UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
CURSO DE PÓS-GRADUAÇÃO EM ENGENHARIA DE ELETRICIDADE

COMPOSIÇÃO DE WEB SERVICES SEMÂNTICOS NO AMBIENTE ICS DE COMÉRCIO ELETRÔNICO

CARLOS ROBERTO BALUZ ALMEIDA

São Luís
2004

COMPOSIÇÃO DE WEB SERVICES SEMÂNTICOS NO AMBIENTE ICS DE COMÉRCIO ELETRÔNICO

Dissertação de Mestrado submetida à Coordenação do curso de Pós-Graduação em Engenharia de Eletricidade da UFMA como parte dos requisitos para obtenção do título de mestre em Ciência da Computação.

Por

CARLOS ROBERTO BALUZ ALMEIDA

Dezembro, 2004

Almeida, Carlos Roberto Baluz.

Composição de Web Services Semânticos no Ambiente ICS de Comércio Eletrônico / Carlos Roberto Baluz Almeida. – São Luís, 2004.

93.:il.

Dissertação (Mestrado em Engenharia de Eletricidade) – Universidade Federal do Maranhão, São Luís, 2004.

1. Inteligência Artificial. 2. Web Semântica. I. Título.

CDU 004.8

COMPOSIÇÃO DE WEB SERVICES SEMÂNTICOS NO AMBIENTE ICS DE COMÉRCIO ELETRÔNICO

CARLOS ROBERTO BALUZ ALMEIDA

DISSERTAÇÃO APROVADA EM __ / __ / 2004

Prof. Dr. Sofiane Labidi
(Orientador)

Profa. Dra. Maria Augusta Soares Machado
IBMEC

Prof. Dr. Edson Nascimento
UFMA

COMPOSIÇÃO DE WEB SERVICES SEMÂNTICOS NO AMBIENTE ICS DE COMÉRCIO ELETRÔNICO

MESTRADO

ÁREA DE CONCENTRAÇÃO: CIÊNCIA DA COMPUTAÇÃO

CARLOS ROBERTO BALUZ ALMEIDA

Orientador: Dr. Sofiane Labidi

Curso de Pós-Graduação em

Engenharia de Eletricidade da

Universidade Federal do Maranhão

AGRADECIMENTOS

Aos meus pais, Yvone e João Inácio, pelo apoio e motivação ao meu sucesso e conquista.

Ao professor Sofiane Labidi, pela credibilidade e confiança em me orientar neste trabalho e pelos muitos ensinamentos que certamente me ajudarão por toda vida.

Aos amigos Ricardo Tomaz e Bernardo Wangoon pela ajuda no desenvolvimento deste trabalho e de muitos outros.

Aos meus irmãos Dedé e Júnior, por tudo, pois qualquer especificação que fizer aqui do incentivo e apoio que me deram e continuam me apoiando, será sempre pouco.

Aos meus irmãos Ricardo e Giovanni, à minha namorada Sônia, à minha cunhada Ivna, aos sobrinhos Lucca e Bruno, à Rosário e Lorena pela confiança e créditos que sempre tiveram em mim e que sempre funciona como motivação.

Dedico este trabalho a

Meus pais João Inácio e Yvone, que meu sucesso continue sempre motivo de orgulho para eles.

Meus irmãos Dedé e Júnior, responsáveis por minha vida de sucesso profissional e acadêmica.

“Faça o necessário, depois o possível, e de repente você vai estar fazendo o impossível”.

São Francisco de Assis

RESUMO

O ICS *(Intelligent Commerce System),* projeto atualmente em desenvolvimento no Laboratório de Sistemas Inteligentes (LSI) na Universidade Federal do Maranhão (UFMA) sob a orientação do Prof. Dr. Sofiane Labidi, é um projeto que tem como objetivo desenvolver um Sistema de Comércio Eletrônico, na categoria B2B, efetivamente Inteligente. Ele é baseado na tecnologia de agentes móveis inteligentes e possui cinco fases no seu ciclo de vida: Modelagem do Usuário, Matchmaking, Negociação, Formação de Contrato e Cumprimento do Contrato. O *Matchmaking* é o processo no qual agentes representando negociantes (compradores e vendedores), que possuem interesse na troca de valores econômicos, são colocados em contato com seus potenciais parceiros de negócios. O enriquecimento do processo de matchmaking é o foco principal deste trabalho. Atualmente no ICS o processo de matchmaking somente emparceira serviços simples. Nossa contribuição é enriquecer o processo de matching atualmente em uso no ICS permitindo a composição de serviços, ou seja, serviços simples somam suas capacidades e formam serviços complexos com a finalidade de retornar maior número de respostas positivas. Para isso utilizamos ontologias aliadas às técnicas de planejamento automático para proporcionar a descoberta de serviços complementares e posterior composição dos mesmos para elaboração de serviços mais complexos.

Palavras-Chave: Planejadores, Serviços Web, Web Semântica, Composição de Serviços Web, Comércio Eletrônico.

