3
Trabalhos Relacionados

Este capítulo apresenta uma discussão sobre os trabalhos relacionados ao proposto nesta tese e está organizado da seguinte forma. A Seção 3.1 apresenta uma visão geral da evolução dos sistemas de workflow em direção à flexibilidade. A Seção 3.2 descreve várias linguagens para composição de workflows/serviços. A Seção 3.3 aborda alguns trabalhos diretamente relacionados à linguagem OWL-S. A Seção 3.4 resume frameworks e protocolos para composição e coordenação de serviços. Por fim, a Seção 3.5 discute duas abordagens para busca de serviços baseada em ontologias. Cada seção termina com uma comparação dos trabalhos apresentados com as abordagens propostas nesta tese. A Seção 3.6 apresenta um resumo deste capítulo.

3.1
Estratégias de Flexibilização

3.1.1
Preliminares

Em 1995 já eram estudadas as questões de flexibilização no contexto de sistemas de gerência de workflow. Em [41] argumenta-se que, em vários ambientes, workflows podem ser de bastante valia se forem dinâmicos, flexíveis e alteráveis em tempo de execução.

Em conseqüência do uso crescente de sistemas de gerência de workflow nos últimos anos, novos requisitos para estes sistemas foram surgindo. A tendência é que estes sistemas evoluam em direção a abordagens de coordenação menos restritivas e cada vez mais flexíveis [25, 89, 40].

Em [40] é brevemente resumida a história dos sistemas de gerência de workflow, enfatizando que a modelagem de workflows não deve ser limitada ao tempo de projeto, mas deve ser feita também em tempo de execução. Pelo fato da separação de tempo de projeto e tempo de execução não acontecer

claramente na prática, sistemas de workflow adaptativos estão se tornando cada vez mais comuns [135].

Em [107] é apresentada uma série de pontos que mostram a importância de se incluir flexibilidade em sistemas de workflow, especialmente no que diz respeito às alterações dinâmicas em tempo de execução. Estas alterações dinâmicas tornam-se cada vez mais freqüentes na medida em que nem todas as situações de um workflow podem ser previstas antecipadamente. Para isso, em [24] é apresentada uma discussão das abordagens de tratamento de exceções em sistemas de workflow, o que contribuiu significativamente para o avanço nos estudos de sistemas adaptativos.

Em [67] são apresentadas as principais características, os requisitos e os objetivos dos *sistemas de workflow adaptativos*, como sistemas projetados para tolerar desvios e exceções não previstas durante o tempo de execução de um workflow. Um sistema adaptativo deve permitir alteração dinâmica da definição do workflow, modelos de execução configuráveis, evolução do modelo e modelos decomponíveis para reuso.

O esforço para alcançar flexibilidade em sistemas de gerência de workflow atuais está dividido em duas abordagens distintas, porém muitas vezes usadas em conjunto. A primeira abordagem trata de se alcançar a flexibilidade por meio de uma modelagem mais flexível, muitas vezes intercalando a fase de modelagem com a fase de execução do workflow. A segunda abordagem trata de flexibilizar a execução do workflow propriamente dita, em especial oferecendo suporte para alterações na estrutura do workflow em tempo de execução, seja no nível do próprio workflow, seja no nível das instâncias correntes em execução. A maioria dos trabalhos encontrados na literatura busca alcançar a flexibilidade com base nesta segunda abordagem.

3.1.2 Intercalando Modelagem e Execução

O projeto WASA [122] utiliza a abordagem de interpretação para alcançar flexibilidade em sistemas de gerência de workflow. Para garantir que modificações sejam possíveis, o servidor de workflow lê modelos de subworkflows apenas quando estes estão para serem iniciados. Isto permite alterar atividades dentro dos subworkflows que ainda não foram "carregados", quando o workflow maior está em execução.

Em [15], é apresentado um formalismo para a definição de workflows, baseado em uma biblioteca de blocos de construção, ou seja, de (sub)workflows, e em uma abordagem de definição dinâmica de manipulação de exceções e eventos. Este formalismo permite flexibilidade na definição dos workflows, que podem ser dinâmica e progressivamente compostos, e o reuso destas definições.

Em [38] é apresentada uma abordagem que permite que as fases de planejamento e de execução em sistemas de gerência de workflow sejam alternadas, aumentando a flexibilidade na coordenação e no controle da execução. Este aumento de flexibilidade é possível pelo uso de conhecimento agregado ao plano de execução lido pelo interpretador, que extrai do modelo as dependências entre tarefas e variáveis. O conceito de método dentro de um plano representa o objetivo deste plano e permite, assim, que soluções de execução para este objetivo sejam encontradas ou que novas soluções sejam criadas em tempo de execução.

Essa abordagem faz parte do escopo do projeto CoMo-Kit, o qual está dividido entre o modelador e o escalonador. O modelador é uma interface gráfica e permite ao projetista modelar o processo sem o conhecimento da linguagem interna na qual estará escrito. Adiar a especificação de partes do plano para o momento da execução tem a vantagem de que o conhecimento específico do projeto pode ser usado para completar e adaptar o plano. Além disso, o sistema apresentado em [38] fornece mecanismos básicos para alteração de planos de projeto como uma reação às alterações ou aos erros não previstos. Através do gerenciamento de dependências causais, os efeitos das alterações podem ser computados e manipulados pelo interpretador.

No âmbito do projeto PoliFlow, o sistema SWATS permite que a definição de um workflow seja iniciada em tempo de projeto e completada em tempo de execução [110]. Neste sistema, modificações podem ser feitas no nível de instâncias, sem que a estrutura do workflow ou de outras instâncias seja afetada. No entanto, para evitar que adaptações sejam necessárias diante das modificações, regras de consistência são definidas em tempo de projeto, garantindo que o workflow sempre saia de um estado consistente, ou seja, um estado que está de acordo com todas as restrições e dependências impostas pelos subworkflows componentes, para outro estado consistente. Regras de integridade, por sua vez, que garantem a correta dependência entre as tarefas de um workflow e entre subworkflows componentes de um workflow maior, são manualmente definidas pelo usuário para assegurar a corretude semântica dos workflows, em especial, devido às alterações manuais.

Em [74] os autores claramente separam a modelagem de workflows em tempo de projeto da modelagem em tempo de execução, usando um conjunto de construtores padrão e um conjunto de restrições. As restrições determinam como cada fragmento pode ser incluído no workflow, sob quais condições a instância do workflow pode ser terminada e quais condições devem ser verdadeiras durante a definição do workflow. Neste caso, a flexibilização é alcançada deixando que a definição do workflow seja completada em tempo de execução, de acordo com as restrições e os construtores definidos.

3.1.3 Alterações Dinâmicas na Estrutura do Workflow

O passo principal na direção da flexibilidade em sistemas de workflow é a consistente e efetiva gerência da evolução dos workflows. Além da evolução natural das organizações e de seus processos, situações inesperadas ou impossíveis de serem previstas podem ocorrer. Em [25] é apresentado um conjunto de primitivas que permitem a modificação de um workflow, preservando o critério de corretude sintática quando as modificações são aplicadas à descrição (estática) do workflow ou às instâncias (dinâmicas) do mesmo. Para o sistema ADEPTflex descrito em [103, 104] é definido um conjunto completo e mínimo de operações que permitem a alteração dinâmica na estrutura de workflows. Dentro deste conjunto, estão operações tais como a de inserção e a de remoção de tarefas.

Em [127] dois conjuntos distintos de operações de flexibilização são definidos: o conjunto de operações para alterações dinâmicas na estrutura do workflow e o conjunto de operações para a intervenção do usuário durante a execução no nível da instância de workflow. Para cada um dos estados de execução, são especificadas quais operações de modelagem dinâmica são válidas. As intervenções dos usuários, por sua vez, permitem que o fluxo pré-definido de atividades seja alterado em tempo de execução. Assim, alterações podem ser feitas em tempo de execução tanto no nível da estrutura do workflow quanto no nível de instâncias.

Em [65] frameworks dirigidos por regras dão suporte às alterações estruturais em um workflow, determinando como tarefas podem ser inseridas ou removidas de um workflow, em tempo de execução. Estas alterações são guiadas por regras do tipo evento-condição-ação (*ECA rules*). Cada primitiva de alteração é encapsulada por uma pré-condição que restringe sua aplicação e levanta um evento correspondente, que é manipulado

pelas instâncias afetadas, a fim de assegurar a consistência dos estados de execução.

Em [29] a abordagem de gerência de alterações em workflows é baseada em uma ontologia de regras, que determina como a migração do antigo para o novo workflow é feita, de modo similar à abordagem apresentada em [65]. Estas regras podem definir que tarefas sejam inseridas ou removidas da estrutura do workflow, mantendo a corretude do mesmo.

Esta tese introduz uma noção de corretude nova, em função da flexibilização, e torna bastante clara a noção de corretude para workflows com execução hierárquica. Geralmente, a noção de corretude em sistemas de workflow está diretamente vinculada à noção de transação.

Em [58] são propostos dois tipos distintos de flexibilidade: por seleção e por adaptação. Flexibilidade por seleção oferece ao usuário caminhos alternativos de execução, mas o permite escolher apenas um. Este tipo de flexibilidade pode ser garantida já em tempo de modelagem, onde são descritos todos os caminhos de execução possíveis, ou em tempo de execução, de forma que da modelagem façam parte caixas-pretas (*black box*) que são conhecidas apenas no momento da execução. Na flexibilidade por adaptação, novos caminhos de execução podem ser adicionados à estrutura do workflow em tempo de execução, tanto no nível da instância em execução quanto no nível do workflow propriamente dito. No contexto da flexibilidade por adaptação, versões da definição do workflow são criadas a cada vez que uma nova alteração é feita. Em [57] é apresentado como estas duas abordagens para gerência flexível de workflow são utilizadas no sistema OPENflow.

