasa"/>
</obj:Imovel>
```

```
<owl:Class rdf:ID="Tarefa">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="envolve">
          <rdfs:domain rdf:resource="#Tarefa"/>
          <rdfs:range rdf:resource="#Objeto"/>
        </owl:ObjectProperty>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:ID="Comprar">
  <rdfs:subClassOf rdf:resource="#Tarefa"/>
</owl:Class>

<task:Comprar rdf:about="OntologiaB#ComprarCasa"/>
  <task:envolve rdf:resource="OntologiaB#Casa"/>
</task:Comprar>
```

Figura 5.3. Definição das ontologias de Objeto e Tarefa.

A Figura 5.3 também apresenta um exemplo de classificação das ontologias de serviços *Web* nas ontologias de domínio comum. As classes *rdf:ID="Imovel"* (derivada de Objeto) e *rdf:ID="Comprar"* (derivada de Tarefa) representam de forma abstrata os conceitos *Imóvel* e *Comprar*, respectivamente. Já a instância *<obj:Imovel rdf:about="OntologiaB #Casa">* classifica o conceito *Casa* no objeto *Imóvel*, enquanto que a instância *<tar:Comprar rdf:about="OntologiaB#ComprarCasa"/>* classifica o conceito *ComprarCasa* na tarefa *Comprar*. Essas instâncias estão associadas através da propriedade *rdf:ID="envolve"*.

A última classe do Repositório Semântico, a **WorkflowOWLS**, é responsável por salvar e recuperar os *workflows* semânticos criados pelo usuário especificador de processos no ambiente WebFlowAH. Na Tabela 5.4 são listados os métodos dessa classe com suas respectivas assinaturas, os quais são implementados utilizando-se a interface *Service* disponibilizada pela OWL-S API.

Tabela 5.4. Métodos da classe WorkflowOWLS

`static WorkflowSemantico`	`recuperar(File file)` Recupera a especificação de um *workflow* semântico
`static void`	`salvar(WorkflowSemantico arg)` Salva a especificação de um workflow semântico

Um *workflow* semântico é representado fisicamente no WebFlowAH por uma ontologia OWL-S, a qual é manipulada pela classe WorkflowOWLS. Essa ontologia contém apenas o *Process Model* como elemento de descrição semântica, que descreve o funcionamento do processo de negócio desejado através de uma instância da classe *CompositeProcess*. O conjunto de passos realizados por essa instância, descritos na respectiva ontologia pela propriedade *Perform*, corresponde aos objetivos de negócio do *workflow* semântico especificado. Cada objetivo é descrito nessa mesma ontologia por uma instância da classe *AtomicProcess* contendo as entradas, saídas, precondições e efeitos requeridos pelo processo de negócio, sendo todos descritos por conceitos do tipo Objeto. Um objetivo de negócio também possui duas variáveis locais (instâncias da classe *Local*) que representam a tupla <*Tarefa*, *Objeto*> definida para esse objetivo pelo especificador de processos. A Figura 5.4 ilustra parte da ontologia de um *workflow* semântico especificado para o processo de reserva de passagens aéreas.

```
<process:CompositeProcess rdf:ID="WorkflowSemantico">
  <process:composedOf>
    <process:Perform>
      <process:process rdf:resource="#Objetivo_1"/>
    </process:Perform>
  </process:composedOf>
</process:CompositeProcess>
```

Figura 5.4. *Workflow* semântico do processo de reserva de passagens aéreas.

Segundo a Figura 5.4, o processo descrito na ontologia do *workflow* semântico é formado por um único objetivo sem necessariamente referenciar a descrição WSDL de um serviço *Web* que implementa esse processo, ao contrário de uma definição de processo de um WMS na linguagem BPEL onde existe uma referência explícita a uma porta de um dado serviço. Portanto, pode-se afirmar que no WebFlowAH um *workflow* semântico é um processo abstrato contendo um conjunto de objetivos que são implementadas por um serviço *Web* desconhecido a priori. Esse nível de abstração permite que o *workflow* semântico não dependa totalmente de um serviço *Web* específico, ou seja, isola a lógica de negócio da implementação. Na Figura 5.5 é ilustrada a especificação do objetivo <*Reservar*, *Voo*> em complemento à ontologia do *workflow* semântico vista anteriormente. Percebe-se que assim como Figura 5.4 não há referência alguma a qualquer serviço *Web*, e sim, a conceitos das ontologias de domínio comum.

```
<process:AtomicProcess rdf:ID="Objetivo_1">
  <process:hasLocal>
    <process:Local rdf:ID="Tarefa_2">
      <process:parameterType
         rdf:datatype="&xsd;anyRI">Conceitos.owl#Reservar</process:parameterType>
    </process:Local>
  </process:hasLocal>
  <process:hasLocal>
    <process:Local rdf:ID="Objeto_3">
      <process:parameterType
         rdf:datatype="&xsd;anyRI">Conceitos.owl#Voo</process:parameterType>
    </process:Local>
  </process:hasLocal>
  <process:hasInput>
    <process:Input rdf:ID="Input_4">
      <process:parameterType
         rdf:datatype="&xsd;anyRI">Conceitos.owl#Origem</process:parameterType>
    </process:Input>
  </process:hasInput>
  <process:hasInput>
    <process:Input rdf:ID="Input_5">
      <process:parameterType
         rdf:datatype="&xsd;anyRI">Conceitos.owl#Destino</process:parameterType>
    </process:Input>
  </process:hasInput>
  <process:hasInput>
    <process:Input rdf:ID="Input_6">
      <process:parameterType
         rdf:datatype="&xsd;anyRI">Conceitos.owl#Partida</process:parameterType>
    </process:Input>
  </process:hasInput>
  <process:hasInput>
    <process:Input rdf:ID="Input_7">
      <process:parameterType
         rdf:datatype="&xsd;anyRI">Conceitos.owl#Retorno</process:parameterType>
    </process:Input>
  </process:hasInput>
  <process:hasOutput>
    <process:Output rdf:ID="Output_8">
      <process:parameterType
         rdf:datatype="&xsd;anyRI">Conceitos.owl#Localizador</process:parameterType>
    </process:Output>
  </process:hasOutput>
</process:AtomicProcess>
```

Figura 5.5. Objetivo <*Reservar, Voo*> do processo de reserva de passagens aéreas.

5.2 COMPOSITOR SEMÂNTICO

O componente Compositor Semântico foi implementado a partir dos trabalhos propostos por Costa, Pires e Mattoso (2004a, 2004b). Apesar de existirem outros trabalhos voltados para a composição dinâmica de serviços *Web* (MEDJAHED; BOUGUETTAYA; ELMAGARMID, 2003; PATHAK *et al.*, 2006; TRAINOTTI *et al.*, 2005), não foi possível no primeiro momento avaliar todas as soluções disponíveis. Portanto, adotou-se a idéia de reusar e melhorar algo já existente. A adaptação de outros algoritmos de composição pode ser realizada em trabalhos futuros onde o WebFlowAH servirá de base para a realização de testes

comparativos. Prevendo a possibilidade de incorporar outros algoritmos, foi projetada uma interface na arquitetura conceitual deste ambiente (ver Figura 4.4), a **ICompositorSemantico**, que estabelece um contrato para o desenvolvimento de novas classes para o Compositor Semântico. Por simplicidade, resolvemos nesta primeira versão do WebFlowAH implementar apenas a classe *Java* **CompositorSemantico** para representar o componente Compositor Semântico, cujos métodos são descritos na Tabela 5.5.

Tabela 5.5. Métodos da classe CompositorSemantico

`List<List<Operacao>>`	`compor(List<URI> entradas, List<URI> saidas, List<Objetivo> objetivos, List<URI> descartes)` Retorna as composições para um dado conjunto de entradas, saídas, objetivos e descartes (saídas não produzidas pelo *workflow* semântico)
`void`	`combinarServicos()` Realiza a combinação dos serviços selecionados para gerar as composições
`void`	`ordenarComposicao()` Ordena os serviços de uma composição com base nos objetivos do *workflow* semântico
`void`	`selecionarServicos()` Seleciona os serviços que implementam o *workflow* semântico

O algoritmo de composição implementado no Compositor Semântico consiste basicamente de três etapas. Na primeira etapa, o método **selecionarServicos** é chamado, onde são selecionados os serviços *Web* semânticos para implementar um dado *workflow* especificado no WebFlowAH, os quais são descobertos pelo componente Seletor Semântico a partir das entradas, saídas e objetivos do *workflow*, caso esses serviços existam no Repositório Semântico (ver Seção 5.1). Na segunda etapa, o método **combinarServicos** é chamado, onde os serviços que foram selecionados anteriormente são combinados de modo a consumir as entradas e a produzir as saídas definidas nos objetivos de negócio, além de atender as precondições e os efeitos dos mesmos. Na última etapa, onde o método **ordenarComposicao** é chamado, cada composição gerada pelo Compositor Semântico é ordenada segundo a seqüência dos objetivos definidos no *workflow* semântico.

Mesmo seguindo a lógica proposta por Costa, Pires e Mattoso (2004a), algumas adaptações foram feitas no algoritmo elaborado pelos autores. Uma delas é que não está sendo criado um grafo com todas as composições geradas pelo Compositor Semântico para que seja implementado um único plano de execução. Isto é, se uma composição envolve a chamada dos serviços **A**, **B** e **C** nesta ordem e se o serviço **D** oferece a mesma funcionalidade de **B**, poderia ser criado um caminho de execução alternativo envolvendo os serviços **A**, **D** e **C** no

mesmo plano de execução. Ao invés disso, resolvemos gerar esses caminhos alternativos em planos separados. Outra adaptação foi implementar como critério de otimização desses caminhos de execução o número de serviços combinados, fazendo com que o componente Executor de *Workflows* (ver Seção 5.4) selecione de preferência um plano de execução que tiver o menor número de serviços combinados toda vez que for solicitado a execução de um *workflow* semântico no WebFlowAH. Tais adaptações visaram principalmente a simplicidade no desenvolvimento do Compositor Semântico neste primeiro momento, deixando para trabalhos futuros a implementação completa do algoritmo em questão.

5.3 SELETOR SEMÂNTICO

O componente Seletor Semântico é responsável por descobrir serviços *Web* no WebFlowAH. Tipicamente, esse componente representa um agente de busca de serviços – mais conhecido por *matchmaker* – que permite a consulta e a localização de serviços *Web* a partir das suas descrições semânticas armazenadas no Repositório Semântico. Existem alguns trabalhos (KIEFER; BERNSTEIN, 2008; KLUSCH; FRIES; SYCARA, 2006; SIRIN; HENDER; PARSIA, 2003) que propõem algoritmos de *matching* e que até produziram versões desses agentes de busca. Entretanto, neste trabalho optou-se por desenvolver um algoritmo próprio ao invés de usar um desses trabalhos referenciados, devido ao uso da classificação das ontologias de serviço nas ontologias de Objeto e Tarefa, ficando a realização de uma análise comparativa de propostas nessa linha de pesquisa para um trabalho futuro.