ABSTRACT

The ICS (Intelligent Commerce System), a developing project of the Intelligent Systems Lab (LSI) at Federal University of Maranhão (UFMA), under Prof. Dr. Sofiane Labidi´s supervision, is a project that has the objective of develop an Electronic Commerce System, in the B2B (Business to Business) category, effectively intelligent. It is based in the technology of intelligent mobile agents and has five phases in its life cycle: User Modeling, Matchmaking, Negotiation, Contract Formation and Contract Fulfillment. The Matchmaking is the process in which agents that represent traders (buyers and sellers), that are interested in the exchange of economic values, are put in touch with their potential business counterparts. The enrichment of the matchmaking process is the main focus of this work. Nowadays in the ICS, the matchmaking process only matches simple services. Our contribution is to enrich the actual matchmaking process allowing the composition of services, that is, simple services can add its capacities and form complex services with the goal to return a greater number of positive responses. To do this, we use ontologies allied to AI planning techniques to provide the discovery of complementary services and the posterior composition of them to elaborate more complex services.

Keywords: AI Planning, Web Services, Semantic Web, Web Services Composition, E-Commerce.

SUMÁRIO

LISTA DE FIGURAS 11

1 INTRODUÇÃO 12

1.1 Objetivo 15

1.2 Justificativa e Relevância 15

1.3 Organização da Dissertação 15

2 ESTADO DA ARTE 16

2.1 Planejamento Automático 16

2.1.1 Conceitos Fundamentais em Planejamento Automático 20
2.1.2 Técnicas de Planejamento Automático 26
2.1.3 STRIPS (STanford Research Institute Problem Solver) 26
2.1.4 ADL (Action Description Language) 27
2.1.5 PDDL (Problem Domain Description Language) 29
2.1.6 HTN (Hierarchical Task Network) ou (Rede Hierárquica de Tarefas) 29

2.2 Web Semântica e Web Services Semânticos 32

2.2.1 Arquitetura da Web Semântica 33
2.2.2 Aplicações da Web Semântica 35
2.2.3 Web Services Semânticos 36
2.2.4 Adequação da Web Semântica e dos Web Services para o desenvolvimento de Ferramentas B2B 39
2.2.5 Conclusão 42

2.3 Ontologias 42

2.3.1 Ontologias e a Web Semântica 45
2.3.2 Linguagens para Representação de Ontologias 47
2.3.3 Conclusão 55

3 INTELLIGENT COMMERCE SYSTEM – ICS 56

3.1 Introdução 56

3.2 Ciclo de Vida do Comércio Eletrônico no ICS 57

3.3 Arquitetura do ICS 59

3.3.1 Marketplace 60
3.3.2 Região 60
3.3.3 Repositório de Ontologias (modelos de domínios) 61
3.3.4 Base de Estereótipos 61
3.3.5 Base de Propagandas 61
3.3.6 Agente Matchmaker 62
3.3.7 Agente Mediador 62
3.3.8 Agente Negociante 63
3.3.9 Agente de Modelagem 63
3.3.10 Conclusão 64

4 COMPOSIÇÃO DE SERVIÇOS NO ICS 66

4.1 Introdução 66

4.2 Cenário de Motivação 67

4.3 SHOP2 (Simple Hierarchical Ordered Planner) 69

4.4 Algoritmo de tradução de DAML-S para SHOP2 72

CONCLUSÃO 86

REFERÊNCIA BIBLIOGRÁFICA 88

URLS 93

Lista de Figuras

Figura 1: Representação da seqüência de execução de um plano. 21

Figura 2: O problema do pneu sobressalente (Stuart, 2000). 29

Figura 3: Arquitetura da Web Semântica (Berners – Lee et al., 2000). 34

Figura 4: SOA – Service Oriented Architecture 37

Figura 5: Tipos de Ontologias e seus relacionamentos (Guarino,1998). 44

Figura 6: Representação RDF 48

Figura 7: Ontologia de Serviço DAML-S (Anupriya, 2002) 51

Figura 8: Service Profile (Anupriya, 2002). 52

Figura 9. Ciclo de vida do comércio eletrônico no ICS. 57

Figura 10: Arquitetura do ICS. 54

Figura 11: Plano de Composição (Baluz et al, 2004) 68

Figura 12: Problema da figura 11 descrito em ADL.. 69

Figura 13: Ferramenta de reconhecimento do Common Lisp para execução do planejador SHOP2 – ALLEGRO COMMON LISP 71

Figura 14: Janela do Allegro Commom Lisp enquanto o SHOP2 está no ar 84

Figura 15: Resposta do Tradutor, uma palavra sinônima em francês com 1 plano. 85

1 INTRODUÇÃO

Na tentativa de construir máquinas que imitam o comportamento humano, a Inteligência Artificial necessita de técnicas especiais para o tratamento dos impasses de processamento gerados pela complexidade computacional e pelo conhecido problema da explosão combinatória em situações onde o processamento do cérebro humano se destaca, como as encontradas no planejamento de ações para a solução de problemas diversos.

Planejar em Inteligência Artificial (IA) significa encontrar a combinação correta de uma seqüência de ações parametrizadas que conduzem os agentes de um estado inicial fornecido para um estado final desejado.