No contexto do sistema de suporte a processos PROSYT (*(PROcess Support sYstem capable of Tolerating deviations*), situações não previstas em tempo de modelagem são tratadas como desvios em tempo de execução [34], realizados por intermédio dos usuários. No entanto, é necessária uma reconciliação manual do modelo de processo com o modelo em execução para evitar inconsistências.

Em [14], workflows são vistos como "fluxos de estados", e não como "fluxos de trabalho". De acordo com a noção de estados válidos, regras indicam como um workflow pode passar de um estado inválido para um estado válido, por exemplo inserindo e removendo tarefas. Assim, é possível alterar a estrutura do workflow dinamicamente, levando em consideração políticas de obrigações, proibições e recomendações. De acordo com a política de obrigações, se uma tarefa precisa estar presente na estrutura de um processo e não está, então ela é incluída. De acordo com a política de proibições, se alguma atividade está presente no processo e não deveria

estar, ela é removida. Por fim, de acordo com a política de recomendações, se algumas atividades não estão incluídas na estrutura do processo, elas são, forte ou fracamente, sugeridas à inclusão. Deste modo, as alterações dinâmicas são guiadas por políticas e pelo conceito de estados válidos.

Em [54] os mecanismos de flexibilização propostos estão centrados na cooperação entre os usuários e na antecipação da execução de tarefas, seguindo a abordagem COO, apresentada em [22]. Esta abordagem oferece um modelo de transação especial para sistemas cooperativos, que é diferente do modelo de sincronização convencional início-fim. Estes mecanismos de flexibilização permitem que algumas atividades sejam antecipadas de acordo com a descrição do workflow, de modo que a ordem de terminação das atividades seja sempre preservada.

Em [53], são distinguidos dois conceitos: capacidade adaptativa e capacidade dinâmica. Capacidade adaptativa é necessária quando uma exceção requer que o trabalho seja realizado em uma seqüência diferente da previamente definida. Capacidade dinâmica é requerida quando a definição da estrutura do workflow é alterada, requerendo a migração do modelo antigo para o novo modelo. Para solucionar o problema da flexibilidade, é proposto o mecanismo de requisitos mínimos, de acordo com o qual a cada definição de processo é associada uma tabela de requisitos mínimos (MRT, do inglês *Minimum Requirements Table*). Assim, quando existe a necessidade de mudança de um modelo de processo, todas as tarefas pendentes para execução na nova definição do modelo são executadas de acordo com os requisitos especificados na MRT. Tarefas não executadas são hierarquicamente executadas até que os requisitos na tabela do novo modelo de processo sejam alcançados e a migração feita com sucesso.

Em [102] é apresentada uma discussão sucinta de alguns sistemas de gerência de workflow, dentre eles o sistema Exotica da IBM, baseado na arquitetura cliente-servidor do FlowMark (produto da IBM responsável por gerenciar filas de mensagens trocadas entre clientes e servidores). Exotica não permite alterações dinâmicas na estrutura dos workflows. Por outro lado, o sistema de gerência de workflows distribuído *RainMan*, também da IBM, permite este tipo de modificação. O *RainMan* é baseado no framework *RainMaker*, que define uma série de interfaces abstratas de serviços implementados em Java para a Internet. O sistema de gerência de workflows da Newcastle-Nortel foi criado com o objetivo de atingir interoperabilidade, escalabilidade e permitir a reconfiguração dinâmica.

InConcert e TeamWare Flow são dois outros sistemas de workflow que permitem alterações dinâmicas no modelo do processo em tempo de

execução. No entanto, este tipo de comportamento é interessante apenas quando as alterações serão de fato permanentes, pois a estrutura do workflow é modificada permanentemente.

3.1.4 Alterações Dinâmicas no Nível das Instâncias de Processo

Tratar desvios como exceções e fornecer mecanismos de tratamento de exceções, ou mesmo modificar o modelo de processos em tempo de execução, muitas vezes não é recomendável dado que desvios são temporários. Ao contrário, PROSYT permite que a execução de um processo divirja de seu modelo estático, sem que o este modelo seja alterado. O desvio é realizado sob o controle do sistema de suporte a processo (*Process Support Systems*, em inglês), mesmo que estes desvios resultem em inconsistências. Estas possíveis inconsistências geradas são posteriormente reconciliadas no modelo de processo através da intervenção do usuário.

3.1.5 Estratégias de Relaxamento de Consultas

CoBase é um sistema de banco de dados centrado na idéia de consultas cooperativas [46]. Quando o usuário submete uma consulta ao sistema, este utiliza uma hierarquia de informação (chamada em [27] de hierarquia de tipos abstratos, TAH, do inglês *Type Abstraction Hierarchy*) para descobrir informação relevante para a consulta original, dentro de uma certa distância semântica da resposta exata. Para realizar esta tarefa, o sistema expande o escopo da consulta, buscando informação em áreas próximas às áreas onde estão contidas as respostas exatas nesta hierarquia. Dessa forma, esta hierarquia (baseada em instâncias) é a principal estrutura de dados para o sistema de respostas a consultas cooperativas no sistema CoBase.

Existem basicamente dois mecanismos para se traduzir uma consulta entre níveis de abstração diferentes, sendo cada nível de abstração representado por um nível da hierarquia. O primeiro mecanismo é o de abstração de consultas, que transforma a representação original de uma consulta em uma representação mais abstrata (em um nível superior da hierarquia). O segundo mecanismo, o de refinamento, é o processo inverso, que converte a consulta em representações mais específicas.

A busca na hierarquia pela informação semanticamente mais próxima pode acontecer por meio de dois métodos distintos: pesquisa por vizinhança

e pesquisa conceitual. Ambos os métodos utilizam tabelas auxiliares que armazenam informação semântica relativa a cada tipo de dado presente na hierarquia.

No mecanismo de pesquisa por vizinhança, a consulta original é generalizada e a hierarquia de dados é percorrida de cima para baixo, até se encontrar informação semanticamente próxima da resposta original. Neste caso, são executados os seguintes passos:

1. busca dos tipos de objetos exatos requisitados na consulta original. Se a busca falha, então o próximo passo é executado;

2. subida na hierarquia de abstração de tipos e reescrita da consulta para uma forma mais abstrata, através de um processo de abstração de consultas;

3. descida na hierarquia e reescrita da consulta original em consultas mais específicas, através de um processo de refinamento de consultas.

Para que esse tipo de pesquisa seja possível, o sistema utiliza 3 tabelas auxiliares, que descrevem informação semântica a respeito do banco de dados (fonte a ser consultada), em diferentes níveis de conhecimento:

1. tabela que faz a ligação entre a hierarquia de abstração e cada uma das relações do banco de dados. Esta tabela descreve os subtipos das relações do banco representando, assim, uma hierarquia de relações;

2. tabela de mapeamento de domínio, que fornece o relacionamento entre nomes de atributos e nomes de domínios;

3. tabela de abstração de atributos e valores, que apresenta o mapeamento de instâncias de cada par de subtipos e supertipos, ou os valores de instância correspondentes entre um supertipo e uma faixa de subtipos.

Na pesquisa conceitual, a consulta original é refinada até se encontrar respostas semanticamente próximas. As explicações formais para cada um dos métodos e do funcionamento do sistema CoBase são apresentadas em [27].

Uma aplicação interessante desse tipo de "relaxamento de consulta" é apresentada em [28]. Neste artigo, esta técnica é utilizada para a construção de condições de disparo de *triggers* em bancos de dados. O usuário especifica as regras em linguagem natural e, através da hierarquia de abstração gerada

automaticamente, o sistema consegue construir as regras que definem sob quais condições o *trigger* deve ser disparada, de modo que elas sejam processáveis pelo sistema de banco de dados.

Em [12] é apresentado um mecanismo para se retornar respostas aproximadas a uma consulta de banco de dados quando a estrutura implícita dos dados não é conhecida. Neste caso, a estrutura da resposta deve estar semanticamente próxima da estrutura informada na consulta. Um mecanismo bastante semelhante para consulta sobre dados semi-estruturados é também apresentado em [93].

3.1.6 Comparação

O mecanismo de tratamento de exceção proposto nesta tese como forma de flexibilizar a execução de workflows utiliza ontologias (incluindo regras) para tomar decisões, durante a execução, sobre como substituir workflows (ou processos, como serão chamados nesta tese) ou recursos, sobre como encontrar instâncias concretas para workflows ou recursos abstratos, sobre como desfazer os efeitos de um workflow e sobre como atribuir valores default a parâmetros.

Alcança, portanto, um efeito similar à intercalação de modelagem e execução ao permitir que workflows sejam modelados com subworkflows e recursos abstratos, substituídos por instâncias concretas, escolhidas em tempo de execução, de acordo com uma ontologia de processos e recursos previamente definida.

O mecanismo de tratamento de exceção desta tese permite que a execução de uma determinada instância não seja interrompida por longos períodos, tratando exceções previamente definidas, de modo semelhante ao formalismo para definição dinâmica de manipulação de exceções e eventos, proposto em [15].

Oferece, ainda, flexibilidade por adaptação, guiada por informação semântica adicional sobre a descrição do workflow. Em particular, o mecanismo tenta evitar que a execução de um subworkflow seja suspensa por um tempo longo demais por falta de informação ou por falta de recursos.

Segundo o mecanismo de tratamento de exceção proposto nesta tese, não são definidas a priori todas as operações que podem ser realizadas sobre os workflows para a alteração de sua estrutura dinâmica, como o são no sistema ADEPTflex, mas sim todos os workflows alternativos a cada

workflow definido e também as alternativas aos recursos. Estas alternativas são portanto escolhidas em tempo de execução.

A estratégia do mecanismo proposto é bastante similar à estratégia de relaxamento de consultas já que utiliza hierarquias semânticas previamente definidas (nas ontologias) para encontrar workflows e recursos alternativos ou instâncias concretas.