```
SELECT ?servico
WHERE
  ( ?servico rdf:type process:Process )
  ( ?tarefa rdf:type conceito:Reservar )
  ( ?objeto rdf:type conceito:Passagem )
  ( ?tarefa conceito:envolve ?objeto )
AND
  ( ?servico = ?tarefa )
USING
  process FOR <http://www.daml.org/services/owl-s/1.1/Process.owl#>
  conceito FOR <http://localhost/webflowah/1.0/Conceitos.owl#>
```

Figura 5.6. Consulta semântica que retorna os serviços para o objetivo <*Reservar, Voo*>.

Implementado pela classe *Java* **SeletorSemantico**, o Seletor Semântico utiliza o raciocinador *Pellet* (CLARK & PARSIA LLC, 2008) para a realização de inferências sobre as ontologias dos serviços *Web* disponíveis no WebFlowAH, descobrindo os serviços que são

adequados a implementar um processo de negócio que foi especificado por um *workflow* semântico. Um exemplo de inferência realizada pela classe em questão é ilustrado na Figura 5.6 que apresenta uma consulta expressa em RDQL para descobrir os serviços (variável *servico* que é uma instância de *Process*) que implementam o objetivo *<Reservar, Voo>* (variáveis *tarefa* e *objeto*, cujo relacionamento é definido por *conceito:envolve*).

Os métodos da classe SeletorSemantico, mostrados na Tabela 5.6, realizam tipos específicos de inferência. Esses métodos são chamados pela classe CompositorSemantico a medida que esta realiza a composição dos serviços para um dado *workflow* semântico. Inicialmente, as inferências desses métodos eram expressas em RDQL, mas devido ao baixo desempenho do *Pellet*, optou-se por utilizar classes da própria OWL-S API para fazer as verificações de equivalência e classificação entre os conceitos descritos nas ontologias. Apesar desse problema de desempenho, segundo Sirin *et al.* (2005) na comparação feita entre alguns raciocinadores existentes, esse mesmo raciocinador demonstrou ser mais expressivo quanto à validação das descrições semânticas em OWL-S dos serviços *Web*.

Tabela 5.6. Métodos da classe SeletorSemantico

`Set<Objetivo>`	`getObjetivoPorServico(String[] list)` Retorna os objetivos implementados por um conjunto de serviços *Web* semânticos
`Set<Operacao>`	`getOperacaoPorEfeito(Operacao[] args, URI uri)` Retorna a partir de um conjunto de serviços *Web* semânticos aqueles que produzem um dado efeito
`Set<Operacao>`	`getOperacaoPorEntrada(Operacao[] args, URI uri)` Retorna a partir de um conjunto de serviços *Web* semânticos aqueles que usam uma dada entrada
`Set<Operacao>`	`getOperacaoPorEntrada(URI uri)` Retorna os serviços *Web* semânticos que usam uma dada entrada
`Set<Operacao>`	`getOperacaoPorPrecondicao(Operacao[] args, URI uri)` Retorna a partir de um conjunto de serviços *Web* semânticos aqueles que exigem uma dada precondição
`Set<Operacao>`	`getOperacaoPorObjetivo(Operacao[] args, Objetivo obj)` Retorna a partir de um conjunto de serviços *Web* semânticos aqueles que implementam um dado objetivo
`Set<Operacao>`	`getOperacaoPorObjetivo(List list)` Retorna os serviços *Web* semânticos que implementam um conjunto de objetivos
`Set<Operacao>`	`getOperacaoPorSaida(Operacao[] args, URI uri)` Retorna a partir de um conjunto de serviços *Web* semânticos aqueles que produzem uma dada saída
`Set<Operacao>`	`getOperacaoPorSaida(URI uri)` Retorna os serviços *Web* semânticos que produzem uma dada saída
`void`	`selectOperacao()` Carrega as ontologias dos serviços *Web* semânticos a partir da ontologia *Serviços.owl*

5.4 EXECUTOR DE WORKFLOWS

O componente Executor de *Workflows* é responsável pela execução e monitoração dos planos de execução no WebFlowAH e foi implementado pela classe *Java* **ExecutorWorkflow**. A OWL-S API também foi utilizada no desenvolvimento desse componente, pois ela provê uma máquina de execução de serviços *Web* semânticos, semelhante a uma máquina de *workflow* de um sistema de w*orkflow*.

```
<process:CompositeProcess rdf:ID="PlanoExecucao">
  <process:composedOf>
    <process:Perform rdf:ID="Perform_1"/>
      <process:process rdf:resource="CompanhiaAerea.owl#ReservarVooProcess"/>
    </process:Perform>
  </process:composedOf>
  <process:hasInput>
    <process:Input rdf:ID="Input_2">
      <process:parameterType rdf:datatype="&xsd;anyRI">Turismo.owl#Cidade</process:parameterType>
    </process:Input>
  </process:hasInput>
  <process:hasInput>
    <process:Input rdf:ID="Input_3">
      <process:parameterType rdf:datatype="&xsd;anyRI">Turismo.owl#Cidade</process:parameterType>
    </process:Input>
  </process:hasInput>
  <process:hasInput>
    <process:Input rdf:ID="Input_4">
      <process:parameterType rdf:datatype="&xsd;anyRI">XMLSchema#string</process:parameterType>
    </process:Input>
  </process:hasInput>
  <process:hasInput>
    <process:Input rdf:ID="Input_5">
      <process:parameterType rdf:datatype="&xsd;anyRI">XMLSchema#string</process:parameterType>
    </process:Input>
  </process:hasInput>
  <process:hasOutput>
    <process:Output rdf:ID="Output_6">
      <process:parameterType rdf:datatype="&xsd;anyRI">XMLSchema#string</process:parameterType>
    </process:Output>
  </process:hasOutput>
  <process:hasResult>
    <process:Result rdf:ID="Result_7">
      <process:withOutput>
        <process:outputBinding>
          <process:toParam rdf:resource="#Output_6"/>
          <process:valueSource>
            <process:valueOf>
              <process:fromProcess rdf:resource="#Perform_1"/>
              <process:theVar rdf:resource="CompanhiaAerea.owl#Localizador"/>
            <process:valueOf>
          </process:valueSource>
        </process:outputBinding>
      </process:withOutput>
    </process:Result>
  </process:hasResult>
</process:CompositeProcess>
```

Figura 5.7. Plano de execução do processo de reserva de passagens aéreas.

Entretanto, antes de ser executado pela classe ExecutorWorkflow, o plano de execução estático é gerado pela classe *Java* **PlanoExecucao** a partir de uma determinada composição produzida pelo Compositor Semântico junto com a especificação do *workflow* semântico. Essa classe é responsável por construir uma ontologia OWL-S para um determinado plano de

execução. Essa ontologia contém o elemento descritivo *Process Model* para descrever o funcionamento do *workflow* semântico, o qual é definido como uma instância da classe *CompositeProcess*, só que fazendo referência a um conjunto de serviços *Web* no lugar dos objetivos. Dessa forma, pode-se dizer que o plano de execução é um processo concreto cujos passos descritos pela propriedade *Perform* são serviços *Web* semânticos. A Figura 5.7 ilustra a ontologia de um plano de execução para o *workflow* semântico exemplificado anteriormente, cujo serviço *Web* semântico selecionado é definido por *ReservarVooProcess*.

Na implementação da classe ExecutorWorkflow foi necessário realizar algumas correções na OWL-S API para a execução correta de um plano de execução. Mesmo sabendo que existem limitações nessa API que não permitem a execução de *workflows* mais complexos, como por exemplo, calcular expressões lógicas envolvendo tipo de dados *XML Schema Definition*[27] (XSD), a decisão de continuar utilizando a OWL-S API foi por causa da ausência de ferramentas para a execução de serviços *Web* semânticos. Uma alternativa seria gerar o plano em outra linguagem de definição de processo como BPEL ou XPDL e submeter à execução de outras máquinas de *workflow* existentes. Outro detalhe a ser dito a respeito do componente Executor de *Workflows* é que este monitora a execução do plano instanciado e caso ocorra alguma falha é criado um novo plano a partir de outra composição. Devido à OWL-S API não suportar um tratamento de falhas na execução de um serviço *Web* semântico, optou-se por instanciar outro plano para que seja executado desde o início. Um aspecto importante a ser comentando é que o cancelamento da execução de um plano não leva em conta o que foi já efetivado por alguns serviços, isto é, se um plano de execução fez a invocação de alguns dos serviços no instante da falha, não se podem desfazer as operações realizadas anterior a essa falha.

[27] http://www.w3.org/XML/Schema

6 ESTUDO DE CASO

Este capítulo apresenta um estudo de caso elaborado com o intuito de comprovar a utilização do WebFlowAH na prática e validar as funcionalidades providas por este ambiente frente aos requisitos previamente descritos na Seção 1.1, os quais são necessários para se permitir a especificação e execução de processos de negócios com base nas descrições semânticas dos serviços *Web*. Tal estudo consistiu em: (i) especificar um dado processo de negócio do mundo real no ambiente proposto, mais precisamente um processo de reserva de hospedagens, analisando-se a capacidade de representá-lo tanto parcialmente (somente objetivos de negócio) quanto inteiramente (com todas as atividades); e (ii) executar a especificação desse processo observando-se os planos de execução gerados com a descoberta e composição dinâmica dos serviços.

A partir de uma série de casos de teste, foram feitas comparações dos resultados obtidos pelo WebFlowAH e algumas ferramentas de composição de serviços *Web* encontradas na literatura como forma de se destacar sua potencialidade perante os demais trabalhos existentes. As ferramentas *Composer* (SIRIN; HENDER; PARSIA, 2003) e *WebComposer* (COSTA; PIRES; MATTOSO, 2004b) foram as escolhidas para a realização dos experimentos por apresentarem um ambiente que permite a especificação de um plano de composição a partir das ontologias de serviços *Web* e a execução desse mesmo plano fazendo a descoberta e composição de forma (semi)automática.

A Seção 6.1 deste capítulo apresenta os serviços *Web* que foram utilizados no estudo de caso. Na Seção 6.2 são descritos os casos de teste elaborados para o estudo de caso enquanto que na Seção 6.3 é apresentada uma análise dos resultados que foram obtidos após a execução desses casos de teste.

6.1 SERVIÇOS WEB UTILIZADOS

Os serviços *Web* construídos para o estudo de caso tiveram suas funcionalidades descritas semanticamente através das ontologias OWL-S. A ferramenta *Protegé*[28], um editor de

[28] http://protege.stanford.edu/

ontologias *open source*, e o *plugin* OWL-S *Editor*[29] foram utilizados na confecção dessas ontologias.

Este trabalho contou com as seguintes ontologias de serviço: *HotelAlfa.owl*, *HotelBeta.owl*, *HotelDelta.owl*, *HotelGama.owl*, *HotelOmega.owl* e *HotelSigma.owl*, todos serviços de reserva de hospedagem; e b) *LocadoraUm.owl* e *LocadoraDois.owl*, ambos serviços de reserva de carro. Na Tabela 6.1 são descritas em resumo as funcionalidades providas em cada um desses serviços.