Para ampliar o grau de complexidade do sistema de planejamento, há uma grande diversidade de domínios que podem ser tratados. Um planejador, cujo objetivo é simular o comportamento humano, não pode depender especificamente de um domínio para funcionar. Várias técnicas de IA trabalham para minimizar os efeitos deste acúmulo de complexidade e tornar tratável este tipo de problema, como acontece no ambiente ICS de Comércio Eletrônico B2B (Labidi et. al, 2003), voltado para negociação na Internet entre possíveis parceiros de negócios.

Na busca do melhor caminho para minimizar custos em diversos domínios de problemas, a IA há anos vem investigando o planejamento de seqüência de ações. Recentemente, estudos nessa sub-área da Inteligência Artificial, o planejamento automático teve um grande avanço. Daí o objetivo do nosso trabalho, estudar e investigar as técnicas de Planejamento, para empregá-las na arquitetura do ICS,

mais especificamente no agente Matchmaker, de modo que este passe a formular planos para composição de serviços simples disponíveis no repositório de propagandas aumentando o número de negociações realizadas dentro do ambiente, uma vez que, a partir de um conjunto limitado de serviços simples disponíveis um grande número de serviços compostos podem surgir.

1.1 Objetivo

O objetivo deste trabalho é de enriquecer o processo de matching dos agentes negociantes atualmente em uso no ICS para permitir a composição de serviços simples para elaboração de serviços mais complexos.

Especificamente, pretende-se:

i) Fazer uma revisão bibliográfica sobre Planejamento Automático, Web Sermântica e Web Services de modo a permitir a compreensão de nossa abordagem;

ii) Fazer uso de técnicas de Composição de Serviços Web como uma alternativa para reduzir o número de *falsos negativos* no atual procedimento de matching;

iii) Utilizar os conceitos, padrões e ferramentas da tecnologia de Planejamento Automático para gerar planos de composição de serviços Web a partir da descrição de suas capacidades utilizando a linguagem DAML-S.

iv) Reduzir custos operacionais e agilizar os processos de negociação entre as empresas dentro do ICS.

1.2 Justificativa e Relevância

As empresas estão em busca de um diferencial para a grande competitividade que existe hoje em dia em qualquer ramo da economia capitalista. A importância de conhecer e possuir ferramentas são diferenciais entre empresas no mundo de hoje e é de grande utilidade para os empresários, porque elas são fontes de informação e dão apoio à tomada de decisão, agregando valores como informações sobre as melhores oportunidades de compra de produtos sem se preocupar com detalhes de negociação, delegadas aos agentes de software, entidades atuando autonomamente (ou semi autonomamente) em favor de empresas em ambientes de negociação virtuais, com baixo custo operacional, rapidez e segurança nas negociações.

Logo, este trabalho se justifica por:

- beneficiar um grande número de empresas através da abordagem inovadora de sua arquitetura;

- reunir referencial teórico necessário ao entendimento de tecnologias como AI Planning, Web Services e Web Semântica, sub-áreas da Inteligência Artificial (IA), para proporcionar descoberta automática de possíveis parceiros de negócios distribuídos pela Web;

- experimentar teorias, técnicas e ferramentas apresentadas no desenvolvimento do projeto.

E finalmente, mostrar que o ICS, diferente de outros sistemas de comércio eletrônico, concretiza todo o ciclo do comércio convencional, desde a negociação até o fechamento do contrato, no mundo eletrônico de forma automatizada.

1.3 Organização da Dissertação

Esta dissertação está organizada em cinco capítulos: o primeiro capítulo é composto por esta introdução. O segundo capítulo apresenta a fundamentação teórica com conceitos e ferramentas relativos a Planejamento Automático para Composição de Web Services e as tecnologias e padrões utilizados.

No capítulo 3 apresentamos o sistema de Comércio Eletrônico ICS, suas principais características e um resumo de cada fase do seu ciclo de vida.

No capítulo 4 apresentamos a Composição de Serviços, as ferramentas usadas para interpretar linguagens de planejamento, um Cenário de Motivação com desenvolvimento de exemplos gerando planos no planejador SHOP2.

Por fim, no quinto capítulo, apresentamos nossas conclusões sobre este trabalho de pesquisa e as sugestões para trabalhos futuros.

2 ESTADO DA ARTE

2.1 Planejamento Automático

Nesta seção são apresentados termos e definições da inteligência artificial(IA) fundamentais para entendimento dos problemas de planejamento automático. É separado um tópico para os conceitos mais importantes da área de IA e por fim as técnicas de planejamento automático necessárias ao desenvolvimento e entendimento da contribuição proposta.

A ação ou efeito de planejar, de trabalhar na preparação de um empreendimento no qual se estabeleçam os objetivos, as etapas, os prazos e os meios para a sua concretização estabelecendo um plano, chama-se de Planejamento (Stuart et al., 1995).

O desenvolvimento de solucionadores de problemas gerais tem sido um dos principais objetivos da área de IA. O desafio consiste em modelar o comportamento dos humanos que, segundo acredita-se, resolvem problemas genéricos, isto é, adaptam-se aos problemas para obterem a melhor solução. Um programa de computador que aceite descrições de alto nível dos problemas e processe automaticamente a sua solução pode ser considerado com um resolvedor geral de problemas. O escopo pode ser definido por meio de uma classe de modelos matemáticos que define os tipos de problemas, as formas de solução e quais soluções são melhores ou ótimas. Um sistema de planejamento em inteligência artificial, dentro desta perspectiva, é um resolvedor de problemas que possui uma representação conveniente e uma solução efetiva para uma certa classe de modelos matemáticos.