O mecanismo proposto não visa permitir alterações dinâmicas na estrutura dos workflows. Não cobre, por exemplo, flexibilidade por seleção. Uma descrição deste mecanismo pode ser encontrada em [119, 121, 120]. Um resumo sobre o início dos estudos sobre sistemas de gerência de workflow, no âmbito desta tese, pode ser encontrado em [32].

3.2 Linguagens para Modelagem de Workflows

3.2.1 Preliminares

Em 1997, Sheth et al. em [80] já apresentavam alguns pontos importantes e tendências da evolução de sistemas de gerência de workflow para o ambiente Web. Atualmente, a "Web de serviços" se apresenta como uma tendência muito forte para as novas gerações de aplicações. Em [114] é apresentada uma visão geral de qual o impacto, as modificações e os avanços decorrentes da tecnologia de serviços em diversas áreas, em especial no que diz respeito aos processos de negócio. Em [117], os autores afirmam que serviços na Web e composições de serviços, em particular, provavelmente serão de grande importância na tecnologia dos sistemas de processos de negócio no futuro.

Diversas são as linguagens e padrões especificados para atender às necessidades dessa tecnologia. Essa grande quantidade de especificações forma atualmente o que é conhecido por *pilha de tecnologia de serviços na Web*, apresentada na Figura 3.1. Assim como são inúmeros os modos de se construir e utilizar os serviços, são diversas as maneiras de se visualizar as tecnologias envolvidas.

Basicamente, a camada inferior desta pilha apresenta as tecnologias de transporte e comunicação de dados, tais como HTTP e FTP. A camada superior compreende tecnologias base como a tecnologia de composição de mensagens SOAP, a linguagem WSDL (*Web Service Description Language*) [123] para a descrição dos serviços e linguagens para descoberta, publicação

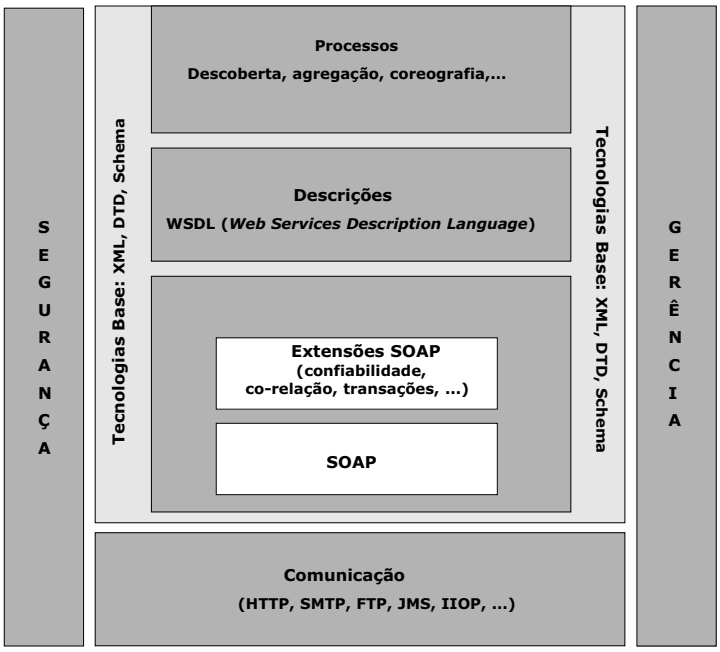

Figura 3.1: Pilha de tecnologias da arquitetura de serviços na Web, segundo o W3C [126].

e composição de serviços. Padrões para segurança na troca de mensagens e para a gerência de processos também estão presentes.

Os serviços na Web estão se movendo da fase de descrição, publicação e interação para uma nova fase na qual interações de negócio robustas são suportadas [35]. Um mecanismo de gerência de workflow é necessário para orquestrar um grupo de serviços na Web especializados no suporte a processos de negócio [68]. Por isso, além de uma camada de representação das linguagens que descrevem os processos de negócio construídos a partir do conceito de serviços na Web, também é necessária uma camada que trate da coordenação destes serviços como em uma transação.

Diversas linguagens de composição de serviços já existem atualmente, tendo sido propostas por organizações e consórcios distintos. Dentre estas linguagens, podem ser citadas as seguintes: WSFL, BPEL4WS, BPML, WSCI, WSCL, WSMF e OWL-S. Normalmente, linguagens de composição de serviços são utilizadas em conjunto com a linguagem WSDL, cujo objetivo é descrever cada um dos serviços utilizados na composição.

Grande parte das linguagens de composição de serviços estão evoluindo no sentido da chamada "Web semântica" [13], na qual dados são semanticamente anotados para que não apenas humanos mas também agentes de software possam processá-los. Em [43] é apresentado um resumo de algumas das linguagens que foram sendo criadas neste contexto desde o começo da veiculação do conceito de "Web semântica".

Conforme [113], o problema da composição de serviços é bastante

similar ao projeto e especificação de protocolos de comunicação de dados, com a diferença essencial de que a semântica das novas linguagens de protocolo de negócio não é definida precisamente.

Diversos artigos têm sido publicados na literatura apresentando uma comparação entre as linguagens existentes. Dentre os pontos de comparação, estão os construtores permitidos pelas linguagens. Em [116] são apresentados 20 padrões de workflows independentes de linguagens como forma de indicar os requisitos destas linguagens de modelagem. Dentre os mais comuns, estão os padrões de seqüência e paralelismo.

3.2.2 WSFL

WSFL (*Web Service Flow Language*) [70] é uma linguagem baseada em XML, proposta pela IBM, para descrever uma composição de serviços na Web através de grafos direcionados, que podem ser aninhados, mas não acíclicos. Ela considera 2 tipos de composição de serviços: os modelos de fluxo e os modelos globais.

Os modelos de fluxo (processos de negócio executáveis) descrevem, a partir de uma composição de serviços (descrição de um processo de negócio), como se alcançar o objetivo de um negócio particular. Os modelos globais (colaboração de negócio) especificam padrões de interação em uma coleção de serviços [92]. Assim como XLANG (linguagem da IBM para modelagem de negócios para BizTalk), este modelo é meramente um mapeamento entre entradas e saídas de processos de negócio.

WSFL é compatível com SOAP, UDDI (*Universal Description, Discovery and Integration*) [1] e WSDL [31]. Apesar de definir a sintaxe do arquivo que descreve a composição de serviços, não há semântica associada. Além disso, segundo apresentado em [96], apesar de WSFL suportar a gerência de exceções, ela não possui suporte direto para transações.

3.2.3 BPEL4WS

A linguagem BPEL4WS (*Business Process Execution Language for Web Services*) [71], definida em conjunto pela BEA, pela IBM e pela Microsoft, tem como base as linguagens WSFL e XLANG, e é tida como a substituta de WSFL. BPEL4WS (ou simplesmente BPEL) está atualmente na versão 1.1 [7] e, como a maioria das linguagens para definição de

processos de negócio através de serviços na Web, depende de especificações tais como WSDL, XML Schema, XPath e WS-Addressing [16]. BPEL é essencialmente uma camada em cima de WSDL, com WSDL definindo as operações específicas permitidas e BPEL definindo como estas operações devem ser ordenadas [96].

BPEL define um modelo de negócio e uma gramática para descrever o comportamento de um processo baseado nas interações entre ele e seus parceiros. Um processo interage com seus parceiros invocando operações que eles suportam e recebendo mensagens através da interface de serviço. Um parceiro pode ser qualquer serviço que o processo invoca ou qualquer serviço que invoca o processo. Por exemplo, uma mensagem é enviada por um processo a um parceiro de negócio através da atividade *invoke*. O processo parceiro, por sua vez, espera que uma operação seja invocada através da atividade *receive*.

No núcleo do modelo de processos de BPEL está a noção de interação *peer-to-peer* entre serviços descritos em WSDL. A interação *peer-to-peer* tratada é de longa duração, *stateful* e possui um início, um comportamento definido durante sua ocorrência e um fim. Os processos executáveis, por sua vez, são como descrições de workflow representadas através de atividades básicas e estruturadas. No entanto, o modelo abstrato de BPEL define apenas o protocolo de negócio sob a perspectiva de uma única entidade na colaboração, deixando a conversação para ser descrita, por exemplo, por linguagens tais como WSCI e WSCL.

BPEL consegue manter o status da interação (e por isso da interação em BPEL ser *stateful*) através do uso do mecanismo de correlação de mensagens. Quando mensagens são trocadas entre parceiros de negócio, elas tipicamente carregam dados que são usados para correlacionar a mensagem com o processo de negócio apropriado. Estes dados são definidos através das chamadas *propriedades* em BPEL.

Além disso, BPEL fornece suporte para tratamento de exceções através do uso de *fault handlers* e *compensation handlers*. Os *fault handlers* funcionam de forma similar aos blocos *try-catch* do Java, dentro dos quais os erros ocorridos são tratados. Os *compensation handlers*, por sua vez, são utilizados para possibilitar o cancelamento dos efeitos das ações que já tinham sido completadas no momento em que o erro ocorreu. Normalmente, os *compensation handlers* são invocados de dentro dos *fault handlers*.

BPEL4WS pode ainda agregar capacidades de coordenação distribuída se combinada a padrões tais como WS-Coordination e WS-Transaction (a serem apresentados a seguir) [112, 2].

O foco de BPEL4WS 1.1 está na definição de um núcleo comum entre os processos de negócio abstratos e os protocolos de negócio. BPEL é também uma linguagem que não permite representação de informação semântica de e entre processos. Em abril de 2003 a OASIS (*Organization for Advanced Structured Information Systems*) anunciou a criação de um comitê para o estudo da linguagem WSBPEL (*Web Services Business Process Execution Language*), como continuação do trabalho apresentado na linguagem BPEL4WS [90].