Tabela 6.1. Funcionalidades do serviços *Web* criados no estudo de caso

HotelAlfa	
`HoteisAchados`	`PesquisarHotel(Destino, Categoria)` Pesquisa por hotéis de uma dada categoria que existem em um destino específico
`QuartosAchados`	`PesquisarQuarto(HotelDesejado, DataChegada, DataSaida, TipoAcomodacao)` Pesquisa por quartos do hotel desejado com base no tipo de acomodação e que estão disponíveis em um dado período
`QuartosAchados`	`ReservarQuarto(QuartoDesejado, DataChegada, DataSaida, Hospede)` Reserva o quarto desejado para um hóspede em um dado período
`HotelSelecionado`	`SelecionarHotel(HoteisExistentes)` Seleciona um hotel dentre um conjunto hotéis existentes
`QuartoSelecionado`	`SelecionarQuarto(QuartosExistentes)` Seleciona um quarto dentre um conjunto de quartos existentes
HotelBeta	
`HotelEscolhido`	`EscolherHotel(ListaHoteis, TipoHotel)` Seleciona um hotel dentre um conjunto hotéis existentes com base no tipo de hotel desejado
`QuartoEscolhido`	`EscolherQuarto(ListaQuartos, TipoQuarto)` Seleciona um quarto dentre um conjunto de quartos existentes com base no tipo de quarto desejado
`HoteisExistentes`	`ListarHotel(Cidade)` Lista os hoteis existentes em uma dada cidade
`QuartosExistentes`	`ListarQuarto(HotelPreferido, DataChegada, DataSaida)` Lista os quartos disponíveis do hotel preferido em um dado período
`ReservaQuarto`	`ReservarQuarto(QuartoPreferido, DataChegada, DataSaida, Cliente)` Reserva um quarto preferido para um cliente em um dado período

[29] http://owlseditor.semwebcentral.org/

HotelDelta	
`HoteisEncontrados`	`EncontrarHotel(Local)` Encontra os hotéis existentes em um dado local
`QuartosEncontrados`	`EncontrarQuarto(Hoteis, DataChegada, DataSaida)` Encontra os quartos disponíveis em um período dentro de um conjunto de hotéis
`ReservaHospede`	`ReservarQuarto(Quartos, DataChegada, DataSaida, Hospede)` Reserva um quarto para um hóspede em um dado período de um conjunto de quartos
HotelGama	
`Hoteis`	`BuscarHotel(Cidade, Categoria)` Busca por hotéis de uma dada categoria que existem em uma cidade específica
`Quartos`	`BuscarQuarto(ListaHoteis, DataChegada, DataSaida, TipoAcomodacao)` Busca por quartos disponíveis em um período dentro de um conjunto de hotéis com base no tipo de acomodação
`ReservaHotel`	`ReservarQuarto(ListaQuartos, DataChegada, DataSaida, Hospede)` Reserva um quarto para um hóspede em um dado período dentro de um conjunto de quartos
HotelOmega	
`ReservaHospede`	`ReservarQuarto(Cidade, DataChegada, DataSaida, Hospede)` Reserva um quarto para um hóspede em um dado período em uma cidade específica
HotelSigma	
`ReservaQuarto`	`ReservarQuarto(Destino, TipoHotel, TipoQuarto, DataChegada, DataSaida, Cliente)` Reserva um quarto para um cliente em um dado período em um destino específico com base nos tipos de hotel e de quarto
LocadoraUm	
`ReservaCarro`	`ReservarCarro(Cidade, DataRetirada, DataDevolucao, TipoCarro)` Reserva um carro para um dado período em uma cidade específica com base no tipo de carro
LocadoraDois	
`CarroSelecionado`	`SelecionarCarro(Categoria)` Seleciona um carro de uma dada categoria
`ReservaCarro`	`ReservarCarro(Local, CarroPreferido, DataDevolucao, DataRetirada)` Reserva um carro para um dado período em um local específico

Conforme visto no Capítulo 4, antes de começarmos a especificação de um *workflow* semântico é preciso classificar no WebFlowAH os conceitos definidos pelas ontologias dos serviços mencionados anteriormente nas ontologias de Objeto e Tarefa do Repositório Semântico. O usuário administrador de serviços solicita, portanto, ao Gestor de *Workflows* que uma ontologia seja carregada a partir de um diretório de arquivos local. Essa solicitação é atendida pelo Gestor de Serviços que realiza também a validação dessa ontologia, ou seja, verifica a consistência do conteúdo semântico da mesma. A partir do *Profile* definido em cada ontologia desses serviços, são selecionadas as entradas, saídas, precondições e efeitos que serão classificados como Objeto. Da mesma forma, os processos definidos pelo *Process Model* dessas ontologias são escolhidos e depois classificados como Tarefa. Além disso, esses conceitos são associados através de suas respectivas propriedades definidas para cada ontologia de domínio comum (*envolve*, *usadoPor*, *exigidoPor*, *criadoPor* e *geradoPor*).

Para exemplificar essa etapa de classificação das ontologias no WebFlowAH, usaremos os serviços *ReservaQuartoAlfaService* e *ReservaQuartoOmegaService* que são descritos respectivamente pelas ontologias *HotelAlfa.owl* e *HotelOmega.owl*. O primeiro serviço é descrito por um *ReservaQuartoAlfaProfile* e por um *ReservarQuartoAlfaProcess*, onde os parâmetros necessários para efetuar uma reserva de quarto de hotel são as entradas *QuartoDesejado*, *DataChegada*, *DataSaida* e *Hospede* e as precondições *CartaoValido* e *QuartoDisponivel*, e os parâmetros produzidos após invocação dessa operação são a saída *ReservaHotel* e o efeito *ReservaConfirmada*. Já o segundo serviço é descrito por um *ReservaQuartoOmegaProfile* e por um *ReservaQuartoOmegaProcess* contendo as entradas *Cidade*, *Hospede*, *DataChegada* e *DataSaida*, e a saída *ReservaHotel*.

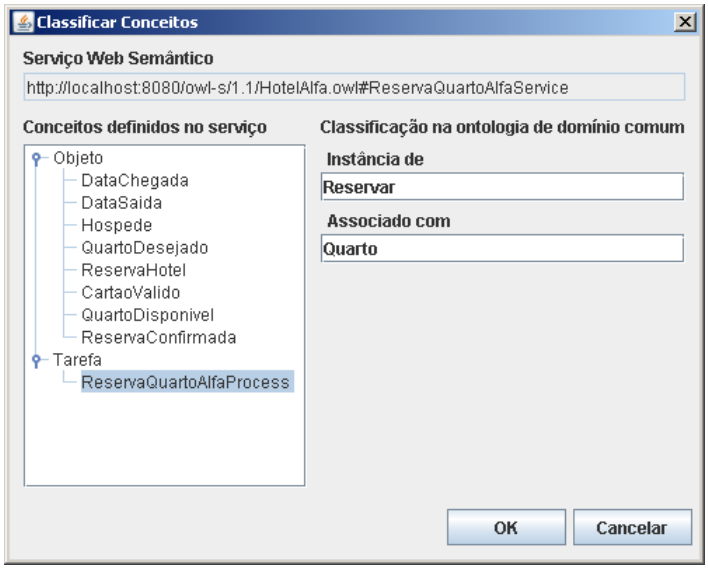


Figura 6.1. Tela Classificar Conceitos do WebFlowAH

Embora esses dois serviços sejam definidos de maneiras diferentes, eles representam o mesmo objetivo de negócio que é o de reservar um quarto de hotel. Esse objetivo é descrito no *Profile* de cada serviço via propriedades *serviceClassification* e *serviceProduct*, sendo que a primeira possui o conceito *Reservar* e a segunda o conceito *Quarto*. Na Figura 6.1 é exibida a tela **Classificar Conceitos** do WebFlowAH contendo a classificação de conceitos para o serviço *ReservaQuartoAlfaService*, cujo processo *ReservaQuartoAlfaProcess* é classificado como o conceito *Reservar* (classificado como Tarefa) e associado ao conceito *Quarto* (classificado como Objeto). Tal classificação aconteceria de modo semelhante com o serviço *ReservaQuartoOmegaService*.

Já os conceitos que representam as entradas, saídas, precondições e efeitos dos serviços *ReservaQuartoAlfaService* e *ReservaQuartoOmegaService* são classificados de maneira diferente dos conceitos de processo. Tanto as entradas quanto as saídas descritas no *Profile* contém uma propriedade *parameterType* que informa o tipo de dado que cada parâmetro representa. Caso um parâmetro seja definido como sendo um tipo de *XML Schema Definition* (XSD), por exemplo, *xsd:integer* ou *xsd:string*, então se cria um conceito Objeto com o mesmo nome desse parâmetro e em seguida classifica-se esse parâmetro no conceito criado. Isto é, se a entrada *DataChegada* do serviço *ReservaQuartoAlfaService* é do tipo *xsd:string* então o WebFlowAH cria o conceito *DataChegada* como Objeto e classifica essa entrada nesse mesmo conceito. Caso a propriedade *parameterType* seja um conceito definido por uma ontologia, então esse conceito é classificado como Objeto e o parâmetro classificado nesse novo conceito. As precondições e efeitos têm conceitos do tipo Objeto criados a partir de seus próprios nomes e depois classificadas nos respectivos conceitos.

No *Composer* essa classificação de conceitos não é necessária, pois ele utiliza a própria hierarquia definida pela ontologia OWL-S dos serviços *Web* (o *ProfileHierarchy*) para a especificação de um serviço composto. Nesse caso, a classe *Profile* do serviço *Web* é utilizada como critério de seleção para outros serviços, e a propriedade *parameterType* como critério de batimento para a composição. Já o *WebComposer*, não possui essa funcionalidade e parte do princípio que as ontologias de serviço foram descritas com os mesmos conceitos. O WebFlowAH possibilita através da classificação de conceitos nas ontologias de domínio comum a unificação dos conceitos definidos pelas ontologias de serviço criando um repositório para a especificação de alto nível de um processo de negócio.

6.2 CASOS DE TESTE

Os casos de teste definidos para comparar as funcionalidades do WebFlowAH com outras ferramentas de composição de serviços consistem basicamente em especificar *workflows* semânticos para um processo de reserva de hospedagens e executar os planos de execução gerados pelo ambiente a partir dos mecanismos de descoberta e composição dinâmicas. Porém, para permitir que esses testes sejam realizados no *Composer* e no *WebComposer*, buscou-se com o estudo de caso reduzir as diversas maneiras de se especificar um mesmo processo de negócio. Portanto, o foco principal nesses experimentos será comparar uma especificação parcial ou incompleta de um *workflow* semântico contendo, por exemplo, um simples objetivo de negócio, e outra especificação mais completa, onde todas as atividades do processo de negócio estão representadas no *workflow* semântico. A seguir temos uma breve descrição de cada um desses casos de teste e os resultados obtidos após a execução dos mesmos.

6.2.1 Definição dos Casos de Teste

O caso de teste **CST01** consiste em especificar um *workflow* semântico com um único objetivo de negócio contendo apenas os parâmetros de entrada informados. Usaremos para representar esse objetivo os conceitos *Reservar* e *Quarto*, e os conceitos *Cidade*, *TipoHotel*, *TipoQuarto*, *DataChegada*, *DataSaida* e *Hospede* para representar essas entradas.

No caso de teste **CST02**, são acrescentados os parâmetros de saídas ao *workflow* especificado no CST01. Dessa forma, foram usados os conceitos foram usados os conceitos *Hotel*, *Quarto* e *ReservaHotel*. para representar essas saídas.

