Implementação de um Sistema de Composição Automática de Serviços Web (SCASW) Desacoplado: Módulos Interface e Tradutor

Juan Carlos Zuñiga[2], Felipe L. Pierin[1], José J. Pérez-Alcázar[1], Sergio. T. Kofuji[2], Adriano V. Fernandes[1]

[1]Escola de Artes Ciências e Humanidades – Universidade de São Paulo (EACH - USP)
Av. Arlindo Bettio, 1000 – CEP 03828-000, São Paulo, SP – Brasil

[2]Escola Politécnica – Universidade de São Paulo (EP - USP)
Avenida Prof. Luciano Gualberto, 158 – CEP 05508-900, São Paulo, SP – Brasil

{juan.carlos,fepierin,jperez}@.usp.br, sergio.kofuji@poli.usp.br

Abstract. *One of the goals in Semantic Web research has been to provide concepts, methods, and tools to combining several web services to provide a new value-added service which satisfy the user requests. Manual composition is an arduous and susceptible to errors task. To avoid this problem, methods and tools have been proposed for automatic composition of web services. This paper aims to describe the implementation of a system of automatic composition of Web services, specifically, two loosely coupled modules: the interface and the translator. These modules were developed to obtain greater flexibility and also to be used by non-technical users, as recommended by the design principles for service-oriented systems.*

Resumo. *Um dos objetivos da Web Semântica é fornecer conceitos, métodos e ferramentas para combinar diversos serviços web para gerar um novo serviço que satisfaça os requerimentos do usuário. Fazer este processo de forma manual é uma tarefa árdua e suscetível a erros. Métodos e ferramentas têm sido propostos para compor serviços web de forma automática. Este trabalho tem como objetivo descrever a implementação de um sistema de composição automática de serviços web, especificamente, a implementação dos módulos: Interface e Tradutor. Estes módulos foram desenvolvidos de forma desacoplada para obter maior flexibilidade e orientado para usuários não-técnicos, como recomendado pelos princípios de design para sistemas orientados a serviços.*

1. Introdução

Sistemas de composição automática de serviços web podem ser vistos como uma evolução dos sistemas baseados em serviços web tradicionais [Rao and Su 2004], oferecendo diversos benefícios em áreas como: e-business, e-government e bioinformática e gestão do conhecimento [Medjahed and Bouguettaya 2011]. O objetivo deste tipo de sistemas é compor (combinar), automaticamente, vários serviços web para produzir um novo serviço web que satisfaça as requisições do usuário.

Diversos trabalhos têm proposto ferramentas para compor serviços de forma automática nos últimos anos [Sycara et al. 2003], [Agarwal et al. 2005],

[Digiampietri et al. 2007] e [Kuzu and Cicekli 2010]. No entanto, estas propostas não trazem soluções completas, desacopladas, interorganizacionais e utilizáveis por usuários finais sem conhecimentos técnicos, como recomendado pelos princípios de design para sistemas orientados a serviços [Erl 2009].

Baseado na carência dessas características, este trabalho objetivou o estudo e desenvolvimento de um Sistema de Composição Automática de Serviços Web Semânticos (SCASWS) seguindo esses princípios de design. Para tal fim, foi estudada a arquitetura apresentada em [Zuñiga et al. 2010] e como contribuição foram implementados os módulos, de forma desacoplada, Interface (orientado para usuários não técnicos) e Tradutor (utilizando a ferramenta ANTLR [Espina 2011]).

O restante deste artigo está organizado da seguinte forma. A Seção 2 apresenta alguns conceitos básicos. A Seção 3 contém uma descrição dos sistemas de composição automática de serviços web. A Seção 4 apresenta o desenvolvimento dos módulos Interface e Tradutor. A Seção 5 contém os trabalhos relacionados e finalmente a Seção 6 apresenta as conclusões e trabalhos futuros.

2. Conceitos Básicos

Entre as soluções propostas para compor serviços web de forma automática destacam-se as baseadas em web semântica e planejamento.

2.1. Web Semântica

A Web Semântica procura associar um significado bem definido (descrições semânticas) às informações, dados e aplicações da web (p. ex. paginas web, serviços web, etc.) para que possam ser automaticamente entendidos e processados pelos computadores [Berners-Lee et al. 2001]. A solução proposta para a estruturação de descrições semânticas é o uso de ontologias. Uma ontologia é uma descrição de conceitos e relações entre conceitos que podem existir para um agente ou uma comunidade de agentes de software [Gruber 1993]. O uso de serviços web em conjunto com as ontologias têm sido uma área de intensa pesquisa, com objetivo de desenvolver modelos ontológicos que permitam a implementação dos chamados "Serviços Web Semânticos".