3.2.4 BPML

BPML (*Business Process Management Language*) é uma meta-linguagem para a modelagem de processos de negócio, definida pela BPMI.org. BPML fornece um modelo de execução abstrato para processos de negócio transacionais e colaborativos, baseado no conceito de uma máquina de estados transacionais finita [91]. Cada definição de processo pode especificar qualquer um dos três modos de instanciar um processo: em resposta a uma mensagem de entrada, em resposta a um sinal levantado ou invocado a partir de uma atividade ou de um escalonamento.

Um processo em BPML é descrito por um conjunto de fluxos de controle, fluxos de dados e fluxos de eventos, acrescido de informações como regras de negócio, segurança e contextos transacionais. BPML é baseada nas tecnologias SOAP, WSDL e UDDI. No contexto de WSDL, operações são especificadas usando tipos de porta e operações. A atividade *action* realiza ou invoca um única operação que envolve a troca de mensagens de entrada e de saída.

BPML suporta construções e atividades de fluxo de processo bastante similares às permitidas em BPEL. BPML suporta, ainda, composição recursiva, ou seja, a construção de processos aninhados, e decomposição recursiva. BPML oferece suporte também à manipulação de exceções e às transações de compensação complexas.

3.2.5 WSCI

A WSCI (*Web Services Choreography Interface*), construída em conjunto pela Sun, BEA e Intalio, é uma linguagem que descreve o fluxo de mensagens trocadas por um serviço durante interações dentro de um

processo, de modo que a troca de mensagens é descrita do ponto de vista de cada serviço (visão do processo orientada a mensagens). WSCI também permite a descrição de mensagens trocadas entre os diversos serviços que compõem o processo. Como cada arquivo WSCI descreve apenas a participação de um parceiro na interação, a descrição de uma coreografia involve diversos documentos WSCI.

Através da especificação das dependências lógicas e temporais entre as mensagens trocadas, serviços podem interagir uns com os outros de forma não ambígua e de acordo com o modo de colaboração definido.

WSCI suporta, dentre outras questões, manipulação de exceções, gerência de transações e colaboração dinâmica. Em WSCI não existe um único processo de controle gerenciando a interação.

Segundo descrito em [96], WSCI parece ser mais robusta no suporte à colaboração quando comparada a BPEL4WS. Conforme descrito no *position paper* [17], WSCI pode ser vista como o "denominador comum" entre BPEL4WS e BPML e, assim como estas, não possui suporte para descrição de relacionamento semântico entre diversos processos.

De acordo com [42], WSCI descreve o comportamento dos serviços, mas não endereça a definição de processos gerenciando as trocas de mensagens, e nem mesmo a definição do comportamento interno de cada serviço. A grande diferença entre WSCI e WSFL é que WSFL apenas descreve um processo envolvendo múltiplos serviços, e não como um serviço na Web pode participar deste processo, nem como ele se expõe para a colaboração.

Entre WSCI e WSDL, a diferença é que WSDL permite a descrição de um serviço em termos das operações que ele suporta e dos *bindings* destas para os protocolos, enquanto WSCI descreve como as operações especificadas em WSDL são coreografadas e quais propriedades destas coreografias são expostas.

Assim, WSCI, WSDL e WSFL podem ser utilizadas em conjunto, com WSDL definindo as operações de cada serviço e WSCI descrevendo o comportamento de cada um deles no fluxo ou processo definido em WSFL.

Se comparada com BPML, WSCI é uma linguagem complementar, já que BPML descreve processos de negócio mapeando atividades de negócio para troca de mensagens.

WSCI é um *note*[1] do W3C desde agosto de 2002.

[1]Um *note* é publicado por um grupo de trabalho registrado no W3C, para indicar que o trabalho foi finalizado para um tópico em particular. Um *Working Group* pode publicar este tipo de documento com ou sem uma publicação prévia de um *Working Draft*. Depois de ter sido amplamente revisado e tendo satisfeito os requisitos técnicos do grupo de

3.2.6 WSCL

A linguagem WSCL (*Web Services Conversation Language*) [11] foi definida pela HP de forma complementar a WSDL, já que WSDL define as mensagens a serem trocadas entre os serviços em um processo de negócio e WSCL define a ordem em que estas mensagens são trocadas. Apesar de conversação ser definida sob a perspectiva do provedor do serviço, ela separa a lógica conversacional da lógica da aplicação ou dos aspectos de implementação do serviço. WSCL não modela explicitamente os papéis na colaboração.

A sintaxe de WSCL é dada por um arquivo XML *Schema*, que representa a linguagem formal na qual uma interação deve ser escrita. Assim como outras linguagens, não há semântica definida em WSCL.

WSCL fornece um modelo de transição de estados simples para organizar a seqüência das operações descritas em WSDL. Ao contrário de WSCI, WSCL não suporta propriedades, contextos, transações, gerência de exceções, time-out, localizadores (*locators*), modelos globais, correlações ou qualquer uma das outras características assim encontradas em WSCI. Apesar de haver alguma intersecção entre estas duas linguagens, WSCI fornece um conjunto de características úteis e necessárias a WSCL.

WSCL é um *note* do W3C desde março de 2002.

3.2.7 WSMF

WSMF (*Web Service Modeling Framework*) [44], ou framework de modelagem de serviços na Web, é um framework (não uma linguagem), proposto pela HP, que fornece um modelo conceitual para o desenvolvimento e a descrição de serviços na Web e suas composições.

Ele está centrado em um forte desacoplamento dos vários componentes que participam do processo de negócio e em um serviço de mediação, responsável por garantir a interação entre os componentes de maneira escalável.

O modelo de WSMF consiste basicamente de 4 elementos distintos:

– ontologias, que definem uma semântica formal para a informação e fornecem a terminologia utilizada por outros elementos;

trabalho que o gerou, o documento torna-se uma *Candidate Recommendation*.

- repositórios globais, que armazenam os objetivos, especificados através de pré- e pós-condições, que o requisitante do serviço tem ao invocar o serviço, ou seja, os problemas que este deve resolver. As pré-condições informam o que deve ser garantido para que um serviço seja invocado. As pós-condições descrevem o efeito da execução do serviço. Como um mesmo serviço pode servir a diferentes propósitos e diferentes serviços podem servir a um mesmo propósito, os objetivos devem ser especificados em separado da descrição do próprio serviço;
- serviços na Web, descritos por um nome, um objetivo, suas pré- e pós-condições, suas estruturas de dados de entrada e de saída, seu código de erro a ser retornado em caso de falha, os serviços que podem invocar para alcançar suas tarefas, suas portas de entrada e de saída (meios de acesso ao serviço), a seqüência das tarefas durante a sua execução (no caso de um serviço composto), a gerência de exceções, seus estados de execução, mensagens de reconhecimento, protocolos de troca de mensagens suportados e outras propriedades não-funcionais; e
- mediador, cuja função é resolver os problemas de interoperabilidade. WSMF considera o mediador como o responsável por permitir a "conversa" entre os parceiros de negócio em um ambiente onde estão presentes diferentes estruturas de dados, lógicas de negócio, protocolos de troca de mensagens e onde é possível a invocação dinâmica de serviços. Mediadores são preferidos aos esquemas de dados globais.

A partir desse modelo, o objetivo do WSMF é permitir que serviços na Web semanticamente descritos sejam automaticamente descobertos, selecionados e executados, inclusive em processos de negócios inter-organizacionais. WSMF pode ser visto como uma extensão de WSFL. Em [18] é apresentada a arquitetura proposta para o framework WSMF.

3.2.8 OWL-S

A arquitetura para serviços na Web semântica, apresentada em [77] e baseada na tecnologia de agentes foi a grande inspiradora de OWL-S (*OWL-based Web Services Ontology*), apresentada em detalhes no Capítulo 4 e definida pela OWL-S Coalition (formalmente DAML-S Coalition).

A OWL-S versão 1.0 foi publicada em novembro de 2003 e, exatamente um ano depois, em novembro de 2004, foi publicada pelo W3C a OWL-S versão 1.1 como uma *W3C Member Submission*[2].

OWL-S é a sucessora de DAML-S [76] e define uma ontologia de serviços desenvolvida como parte do projeto *DARPA Agent Markup Language* [37]. Apesar de bastante similar às idéias descritas em [18], OWL-S não está centrada em um serviço de mediação, mas fornece uma especificação declarativa de pré-requisitos e resultados de serviços individuais.

OWL-S possui um sistema de tipos muito mais rico do que o permitido pela linguagem XML e do que o usado nas diversas tecnologias para serviços na Web até o momento, o que permite descrever e restringir as capacidades de um serviço. Ainda, OWL-S consegue representar semanticamente as atividades de um processo, o que não é possível em BPEL4WS e BPML, porque elas não se utilizam de ontologias.

OWL-S oferece uma abordagem canônica, geral, através do uso de um vocabulário e de propriedades para um modelo de processo permanecer compatível com qualquer padrão de modelagem de processos. Em OWL-S, um serviço pode ser visto como um *black box* cujos componentes internos e detalhes de operação são escondidos do usuário final.

3.2.9 Comparação

A Tabela 3.1 apresenta uma comparação entre as linguagens descritas anteriormente.

Esta tese utiliza OWL-S, em especial a sua ontologia de processos, porque ela permite separação total entre a lógica do workflow e a tecnologia de implementação, viabilizada através do documento *ServiceModel*. Esta separação facilita encontrar e substituir workflows e recursos concretos ou abstratos por instâncias concretas em tempo de execução. Além disso, a ontologia de processos de OWL-S possui muitos dos conceitos que foram reconhecidos como necessários neste trabalho, dentre eles os conceitos de processos concretos e abstratos, de recursos, de pré-condições e de fluxo de dados. De fato, estas características de OWL-S motivaram a utilização

[2]Uma W3C *Member Submission* é simplesmente um conjunto de documentos produzidos por um grupo de trabalho fora do contexto do W3C e a ele submetido. Não significa, portanto, que a tecnologia será aprovada pelo W3C. Ela simplesmente torna pública a requisição de submissão pelo grupo que a realizou, o que provavelmente garante a leitura por diversos interessados e o possível alcance de um padrão W3C.

da ontologia de processos para a definição dos workflows, assim como no trabalho [36].