Já o caso de teste **CST03** adiciona precondições ao *workflow* semântico do caso de teste anterior. Para representar tais precondições, foram usados os conceitos *CartaoValido* e *QuartoDisponivel*.

O *workflow* semântico do caso de teste **CST04** tem o mesmo objetivo dos casos antecessores, mas se diferencia por ter todos os parâmetros definidos (entradas, saídas, precondições e efeitos). Para tal, reutilizamos o CTS03 adicionando o conceito *ReservaConfirmada* como o efeito.

Para o caso de teste **CST05**, o *workflow* semântico tem dois objetivos especificados. Nesse caso, usamos os conceitos *Procurar* e *Hospedagem*, *Escolher* e *Quarto* para representar esses objetivos. Já os conceitos *Cidade*, *DataChegada*, *DataSaida* foram as entradas, e os conceitos *ListaHotel* e *Quarto* as saídas.

No caso de teste **CST06** foi necessário especificar um *workflow* semântico completo, onde as atividades do processo de reserva de hospedagens foram descritas por uma seqüência de objetivos. Somente os parâmetros de entrada e saída foram informados para esse *workflow*.

Finalmente, no caso de teste **CST07** especificou-se um *workflow* semântico de forma similar a especificação de um processo de negócio em um sistema de *workflow*, onde foram utilizados fluxos de controle típicos (ex.: fluxo condicional, fluxo de repetição) de uma linguagem de definição de processo.

Todos os conceitos mencionados nos casos de teste descritos anteriormente foram escolhidos com a ajuda de um assistente do ambiente WebFlowAH, que permite a realização de consultas sobre as ontologias de domínio comum.

6.2.2 Execução dos Casos de Teste

Seguindo a ordem de definição dos casos de teste apresentada neste trabalho, o primeiro experimento executado foi o CST01. Através de um assistente de construção de *workflows* disponibilizado pelo componente Interface com Usuário do ambiente WebFlowAH (ver Figura 6.2), foi feita a especificação do *workflow* semântico exigido por esse caso de teste. Depois, foi solicitada neste ambiente, via esse mesmo componente, a execução do *workflow* semântico especificado, ocasionando a descoberta e composição dinâmicas dos serviços existentes no Repositório Semântico. O resultado obtido no final desse experimento foi a descoberta dos serviços *HotelAlfa*, *HotelBeta*, *HotelDelta*, *HotelGama*, *HotelOmega* e *HotelSigma* realizada pelo Seletor Semântico, e a geração de um plano de execução para cada serviço feita pelo Gestor de *Workflows*, os quais são ilustrados a seguir nas Figuras 6.3, 6.4, 6.5 e 6.6. No caso do *Composer*, a especificação do serviço *Web* semântico (que corresponderia ao *workflow* semântico no WebFlowAH) foi realizada selecionando-se o *Profile* referente ao serviço de reserva de quartos. Como resultado foi descoberto apenas o serviço *HotelAlfa* para implementar o serviço especificado. Já no *WebComposer*, o *workflow* foi especificado utilizando os conceitos definidos para representar os serviços de reserva de

quartos e seus respectivos parâmetros de entrada e saída. O resultado foi a composição dos serviços *HotelAlfa*, *HotelBeta*, *HotelDelta*, *HotelGama*, *HotelOmega* e *HotelSigma* além dos mesmos planos obtidos pelo WebFlowAH.

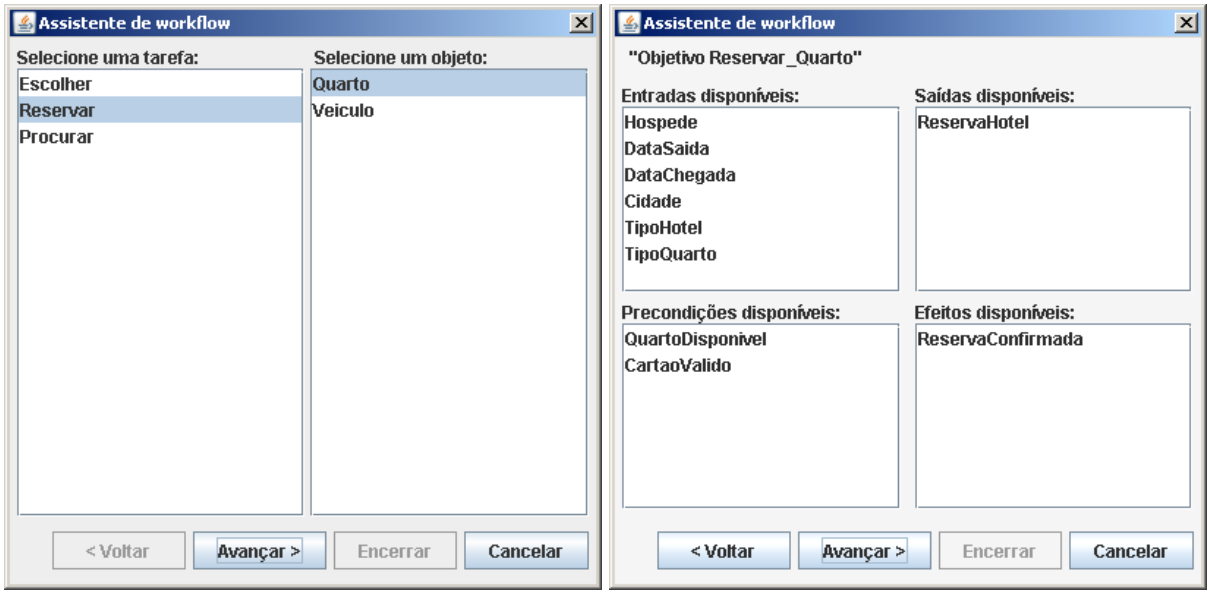


Figura 6.2. Tela Assistente de *workflow* do WebFlowAH

Os planos de execução PLAN1 e PLAN2 gerados pelo WebFlowAH no caso de teste CST01 utilizam respectivamente os serviços *HotelOmega* e *HotelSigma*, ambos serviços atômicos que provêem uma única operação. No PLAN1 a operação do serviço *HotelOmega* é descrita pelo conceito *ReservaQuartoOmegaProcess*, enquanto que no PLAN2 a operação do serviço *HotelSigma* é descrita pelo conceito *ReservaQuartoSigmaProcess*. A Figura 6.3 ilustra claramente a estrutura desses planos de execução.

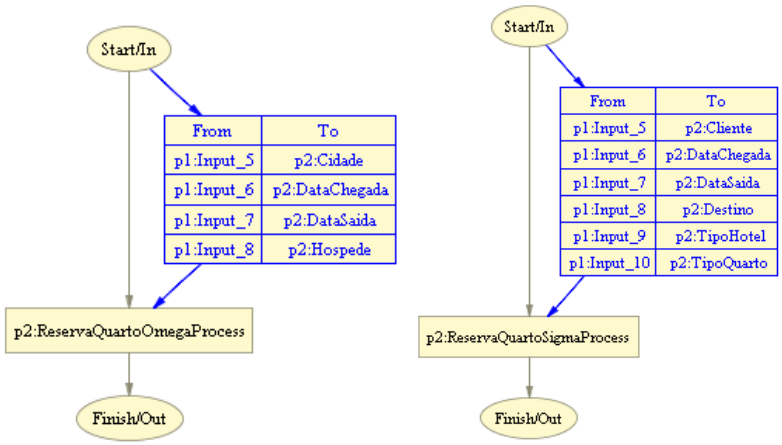


Figura 6.3. Planos de execução PLAN1 e PLAN2 do caso de teste CST01

O plano de execução PLAN3 do WebFlowAH implementa o *workflow* semântico do caso de teste CST01 através das operações do serviço *HotelDelta*, as quais são descritas pelos conceitos *EncontraHotelDeltaProcess*, *EncontraQuartoDeltaProcess* e

ReservaQuartoDeltaProcess. Já o plano PLAN4 do WebFlowAH implementa esse mesmo *workflow* com as operações do serviço *HotelGama*, que são descritas pelos conceitos *BuscaHotelGamaProcess*, *BuscaQuartoGamaProcess* e *ReservaQuartoGamaProcess*. Na Figura 6.4 tem-se uma representação gráfica desses planos.

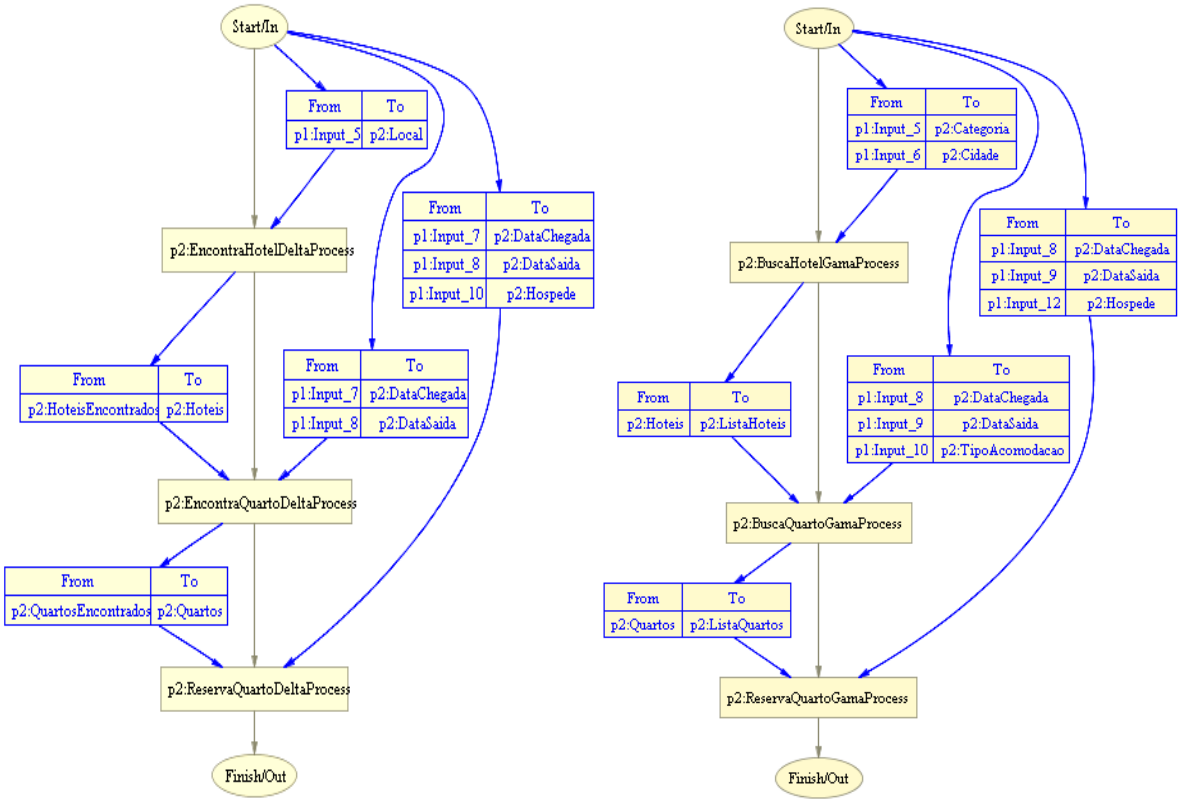


Figura 6.4. Planos de execução PLAN3 e PLAN4 do caso de teste CST01

O plano de execução PLAN5 gerado pelo WebFlowAH no CST01 é composto pelas operações do serviço *HotelAlfa*, conforme ilustra a Figura 6.5. Essas operações são descritas através dos conceitos *PesquisaHotelAlfaProcess*, *SelecionaHotelAlfaProcess*, *PesquisaQuartoAlfaProcess*, *SelecionaQuartoAlfaProcess* e *ReservaQuartoAlfaProcess*. Um detalhe interessante nesse plano é que a operação referente ao conceito *PesquisaHotelAlfaProcess* pode ser substituída pela operação do conceito *BuscaHotelGamaProcess* para se construir outro plano de execução, uma vez que ambos os conceitos têm o mesmo significado, isto é, semanticamente definem uma mesma funcionalidade (no caso, a funcionalidade de procurar por hotéis de uma categoria específica em uma determinada cidade), mas de serviços diferentes. Essa nova composição, gerada nos testes com a ferramenta *WebComposer*, é considerada incoerente, pois em termos de negócio não é possível reservar um quarto de um dado hotel pelo serviço de reserva de outro hotel. Portanto, no WebFlowAH, essa composição é descartada.

