9
Conclusões e Trabalhos Futuros

Este capítulo apresenta as conclusões desta tese e as sugestões de trabalhos futuros.

9.1
Conclusões

Esta tese endereçou um requisito de sistemas de workflow aqui chamado de *execução flexível*. Geralmente, sistemas de gerência de workflow interpretam rigidamente a definição do workflow, não permitindo qualquer tipo de desvio. No entanto, existem situações reais em que usuários devem poder desviar do fluxo estático especificado na definição do workflow por diversas razões, incluindo a falta de informação e a indisponibilidade dos recursos necessários à execução.

A maioria dos sistemas de workflow atuais, quando abordam a questão de flexibilização, o fazem através da alteração da descrição do workflow em tempo de execução. No entanto, nem sempre os desvios ocorridos em tempo de execução se repetem a ponto de justificarem uma alteração na estrutura do workflow.

Esta tese propôs então um enfoque, aqui chamado de *mecanismo de tratamento de exceção para a flexibilização*, permitindo basicamente que:

- a especificação completa de um workflow, chamado nesta tese de "processo", por analogia aos termos utilizados em OWL-S, seja adiada para o tempo de execução, ao incluir referências a processos ou recursos abstratos;
- a execução de um processo continue mesmo quando não estão disponíveis todos os valores de parâmetros necessários à execução, pelo uso de valores default;
- a execução de um processo continue mesmo quando os recursos necessários estão indisponíveis, pelo uso de recursos ou processos alternativos.

O mecanismo implementa uma estratégia de substituição de componentes, aplicada à execução de processos, onde subprocessos e recursos fazem o papel dos componentes [9]. Ele representa um meio de se continuar a execução mesmo diante de situações em que ela deveria ser interrompida, sem que seja necessária qualquer alteração na definição do processo.

O mecanismo utiliza uma ontologia de processos e recursos para encontrar:

- recursos ou processos concretos equivalentes a um recurso ou processo abstrato, quando um objeto abstrato é reconhecido em tempo de execução;
- valores default para parâmetros de processos, quando seus valores são desconhecidos;
- recursos ou processos alternativos, quando os recursos necessários estão indisponíveis;
- processos que tratem de outras exceções levantadas por uma instância de processo em execução.

Por fim, a semântica da linguagem de processos adotada captura tanto as características dos construtores convencionais, quanto o comportamento do mecanismo de tratamento de exceção para a flexibilização. Em particular, a semântica define um comportamento transacional para a execução de uma instância de processo, no sentido de garantir que ou todas as ações da instância terminam corretamente, ou todas são abandonadas e identificadas como tal.

Naturalmente, como o mecanismo não tem controle estrito sobre as ações atômicas, ele não desfaz automaticamente os efeitos das ações abandonadas, como em um sistema de gerência de banco de dados. Apenas nos casos em que o projetista previu de antemão, o mecanismo pode de fato desfazer os efeitos de uma ação atômica.

Em resumo, este trabalho definiu um enfoque permitindo que a modelagem de processos seja completada em tempo de execução, e que a execução das instâncias de processo fosse flexibilizada, permitindo sua continuação quando normalmente ela seria interrompida. O enfoque adotado definiu situações em que é importante flexibilizar e criou os meios para tratamento das exceções levantadas.

Esta tese possui duas contribuições centrais:

- o mecanismo de tratamento de exceção para a flexibilização;

– uma arquitetura distribuída para um sistema de gerência de workflow, incorporando o mecanismo.

A definição do mecanismo desdobra-se em duas contribuições:

– uma extensão das ontologias de processos e de recursos de OWL-S, para incorporar flexibilização à definição de um processo e guiar o mecanismo;
– a especificação de uma semântica formal para um fragmento da linguagem OWL-S, ainda não existente oficialmente na literatura, incluindo as extensões propostas, que possibilita a fácil compreensão da semântica da linguagem.