Porém, algumas extensões a OWL-S se tornaram necessárias:

- tratamento de exceções levantadas por *timeout*;
- inclusão de valor default para parâmetros de processos;
- inclusão de construções sintáticas adicionais (classes e propriedades) para uso pelo mecanismo de tratamento de exceção, atribuindo valores a relacionamentos;
- inclusão de um novo construtor, *ForAll*.

As extensões realizadas à linguagem OWL-S e a definição de uma semântica formal para a linguagem estão entre as principais contribuições desta tese.

3.3 OWL-S: Funcionalidades e Semântica

3.3.1 Preliminares

Existem basicamente duas linhas de esforços distintas no que diz respeito ao estudo da linguagem OWL-S. A primeira delas está focada em propostas de melhorias para evitar inconsistências e levantar limitações. A outra linha de estudo está focada na semântica de OWL-S, até agora não definida oficialmente.

Balzer et al. em [10] discutem algumas das limitações e dos problemas da linguagem OWL-S versão 1.0, dentre os quais podemos citar:

	WSFL	**BPEL4WS**	**BPML**	**WSCI**	**WSCL**	**WSMF**	**OWL-S**
Quem propôs	IBM	IBM, BEA e Microsoft	BPMI.org	Sun, BEA e Intalio	HP	HP	OWL-S Coalition (formalmente, DAML-S Coalition)
Conceito básico	Composição de serviços como grafos direcionados, que podem ser aninhados mas não acíclicos	Modelo de negócio e uma gramática para descrever o comportamento de um processo baseado nas interações entre os processos e seus parceiros (interação peer-to-peer)	Máquina de estados transacionais finita	Troca de mensagens é descrita do ponto de vista de cada serviço (visão do processo orientada a mensagens)	Conversação é definida sob a perspectiva do provedor do serviço. Define a ordem em que as mensagens são trocadas	Centrado no forte desacoplamento dos vários componentes que participam do processo de negócio e em um serviço de mediação	Cada composição de serviços pode ser vista como um processo
Compatibilidade com	SOAP, UDDI e WSDL	XMLSchema, XPath, WS-Addressing, WSDL	SOAP, UDDI, WSDL, XMLSchema e XPath	WSDL	WSDL, XMLSchema		WSDL
Semântica associada	Não	Não	Não	Não	Não	Sim, através do uso de ontologias	Sim, através do uso de ontologias
Gerência de exceções	Sim	Sim, através de *fault handlers* e *compensation handlers*	Sim	Sim	Não	-	Não
Suporte a transações	Não	-	Sim	Sim	Não	-	Não

Tabela 3.1: Comparação entre as linguagens de composição de serviços descritas.

(i) nas restrições de cardinalidade impostas às propriedades, as especificações de tipo são inexistentes;

(ii) as especificações de propriedades muitas vezes estão de acordo com a linguagem RDF, e não com OWL-DL, como deveria;

(iii) a definição de propriedades funcionais está incorreta;

(iv) muitas construções tornam a ontologia de OWL uma ontologia OWL-Full, o que impede a utilização dos mecanismos de verificação automática desenvolvidos para OWL-DL. Na linguagem OWL-Full, uma classe pode também ser um indivíduo e é permitido o uso de construções mais sofisticadas se comparadas com as construções permitidas em OWL-DL. No entanto, uma ontologia em OWL-Full pode não ser decidível;

(v) a impossibilidade de referenciar parâmetros particulares de um processo para descrever o fluxo de dados, pela falta do conceito de variáveis em OWL;

(vi) a falta de um mecanismo de tratamento de exceção.

Alguns destes problemas já foram sanados na versão 1.1 de OWL-S. Por exemplo, a ontologia OWL-S 1.1 classifica-se como OWL-DL, resolvendo o problema (iv) e, conseqüentemente, o problema (ii). O problema de referenciar um parâmetro de uma determinada instância de processo também foi parcialmente resolvido em OWL-S 1.1 através do conceito de *TheParentPerform*. Uma discussão mais detalhada sobre este problema pode ser encontrada no Capítulo 8. O problema de tratamento de exceção, no entanto, não foi abordado na versão 1.1 da linguagem. Este trabalho apresenta uma proposta neste sentido, como será visto nos próximos capítulos.

Em [79] são identificados diversos problemas semânticos com relação à linguagem OWL-S:

- possui ambiguidade conceitual;
- não possui uma axiomatização concisa;
- foi definida de forma imprecisa;
- oferece uma visão bastante limitada de serviços na Web.

Segundo descrito em [79], a ambiguidade conceitual afeta sobretudo a ontologia de serviços de OWL-S. Apesar do conceito de serviço estar vinculado a sites Web, a programas e a dispositivos acessíveis através da

Web, na ontologia o conceito "Web" ou de "site Web" não aparece, em uma demonstração de uma ontologia mal projetada.

Além disso, os autores apontam que a ontologia, além de não descrever conceitos precisamente, não possui uma hierarquia bem definida, sendo muitos conceitos subconceitos diretos de conceitos do mais alto nível. Ainda, o problema (v) apontado em [10] também é apontado em [79].

Um outro problema apontado é o fato do conceito *Service* estar relacionado ao conceito *ServiceModel* com cardinalidade de 1 para 1. Para solucionar este problema, os autores de [79] propõem um alinhamento da ontologia de OWL-S com uma ontologia de referência, no caso a ontologia DOLCE.

Por fim, não existe oficialmente, até o momento, nenhum documento que defina a semântica de OWL-S. No entanto, alguns trabalhos têm sido publicados na literatura no intuito de definir uma semântica para a linguagem. Em [8] é definida uma sintaxe formal para OWL-S e uma semântica operacional baseada em Erlang e Concurrent Haskell.

Em [86] é definida a semântica para um subconjunto da linguagem OWL-S em Lógica de Primeira Ordem. Posteriormente, é apresentado um modelo de OWL-S em Redes de Petri estendidas, escolhidas por permitirem modelar eventos e estados em um sistema distribuído, e modelar controle seqüencial, concorrente e baseado em eventos assíncronos.

3.3.2 Comparação

Alguns problemas sintáticos da linguagem OWL-S foram listados em [10, 79]. Esta tese define extensões para a linguagem que possam cobrir as limitações existentes, conforme apresentado no Capítulo 6. Em particular, definimos um outro construtor, chamado *ForAll*, e diversas outras propriedades para tratarem da flexibilização da execução pelo tratamento de exceções provenientes de situações diversas.

Diferentemente de [8, 86], este trabalho apresenta uma semântica operacional para OWL-S, sobretudo para o *ServiceModel*, baseada na noção de uma máquina abstrata.

3.4 Frameworks e Protocolos para Composição e Coordenação de Serviços

3.4.1 Preliminares

Não apenas a composição, mas também a coordenação de serviços na Web é necessária. Para isso, estão sendo desenvolvidos novos protocolos de coordenação, já que transações envolvendo serviços são diferentes das transações convencionais porque são normalmente de longa duração e necessitam de níveis de isolamento mais relaxados, como apresentado em [72].

Por exemplo, imagine que um usuário Web deseje efetuar reserva de passagens aéreas e de hotel para uma viagem que vá realizar. No entanto, a reserva de passagens aéreas é oferecida como um serviço separado do serviço de reserva de hotel. Deste modo, a reserva de passagens pode, por exemplo, ser efetuada antes que seja confirmada a reserva de hotel. Considere então que a reserva de hotel falha, ou seja, que não existem mais vagas no hotel e na data especificados pelo usuário. Neste caso, como o ambiente é a Web e o usuário está rodando serviços remotos espalhados pela rede, ele não tem acesso ao banco de dados que diz respeito à reserva de passagens e, por isso, não pode cancelá-las. Assim, é necessário que exista um serviço compensatório que o usuário possa acessar e efetuar o cancelamento da reserva das passagens aéreas. Além disso, nenhum serviço sabe a priori de que transação vai participar, e nem mesmo a transação que os invoca tem poder sobre a sua execução. Um serviço invocado é executado na máquina de quem o disponibiliza e o cliente deste serviço apenas recebe a resposta de sua execução.

Nesse contexto, mecanismos específicos de coordenação de serviços devem ser adotados na Web para permitir que um conjunto de serviços possam participar de transações e que o resultado final das transações seja consistente.

3.4.2 METEOR-S WSCF

O framework WSCF (*Web Service Composition Framework*) de composição de serviços descrito em [23, 111, 3] faz parte do projeto METEOR-S (*Managing End-To-End OpeRations for Semantic web services*). Um dos aspectos principais deste framework de composição

está na definição de templates de processos semânticos (SPTs, em inglês), que nos permitem semanticamente definir cada atividade envolvida em um processo. De acordo com um template, diversos processos distintos podem ser gerados a partir de implementações diferentes de serviços na Web para cada uma das atividades especificadas. Um template indica as características do serviço na Web necessárias para a execução do processo.

O framework do METEOR-S realiza a composição de serviços não apenas de acordo com os parâmetros de entrada e saída de cada serviço (*matching* sintático), mas também de acordo com as suas especificações de qualidade de serviço (QoS). METEOR-S leva com consideração um modelo baseado nos seguintes aspectos:

- tempo de execução, como medida de performance;
- custo associado à execução do serviço;
- confiabilidade; e
- disponibilidade, que corresponde à probabilidade dos serviços serem executados assim que forem demandados.

Serviços no METEOR-S são semanticamente anotados e publicados para posterior descoberta. Processos abstratos são definidos sem *binding* para serviços na Web implementados e serviços específicos são buscados em tempo de execução. O *matching* é feito mantendo a descrição do template do processo abstrato e de acordo com as restrições impostas pelos processos. A abordagem de METEOR-S é representar todos os critérios que afetam a seleção de serviços como restrições ou objetivos, transformando o problema para um problema de otimização ou satisfação de restrições. Estas restrições, por sua vez, podem dizer respeito à qualidade do serviço e podem estar vinculadas a custos, de acordo com um modelo de custos definido no sistema, para dar prioridade maior ou menor para cada fator no momento da seleção dos serviços.