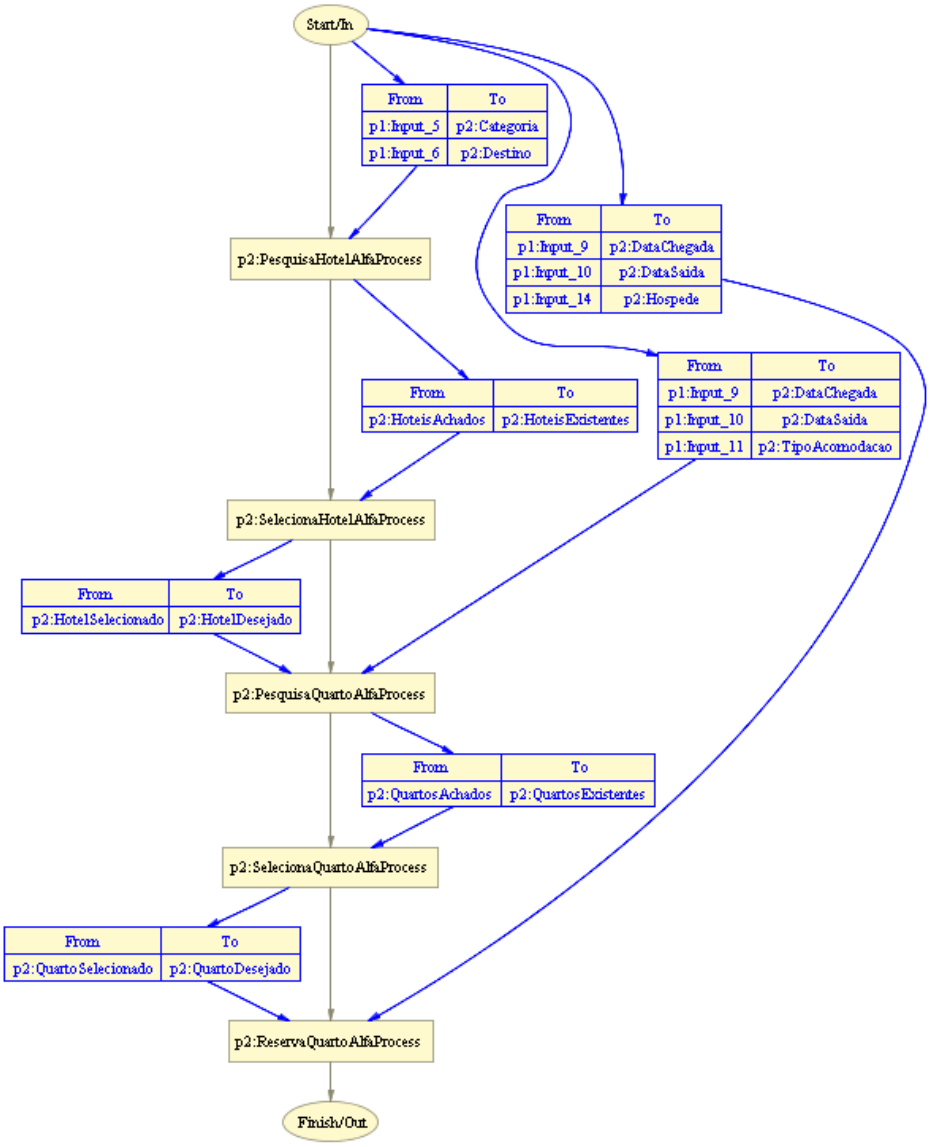


Figura 6.5. Plano de execução PLAN5 do caso de teste CST01

O último plano de execução gerado pelo WebFlowAH no CST01, o PLAN6, utiliza as operações do serviço *HotelBeta*, as quais são descritas pelos conceitos *ListaHotelBetaProcess*, *EscolheHotelBetaProcess*, *ListaQuartoBetaProcess*, *EscolheQuartoBetaProcess* e *ReservaQuartoBetaProcess* (ver Figura 6.6). Com base nesses resultados, pode-se afirmar que o ambiente WebFlowAH permite que os serviços sejam descobertos, selecionados, combinados e orquestrados a partir de uma especificação abstrata e incompleta, pois não foi necessário especificar todas as atividades do processo de reserva de hospedagens, apenas o objetivo de negócio <*Reservar*, *Quarto*>.

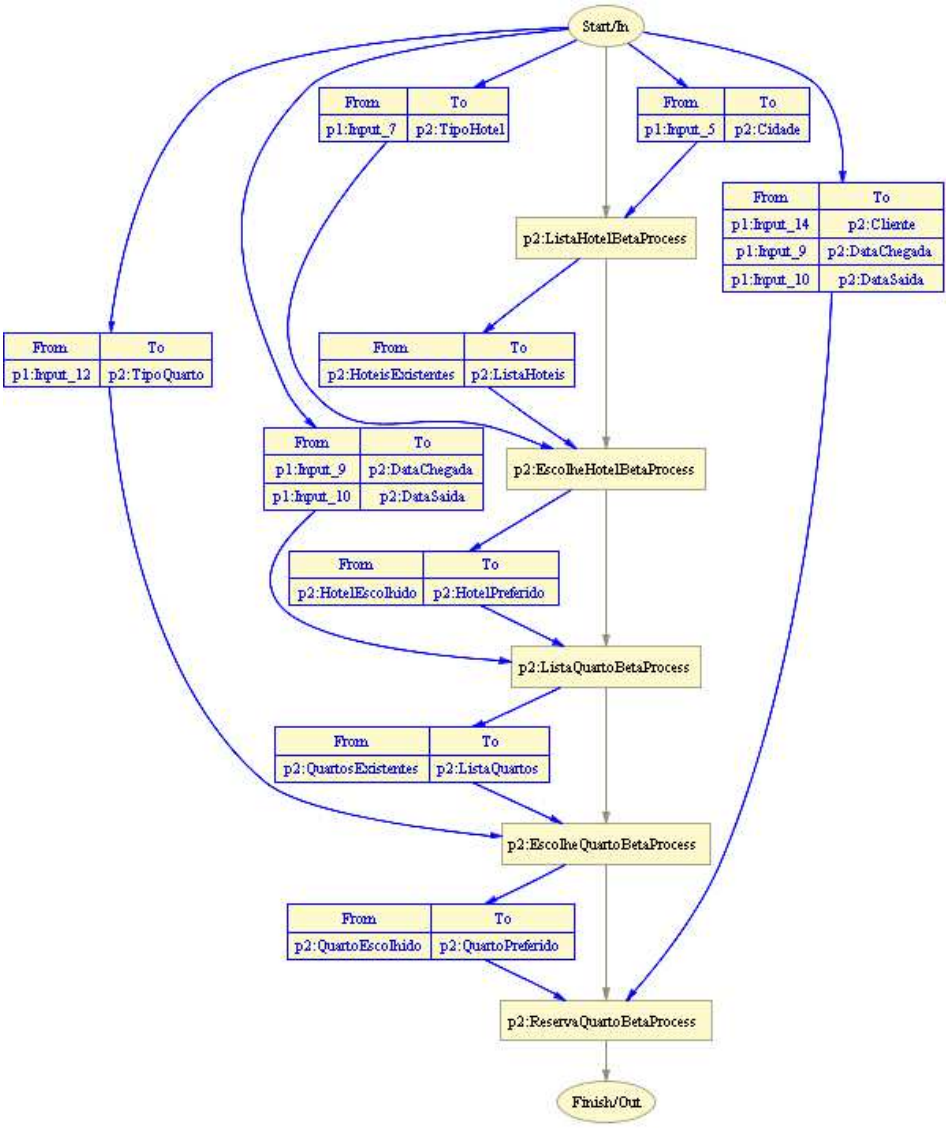


Figura 6.6. Plano de execução PLAN6 do caso de teste CST01

O próximo caso de teste, o CST02, teve como resultado no WebFlowAH a geração de dois planos de execução, que correspondem aos planos PLAN5 e PLAN6 vistos anteriormente. Embora o *workflow* semântico especificado nesse caso de teste seja semelhante ao CST01, o simples fato de terem sido acrescentados parâmetros de saída fizeram com que os serviços *HotelDelta*, *HotelGama*, *HotelOmega* e *HotelSigma* fossem descartados pelo Seletor Semântico. Tanto no *Composer* quanto no *WebComposer*, o resultado desse caso de teste foi idêntico ao que cada um gerou no CST01.

Nos casos de teste CST03 e CST04, onde foram adicionadas as precondições e os efeitos, o resultado no WebFlowAH foi ainda menor gerando-se apenas um único plano de execução (o mesmo PLAN5 do CST01), pois apenas o serviço *HotelAlfa* atendia aos requisitos exigidos pelo *workflow* semântico. Sendo assim, pode-se dizer que quanto mais detalhada é a especificação do processo de negócio no WebFlowAH, mais reduzido é o

conjunto de serviços *Web* selecionados pelo Seletor Semântico durante a composição dos serviços. Não foi possível realizar os casos de teste CST03 e CST04 no *Composer*, pois tal ferramenta não suporta a especificação de precondições e efeitos. Já o *WebComposer*, não gerou planos de composição em ambos os casos.

O caso de teste CST05 é um pouco diferente dos casos anteriores, pois exige que o *workflow* semântico tenha dois objetivos especificados. Nesse ponto, o assistente do Gestor de *Workflows* do WebFlowAH auxilia o usuário especificador de processos na escolha do objetivo de negócio subseqüente identificando os parâmetros de entrada e saída que favorecem a combinação de serviços para o *workflow* semântico em questão. Essa característica é comum em ambientes de composição semi-automática, permitindo a um usuário do ambiente descobrir quais objetivos de negócio podem ser utilizados para a especificação de um determinado *workflow* semântico. Como resultado, o WebFlowAH gerou para o caso CST05 os planos de execução PLAN7 e PLAN8 ilustrados pelas respectivas Figura 6.7 e 6.8. O primeiro plano utiliza as operações do serviço *HotelBeta* enquanto que o segundo plano utiliza as operações do serviço *HotelAlfa*. Novamente, não foi possível especificar um serviço semântico no *Composer* para o caso de teste CST05. Já o *WebComposer*, gerou planos de composição que foram obtidos pelo WebFlowAH, porém, apresentou o mesmo comportamento no caso de teste CST01, onde combinou serviços diferentes criando planos inválidos.