Na metodologia aceita atualmente pela comunidade que estuda o problema de planejamento, um planejador deve ser capaz de ler um problema de planejamento descrito no formato PDDL (*Planning Domain Definition Language* - Linguagem de Definição de Domínios de Planejamento), que consiste de uma descrição do comportamento das ações e uma descrição do estado inicial e final. Deve retornar dentro de um limite de tempo previamente especificado e consumindo, no máximo, uma certa quantidade de memória, uma seqüência de ações que, quando executadas de uma dada ordem sobre o estado inicial, transforma-o no estado final. Qualquer seqüência de ações que faça isto é considerada um plano.

Infelizmente, não existe um tamanho para o plano. Na maioria das situações, é suposto que um planejador deve retornar um plano ótimo, isto é, um plano de tamanho mínimo. Por outro lado, como isto é difícil de acontecer, apenas encontrar um plano é, em geral, suficiente. (Lecheta, 2004)

Conhecer o estado atual do ambiente nem sempre é suficiente para se decidir o que fazer. Por exemplo, em um entroncamento de estradas, o táxi pode virar à esquerda, virar à direita ou seguir em frente. A decisão correta depende de onde o táxi está tentando chegar. Ou seja, da mesma forma que o agente precisa de uma descrição do estado atual, ele também precisa de alguma espécie de informação sobre objetivos que descreva situações desejáveis – por exemplo, estar no destino do passageiro. O programa de agente pode combinar isso com informações sobre os resultados de ações possíveis (as mesmas informações que foram usadas para atualizar o estado interno no agente ativo), a fim de escolher ações que alcancem o objetivo.

Entre os diferentes tipos de planejamentos estudados em IA, destacamos três tipos de planejamento, dado a sua aplicação recente na resolução e otimização de problemas do mundo real:

Planejamento autônomo e escalonamento: A uma centena de milhões de quilômetros da Terra, o programa *Remote Agente* da NASA se tornou o primeiro programa de planejamento autônomo de bordo a controlar o escalonamento de operações de uma nave espacial (Jonsson et. al., 2000). O Remote Agent gerou planos de metas de alto nível especificadas a partir do solo e monitorou a operação da nave espacial à medida que os planos eram executados – efetuando a detecção, o diagnóstico e a recuperação de problemas conforme eles ocorriam. (Stuart et al., 1995).

Planejamento logístico: Durante a crise do Gosto Pérsico em 1991, as forças armadas dos Estados Unidos distribuíram uma ferramenta denominada *Dynamic Analysis and Replannig Tool, ou DART* (Cross e Walker, 1994), a fim de realizar o planejamento logístico automatizado e a programação de execução do transporte. Isso envolveu até 50.000 veículos, transporte de carga aérea e pessoal ao mesmo tempo, e teve de levar em conta pontos de partida, destinos, rotas e resolução de conflitos entre todos os parâmetros. As técnicas de planejamento da IA permitiram a geração em algumas horas de um plano que exigiria semanas com outros métodos. A *Defense Advanced Research Project Agency* (DARPA) declarou que essa única aplicação compensou com folga os 30 anos de investimento da DARPA em IA (Stuart et al., 1995).

Planejamento Clássico: São ambientes completamente observáveis, determinísticos, finitos, estáticos (a mudança só acontece quanto o agente age) e

discretos (em tempo, ação, objetos efetivos). É neste tipo de planejamento que se enquadra o nosso estudo. Ou seja, o ICS é um ambiente determinístico, observável e finito, logo, o Planejamento Clássico é a técnica usada para compor serviços Web dentro do Matchmaking do ICS.

Os subcampos da IA dedicados a encontrar estas seqüências de ações que alcançam os objetivos do agente no planejamento clássico são a busca e planejamento. Busca é o processo de procurar por tal seqüência. Um algoritmo de busca recebe um problema como entrada e retorna uma solução sob a forma de uma seqüência de ações. Depois que uma solução é encontrada, as ações que ela recomenda podem ser executadas. Isso se chama fase de execução**.** Desse modo, temos um simples projeto de "formular, buscar, executar" para o agente, como mostra a figura 1. Depois de formular um objetivo e um problema a resolver, o agente chama um procedimento de busca para resolvê-lo. Em seguida, ele utiliza a solução para orientar suas ações, fazendo o que a solução recomendar como a próxima ação – em geral, a primeira ação da seqüência – e então removendo esse passo da seqüência. Depois que a solução for executada, o agente formulará um novo objetivo.

função AGENTE-DE-RESOLUÇÃO-DE-PROBLEMAS-SIMPLES*(percepção)* **retorna** uma ação

entradas: *percepção:* uma *percepção*

variáveis estáticas: *seq:* uma seqüência de ações, inicialmente vazia

estado: alguma descrição do estado atual do mundo

objetivo: um objetivo, inicialmente nulo

problema: uma formulação de problema

estado ← ATUALIZAR-ESTADO*(estado,percepção)*

se seq está vazia **então faça**

objetivo ← FORMULAR-OBJETIVO (*estado*)

problema ← FORMULAR-PROBLEMA (*estado,objetivo)*

seq ← BUSCA (*problema)*

ação ← PRIMEIRO (*seq)*

seq ← RESTO *(seq)*

retornar ação

Figura 1: Um agente simples de resolução de problemas.