Dentre os modelos ontológicos mais citados na literatura estão: OWL-S (*Web Ontology Language for Services*) [Martin et al. 2004], WSDL-S (*Web Service Description Language for Services*) [Akkiraju et al. 2005] e WSMO (*Web Services Modelling Ontology*) [Lausen et al. 2005]. OWL-S é uma ontologia baseada em OWL (*Web Ontology Language*) [McGuinness and van Harmelen 2004] para a definição de serviços. Já WSDL-S permite adicionar anotações semânticas a documentos WSDL (*Web Services Description Language*) para descrever semanticamente as entradas, saídas e operações de um serviço web. Finalmente, WSMO provê um framework mais completo e claro para a descrição de serviços web semânticos, o que possibilita a automatização (total ou parcial) de tarefas como: descoberta, seleção, composição, execução e monitoração da execução de serviços. Além disso, o WSMO oferece uma maior expressividade na descrição dos serviços web semânticos, fatos que justificam a escolha deste modelo ontológico para nossa implementação. Nesse sentido, a continuação detalha-se os principais conceitos deste modelo ontológico.

2.1.1. WSMO

WSMO é um framework que implementa uma ontologia formal e conjunto de linguagens que permitem descrever todos os aspectos relevantes para acessar um serviço web de forma automática (total ou parcial). O modelo ontológico WSMO é composto por quatro elementos [Fensel et al. 2007]:

Ontologia: As ontologias fornecem a semântica formal das terminologias utilizadas por todos os outros elementos especificados no modelo ontológico WSMO;

Serviços Web: Um serviço web está associado a uma determinada ontologia de domínio. Serviços web são compostos de uma descrição formal que descreve: as funcionalidades do serviços, as propriedades não-funcionais e um método que permite interagir com o serviço web (interface). As funcionalidades do serviço web são descritas como capacidades (*capabilities*) e utilizam os seguintes conceitos para expressar restrições e relações entre serviços web:

- **Pré-Condições:** As pré-condições especificam as informações (parâmetros ou variáveis) requeridas antes da execução do serviço web, por exemplo, para comprar uma passagem de avião precisa-se de que o cliente tenha um Id de usuário e uma senha validos;
- **Suposições:** As suposições descrevem o estado suposto do serviço web antes da sua execução. Diferente das pré-condições, suposições não são necessariamente verificadas pelo serviço. Por exemplo, para comprar uma passagem de avião suponha-se que o cliente já esteja logado no sistema;
- **Pós-condições:** As pós-condições descrevem as informações (parâmetros ou variáveis) resultantes após a execução bem sucedida do serviço web, por exemplo, após comprar uma passagem de avião o cliente deverá receber o número do assento reservado;
- **Efeitos:** Os efeitos descrevem o estado resultante do serviço web após o sucesso da execução, por exemplo, após comprar uma passagem de avião e confirmar a reserva do assento o cliente é desconectado do sistema.

Metas (*Goals*): As Metas descrevem formalmente a funcionalidade desejada pelo usuário. Isto é, uma meta é o desejo do usuário que deveria ser satisfeito pelas funcionalidades de um serviço web ou pela combinação de diversos serviços web.

Mediadores: Os mediadores são responsáveis por lidar com problemas de interoperabilidade entre os diferentes elementos do WSMO. Mediadores podem ser de quatro tipos: **ggMediator**, mediador que liga duas metas. Esta ligação representa o refinamento de uma meta origem em uma meta alvo; **ooMediator**, mediador que importa ontologias e resolve possíveis combinações entre estas; **wgMediator**, mediador que liga serviços web a metas, significando que um serviço Web satisfaz (totalmente ou parcialmente) a meta com a qual ele é ligado; e **wwMediator**, mediador que liga dois serviços Web.

2.1.2. WSML

A implementação formal dos elementos do WSMO é feita através do *Web Service Modeling Language* (WSML), que é um conjunto de linguagens que foram projetados para descrever os aspectos funcionais, comportamentais e não-funcionais de todos os elementos

do WSMO [de Brujin et al. 2008]. O WSML é baseado em dois paradigmas: *Description Logic* e *Logic Programming*. Estes paradigmas permitem ao WSML descrever diversas variantes da linguagem (WSML-Core, WSML-DL, WSML-Flight, WSML-Full), que diferem em grau de expressividade lógica e complexidade computacional.