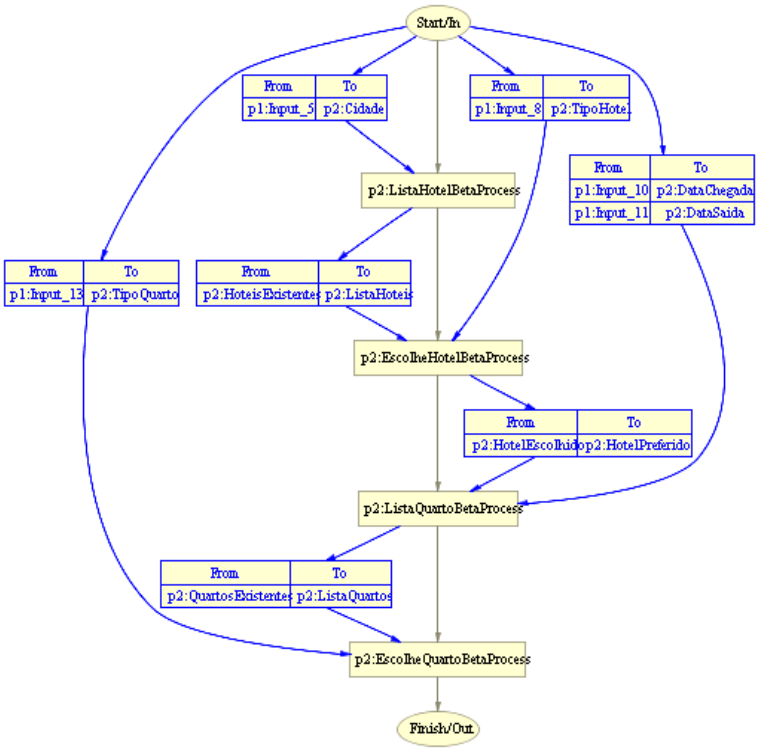


Figura 6.7. Plano de execução PLAN7 do caso de teste CST05

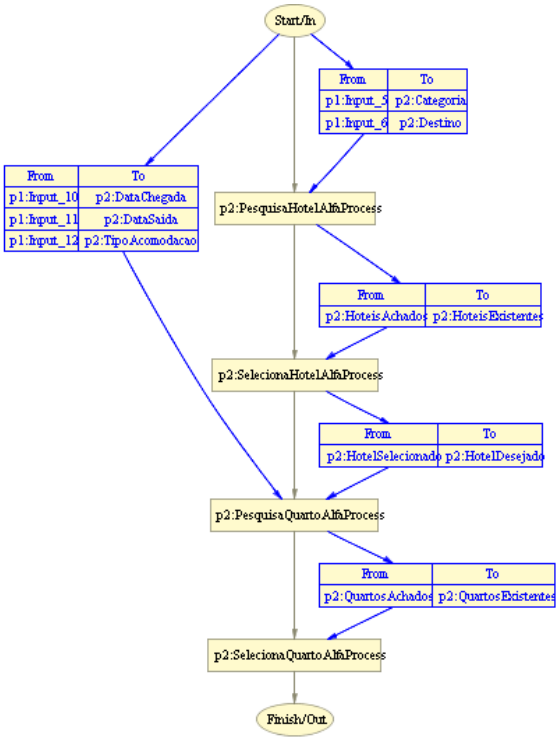


Figura 6.8. Plano de execução PLAN8 do caso de teste CST05

No caso de teste CST06, foi criada uma especificação completa de um *workflow* semântico, cujo processo de negócio descrito nessa especificação é formado pelos seguintes passos: 1) procurar um hotel em uma determinada cidade; 2) escolher um hotel a partir do resultado de tal busca; 3) procurar por quartos disponíveis no hotel preferido; 4) escolher um quarto disponível; e 5) reservar o quarto preferido. Esse tipo de especificação é muito comum nas atuais ferramentas de composição de serviços *Web* e exige por parte do usuário um conhecimento prévio das atividades do processo de negócio. No WebFlowAH, cada passo do respectivo processo de negócio foi mapeado para um objetivo do *workflow* contendo apenas os parâmetros de entrada e de saída informados. Como resultados da execução desse *workflow* foram gerados os mesmos planos PLAN5 e PLAN6 do CST01. Porém, o esforço exigido para especificar o *workflow* do CST06 é extremamente superior em relação ao CST04, que também apresentou o mesmo resultado. Tal experimento veio, portanto, comprovar a vantagem de uma especificação incompleta de um processo de negócio no lugar de uma mais detalhada, onde a partir de um simples objetivo de negócio descrito por conceitos obtidos pelas ontologias dos serviços é possível via mecanismos de descoberta e composição dinâmicas de serviços *Web* implementar esse processo. Entretanto, é preciso que as ontologias desses serviços sejam classificadas nas ontologias de domínio comum do WebFlowAH permitindo a realização de inferências sobre as ontologias dos serviços para suportar a especificação de *workflows* semânticos. No *Composer*, a especificação do serviço semântico

para ao CST06 contou com a ajuda da ferramenta para selecionar o próximo serviço a ser combinado, e como resultado, teve-se a geração de um único plano. O *WebComposer*, por sua vez, gerou os mesmos planos de composição para o caso de teste CST02.

No último caso de teste executado, o CST07, buscou-se realizar a especificação de um processo de negócio no WebFlowAH de forma análoga a um sistema de *workflow*. Nesse caso, foi criado um *workflow* semântico que atendesse as seguintes regras de negócio: se o hotel escolhido para a hospedagem oferece os serviços de café da manhã e de acesso à *Internet* então se deve efetuar a reserva do quarto, e se esse mesmo hotel possui estacionamento então providenciar a reserva de um carro. Para tal, foram utilizados fluxos de controle condicional para especificar essas regras. Ademais, todos os parâmetros listados pelo assistente de *workflow* o WebFlowAH foram informados no *workflow* semântico. O resultado obtido com a execução desse *workflow* foi a geração do plano PLAN9 (conforme Figura 6.9) que é composto pelos serviços *HotelAlfa* e *LocadoraDois*. O *Composer*, por não suportar fluxos de controle, não permitiu a especificação de um serviço semântico para tal situação. Já o *WebComposer*, gerou planos de composição inválidos além daqueles que foram gerados pelo WebFlowAH.

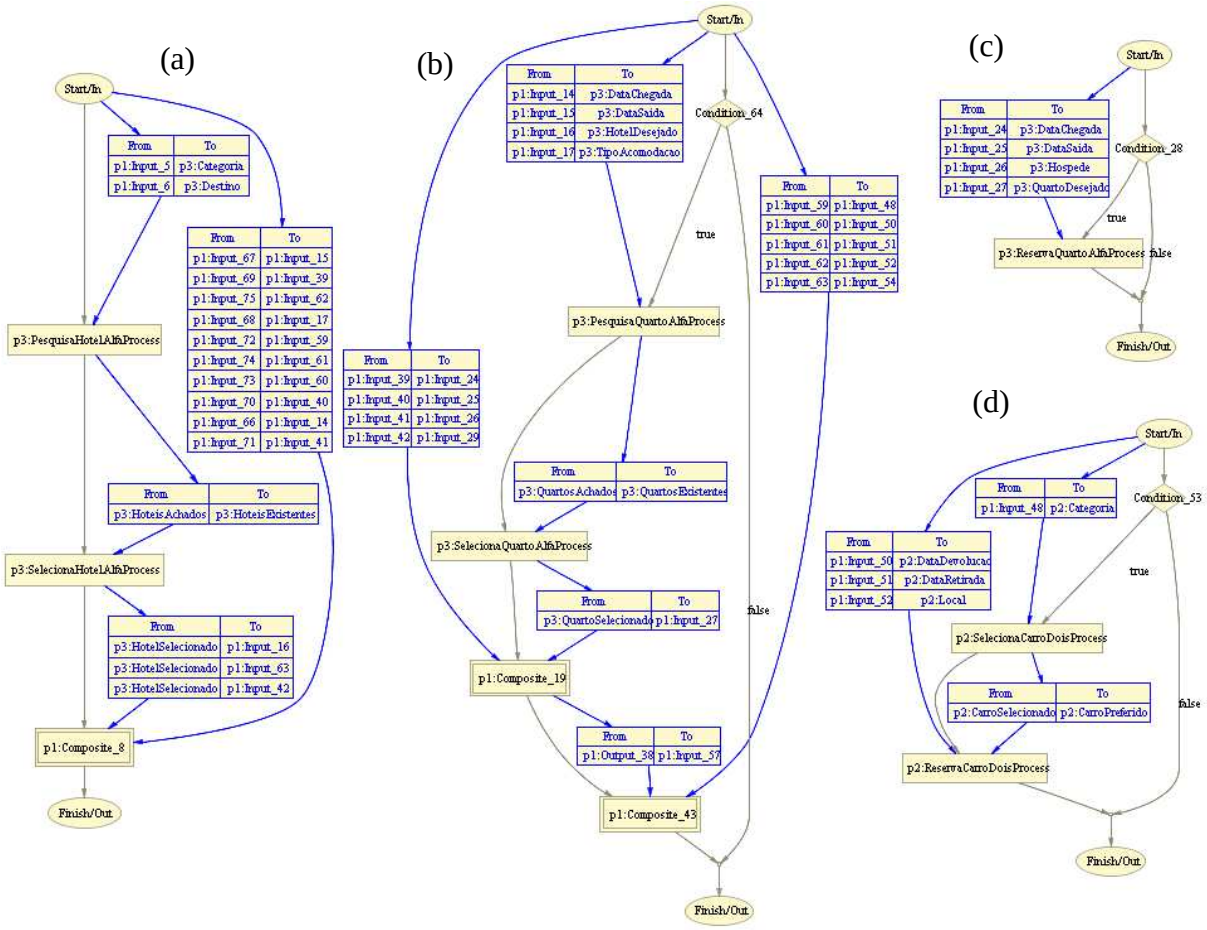


Figura 6.9. Plano de execução PLAN9 do caso de teste CST07

6.3 ANÁLISE DOS RESULTADOS

A partir dos resultados obtidos em cada experimento realizado neste estudo de caso foi elaborada uma análise comparativa, a qual é representada pela Tabela 6.2. Tal análise consistiu em avaliar basicamente se foi possível especificar um *workflow* para cada caso de teste, e se foram gerados planos de composição para os mesmos. Essa limitação foi necessária, pois não haveria ferramentas suficientes para podermos comparar com o ambiente proposto neste trabalho.

Tabela 6.2. Análise dos resultados do estudo de caso

	WebFlowAH	**Composer**	**WebComposer**
CST01	Foi executado. Gerou planos de execução com a orquestração de processos atômicos de um mesmo serviço.	Foi executado. Gerou um plano de composição contendo o serviço selecionado pelo usuário.	Foi executado. Gerou planos de composição com a orquestração de operações de serviços diferentes.