A definição da semântica operacional para parte de OWL-S, por meio do conceito de máquina abstrata, é atrativa, em especial, por dois motivos: (1) a máquina abstrata é de fácil entendimento, utilizando apenas conceitos simples; e (2) a definição pode ser feita de forma incremental, através de sucessivas extensões para acomodar construtores mais sofisticados.

Em particular, na versão 1.1 da linguagem OWL-S, um processo abstrato pode ter parâmetros e recursos, mas não existe ainda uma forma de especificar o mapeamento de parâmetros e de recursos entre um processo abstrato e os seus processos concretos relacionados. Uma contribuição subsidiária desta tese consistiu, então, na definição de:

– um mapeamento entre parâmetros de processos abstratos e parâmetros de processos concretos por meio das classes *pr:Relation_for_value* e *pr:ParameterMap* e de propriedades adicionais;
– um mapeamento entre recursos abstratos de processos abstratos e recursos abstratos de processos concretos por meio das classes *pr:Relation_for_value* e *pr:ResourceMap* e de propriedades adicionais.

As idéias apresentadas nesta tese são úteis a qualquer domínio de aplicação onde OWL-S seja utilizado. Além disso, são também úteis em qualquer sistema de gerência de workflow, desde que haja a tradução das idéias expressas em OWL-S para a linguagem de modelagem de workflow deste sistema.

9.2 Trabalhos Futuros

Diversos são os trabalhos futuros a serem propostos. Pelo fato desta tese ter utilizado a linguagem OWL-S para a descrição dos workflows, o primeiro trabalho futuro proposto é estudar com maior rigor a linguagem, identificando seus mais diversos problemas sintáticos e definindo semântica para construtores e propriedades ainda não claramente definidas. Problemas com OWL-S vão desde erros de definição de restrições de cardinalidade até erros lógicos, como definição de uma propriedade como uma *FunctionalProperty* de OWL. É importante que este estudo seja feito sobre a última versão disponível da linguagem, no momento, a versão 1.2 (disponível no endereço `http://www.daml.org/services/owl-s/1.2/`).

Outra questão relativa à OWL-S que deve ser estudada é o controle da execução de processos. A DAML-S Coalition está prevendo para as próximas versões da linguagem uma ontologia de controle de processos, que ajude na tarefa de monitoramento da execução. Para permitir o controle da execução de processos, propomos, em primeira instância, a criação de um log, para o armazenamento dos passos tomados por uma instância de processo.

Como segundo trabalho, propomos melhorar a definição da ontologia *pr* de processos e recursos para que ela possa ser escrita na linguagem OWL-DL. Atualmente, esta ontologia encontra-se em OWL-Full, o limita o uso de certos provadores automáticos disponíveis.

Um terceiro trabalho futuro é a aplicação de um mecanismo para refazer os efeitos da execução de um processo para o caso de conflitos detectados pelo uso de valores default de parâmetros por instâncias de processo. Conforme explicado nos Capítulos 7 e 8, quando um conflito é detectado, deveria ser empregado um mecanismo para refazer os efeitos da execução do processo que gerou o conflito, refazendo-o de acordo com o valor correto do parâmetro. No entanto, este mecanismo é bem complicado se considerarmos a hierarquia de coordenação formada durante a execução das instâncias e as dependências entre os processos envolvidos.

Como quarto trabalho futuro, propomos o emprego de uma abordagem de execução segundo a qual seja possível mudar o curso da execução de acordo com valores de variáveis que se alteram no tempo. Por exemplo, no caso de uma limpeza de áreas costeiras afetadas por derramamento de óleo, a previsão de quais áreas serão afetadas pode não ser confirmada porque no momento da previsão a direção do vento era uma, e logo após esta direção se alterou, fazendo com que outras áreas fossem afetadas.