Entre as variantes, o WSML-Core é a menos expressiva, porém, permite a modelagem de classes, atributos, relações binárias, instâncias, hierarquias de classe e de relações de hierarquias. O WSML-DL permite a interoperabilidade com ontologias OWL. Por sua vez, o WSML-Flight é uma extensão do WSML-Core e acrescenta suporte a metamodelagem, restrições, negação não monotônica e interoperabilidade com RDFS. Já o WSML-Rule é uma extensão do WSML-Flight com maior expressividade lógica acrescentando suporte para a Lógica de Horn. Finalmente, o WSML-Full é a variante que generaliza o WSML-DL e WSML-Rule, porém, a semântica desta variante ainda não foi completamente especificada na versão atual do WSML [Medjahed and Bouguettaya 2011].

2.2. Planejamento

Entre as técnicas da inteligência artificial (IA) que permitem compor de serviços web podemos destacar a baseada em Planejamento, a qual estuda o processo de deliberação que escolhe e organiza ações, prevendo os efeitos da execução dessas ações buscando atingir da melhor forma possível alguns objetivos pré-definidos [Russel and Norvig 2003].

Um planejador é uma ferramenta de software que implementa uma técnica de planejamento. Para construir um planejador que seja capaz de resolver um problema de planejamento, é necessária uma linguagem formal para a representação: das ações (domínio de planejamento); e do estado inicial e meta (problema de planejamento). O domínio de planejamento descreve através de uma sintaxe formal a representação dos estados do mundo em ações dentro de um contexto específico. Já um problema de planejamento descreve um estado inicial do mundo, que é ponto de partida para chegar à meta especificada [Ghallab et al. 2004].

A linguagem padrão na área de planejamento é PDDL (*Planning Domain Definition Language*) [GHALLAB et al. 1998]. A descrição de um problema de planejamento a partir da linguagem PDDL acontece a partir de duas entradas: a definição do domínio e a definição do problema. A definição do domínio em linguagem PDDL é composto de requerimentos (requirements), tipos (types), constantes (constants), predicados (predicates) e ações (action). Os requerimentos são os subconjuntos de funcionalidades que um domínio demanda de suporte por parte dos planejadores. A importância dos requerimentos são, de acordo com [GHALLAB et al. 1998], uma forma de permitir com que um planejador descarte rapidamente um domínio que ele não seja capaz de atender. Ações avaliam parâmetros, pré-condições e efeitos a fim de determinar a execução das mesmas. A definição do problema descreve um estado inicial do mundo, que é ponto de partida da deliberação que escolhe e organiza ações para chegar à meta especificada [Russel and Norvig 2003].

3. Composição Automática de Serviços Web

Nesse sentido, este trabalho objetivou o estudo e desenvolvimento de um Sistema de Composição Automática de Serviços Web Semânticos (SCASWS) seguindo os princípios de design para sistemas orientados a serviços [Erl 2009]. Para tal fim, estudamos a arquitetura apresentada em [Zuñiga et al. 2010] e descrita na figura 1.

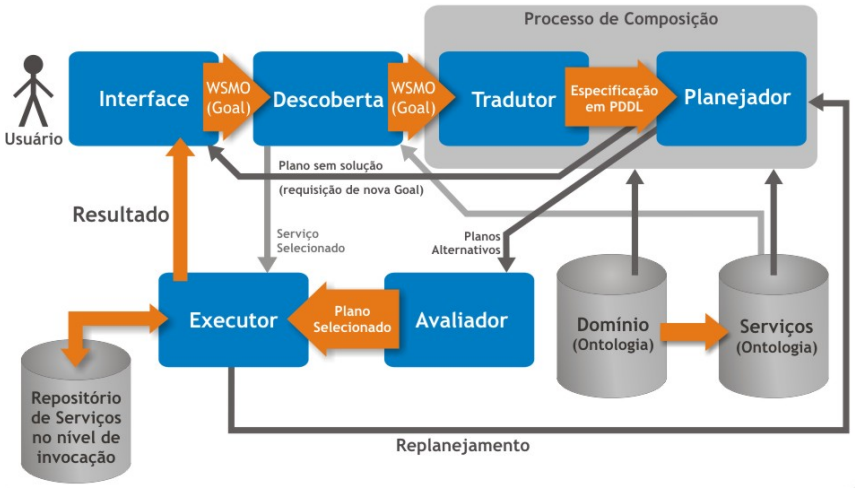

Figure 1. Arquitetura de Composição de Serviços Web proposta em [Zuñiga et al. 2010].