Este framework é bastante interessante porque além do conceito de template de processo, ele permite o *binding* dinâmico de serviços e ainda trata não apenas do *matching* sintático, mas também do *matching* de serviços de acordo com restrições impostas pelos processos, restrições estas que podem ser semanticamente definidas.

3.4.3 WebComposer

WebComposer é uma ferramenta para a implementação e a execução automáticas de workflows baseados em serviços na Web [36]. Nesta ferramenta, workflows são descritos através do documento *ServiceModel* de OWL-S, o qual informa as entradas, saídas, pré-condições e efeitos de cada processo que é implementado por uma operação. Como o *ServiceModel* não possui informação de implementação, ele permite que uma descrição seja compartilhada entre diferentes implementações de um mesmo processo abstrato. Esta característica do documento *ServiceModel* de OWL-S torna a descrição do workflow independente de alterações na interface dos serviços na Web disponíveis e das tecnologias utilizadas na sua invocação.

A execução dinâmica de workflows descrita em [36] pode ser dividida em três fases: (i) descoberta de serviços; (ii) seleção de serviços dentre os serviços na Web encontrados na fase anterior; e (iii) composição de chamadas. A especificação de um workflow através da ferramenta WebComposer pode ser obtida como um programa Java.

3.4.4 WebTransact

O framework *WebTransact* trata da construção de composições de serviços na Web, autônomos e heterogêneos [97, 98]. Basicamente, a arquitetura de *WebTransact* é composta de 4 camadas:

- camada de descrição de serviços;
- camada de integração de serviços, composta por serviços remotos que representam o mapeamento de serviços nas mais diferentes interfaces para serviços cujo formato das mensagens é o formato entendido pelo mediador;
- camada de agregação, que compreende serviços mediadores que integram serviços remotos semanticamente equivalentes, fornecendo uma visão homogênea de serviços na Web heterogêneos;
- camada de composição, que representa os padrões de interação transacionais entre os serviços mediadores, gerando serviços mediadores compostos que podem, por sua vez, ser usados por programas de aplicações ou expostos como novos serviços complexos na Web.

Esse framework está baseado em duas linguagens: WSDL, através da qual um serviço remoto entende como interagir com um serviço na Web, e WSTL (*Web Services Transaction Language*), através da qual o serviço remoto conhece o comportamento transacional de um serviço na Web. WSTL é uma linguagem proposta dentro do escopo do framework *WebTransact*, e que se diferencia da maioria das demais citadas, por exemplo de WSFL, uma vez que estas não tratam do problema de homogeneização de serviços nem mesmo consideram a coordenação e a mediação de serviços na Web com suporte transacional diferente. WSTL, ainda, se diferencia do protocolo BTP a ser apresentado adiante por não definir padrões de mensagens específicos, fornecendo maior flexibilidade enquanto preserva a autonomia dos serviços. A arquitetura distribuída de *WebTransaction* separa a tarefa de agregação e homogeneização de serviços na Web da tarefa de especificação de padrões de interação transacionais, fornecendo assim um mecanismo geral para lidar com a complexidade introduzida por um grande número de serviços heterogêneos.

3.4.5 WS-Coordination / WS-Transaction

O WS-Coordination e o WS-Transaction são propostas de padrão de coordenação de serviços definidas em conjunto pela Microsoft, pela IBM e pela BEA.

Resumidamente, WS-Coordination é um framework extensível para coordenação de ações em aplicações distribuídas, enquanto WS-Transaction é o framework responsável por fornecer os protocolos de coordenação propriamente ditos, de acordo com o tipo de atividade a ser coordenada. Para o trabalho de coordenação, o WS-Coordination conta com:

- um *serviço de ativação*, que permite a uma aplicação criar um novo contexto de coordenação ou propagar um contexto já existente, resultando em níveis de coordenação aninhados (de forma similar ao mecanismo de coordenação em transações aninhadas);
- um *serviço de registro*, que permite que uma aplicação se registre para um protocolo de coordenação, que deve ditar o modo como vai se comportar durante a execução;
- dois tipos de coordenação específicos e um conjunto de *protocolos de coordenação* associado. O tipo de coordenação "transação atômica" (AT, do inglês *Atomic Transaction*) é usado para coordenar atividades

de pequena duração, limitadas a domínios confiáveis e com a propriedade *all or nothing*. O outro tipo de coordenação, a "atividade de negócio" (BA, do inglês *Business Activity*), é usado para coordenar atividades de longa duração, cujos requisitos não permitem a aplicação das propriedades ACID e cuja lógica de negócio é utilizada para manipular exceções durante a execução. Atividades de negócio, ao contrário de atividades atômicas, não bloqueiam recursos e suas ações são aplicadas imediatamente, sendo permanentes. Como não são tentativas, estas ações nas atividades de negócio apenas podem ter seus efeitos semanticamente desfeitos através de ações compensatórias.

A Figura 3.2 ilustra como pode ocorrer a coordenação de dois serviços distintos. Conforme o exemplo da figura, o serviço Web representado pela "Aplicação 1" envia uma mensagem ao seu serviço de ativação, requisitando a criação do contexto da transação. Este serviço de ativação manda a resposta à "Aplicação 2" requisitante, contendo tanto o contexto da transação quanto a identificação do serviço de registro e a identificação do próprio serviço que se está ativando. A informação de contexto é a informação que descreve de qual transação o serviço está fazendo parte.

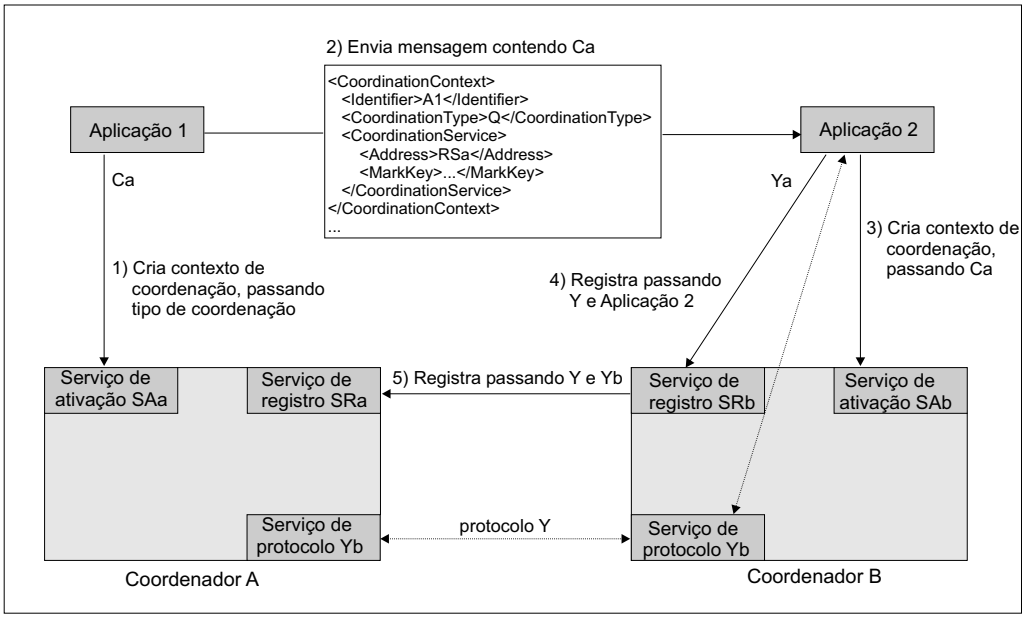

Figura 3.2: Exemplo de coordenação envolvendo dois serviços distintos, cada qual com o seu próprio coordenador [20].

Ao invés de se registrar junto ao mesmo coordenador da "Aplicação 1", a "Aplicação 2" prefere se registrar junto ao seu próprio coordenador. Assim, esta última aplicação envia uma mensagem ao seu serviço de ativação, informando o contexto da coordenação como sendo o contexto da

coordenação que recebeu da "Aplicação 1". Este serviço de ativação então cria sua própria informação de contexto, contendo o mesmo serviço passado pela "Aplicação 1", o mesmo identificador da transação (com o mesmo tipo de coordenação), mas com seu próprio serviço de registro.

A "Aplicação 2" determina então os protocolos de coordenação suportados pelo serviço correspondente. O registro através da "Aplicação 2" se propaga até o serviço de registro da "Aplicação 1", evidenciando uma ligação lógica entre as duas aplicações, que são agora coordenadas pelos seus respectivos coordenadores, estando o coordenador da "Aplicação 2" submetido ao da "Aplicação 1".

Mensagens de erros padrões são definidas em WS-Coordination para que tanto o participante quanto o coordenador possam avisar às partes envolvidas sobre a ocorrência de falhas (por exemplo, protocolo ou parâmetro inválido).

A especificação de WS-Transaction [19, 21], por sua vez, fornece a definição de protocolos de coordenação para os dois tipos de coordenação definidos:

- as atividades atômicas, nas quais tipicamente o participante bloqueia os itens de dados envolvidos na transação para prevenir alterações enquanto a transação está sendo processada; e
- as atividades de negócio, para as quais o principal conceito é o de compensação. Ao invés de cada participante da transação bloquear os itens de dados e liberá-los assim que tornarem permanentes as suas alterações, a compensação assume que todas as atualizações obterão sucesso e as alterações são imediatamente feitas permanentes. No entanto, é possível desfazer as alterações compensando a ocorrência de alguma falha não prevista.

Pelo fato de alterações serem persistidas imediatamente, transações de atividades de negócio tornam resultados intermediários visíveis a outros participantes enquanto transações atômicas oferecem um alto nível de isolamento entre as transações em execução.