CST02	Foi executado. Gerou um subconjunto de planos do CST01.	Foi executado. Teve o mesmo resultado do CST01.	Foi executado. Gerou os planos de composição do CST01.
CST03	Foi executado. Gerou um subconjunto de planos do CST02.	Não foi executado. A ferramenta não permite a escolha de saídas.	Foi executado. Não gerou planos de composição.
CST04	Foi executado. Teve o mesmo resultado do CST03.	Não foi executado. A ferramenta não permite a escolha de precondições e efeitos.	Foi executado. Não gerou planos de composição.
CST05	Foi executado. Gerou planos de execução com a orquestração de processos atômicos de um mesmo serviço.	Não foi executado. A ferramenta não permite a especificação parcial do plano de composição.	Foi executado. Gerou planos de composição com a orquestração de operações de serviços diferentes.
CST06	Foi executado. Gerou os planos do CST02.	Foi executado. Gerou um plano de composição contendo o serviço selecionado pelo usuário.	Foi executado. Gerou os planos de composição do CST02.
CST07	Foi executado. Gerou planos de execução com a orquestração de processos atômicos de serviços diferentes.	Não foi executado. A ferramenta não permite fluxos de controle condicional.	Foi executado. Gerou planos de composição com a orquestração de operações de serviços diferentes.

Os resultados dos casos de teste realizados no *Composer* mostraram que essa ferramenta apresenta limitações quanto à especificação abstrata de *workflows*. Em primeiro lugar, não é possível especificar parcialmente um *workflow* para um dado serviço composto. Ou seja, a especificação deve ser completa contendo todas as atividades realizadas pelo *workflow*, as quais são definidas pelo usuário da própria ferramenta. Em segundo lugar, a combinação de serviços considera apenas as entradas e saídas dos mesmos, não permitindo a especificação de precondições e efeitos desses serviços. Por fim, a composição utiliza apenas fluxos de controle seqüencial não possibilitando a especificação de *workflows* mais complexos (ex.: condições, repetições, etc.).

No caso do *WebComposer*, os resultados dos respectivos casos de teste mostraram que essa ferramenta é superior em relação à ferramenta anterior. Além de permitir a especificação parcial de *workflows*, cuja completude é feita com a ajuda dos mecanismos de descoberta e composição dinâmicas de serviços, a ferramenta *WebComposer* suporta a definição de precondições, efeitos e outros tipos de fluxo de controle. Porém, os mesmos resultados também mostraram que essa ferramenta apresenta erros na composição de serviços, pois gera composições inválidas em certas situações (ver resultado do CST01 na Seção 6.2.2). Ademais, a especificação do *workflow* nessa ferramenta requer um conhecimento avançado por parte do usuário da sintaxe da linguagem adotada para tal especificação.

Com base nessa análise, podemos afirmar que o WebFlowAH atende aos requisitos de um ambiente de especificação e execução *ad-hoc* de processos de negócios baseados em serviços *Web*. Ao prover uma classificação das ontologias dos serviços *Web* nas ontologias de domínio comum, este ambiente oferece um assistente que auxilia um usuário a especificar *workflows* consultando-se os conceitos definidos nas ontologias de Objeto e Tarefa. Além disso, a partir da especificação incompleta contendo apenas os objetivos de negócio do *workflow* semântico é possível gerar um ou mais planos de execução através dos mecanismos de descoberta e composição dinâmicas suportados pelo ambiente. Por fim, esses planos podem ser executados no WebFlowAH fazendo a invocação dos serviços *Web* combinados para implementar um dado processo de negócio.

7 CONCLUSÃO E TRABALHOS FUTUROS

Neste trabalho vimos que a tecnologia de Serviços *Web* tem contribuído para alavancar a gerência de processos de negócio uma vez que favorece o desenvolvimento de novos processos a partir da composição de serviços *Web* básicos, gerando, dessa forma, serviços compostos e de elevado valor agregado. Essa composição é comumentemente realizada de forma manual, onde o próprio projetista é responsável por descobrir, selecionar e combinar os serviços que irão compor um processo de negócio. A falta de semântica das descrições WSDL dos serviços *Web* é razão pela qual essas tarefas são realizadas apenas por pessoas e das mesmas não serem automatizadas por agentes de *software*. Em ambientes de negócios considerados dinâmicos (*on the fly*), cujas mudanças são muito freqüentes, essa composição de serviços acaba sendo uma atividade custosa, inflexível e demorada.

A *Web* Semântica vem através das ontologias de serviço (tais como OWL-S e WSMO), agregar significados às descrições sintáticas dos serviços *Web* tornando-as passíveis de serem compreendidas por agentes de *software* e, com isso, permitir a automação de tarefas como a descoberta e composição de serviços. Projetos com grande visibilidade nessa área como INFRAWEBS (NERN *et al.*, 2004) e IRS-III (DOMINGUE *et al.*, 2004) provêem uma plataforma de desenvolvimento para serviços *Web* semânticos, na qual se pode descobrir, combinar e invocar serviços *Web* de forma automática. No entanto, o ferramental disponível atualmente por tais projetos exige que o usuário projetista tenha um conhecimento específico sobre as tecnologias da *Web* Semântica para realizar a especificação e execução de processos de negócios a partir da descoberta e composição dinâmicas. É preciso implementar mecanismos que permitam a especificação abstrata dos objetivos de um processo de negócio, a descoberta e seleção automática de serviços disponíveis que atendam aos objetivos especificados, a geração do plano de composição e a orquestração dos serviços selecionados. O ambiente proposto neste trabalho apresenta esses requisitos e permite tanto a especificação quanto a execução *ad hoc* de processos de negócio baseados em serviços *Web*.

Portanto, o WebFlowAH pode auxiliar na gerência de processos de negócio, uma vez que provê a descoberta, composição e invocação automáticas de serviços heterogêneos, descritos semanticamente com o uso de ontologias OWL-S. Ao classificar as descrições semânticas dos serviços *Web* nas ontologias de domínio comum, o WebFlowAH permite a representação abstrata de objetivos através de conceitos obtidos nessas ontologias, não havendo uma dependência entre a especificação do processo de negócio e as operações

providas pelos serviços *Web* existentes, ao contrário do que acontece em ambientes de composição manual. O estudo de caso descrito no Capítulo 6 mostrou como essa classificação de conceitos é fundamental para a especificação de um *workflow* semântico. Ademais, o WebFlowAH oferece um assistente que auxilia o usuário a especificar os objetivos para um dado processo de negócio. Ao contrário das abordagens de composição estática, tais como BPEL, onde é preciso saber a priori todas as atividades do processo de negócio de modo a criar uma especificação completa de um *workflow*, essa especificação dos objetivos de negócio apoiada pelo WebFlowAH permite que a mesma seja feita parcialmente ou de forma incompleta. A partir dessa especificação o próprio ambiente é capaz de concluir, com ajuda de mecanismos de inferências, quais serviços *Web* precisam ser selecionados e combinados para atender aos respectivos objetivos especificados, e posteriormente, gerar planos de execução contendo chamadas a esses serviços. Além disso, a composição de serviços com planos de execução alternativos vem trazer uma maior tolerância a falhas a esse ambiente, conforme visto nos resultados dos casos de teste.

7.1 CONTRIBUIÇÕES

Como contribuições deste trabalho, podemos destacar: (i) o modelo de classificação das ontologias de domínio comum Objeto e Tarefa que foi especificado e implementado para nivelar os conceitos definidos pelas ontologias dos serviços *Web* disponíveis, permitindo, dessa maneira, a especificação de processos de negócio a partir desses conceitos sem criar dependências desses serviços; (ii) a disponibilização de um assistente para guiar um usuário na especificação de um processo de negócio apenas com a definição dos objetivos descritos por uma ontologia de domínio comum, permitindo também que tal especificação seja parcial ou incompleta sem necessariamente informar todos os passos para a realização desse processo; (iii) o modelo de especificação de um *workflow* semântico em OWL-S para definição abstrata de processos de negócio não precisando criar novas linguagens de definição de processo ou estender as linguagens existentes; (iv) a geração automática de um processo executável em OWL-S; (v) a integração de diferentes ferramentas e tecnologias no desenvolvimento dos componentes do ambiente proposto (principalmente aqueles diretamente relacionados à *Web* Semântica como o *Compositor Semântico* e o *Seletor Semântico*), e

conseqüentemente, (vi) a implementação de um ambiente único para a especificação e execução de processos de negócio.

7.2 TRABALHOS FUTUROS

Muitas outras pesquisas poderão ser feitas a partir do ambiente WebFlowAH, haja vista as possibilidades de melhorias nas técnicas de descoberta e composição de serviços, e no modelo de especificação de *workflows* semânticos. Nesse aspecto, podemos destacar como trabalhos futuros a implementação de outros algoritmos de composição (AGRE *et al.*, 2007; PATHAK *et al.*, 2006; PISTORE *et al.*, 2004), bem como a complementação do algoritmo adotado pelo WebFlowAH, com o intuito de compará-los. Existe ainda a questão da validação das composições geradas por tais algoritmos onde alguns trabalhos (FOSTER *et al.*, 2007, MEDJAHED; BOUGUETTAYA; ELMAGARMID, 2003; NARAYANAN; MCILRAITH, 2002) estão sendo realizados nessa direção. A análise comparativa entre os algoritmos de *matching* (KIEFER; BERNSTEIN, 2008; KLUSCH; FRIES; SYCARA, 2006; SIRIN; HENDER; PARSIA, 2003) para a escolha de serviços *Web* e a compatibilização de ontologias também podem ser estudados futuramente. Além disso, pretende-se explorar a aplicação de lógica nebulosa ou *fuzzy* (STOILOS *et al.*, 2005) à descrição dos serviços *Web* de modo a permitir a seleção de serviços com base em informações não precisas como “mais barato” ou “mais caro”, as quais não podem ser representadas pela linguagem OWL. Outra oportunidade de trabalho é exportação dos planos de execução em outras linguagens de definição de processo como BPEL, por exemplo, e execução desses planos em outras máquinas de *workflow*. Finalmente, pode-se aperfeiçoar a especificação de *workflows* semânticos com a representação de um objetivo de negócio em termos de precondições e pós-condições, mesclando as metas do WSMO com os conceitos de Tarefa e Objeto.

REFERÊNCIAS

AGARWAL, V., CHAFLE, G., MITTAL, S., SRIVASTAVA, B. *Understanding approaches for web service composition and execution*. In Proceedings of the 1st Bangalore Annual Compute Conference (Compute '08). Bangalore, India, 2008. p. 1-8.

AGRE, Gennady, MARINOVA, Zlatina, PARIENTE, Tomas, MICSIK, Andras. *Towards Semantic Web Service Engineering*. Proceedings of the Workshop on Service Matchmaking and Resource Retrieval in the Semantic Web (SMR2 2007), 2007.

ALBERTONI, R., MARTINO, M. *Asymmetric and context-dependent semantic similarity among ontology instances*. Journal on Data Semantics X, v. LNCS4900, p. 1-30, 2008.