Cada caixa da figura 1 representa um módulo do sistema e as setas correspondem a interações entre os módulos. O módulo Interface é responsável pela interação entre o usuário e o resto do sistema. Ele permite ao usuário especificar suas requisições. Essas requisições são modeladas na forma de metas ou *goals* utilizando a linguagem WSML.

O Módulo Descoberta, encarrega-se de procurar um serviço adequado para a meta especificada. Este módulo executa algoritmos de casamento entre a *goal* e os serviços existentes no repositório de serviços. Se algum serviço satisfaz a requisição, este é enviado para o módulo de Execução. Caso contrario, a requisição é envida para o módulo Tradutor. O módulo Tradutor recebe a requisição do usuário e a traduz para a linguagem entendida pelo planejador, neste caso a linguagem PDDL, para construir a definição do domínio e definição do problema de planejamento.

O Planejador é o módulo de raciocínio do sistema. Ele recebe a definição do domínio e do problema de planejamento e através do algoritmo de planejamento implementado nele, raciocina para produzir um ou mais planos solução (que são compostos pelos serviços web satisfazem a requisição do usuário). Caso nenhum plano seja gerado, solicita-se uma nova requisição ao usuário. Sempre que mais de um plano solução é produzido, o módulo Avaliador analisa as propriedades não-funcionais descritas nos serviços web que compõem o plano, para determinar qual se adapta melhor à solicitação do usuário.

O módulo Executor recebe o plano solução e faz o mapeamento para os serviços web (nível de invocação) a serem executados. Após a execução, os resultados são apresentados para o usuário. Se uma falha ocorrer durante a execução dos serviços web, um processo de replanejamento pode ser solicitado pelo Executor ao Planejador.

4. Desenvolvimento dos Módulos Desacoplados

A implementação do módulo Interface e Tradutor foi utilizada a linguagem Java, o ambiente de desenvolvimento ECLIPSE, o ANTLR IDE [Espina 2011] que é um plug-in para o ECLIPSE.

4.1. Módulo Interface

O módulo Interface foi projetado para interagir de forma simples e dinâmica com usuários não técnicos, e como resultado dessa interação abstrair a funcionalidade desejada pelo usuário e transformá-la em uma requisição formalmente especificada (meta), que neste caso é feito através de linguagem de modelagem semântica WSML. Além disso, o módulo Interface deve interagir com os usuários de forma que permita: gerenciar o sistema (usuário autorizado); e apresentar o resultado gerado pelo sistema para uma requisição solicitada.

4.1.1. Interface Gráfica de Gerenciamento

Esta interface gráfica permite que um usuário autorizado (p. ex. administrador) faça o upload e armazenamento de documentos WSML (ontologias, serviços web e mediadores) nos repositórios (banco de dados) do SCASW. Estes documentos WSML são associados a um domínio de aplicação (p. ex. viagens). Um domínio de aplicação é como um tema (categoria) de aplicações que tratam informações e funcionalidades similares. Para facilitar o uso do SCASW por usuários não técnicos, as ontologias submetidas devem ser associadas a ícones representativos, desta forma a interface gráfica de requisição de metas torna-se graficamente mais intuitiva. A figura 2 representa a interface gráfica gerenciamento.

O armazenamento dos documentos WSML segue um processo de criação de objetos Java que representam os tipos de dados dos conceitos, variáveis, propriedades funcionais e não funcionais, contidas nestes documentos WSML. Estes objetos Java são armazenados no banco de dados, o que permite a criação dinâmica dos formulários das interfaces gráficas de requisição de metas e apresentação de resultados.

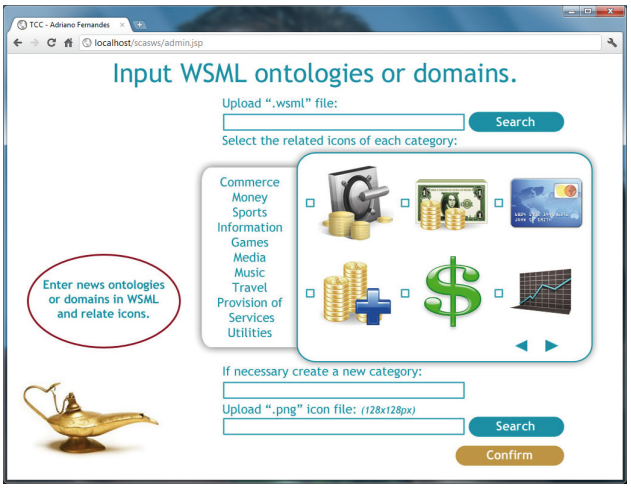

Figure 2. Interface Gráfica de Gerenciamento

4.1.2. Interface Gráfica de Requisição de Metas

A interface gráfica de requisição de metas permite que qualquer usuário especifique uma requisição ao SCASW. Esta interface gráfica deve ser: dinâmica, isto é, o formulário

apresentado deve corresponder ao domínio de aplicação escolhido; e com um alto grau de abstração, isto é, que usuários não técnicos possam especificar suas requisições de forma simples e clara.