2.1.1 Conceitos Fundamentais em Planejamento Automático

A seguir apresentamos os conceitos necessários ao entendimento de Planejamento Automático.

Há muitos anos os cientistas tentam descobrir como pensamos. A Inteligência Artificial não só tenta entender, compreender como também construir entidades inteligentes.

O Planejamento Automático (AI Planning) é uma área de pesquisa bastante ativa dentro da Inteligência Artificial. O objetivo principal do planejamento automático é a identificação da seqüência de ações que devem ser executadas para atingir um objetivo a partir de condições iniciais específicas, criando assim entidades inteligentes.

A idéia do planejamento automático é simular o comportamento de um agente dentro de um ambiente específico e enumerar suas ações para alcançar seus

objetivos. A descrição de ambiente em um dado instante de tempo é chamada de estado do sistema. O primeiro estado é chamado estado inicial e o último estado é chamado de estado objetivo. Nós assumimos que existe um finito (mais amplo) conjunto de estados e um finito conjunto de ações que provocam a mudança de estado. O objetivo do plano é permitir que um agente vá do estado inicial para o estado final (objetivo) como mostrado na figura abaixo.

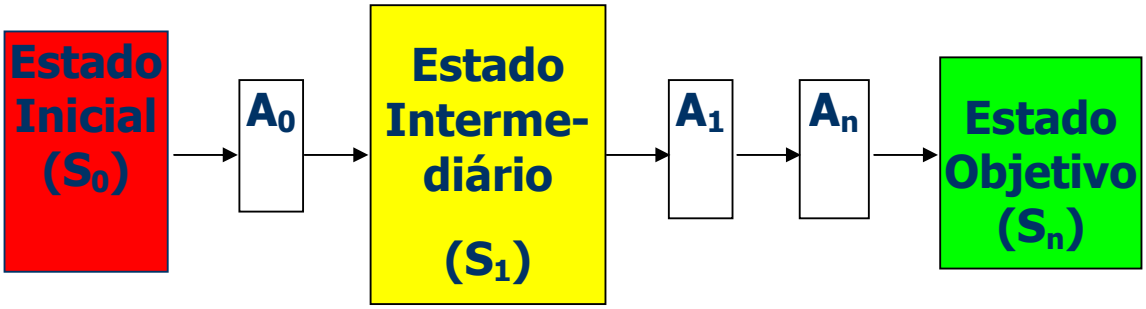

Figura 1: Representação da seqüência de execução de um plano.

Através da execução das ações {A_0, A_1, A_2,…, A_{n-2}, A_{n-1}, A_n}, o agente muda do estado inicial para o estado objetivo. A tarefa de apresentar uma seqüência de ações que alcançam um objetivo é chamada planejamento.

Diante da representação de uma seqüência de execução de um plano, é preciso conceituar os termos utilizados para investigar uma perfeita seqüência de ações.

Fluents: na Lógica clássica, caracterizamos como os objetos que compõem um mundo através de fatos. No planejamento automático, entretanto, essas informações são associadas a instantes de tempo, ou seja, o que é válido em um momento pode ser inválido no momento seguinte. Essa não-monotonicidade é identificada nos problemas de planejamento automático pelo termo *fluent*, ou seja,

um *fluent* é uma proposição cujo valor verdade é dado em função do tempo. Outro aspecto interessante é a imprevisibilidade associada a um *fluent*, que pode aparecer em intervalos intermitentes de tempo, dependendo dos operadores instanciados pelo planejador. Alguns autores usam a noção de visibilidade para descrever o comportamento dos *fluents*, ou seja, visíveis quando o *fluent* é verdadeiro em um determinado instante de tempo e não visíveis caso contrário.

Estados: sendo $\mathcal{L}$ uma linguagem da Lógica Clássica de primeira ordem, um estado S_t é um conjunto finito de fluents pertencentes a $\mathcal{L}$ e que são verdadeiros (visíveis) no instante de tempo t. Note-se que essa definição assume um mundo fechado, ou seja, todo fluent $f \notin S_t$ é considerado falso no instante de tempo t.

Operadores: também chamados de ações, os operadores são esquemas que representam regras de inferência não instanciadas. Ou seja, uma regra do tipo $\alpha \supset \beta$ onde α e β são conjunções de variáveis. Durante o processo de planejamento, o sistema deve instanciar essas variáveis com fluents selecionados a partir de algum critério - esse processo de instanciação e seleção dos operadores é o cerne da complexidade associada aos problemas de planejamento.

Domínio: o domínio é composto pela enumeração dos predicados e do conjunto de operadores aplicáveis a um determinado ambiente.

Problema: o problema é composto pela descrição do estado inicial e do estado objetivo associado a um ambiente. Ou seja, descreve-se o estado atual de um mundo (ou visão atual), e o estado esperado após a aplicação de um plano. Por

exemplo, *Em* é a função que indica o estado atual, considerando o ponto de partida ou estado inicial do problema *w*, então o agente pode ser descrito como *Em(w)*.