ALLEN, R. *Workflow: An Introduction*. Workflow Handbook, 2001. p.17-38. Disponível em: <http://www.wfmc.org/standards/docs/Workflow_An_Introduction.pdf>. Acesso em: 25.07.2007

ARPINAR, I. B., ALEMAN-MEZA, B., ZHANG, R., MADUKO, A. *Ontology-Driven Web Services Composition Platform*. In Proceedings of the IEEE international Conference on E-Commerce Technology, CEC. IEEE Computer Society, Washington, DC, 2004, p. 146-152.

BAADER F., NUTT, W. *Basic Description Logics*. In the Description Logic Handbook, Cambridge University Press, 2002, p. 47-100.

BROWN, Tony M. *The Road to BPM*. BPM.com, 2005. Disponível em: <http://www.bpm.com/>. Acesso em: 25.07.2007

BURBECK, S. *The Tao of e-business services – The evolution of Web applications into service-oriented components with Web services*. IBM Software Group, Oct 2000. Disponível em: <http://www.ibm.com/developerworks/webservices/library/ws-tao/>. Acesso em: 26.07.2007

BURLTON, R. *Business Process Management: Profiting From Process*. SAMS, USA, 2001.

CAMPOS, L.M., PIRES, P. F., CAMPOS, M. L. M. *OntoGuide-4WS: Uma Abordagem Sistemática para Descrição de Serviços Web*. 2nd Latin American Web Congress and 10th Brazilian Symposium on Multimedia and the Web: Anais WebMedia 2004, 2.

CLARK & PARSIA LLC. *Pellet: The Open Source OWL DL Reasoner*. Disponível em: <http://pellet.owldl.com>. Acesso em: 26.01.2007

COSTA, L., PIRES, P. F., MATTOSO, M. *Automatic Composition of Web Services with Contingency Plans*. In Proceedings of 2004 IEEE International Conference on Web Services, IEEE Computer Society, 2004a. p.454-461.

COSTA, L., PIRES, P. F., MATTOSO, M. *WebComposer: a tool for the composition and execution of Web Service-based Workflows*. In Proceedings of the Joint WebMedia/LA-Web 2004 Conference, IEEE Computer Society, 2004b, p30-37.

DAML Services Coalition. *OWL-S 1.1 Release*. 2004. Disponível em: <http://www.daml.org/services/owl-s/1.1/>. Acesso em: 26.09.2006

DUSTDAR, S., SCHREINER W. *A Survey on Web services Composition*. International Journal of Web and Grid Services, v.1, n.1, p. 1-30, 2005.

FALCONER, S. M., STOREY, M. A. *A Cognitive Support Framework for Ontology Mapping*. In Proc. 6th International Semantic Web Conference (ISWC 2007), Busan, South Korea, 2007, p. 114-127.

FOSTER, H., EMMERICH, W., KRAMER, J., MAGEE, J., ROSENBLUM, D., UCHITEL, S. *Model checking service compositions under resource constraints*. In Proceedings of the the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC-FSE '07), Dubrovnik, Croatia, 2007, p. 225-234.

GLASS, Graham. *The Web services (r)evolution Part 1: Applying Web services to applications*. IBM Software Group, Nov 2000. Disponível em: <http://www.ibm.com/developerworks/webservices/library/ws-peer1.html>. Acesso em: 26.07.2007

GRUBER, Thomas R. *A Translation Approach to Portable Ontology Specifications*. Journal of Knowledge Acquisition, v.5, n.2, 1993, pp. 199-220.

GUARINO, N. *Formal Ontology and Information Systems*. In Proc. of International Conference on Formal Ontology in Information Systems. IOS Press, 1998, p. 3-15.

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

HP Labs Semantic Web Programme. *Jena – A Semantic Web Framework for Java*. Disponível em: <http://jena.sourceforge.net>. Acesso em: 26.01.2007

HUANG, J., DANG, J., HUHNS, M. N. *Ontology Reconciliation for Service-Oriented Computing*. In Proceedings of the IEEE International Conference on Services Computing (ICSC 2006). IEEE Computer Society, Washington, DC, 2006, p. 3-10.

IBM Software Group. *Business Process Execution Language for Web Services version 1.1*. 2002. Disponível em: <http://www.ibm.com/developerworks/library/ws-bpel/>. Acesso em: 26.09.2006.

MENDES JR, J. R. ; PIRES, P. F. ; DELICATO, F. C. ; Batista, T. V. . *Um Ambiente para Especificação e Execução ad-hoc de Processos de Negócio Baseados em Serviços Web*. In Proceedings of the XIII Brazilian Symposium on Multimedia and the Web (WebMedia 2007), SBC, Gramado, RS, 2007, p. 238-245.

KHALAF, R., MUKHI, N., WEERAWARANA S. *Service-Oriented Composition in BPEL4WS*. Proceedings of the Twelfth International World Wide Conference (WWW2003), Web Services Track, Budapest, Hungary, 2003, Kluwer Academic Publishers, Norwell, MA.

KIEFER, Christoph, BERNSTEIN, Abraham. *The Creation and Evaluation of iSPARQL Strategies for Matchmaking*. Proceedings of the 5th European Semantic Web Conference (ESWC 2008). Tenerife, Spain, 2008.

KLUSCH, M., FRIES, B., SYCARA, K. *Automated semantic web service discovery with OWLS-MX*. In Proceedings of the Fifth international Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), Hakodate, Japan, 2006. p. 915-922.

LAUKKANEN, Mikko, HELIN, Heikki. *Composing Workflows of Semantic Web Services*. Proceedings of the Workshop on Web-Services and Agent-based Engineering, 2003.

MAJITHIA, Shalil, WALKER, David W., GRAY, W. A. *Automated Web Service Composition Using Semantic Web Technologies*. 1st International Conference on Autonomic Computing (ICAC 2004), New York, USA, 2004, p. 306-307.

MCBRIDE, B. *Jena: Implementing the RDF model and syntax specification*. In Proceedings of the 2nd International Workshop on the Semantic Web, 2001.

MEDJAHED, B., BOUGUETTAYA, A., ELMAGARMID, A. *Composing Web services on the Semantic Web*. The VLDB Journal. v.12, n.4, p. 333-351, 2003.

MILANOVIC, N., MALEK, M. *Search Strategies for Automatic Web Service Composition*. International Journal of Web Services Research, v.3, n.2, p. 1-32, 2006.

MINDSWAP Group. *OWL-S API*. Disponível em: <http://www.mindswap.org/2004/ owl-s/api/>. Acesso em: 30.03.2007

NextGRID Project. Disponível em: <http://www.nextgrid.org>. Acesso em: 09.06.2007

PAPAZOGLOU, Michael P., DUBRAY, Jean-jacques. *A Survey of Web service technologies*. Technical Report DIT-04-058, Informatica e Telecomunicazioni, University of Trento, 2004.

PATHAK, J., BASU, S., LUTZ, R., HONAVAR, V. *Parallel Web Service Composition in MoSCoE: A Choreography-based Approach*. 4th IEEE European Conference on Web Services (ECOWS 2006), Zurich, Switzerland, 2006, p. 3-12.

RAO, J., SU X. *A Survey of Automated Web Service Composition Methods*. In Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004), San Diego, California, USA, 2004.

ROMAN, D., KELLER, U., LAUSEN, H., BRUIJN, J., LARA, R., STOLLBERG, M., POLLERES, A., FEIER, C., BUSSLER, C., FENSEL, D. *Web Service Modeling Ontology*. Journal of Applied Ontology, v.1, n.1, p. 77-106, 2005.

SRIVASTAVA, B., KOEHLER, J. *Web service composition: Current solutions and open problems*. In ICAPS 2003 Workshop on Planning for Web Services, 2003, p. 28-35.

SIRIN, Evren, PARSIA, Bijan, GRAU, Bernardo Cuenca, KALYANPUR, Aditya, KATZ, Yarden. *Pellet: A practical OWL-DL reasoner*. UMIACS Technical Report, 2005-68, 2005.

STANFORD, Medical Informatics. *The Protégé Ontology Editor and Knowledge Acquisition System*. Disponível em: <http://protege.stanford.edu> . Acesso em: 04.07.2007

STOILOS, G., SIMOU, N., STAMOU, G., KOLLIAS, S. *Uncertainty and the Semantic Web*. IEEE Intelligent Systems, v.21, n.5, p. 84-87, 2006.

TRAINOTTI, M., PISTORE, M., CALABRESE, G., ZACCO, G., LUCCHESE, G., BARBON, F., BERTOLI, P., TRAVERSO, P. *ASTRO: Supporting Composition and Execution of Web Services*. 3rd International Conference Service-Oriented Computing (ICSOC 2005), Amsterdam, The Netherlands, 2005, p. 495-501
UDDI. *The Universal Description, Discovery and Integration protocol*. Version 3, 2003. Disponível em: <http://www.uddi.org/>. Acesso em: 26.09.2006

W3C Recommendation. *WSDL Specification*. 2003. Disponível em: <www.w3.org/TR/wsdl>. Acesso em: 26.09.2006

W3C Recommendation. *Web Services Architecture*. 2004a. Disponível em: <http://www.w3.org/TR/ws-arch/>. Acesso em: 27.07.2007

W3C Recommendation. *OWL Web Ontology Language*. 2004b. Disponível em: <http://www.w3.org/TR/owl-features>. Acesso em: 26.09.2006

W3C Recommendation. *Resource Description Framework (RDF): Concepts and Abstract Syntax*. 2004c. Disponível em: <http://www.w3.org/TR/rdf-concepts/>. Acesso em: 26.09.2006

W3C Submission. *RDQL - A Query Language for RDF*. 2004d. Disponível em: <http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/>. Acesso em: 26.09.2006

WfMC. *Workflow Management Coalition*. 2007. Disponível em: <http://www.wfmc.org>. Acesso em: 25.07.2007

WfMC. *Workflow and Internet: Catalysts for Radical Change*. 1998. Disponível em: <http://www.wfmc.org/standards/docs/Workflow_Internet_catalysts_for_change.pdf>. Acesso em: 25.07.2007

WSMO. *Web Service Modeling Ontology*. 2005. Disponível em: <http://www.wsmo.org>. Acesso em: 26.09.200

Livros Grátis

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

Milhares de Livros para Download:

Baixar livros de Administração
Baixar livros de Agronomia
Baixar livros de Arquitetura
Baixar livros de Artes
Baixar livros de Astronomia
Baixar livros de Biologia Geral
Baixar livros de Ciência da Computação
Baixar livros de Ciência da Informação
Baixar livros de Ciência Política
Baixar livros de Ciências da Saúde
Baixar livros de Comunicação
Baixar livros do Conselho Nacional de Educação - CNE
Baixar livros de Defesa civil
Baixar livros de Direito
Baixar livros de Direitos humanos
Baixar livros de Economia
Baixar livros de Economia Doméstica
Baixar livros de Educação
Baixar livros de Educação - Trânsito
Baixar livros de Educação Física
Baixar livros de Engenharia Aeroespacial
Baixar livros de Farmácia
Baixar livros de Filosofia
Baixar livros de Física
Baixar livros de Geociências
Baixar livros de Geografia
Baixar livros de História
Baixar livros de Línguas

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