Nesse sentido, a primeira tela (formulário) desta interface gráfica permite a escolha de uma categoria (domínio de aplicação). A escolha da categoria gera uma lista de ícones representativos das ontologias e serviços web associados a esta categoria, como representado na figura 3A. A seleção destes ícones representa o refinamento da requisição do usuário. Por exemplo, para descrever que o usuário quer viajar, ele escolhe a categoria correspondente e entre os ícones associados ele pode refinar seu desejo indicando que quer pagar com cartão de credito, que o médio de transporte seja de trem e que as passagens sejam enviadas pelo serviços de postagem local, como pode ser observado na parte superior central da figura 3A.

Ao ser selecionado o botão "Next Step", o módulo Interface gera uma segunda tela (formulário) a qual permite especificar (escolher em alguns casos) os valores (instâncias das ontologias) dos tipos de dados associados aos ícones escolhidos. Por exemplo, para o domínio de aplicação "Viajar", e a seleção dos ícones viagem de trem e modo de entrega das passagens, o formulário gerado contem informações como a cidade origem, cidade destino, horário da viagem e serviço de entrega das passagem, como pode ser observado na figura 3B.

Este segundo formulário é gerado de forma dinâmica em base aos objetos Java associados ao domínio de aplicação e as ontologias (neste caso representadas pelos ícones) escolhidos. Finalmente, ao selecionar o botão "Next Step" o processo de composição é iniciado, uma meta (requisição do usuário) é formalmente especificada na linguagem WSML gerando um documento WSML do tipo meta (*goal*). Note-se que a cada ícone selecionado, o refinamento da requisição cresce e consequentemente a complexidade do serviço web a ser composto.

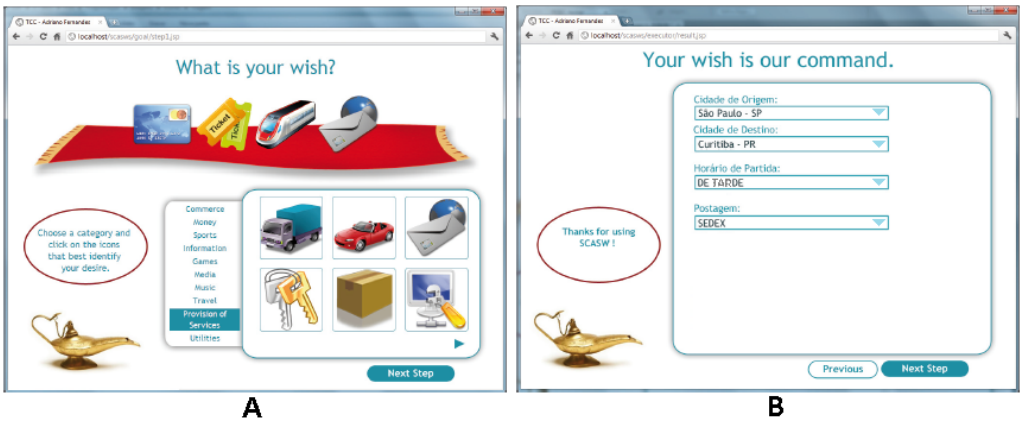

Figure 3. Interface Gráfica de Requisições de Metas

4.1.3. Interface Gráfica de Resultados

Em base à arquitetura do SCASW, a interface gráfica de resultados é responsável por apresentar ao usuário o resultado gerado pelo módulo Executor. A construção da tela

(formulário) que contem as informações dos serviços web executados, segue o mesmo conceito da interface gráfica de requisição de metas, isto é, são utilizados os objetos Java associados às pós-condições e efeitos dos serviços web inclusos no plano solução e executados pelo módulo Executor.

Para o exemplo da subseção anterior, o resultado gerado informa as cidades de origem e destino, o horário da viagem, o valor e forma de pagamento da passagem, e o valor e o forma de entrega das passagem, como pode ser observado na figura 4. Caso o módulo Planejador ou Executor não consigam gerar ou executar o plano, respectivamente, o fluxo do processo é devolvido à interface gráfica de requisição de metas.