Para as atividades atômicas, são definidos os protocolos *Completion*, *Volatile 2PC* e *Durable 2PC*, descritos em [19]. O protocolo *Completion* é usado por uma aplicação para informar ao coordenador para tentar terminar a transação ou abortá-la. Segundo o protocolo *Volatile 2PC*, quando o coordenador recebe uma notificação de *Commit*, ele inicia a fase de preparação e todos os participantes registrados para este protocolo devem responder antes que a fase de preparação se inicie para os participantes

registrados para o protocolo *Durable 2PC*. O contrário acontece com relação ao protocolo *Durable 2PC*.

Para as atividades de negócio, são definidos no framework WS-Transaction dois protocolos: o *BusinessAgreementWithParticipantCompletion* e o *BusinessAgreementWithCoordinatorCompletion*. Segundo o primeiro protocolo, o participante deve saber quando todo o trabalho foi completado pela atividade de negócio, enviando assim a notificação de *Completed*. De acordo com o segundo protocolo, o coordenador é quem avisa ao participante que a atividade de negócio foi finalizada.

O framework WS-Coordination é importante para o funcionamento do WS-Transaction, pois permite que:

- seja criado um novo contexto de coordenação para uma transação atômica;
- seja adicionado um coordenador sob os cuidados do coordenador já existente da transação;
- seja propagado o contexto da coordenação através da troca de mensagens;
- um participante se registre junto a um protocolo de coordenação, dependendo de seu papel na atividade. Cada um destes protocolos é um dos protocolos de coordenação de transações atômicas definidos na especificação: *Completion*, *Volatile 2PC* and *Durable 2PC*.

De fato, WS-Transaction define como serviços na Web registram sua intenção de usar os protocolos de coordenação e como eles comunicam a informação de status (tais como as decisões de vetar a continuação da transação da qual participa) para outros serviços [134]. Ele define ainda como um serviço na Web pode requisitar a outro serviço para compensar alterações previamente realizadas [134].

A Figura 3.3 apresenta uma visão geral do funcionamento de WS-Coordination e de WS-Transaction. De acordo com esta figura, a aplicação troca informação de contexto com os serviços que invoca. Os módulos coordenadores trocam mensagens SOAP para a coordenação da transação e fazem a seleção do protocolo de coordenação a ser utilizado de acordo com a escolha dos participantes. A descrição dos serviços é geralmente realizada através de WSDL.

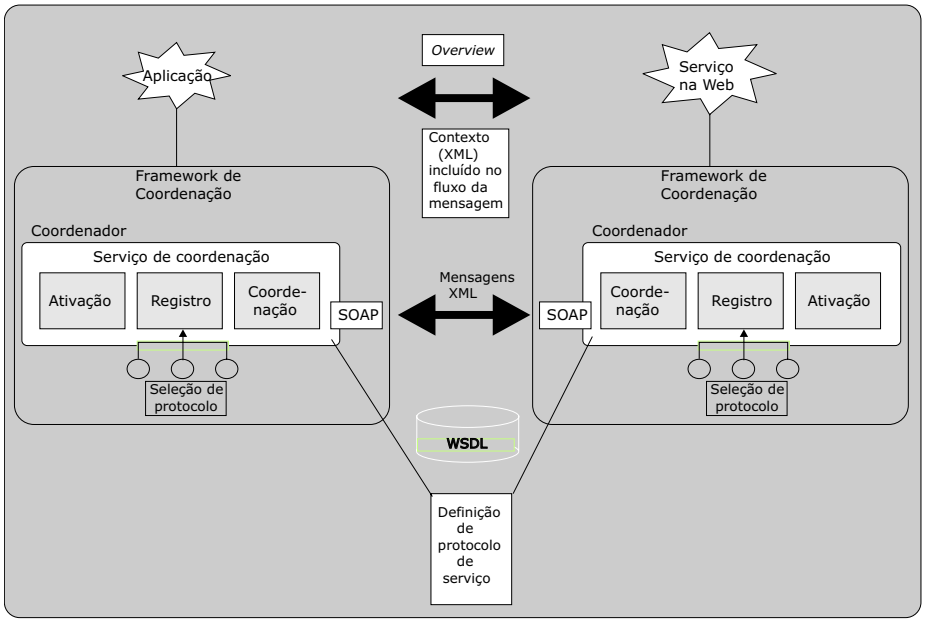

Figura 3.3: Visão geral do funcionamento de WS-Coordination e de WS-Transaction [45].

3.4.6 BTP

O protocolo BTP (*Business Transaction Protocol*) [72, 95], da OASIS, destina-se a atender, ao contrário dos frameworks WS-Coordination e WS-Transaction, não apenas aplicações de serviços na Web, mas qualquer tipo de aplicação distribuída. Com o uso do BTP, as aplicações têm completo controle sobre as fases de preparação e terminação e podem determinar quais transações devem ser completadas e quais devem ser canceladas, de acordo com a lógica do negócio.

Em [73, 33] é apresentada uma comparação entre o BTP e os frameworks de WS-Coordination e WS-Transaction. Dentre as diversas diferenças, é possível citar, por exemplo, que:

- o BTP não é apenas destinado ao ambiente de serviços na Web;
- no BTP coordenação e transação são tratados em conjunto, enquanto WS-Coordination trata da coordenação e WS-Transaction dos aspectos transacionais de uma aplicação distribuída. Não existe equivalente ao WS-Coordination no BTP;
- no framework WS-Transaction, se uma falha ocorre, uma atividade de negócio precisa ser compensada para deixar os dados em um estado consistente, uma vez que as finalizações são feitas imediatamente após o recebimento das mensagens associadas, ao contrário do que ocorre no BTP;

– BTP é uma especificação "fechada", restrita ao protocolo 2PC, enquanto WS-Transaction permite que novos protocolos sejam facilmente incorporados; e
– BTP não permite escopos aninhados (composição hierárquica de serviços), o que é permitido em WS-Transaction.

3.4.7 Comparação

Esta tese adota a estratégia de separação total entre a lógica do workflow e a tecnologia de implementação, viabilizada através do documento *ServiceModel* de OWL-S. Esta estratégia facilita encontrar e substituir workflows e recursos concretos e abstratos por instâncias concretas em tempo de execução. De fato, esta característica motivou a utilização da ontologia de processos de OWL-S para a definição dos workflows, à semelhança do WebComposer.

É importante frisar que esta tese não está direcionada para a orquestração de serviços na Web. Desta forma, possui objetivos distintos das várias propostas resumidas nesta seção, incluindo o *WebTransact*.

Porém, esta tese beneficia-se da pesquisa sobre protocolos de coordenação e orquestração de serviços. Ela propõe um protocolo de terminação que inclui testes de consistência pertinentes às decisões de flexibilização tomadas durante a execução, que é semelhante aos protocolos incluídos no framework WS-Transaction.

3.5 Matching Semântico de Serviços

3.5.1 TBPM

A abordagem de flexibilização adotada pelo projeto TBPM (do inglês *Task-Based Process Management*) [85] está centrada em uma máquina de workflow inteligente e no uso de ontologias de domínio. Foram identificadas neste projeto 7 ontologias, não independentes, como segue:

1. ontologia de artefatos: recursos físicos e componentes;
2. ontologia de informação: os tipos de informação, seu conteúdo e formato, que são usados, gerados ou transferidos durante o processo;

3. ontologia de tarefas: descreve de maneira genérica tarefas e processos que são comumente executados como parte do processo maior;

4. ontologia da estrutura organizacional: inclui a organização estática do negócio e dos participantes, além da especificação dos papéis que podem exercer cada indivíduo dentro da organização;

5. ontologia de capacidades ou de habilidades: descreve as habilidades específicas do domínio que cada participante pode ter;

6. ontologia de agentes: descreve os participantes no processo, tanto humanos quanto agentes de software inteligentes. Possui um forte relacionamento com a ontologia de capacidades - agentes possuem capacidades que os possibilitam participar de um processo. Esta ontologia de agentes pode ajudar o sistema a inferir as capacidades e interesses dos agentes através de seu tipo;

7. ontologia de suporte computacional: descreve a estrutura de hardware e software na qual as partes do processo centradas na informação são conduzidas.

De posse destas ontologias, o sistema do TBPM é capaz, por exemplo, de em tempo de execução determinar como a delegação de tarefas será feita, assegurando que os agentes mais adequados sejam selecionados.

Em [84], também dentro do contexto do projeto TBPM, os autores dão ênfase no uso de uma biblioteca de planos, que nada mais é do que um banco de dados de estruturas de processos e um conjunto de relações entre estruturas e tipos de tarefas. Assim, cada plano definido a partir da biblioteca de planos especifica um conjunto de tarefas e as restrições de ordem e fluxos de objetos entre elas. Esta capacidade de se construir modelos de processos em tempo de execução combinando os planos presentes na biblioteca introduz um alto nível de flexibilidade ao sistema de workflow. Junto à biblioteca, também estão definidas ontologias de domínio que contribuem com o conhecimento necessário para o auxílio ao usuário na tomada de decisões.

A flexibilidade proposta em [63] leva em consideração o fato de que sistemas de gerência de workflows devem dar suporte à tomada de decisões por parte dos usuários. Dessa forma, usuários podem, por exemplo, escolher as atividades necessárias para se alcançar uma tarefa e os agentes que deverão executá-la. Estas decisões podem ocorrer em três fases distintas: na inicialização de tarefas, no planejamento de cada tarefa e no escalonamento

das atividades. A abordagem é chamada de *knowledge-based capability matching* e está diretamente relacionada com o uso de conhecimento a respeito de capacidades, tanto de atividades quanto de agentes, para a tomada de decisões. Estas capacidades estão descritas em uma ontologia, chamada *Enterprise Ontology*. O *matching* de capacidades retorna uma lista ordenada dos agentes disponíveis no momento, de acordo com o quão próximo estão das capacidades requeridas pela atividade.