Formulação de Problemas é o processo de decidir que ações e estados devem ser considerados, dado um objetivo. A formulação mais comum utiliza uma função SUCESSOR. Dado um estado particular *x*, SUCESSOR(*x*) retorna um conjunto de pares ordenados (ação, sucessor), em que cada ação é uma das ações válidas no estado *x* e cada sucessor é um estado que pode ser alcançado a partir de x aplicando-se a ação. Por exemplo, a partir do estado *Em(w),* a função SUCESSOR para a letra do alfabeto depois de *w* é:

$$\{<Ir(x),Em(x)>,<Ir(y),Em(y)>,<Ir(z),Em(z)>\}.$$

Juntos, o estado inicial e a função *SUCESSOR* definem implicitamente o espaço de estados do problema – o conjunto de todos os estados acessíveis a partir do estado inicial.

Problema do mundo real é aquele cujas soluções de fato preocupam as pessoas e tendem a não apresentar uma única descrição consensual. Exemplos de problema do mundo real.

Exemplo1:

Problema do Caixeiro Viajante - O **PCV** é um problema que um caixeiro viajante deve visitar um conjunto de cidades, visitando cada cidade exatamente uma vez. O objetivo é encontrar o percurso *mais curto.*

Exemplo 2:

Problema de Roteamento - O problema é definido em termos de posições especificadas e transições ao longo de ligações entre elas. Os algoritmos de roteamento são utilizados em uma grande variedade de aplicações, como o roteamento em redes de computadores, planejamento de operações militares e sistemas de planejamento de viagens aéreas.

Representação das etapas no exemplo simplificado de um problema de viagens aéreas:

- Estados: Cada um é representado por uma posição (por exemplo, um aeroporto e pela hora atual).

- Estado Inicial: É especificado pelo problema.

- Função sucessor: Retorna os estados resultantes de tomar qualquer vôo programado (talvez especificado com mais detalhes pela classe e pela posição da poltrona) que parte depois da hora atual somada ao tempo de trânsito no aeroporto, desde o aeroporto atual até outro.

- Teste objetivo: Estamos no destino após algum tempo previamente especificado?

- Custo de caminho: Depende do custo monetário, do tempo de espera, do tempo de vôo, dos procedimentos alfandegários e de imigração, da qualidade da poltrona, da hora do dia, do tipo da aeronave, dos prêmios por milhagem em vôos freqüentes e assim por diante (Stuart et al., 1995).

Os sistemas comerciais de informações para viagens utilizam uma formulação de problema desse tipo, com muitas complicações adicionais para manipular as estruturas de tarifas que as empresas aéreas impõem. Porém, nem toda viagem

aérea transcorre de acordo com os planos. Um sistema realmente bom deve incluir planos de contingência – como reservas substitutas em vôos alternativos – até o ponto em que esses planos possam ser justificados em função do custo e pela probabilidade de fracasso no plano original.

Plano de Contingência é a opção de outro plano para lidar com circunstâncias desconhecidas que podem surgir durante a execução do plano original.

Plano: é a lista ordenada de operadores instanciados por um planejador e que permite a transição do estado atual ao estado objetivo de um problema.

Esta seqüência de passos ou ações chamada plano contribuiu bastante para o planejamento automático e a primeira seqüência automática de montagem de objetos complexos por um robô foi demonstrada por FREDDY (Michie, 1972). Recentemente, houve um aumento da demanda por robôs de software que executam pesquisas na Internet procurando respostas para perguntas, informações inter-relacionadas ou oportunidades de compras (Stuart et al., 1995). O ICS é um exemplo de ambiente que usa tais tipos robôs de software (agentes) que descobre serviços com afinidade para realizar negócios, compõem serviços a fim de realizar negócios, negocia autonomamente e forma contratos.

Assim, existem métodos que um agente pode usar para selecionar ações em ambientes determinísticos, observáveis, estáticos e completamente conhecidos (planejamento clássico). Em tais casos, o agente pode construir seqüências de ações que alcançam seus objetivos. E antes de um agente poder começar a procurar soluções, ele deve formular um objetivo, e depois usar o objetivo para formular o problema, até chegar na solução do problema.

Solução é um plano desde o estado inicial até um estado objetivo.

Com base nos conceitos apresentados acima podemos definir com maior precisão planejamento automático, como sendo uma enumeração do conjunto de instâncias de operadores aplicáveis ao domínio de um problema e que permitem a transição do estado inicial, dito S_0, para o estado desejado e conhecido como Estado Objetivo (S_n) deste problema(Silva, 2003).

2.1.2 Técnicas de Planejamento Automático

Para permitir o planejamento automático de um problema específico, estado, ações e objetivos devem tornar possível a criação de algoritmos de planejamento com base na estrutura lógica do problema. Para isso, é necessário encontrar uma linguagem suficientemente expressiva para descrever uma ampla variedade de problemas, mas restritiva o bastante para permitir que algoritmos eficientes operem sobre ela.

Esboçamos a linguagem básica de representação de planejadores clássicos, depois destacamos algumas das variações possíveis.