Figure 4. Interface Gráfica de Resultados

4.2. Módulo Tradutor

O módulo Tradutor permite que especificações semânticas em WSML (ontologias e metas) sejam traduzidas para a linguagem PDDL. Este módulo foi desenvolvido baseado no conceito de compiladores através do uso da ferramenta ANTLR [Parr 2007]. Nesse contexto, três gramáticas foram construídas: a de representação da linguagem WSML, a de representação do mecanismo de tradução e a de representação da linguagem PDDL.

A gramática capaz de representar a linguagem WSML foi construída para que, quando interpretada pelo compilador de compiladores (ANTLR), gere o analisador léxico e sintático para a linguagem WSML. Quando um documento WSML (neste caso uma ontologia ou uma meta) é submetido à gramática construída, a ferramenta Antlr IDE constrói uma representação gráfica da árvore sintática em conformidade com a entrada recebida, validando a gramática construída.

A gramática de representação do mecanismo de tradução mapeia a equivalência de termos de uma linguagem para a outra, definindo as regras da gramática WSML e a forma como elas deveriam ser traduzidas. Essa gramática, junto ao ANTLR é responsável pela criação do analisador semântico e a tradução de uma ontologia para um domínio PDDL e a tradução da meta para uma especificação de um problema de planejamento.

Para validar os arquivos PDDL gerados, foi projetada uma gramática que representa a linguagem PDDL. Assim, os arquivos PDDL gerados são submetidos à ferramenta

Antlr IDE, a qual produz uma árvore sintática em conformidade aos arquivos submetidos, validando a tradução realizada. Todo este fluxo de processamento é representado na figura 5.

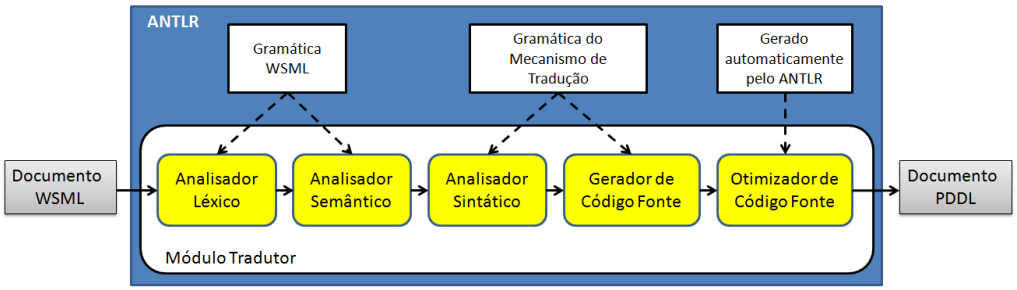

Figure 5. Mecanismo de tradução do módulo tradutor.

5. Trabalhos Relacionados

Diversos trabalhos têm estudado o processo de composição automática serviços web [Sycara et al. 2003], [Agarwal et al. 2005], [Digiampietri et al. 2007] e [Kuzu and Cicekli 2010]. Estes trabalhos focaram seus esforços no desenvolvimento e otimização de diversos tipos de algoritmos e procedimentos para tratar, especificamente, o processo de composição, desconsiderando a importância da interface do sistema.

Já em [Chahoud 2006] e [Silva 2008] apresentam-se duas ferramentas de composição automática de serviços web com interfaces gráficas. Estas interfaces gráficas requerem que os usuários tenham conhecimento técnico da ferramenta, o que limita o uso deste tipo de ferramentas por usuários comuns. Além disso, estas interfaces gráficas foram projetas, em ambos casos, para especificar a requisição do usuário (meta) para um único estudo de caso específico.

Em [Santos et al. 2011] é apresentada uma ferramenta que traz as devidas preocupações para interação com usuários não técnicos, porém, esta ferramenta foi projetada para prover serviços web de forma dinâmica, apresentando os serviços web disponíveis para uma dada situação, nesse sentido, não proporciona nenhuma solução a nível de composição.

Já entre as poucas ferramentas que permitem a tradução de linguagens semânticas (como OWL-S e WSML) para a linguagem PDDL, entre elas destacam-se o OWLSX-Plan [Klusch et al. 2005] que é uma ferramenta de composição que implementa um componente chamado OWLS2PDDL [Gerber and Klusch 2009] que faz a tradução entre a linguagem OWL-S e a linguagem PDDL (entendida pelos planejadores).