Como outro trabalho futuro, propomos a elaboração de um algoritmo de otimização do uso de recursos. Por exemplo, o algoritmo pode evitar substituir um processo A por um processo alternativo B, se B for bloquear outro processo C que será executado em paralelo com A. Como outro exemplo, se a alternativa B escolhida for seguida por um processo D, então, se possível, os recursos que B usa devem ter uma intersecção não-trivial com os recursos requeridos por D (intuitivamente, B será capaz de deixar disponível para D o número de recursos necessários, reduzindo o tempo de D e evitando que D sofra exceção temporal de inicialização devido à indisponibilidade de recursos).

Um outro trabalho futuro proposto é a implementação da arquitetura proposta para um sistema de gerência de workflow. Esta implementação não necessariamente precisa ser iniciada no zero, pela construção de um novo sistema de workflow. Na verdade, poderia ser aplicado o mecanismo de tratamento de exceção para a flexibilização sobre um sistema de gerência de workflow já existente. Neste caso, as modificações tornariam o sistema existente um sistema dirigido por ontologia (do inglês *ontology-driven*), no qual a ontologia guiaria o funcionamento do mecanismo para a flexibilização.

Ainda, o mecanismo poderia ser aplicado utilizando uma linguagem de modelagem de processos/workflows diferente da linguagem OWL-S. Este mecanismo é independente de linguagem e deve poder ser aplicado em sistemas que descrevam processos em qualquer linguagem. Seria preciso apenas avaliar quais mudanças se tornariam necessárias na linguagem de descrição de processos para suportar o mecanismo proposto.

Como outro trabalho futuro, propomos a aplicação de um mecanismo que garanta que a execução de uma instância de processo continue mesmo depois da ocorrência de alguma falha no sistema, a partir do ponto em que a falha ocorreu.

Sugerimos como trabalho futuro também uma discussão mais aprofundada sobre a modelagem da classe *pr:Relation_for_value* e das classes e propriedades relacionadas, e sobre os impactos à ela relacionados.

Podemos ainda sugerir mecanismos para gerar explicações sobre as alternativas adotadas pelos mecanismos de flexibilização, como uma forma de melhorar a comunicação com o usuário. Tais mecanismos seriam baseados nas próprias ontologias e no log de execução.

Podemos sugerir como trabalho futuro também a análise em tempo de execução de alternativas mais econômicas, em termos dos custos definidos, para se executar determinados processos. Por exemplo, suponha que exista uma processo p definido como uma seqüência de dois subprocessos: p_1,

que determina a compra de carne em um açougue, e p_1, que determina a compra de pasta de dente em uma farmácia. Imagine que a farmácia fique no caminho do açougue. Portanto, seria melhor a seqüência p_2;p_1. Para que esta reestruturação dinâmica seja possível, é necessário um cálculo dinâmico a priori, de custos, e uma análise cuidadosa das pré-condições e recursos envolvidos.

Imagine, ainda, que existam dois outros subprocessos, p_1' e p_2', que determinem a compra de carne no supermercado e de pasta de dente no supermercado, respectivamente, e que sejam, nesta ordem, semanticamente próximos a p_1 e p_2. Considere que o custo definido do processo p_1 seje menor que o custo do processo p_1' e, da mesma forma, que o custo de p_2 seje menor que o custo de p_2'. No entanto, se for realizado um cálculo dinâmico de custos, levando em consideração as alternativas p_1' e p_2', respectivamente, aos processos p_1 e p_2, pode ser verificado que o custo da execução da seqüência p_1;p_2 é maior do que o custo da execução da seqüência p_1';p_2', mesmo considerando os custos individuais dos processos. Neste caso, o cálculo dinâmico indicaria a execução desta última seqüência ao invés da primeira seqüência, em razão dos menores custos envolvidos.

Por fim, sugerimos como trabalho futuro estudar a possibilidade de adotar uma variação da política otimista para a alocação de recursos, que calculasse o custo da perda dos recursos e utilizasse esse no cálculo de custo de utilização de outro processo.