2.1.3 STRIPS (STanford Research Institute Problem Solver)

Em 1971, foi lançado o planejador automático baseado nos operadores STRIPS (STanford Research Institute Problem Solver). Este planejador realiza a representação de ações baseado nos operadores Pré-Condição, Adição e Eliminação. Na linguagem STRIPS o mundo é decomposto em condições lógicas e um estado é representado como um conjunto de literais positivos. A linguagem

STRIPS trabalha com a hipótese de mundo fechado, ou seja, quaisquer condições não mencionadas em um estado são consideradas falsas.

A linguagem STRIPS descreve ações em termos de suas precondições e seus efeitos, e descreve os estados inicial e objetivo como conjunções de literais positivos. Tem bastante influência devido a sua representação de ações.

Uma ação é especificada em termos das pré-condições que devem ser válidas, antes de ser possível executá-la e dos efeitos que resultam de sua execução.

Pré-condição (PC): é um conjunto de literais presentes no estado atual que justificam a aplicação da ação. As pré-condições declaram o que deve ser verdadeiro em um estado antes da ação poder ser executada.

Efeito: é um conjunto de literais que descrevem como o estado se altera quando a ação é executada. Alguns sistemas de planejamento dividem o efeito em lista de adição para literais positivos e lista de eliminação para literais negativos.

Com o tempo surgiu uma compreensão aprimorada das limitações e dos compromissos entre os formalismos, a ADL.

2.1.4 ADL (Action Description Language)

A ADL veio com alguns melhoramentos em relação à STRIPS. A Action Description Language (ADL), relaxou algumas das restrições da linguagem STRIPS e tornou possível codificar problemas mais realistas.

Além de literais positivos, a ADL também aceita literais negativos em estados: ¬Rico e ¬Famoso. Aceita a hipótese de mundo aberto: Literais não-mencionados são desconhecidos. Os objetivos permitem conjunção e disjunção; ¬Pobre e (Famoso ou Inteligente), dentre outras diferenças.

Para melhor esclarecer o funcionamento da linguagem ADL, utilizamos o exemplo clássico do problema do pneu sobressalente (Stuart et al., 1995). Notemos que a descrição da linguagem ADL vai além da linguagem STRIPS pelo fato de usar uma pré-condição negada.

Iniciar(Em(Furado,Eixo) ^ Em(Sobressalente, PortaMalas))

Objetivo (Em (Sobressalente, Eixo))

1. *Ação(Remover(Sobressalente,PortaMalas)),*

    PRECOND*: Em (Sobressalente,PortaMalas)*

    EFFECT: *¬Em(Sobressalente,PortaMalas) ^ Em(Sobressalente,Chão)*

2. *Ação(Remover(Furado,Eixo)),*

    PRECOND*:(Em(Furado,Eixo))*

    EFFECT: *¬Em(Furado,Eixo) ^ Em(Furado,Chão)*

3. *Ação(Montar(Sobressalente,Chão),*

    PRECOND:*(Em(Sobressalente,Chão) ^ ¬Em(Furado,Eixo)*

    EFFECT: *¬Em(Sobressalente,Chão) ^ Em(Sobressalente,Eixo)*

4. *Ação(DeixarDuranteNoite),*

    PRECOND:

EFFECT: *(¬Em(Sobressalente,Chão)) ^ (¬Em(Sobressalente,Eixo)) ^ (¬Em(Sobressalente,PortaMalas)) ^ (¬Em(Furado,Chão)) ^ (¬Em(Furado,Eixo))*

Figura 2: O problema do pneu sobressalente (Stuart et al., 1995).

2.1.5 PDDL (Problem Domain Description Language)

A PDDL foi introduzida como uma sintaxe padronizada e analisável por computador para representar STRIPS, ADL e outras linguagens. PDDL foi usada como linguagem-padrão para as competições de planejamento na conferência AIPS, a partir de 1998.

Atualmente, estudiosos da área de planejamento automático dizem que um planejador deve ser capaz de ler um problema de planejamento descrito no formato PDDL (Planning Domain Definition Language. Linguagem de Definição de Domínios de Planejamento).

2.1.6 HTN (Hierarchical Task Network) ou (Rede Hierárquica de Tarefas)

Uma das idéias mais difundidas ao se lidar com a complexidade é a decomposição de tarefas, onde produtos de software complexos são criados a partir de uma hierarquia de sub-rotinas ou classes de objeto. O benefício fundamental da estrutura hierárquica é que, em cada nível da hierarquia, uma tarefa computacional é reduzida a um pequeno número de atividades no nível mais baixo seguinte, de forma que o custo computacional de se encontrar a maneira correta de organizar essas atividades para o problema atual é pequeno.

O método de decomposição de tarefas auxilia o planejamento de HTN (Hierarchical Task Network). O HTN é uma técnica de planejamento que cria planos de resolução de problemas no mundo real através da decomposição de tarefas. No planejamento de HTN, o plano inicial que descreve o problema, é visualizado como uma descrição de nível muito alto do que deve ser feito, logo os planos são refinados pela aplicação da decomposição de tarefas. Cada decomposição de ação reduz uma ação de alto nível a um conjunto parcialmente ordenado de ações de nível mais baixo.