Já em [Sirin et al. 2004] é apresentado um algoritmo que permite a tradução da linguagem OWL-S para a sintaxe de planejamento hierárquico ou *Hierarchical Task Network* (HTN) [Sirin et al. 2004] entendida pelo planejador JShop2 [Ilghami 2006]. O AIMO Translator [Tabatabaei et al.] é uma ferramenta que permite a tradução da linguagem WSML para a sintaxe HTN. Em [Zuñiga et al. 2010] é implementado um procedimento que permite a tradução da linguagem WSML para PDDL.

No entanto, todas estas ferramentas implementam algoritmos que fazem a leitura da sintaxe de uma linguagem semântica (OWL-S e WSML) e a traduzem linha por linha,

utilizando tabelas de equivalências, para gerar o documento equivalente na linguagem de planejamento especificada (HTN e PDDL).

6. Conclusões e Trabalho Futuros

O objetivo deste trabalho foi atingido ao descrever, desenvolver e avaliar os módulos Interface e Tradutor de forma desacoplada. Nesse sentido, este trabalho estimula e contribui com os esforços de pesquisa sobre sistemas de composição automática de serviços web.

Ao compararmos as interfaces propostas nos trabalhos correlatos com o nosso módulo Interface, nós apresentamos uma contribuição em base ao nível de abstração da interface gráfica, sendo esta mais intuitiva e simples para usuários sem conhecimentos técnicos, e mais dinâmica no sentido que nossa interface gráfica pode construir de forma automática as telas (formulários) para os diversos domínios de aplicação suportados pelo sistema, não ficando pressa a um único estudo de caso e nem precisando de implementar interfaces gráficas especificas para cada estudo de caso.

Além disso, nosso módulo Interface é composto por três interfaces gráficas, as quais dividem, claramente, a interação para atendam usuários finais (incluindo usuários não-técnicos) e a usuários autorizados que são os encarregados de dar suporte e manutenção ao SCASW. O resultado gerado por nosso módulo Interface é uma especiação formal, na linguagem semântica WSML, da requisição ou meta feita pelo usuário de forma visual e intuitiva.

A construção do nosso módulo Tradutor em base à abordagem de compiladores, demonstrou que é possível realizar a tradução de uma linguagem semântica para uma linguagem de planejamento de forma diferente à abordagem utilizada pelo trabalhos correlatos, que utilizam tabelas de equivalências. Esta abordagem permite que além da tradução dos documentos, também, seja feita uma validação dos mesmos através da análise léxica e sintática.

Além disso, o uso de gramaticas permite o reúso das mesma para a extensão do módulo de forma que possa suportar a tradução de linguagem semântica OWL-S e de extensões do PDDL como o PPDDL que da suporte para problemas de planejamento não-determinístico.

O resultado do módulo Tradutor são os documentos especificados na linguagem de planejamento PDDL. A disponibilização módulo de forma online[1] permite a experimentação da ferramenta. Nesse sentido, podem ser utilizados documentos WSML (ontologias e *goals*) disponíveis de forma online no site oficial [2] dos criadores da modelo ontológico WSMO e da linguagem semântica WSML.

A continuidade do presente trabalho foca o interesse em aperfeiçoar o módulo Interface para suportar de forma simples e intuitiva a representação de axiomas, que são expressões logicas que permitem refinar a especificação de conceitos e relações entre conceitos em uma ontologia. Além disso, o módulo Tradutor deve ser estendido para suportar a tradução para a linguagem de planejamento PPDDL, desta forma teríamos o suporte para criar domínios de aplicação mais complexos que abordem aspectos de não-

[1] Acessível em: http://wsml2pddl.appspot.com
[2] Acessível em: http://wsmo.org

determinismo como observabilidade parcial e nula, já que estes modelos representam de melhor forma a característica dinâmica dos serviços e aplicações web.

References

Agarwal, V., Chafle, G., Dasgupta, K., Karnik, N., Kumar, A., Mittal, S., and Srivastava, B. (2005). Synthy: A system for end to end composition of web services. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(4):311 – 339.

Akkiraju, R., Farell, J., Miller, J., Nagarajan, M., Sheth, A., and Verma, K. (2005). Wsdl-s: Web service semantics. Technical report, World Wide Web Consortium - W3C.

Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The semantic web. *Scientific American*, 284(5):35–43.

Chahoud, J. J. (2006). Planejamento para serviços web semânticos. Master's thesis, Instituto de Matemática e Estatística da Universidade de São Paulo.

de Brujin, J., Fensel, D., Kerrigan, M., Keller, U., Lausen, H., and Scicluna, J. (2008). *Modeling Semantic Web Services*. Springer-Verlag, Berlin, Heidelberg.