3.5.2 Vispo

Em [69] é apresentado um sistema baseado no conceito de agentes, capaz de compor serviços na Web e usar descrição semântica na descoberta de serviços. Neste sistema, questões relacionadas à não-disponibilização de um serviço em determinado momento são tratadas. Neste caso, serviços semanticamente equivalentes são procurados, de forma semelhante a que ocorre no trabalho anteriormente descrito, apresentado em [98].

De fato, de acordo com [69], para se encontrar um serviço na Web para substituir um outro serviço que falhou, são necessários os seguintes passos:

- identificação da funcionalidade requerida do serviço;
- *matching* semântico entre os serviços;
- criação ou atualização do workflow;
- execução e monitoramento do serviço escolhido dentro do contexto de execução do workflow.

Em [94], o *matching* de capacidades é feito a partir da informação do que o serviço faz. Na comparação entre o serviço buscado e os serviços registrados, são levados em consideração os parâmetros de entrada e de saída do serviço. Para estes parâmetros, existe uma hierarquia de conceitos que define o relacionamento semântico entre eles, de modo que inferência pode ser feita sobre a hierarquia. Ainda, este trabalho propõe a adição de uma camada semântica em UDDI, para que serviços possam ser buscados não apenas por palavras-chave, mas também pelo que eles de fato fazem. O *matching* não precisa ser exato: o grau de *matching* é dado de acordo com a distância semântica mínima na hierarquia de conceitos entre os parâmetros do serviço buscado e os parâmetros dos serviços registrados. A proposta é de um *matching* flexível.

Um modelo de análise de compatibilidade para suporte à substituição de serviços na Web dentro de processos cooperativos também é apresentado

em [9]. Este modelo está descrito dentro do escopo do projeto italiano VISPO (*Virtual district Internet-based Service PlatfOrm*), cujo objetivo é oferecer um ambiente para execução de processos cooperativos, com suporte a composições dinâmicas de serviços na Web, considerando não apenas o projeto de serviços complexos a partir de serviços simples, mas também a substituição dinâmica e semi-automática de serviços na Web que falharam ou que foram modificados.

Dentro do escopo desse projeto VISPO, um processo cooperativo é definido como um conjunto de descrições abstratas de serviços na Web, que indicam as funcionalidades necessárias no processo. Em tempo de execução é buscado o serviço concreto a ser executado, com base na descrição abstrata do serviço.

No entanto, a busca por serviços na Web compatíveis com a descrição abstrata também pode se fazer necessária quando um serviço falha durante execução, quando uma funcionalidade é garantida por um melhor serviço ou mesmo quando uma nova versão do serviço se torna disponível. Assim, uma classe de compatibilidade é definida como o conjunto de serviços na Web que podem ser mutuamente substituídos uns pelos outros, dada uma definição de serviço abstrata.

A análise de compatibilidade apresentada em [9] é feita com relação às informações de entrada e de saída dos serviços e também às operações que eles suportam. Para esta análise, é utilizado o conceito de descritor de serviços, que é representado por uma tripla (operações, conjunto de entrada, conjunto de saída). Esta tripla é diretamente derivada da descrição do serviço na linguagem WSDL.

Informações de mapeamento são necessárias para o *matching* de serviços porque serviços compatíveis podem ter diferentes sintaxe e semântica. Dessa forma, é preciso identificar todos os serviços na Web pertencentes a uma classe de compatibilidade e definir o mapeamento para a substituição dos serviços.

Essa análise de compatibilidade ocorre em duas fases. Na fase de análise em alto nível, através dos descritores dos serviços e da informação semântica que os descreve, são encontrados os possíveis serviços compatíveis, dada uma determinada definição abstrata. Esta análise é feita com base em 3 coeficientes: o coeficiente de similaridade baseado na entidade, o coeficiente de similaridade baseado na funcionalidade e o coeficiente de similaridade global.

O primeiro deles é avaliado com base na afinidade entre os nomes dos parâmetros de entrada e de saída das operações do serviço. Esta afinidade é

computada com base em uma ontologia de relacionamentos terminológicos, enriquecida de conhecimento específico do domínio. O coeficiente de similaridade com base na funcionalidade é computado comparando-se a afinidade das operações nos seus descritores correspondentes. Duas operações são similares se suas ações, suas entidades de informação de entrada e de saída têm afinidade na ontologia terminológica definida. Por fim, o coeficiente de similaridade global de dois serviços é a medida de seu nível de similaridade total computada como a soma ponderada dos dois outros coeficientes.

Na fase de análise estrutural, para cada serviço na Web selecionado na fase anterior, é analisada sua estrutura em termos das operações que podem ser invocadas e das mensagens e dados que podem ser trocados. Ao final, são geradas as informações de mapeamento necessárias para passar de uma visão abstrata para cada visão concreta correspondente.

3.5.3 Comparação

Conforme anteriormente mencionado, o mecanismo de tratamento de exceção que viabiliza a flexibilização da execução proposto nesta tese utiliza ontologias (incluindo regras) para tomar decisões, durante a execução, sobre como substituir workflows (chamados nesta tese de processos, por analogia a OWL-S) ou recursos, sobre como encontrar instâncias concretas para workflows ou recursos abstratos, sobre como desfazer os efeitos de um workflow e sobre como atribuir valores default a parâmetros.

O mecanismo proposto assemelha-se às abordagens para alteração de workflows encontradas em [29, 65]. Estas abordagens baseiam-se em uma ontologia de regras para determinar como alterar a estrutura de um workflow, mantendo a sua corretude. A semelhança destas abordagens com o mecanismo apresentado nesta tese está no uso de informação adicional, descrita por meio de ontologias, para guiar as alterações ocorridas. No caso do mecanismo proposto, a ontologia de workflows não auxilia a modificação estrutural do workflow, mas a escolha de componentes alternativos em tempo de execução.

Considera-se que o mecanismo proposto é também próximo à estratégia adotada pelo TBPM para construir modelos de processos em tempo de execução, combinando os planos presentes em uma biblioteca de planos, utilizada em conjunto com uma ontologia de domínio. De fato, a estratégia do TBPM pode ser comparada ao mecanismo de tratamento

de exceção, quando a exceção gerada está relacionada à concretização de objetos (recursos ou workflows) abstratos, já que um workflow contendo subworkflows abstratos assemelha-se à definição de um plano que deve ser concretizado em tempo de execução com auxílio de uma ontologia.

O mecanismo proposto também pode ser comparado com os mecanismos para substituição de serviços, apresentados em [69, 94, 9]. Genericamente, um processo de substituição semântica de serviços primeiro identifica a funcionalidade do serviço a ser substituído, em seguida pesquisa um serviço alternativo semelhante ao serviço inicial e, por fim, substitui o serviço inicial pelo seu alternativo. Da mesma forma, o mecanismo de tratamento de exceção proposto nesta tese, quando invocado para a substituição ou concretização de workflows ou recursos, escolhe alternativas (ou instâncias concretas) destes objetos (workflows ou recursos), mantendo proximidade semântica, conforme previamente definido na ontologia.

No entanto, é importante deixar claro que a busca por workflows alternativos nesta tese leva em consideração o relacionamento de proximidade semântica definido, mas não realiza, por exemplo, análise de compatibilidade entre os parâmetros de entrada e saída. Nesta tese, se dois workflows são ditos semanticamente próximos na ontologia, é porque foram analisados em uma etapa anterior, fora do escopo desta tese, e assim considerados. O que existe, conforme será visto nos próximos capítulos, é uma informação de mapeamento entre parâmetros e recursos de processos considerados compatíveis.

É importante também frisar que o trabalho aqui apresentado não depende de forma alguma da tecnologia de serviços na Web. O mecanismo de tratamento de exceção proposto se aplica a sistemas de gerência de workflow de uma maneira geral e neles workflows podem estar ou não descritos como composições de serviços na Web. No entanto, os problemas relativos às composições de serviços e ao sistema aqui proposto possuem grande semelhança, conforme apresentado.

3.6 Resumo

As propostas existentes para a flexibilização da execução de workflows são geralmente voltadas para permitir a reestruturação dos workflows em tempo de execução.

A idéia deste trabalho é permitir que situações não previstas na modelagem possam ser acomodadas pela máquina de execução durante a

execução de uma instância de workflow, sem que a estrutura estática deste workflow seja modificada.

Além do mais, o uso de ontologias no processo de flexibilização possibilita construções semânticas que facilitam, muitas vezes, a execução de uma instância pela máquina de execução, através da tomada dinâmica de decisões sobre como substituir workflows ou recursos, sobre como encontrar instâncias concretas para workflows ou recursos abstratos, sobre como desfazer os efeitos de um workflow e sobre como atribuir valores default a parâmetros.

Diante das diversas linguagens existentes para a definição de workflows, OWL-S se apresenta, no contexto deste trabalho, como a linguagem mais adequada, uma vez que já possui um conjunto importante de definições necessárias à abordagem de flexibilização e, mais do que isso, pois permite novas extensões.

Ainda, este trabalho apresenta uma semântica operacional para OWL-S, baseada na noção de uma máquina abstrata, uma vez que não existe semântica formalmente definida para esta linguagem na literatura.

Por fim, observamos que os problemas inerentes ao ambiente de serviços na Web são semelhantes aos problemas relativos a sistemas de workflow. Em particular, as propostas relativas a composição e coordenação de serviços na Web podem ser generalizadas para sistemas de workflow. Em adição, a abordagem de *matching* de serviços para encontrar serviços equivalentes assemelha-se à abordagem de *matching* de processos e recursos que tratamos neste trabalho. No entanto, vale ressaltar que a proposta de flexibilização de workflows apresentada não depende da tecnologia de serviços na Web.

É importante também deixar claro que a busca por workflows alternativos nesta tese leva em consideração o relacionamento de proximidade semântica definido. Se dois workflows são ditos semanticamente próximos na ontologia, é porque foram analisados em uma etapa anterior, fora do escopo desta tese, e assim considerados.