Digiampietri, L. A., Pérez-Alcázar, J. J., and Medeiros, C. B. (2007). Ai planning in web services composition: A review of current approaches and a new solution. In *VI Encontro Nacional de Inteligência Artificial, Anais do XXVII congresso da SBC*.

Erl, T. (2009). *SOA Design Patterns*. Prentice Hall PTR.

Espina, E. (2011). Antlr ide. *Disponível em: <http://antlrv3ide.sourceforge.net/> Acesso em Outubro, 18*.

Fensel, D., Lausen, H., Polleres, A., Bruijn, J. d., Stollberg, M., Roman, D., and Domingue, J. (2007). *Enabling Semantic Web Services: The Web Service Modeling Ontology*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Gerber, A. and Klusch, M. (2009). Owls2pddl. Technical Report URL: http://projects.semwebcentral.org/projects/owls2pddl/, Sem Web Central.

GHALLAB, M., HOWE, A., KNOBLOCK, C., MCDERMOTT, D., RAM, M., VELOSO, M., WELD, D., and WILKINS, D. (1998). Pddl - the planning domain definition language. In *AIPS-98 Planning Committee*.

Ghallab, M., Nau, D., and Traverso, P. (2004). *Automated Planning, Theory and Practice*. Morgan Kaufmann Publishers, Elsevier.

Gruber, T. R. (1993). A translation approach to portable ontology specifications. Technical Report KSL 92-71, Knowledge Systems Laboratory, Stanford University.

Ilghami, O. (2006). Documentation for jshop2. Technical report, Department of Computer Science University of Maryland.

Klusch, M., Gerber, A., and Schmidt, M. (2005). Semantic web service composition planning with owls-xplan. In *Proceedings of the 1st Intl. AAAI Fall Symposium on Agents and the Semantic Web, Arlington VA, USA, AAAI Press*.

Kuzu, M. and Cicekli, N. K. (2010). Dynamic planning approach to automated web service composition. *Applied Intelligence*, pages 1–28.

Lausen, H., Polleres, A., and Roman, D. (2005). Wsmo: Web service modelling ontology. Technical report, World Wide Web Consortium - W3C.

Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., and Sirin, E. (2004). Owl-s: Semantic markup for web services. Technical report, World Wide Web Consortium - W3C.

McGuinness, D. L. and van Harmelen, F. (2004). Owl: Web ontology language. Technical report, World Wide Web Consortium - W3C.

Medjahed, B. and Bouguettaya, A. (2011). *Service Composition for the Semantic Web*. Springer New York.

Parr, T. (2007). *The definitive ANTLR reference: building domain-specific languages*. Pragmatic Bookshelf Series. Pragmatic.

Rao, J. and Su, X. (2004). A survey of automated web service composition methods. In *Proceedings of The 1st International Workshop on Semantic Web Services and Web Process Composition (SWSWPC)*, San Diego, USA. Springer.

Russel, S. and Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*. Prentice-Hall, 2nd edition edition.

Santos, L., Sorathia, V., Pires, L., and Sinderen, M. (2011). Towards a conceptual framework to support dynamic service provisioning for non-technical service clients. *Journal of Software*, pages 564–573.

Silva, M. V. A. (2008). Transplan: Uma solução para mapear e planejar serviços web semânticos. Master's thesis, Universidade Federak da Bahia, Laboratório de Sistemas Distribuídos.

Sirin, E., Parsia, B., Wu, D., Hendler, J., and Nau, D. (2004). Htn planning for web service composition using shop2. *Journal of Web Semantics*, 1(4):377–396.

Sycara, K., Paolucci, M., Ankolekar, A., and Srinivasan, N. (2003). Automated discovery, interaction and composition of semantic web services. *Web Semantics: Science, Services and Agents on the World Wide Web*.

Tabatabaei, S. G. H., Kadir, W. M. N. W., Ibrahim, S., and Dastjerdi, A. V. Aimo translator: Bridging the gap between semantic web service discovery and composition. In *Proceedings of the 2010 Fifth International Conference on Internet and Web Applications and Services*, pages 268–273. IEEE Computer Society.

Zuñiga, J. C., Perez-Alcazar, J. J., and Digiampietri, L. (2010). Implementation issues for automatic composition of web services. In *Proceedings of the 2010 Workshops on Database and Expert Systems Applications*, DEXA '10, pages 201–205.