O HTN possui dois tipos de operadores:

- primitivas: são operadores diretamente executáveis;

- abstratos: são operadores que precisam ser decompostos em operadores primitivos para serem executados.

O HTN é um processo no qual o planejador decompõe as tarefas em sub-tarefas cada vez menores até que tarefas do tipo primitivas sejam encontradas e executadas diretamente.

O HTN possui três tipos de tarefas:

- tarefa objetivo(operador abstrato): tem a forma **achieve[L],** onde L é uma literal. Por exemplo: achieve[clear(blockA)]

- tarefa composta(operador abstrato): tem a forma **perform[t(x1,...,xk)]**, onde t é o nome de uma tarefa e x*i*, seus termos. Por exemplo: perform[go(Brazil, France)]

- tarefa primitiva(operador primitivo): tem a forma **do[f(x1,...,xk)]**, onde f é o símbolo de uma tarefa primitiva e x*i*, seus termos. Por exemplo: do[move(blockA,blockB)].

O planejamento segue a seguinte seqüência: o planejador primeiro descobre os serviços primitivos(tarefas), gerando os Operadores(serviços atômicos) *(S1, S2, ... Sn)*. A seguir estes serviços são combinados na geração de um Plano levando em conta as pré-condições e os efeitos.

Para clarear este processo vamos considerar o seguinte exemplo: para viajar de um local X para um local Y é necessário executar algumas tarefas. O primeiro passo é a enumeração dessas tarefas. Essas tarefas precisam ser organizadas numa seqüência de cooperação*(fluxo)* entre os serviços que as realizam. Uma ordem cronológica simples de tarefas poderia ser:

- comprar passagem (aeroporto(X), aeroporto(Y));

- conseguir taxi;

- ir de táxi (X, aeroporto (X));

- pagar táxi;

- usar passagem(aeroporto(X), aeroporto(Y));

- conseguir taxi;

- ir de táxi (aeroporto (Y),Y);

- pagar táxi;

Para determinar o fluxo de interações, o planejador precisa levar em conta algumas pré-condições e efeitos. Por exemplo, para executar a tarefa usar passagem((aeroporto(X), aeroporto(Y)), é necessário antes realizar a tarefa comprar passagem (aeroporto(X), aeroporto(Y)), assim como para executar a tarefa - ir de táxi (X, aeroporto (X)), é preciso antes realizar a tarefa - conseguir táxi. Além disso, ao executar uma tarefa, é necessário que o seu efeito contribua para alcançar o estado objetivo. Desta maneira o planejador gera a seqüência de ações que permitem alcançar este estado objetivo.

Na próxima seção, conceitos e aplicações de Web Semântica e Web Services Semânticos, seus padrões e ferramentas no contexto da composição de serviços no ICS.

2.2 Web Semântica e Web Services Semânticos

É apresentado nesta seção os principais conceitos relacionados à Web Semântica e à tecnologia de Web Services Semânticos, bem como a contribuição e adequação do uso de cada uma destas tecnologias para o desenvolvimento e entendimento deste trabalho.

A Web Semântica, uma evolução da Web atual, proposta por Tim Berners-Lee em sua visão sobre o futuro da Web (Berners-Lee, 1998), proporciona um certo grau de estrutura para os dados disponibilizados na Web, adicionando metadados e ontologias, o que possibilita a recuperação automática de informações por agentes inteligentes de software.

A idéia principal da Web Semântica é a de uma Web no qual os recursos disponíveis (dados e informações) são acessíveis não somente por seres humanos,

mas também por processos automatizados que podem ser agentes de software. A Web Semântica visa atribuir estrutura aos dados armazenados na Web, juntamente com relações semânticas entre eles que permitam recuperá-los com maior eficiência. A automação da Web passa pela elevação de seu status de lida automaticamente "*machine-readable"* para entendida automaticamente "*machine understandable"* (Horrocks, 2002).

Apesar de relativamente nova, a idéia da Web Semântica está embasada em pesquisas muito consolidadas na área da IA (Inteligência Artificial) que são a Engenharia do Conhecimento (Stanley, 1999) e a tecnologia de Agentes Inteligentes (Jennings et al., 2000). A idéia é modelar e representar o conhecimento disponível na Web e aplicar mecanismos de inferência (raciocínio) fazendo uso de agentes inteligentes para recuperá-lo eficientemente.

Um dos pilares da Web Semântica são os Metadados, que tornam possível atribuir significado aos conteúdos e serviços disponíveis na Web. Metadados são *"dados sobre os dados"*, ou seja, metadado é uma estrutura descritiva da informação sobre outro dado. Os metadados podem ser estruturais ou semânticos. O metadado estrutural descreve a organização e a estrutura dos dados armazenados (formato, tipo de dado e relacionamentos sintáticos). O metadado semântico descreve o significado e os relacionamentos semânticos dos dados armazenados.

2.2.1 Arquitetura da Web Semântica

A Web Semântica foi idealizada em camadas (Berners–Lee et al., 2000). O W3C (URL 1), entidade certificadora dos padrões e ferramentas desenvolvidas para

construção da Web Semântica, propôs uma visão inicial em três camadas como mostra a figura 2.

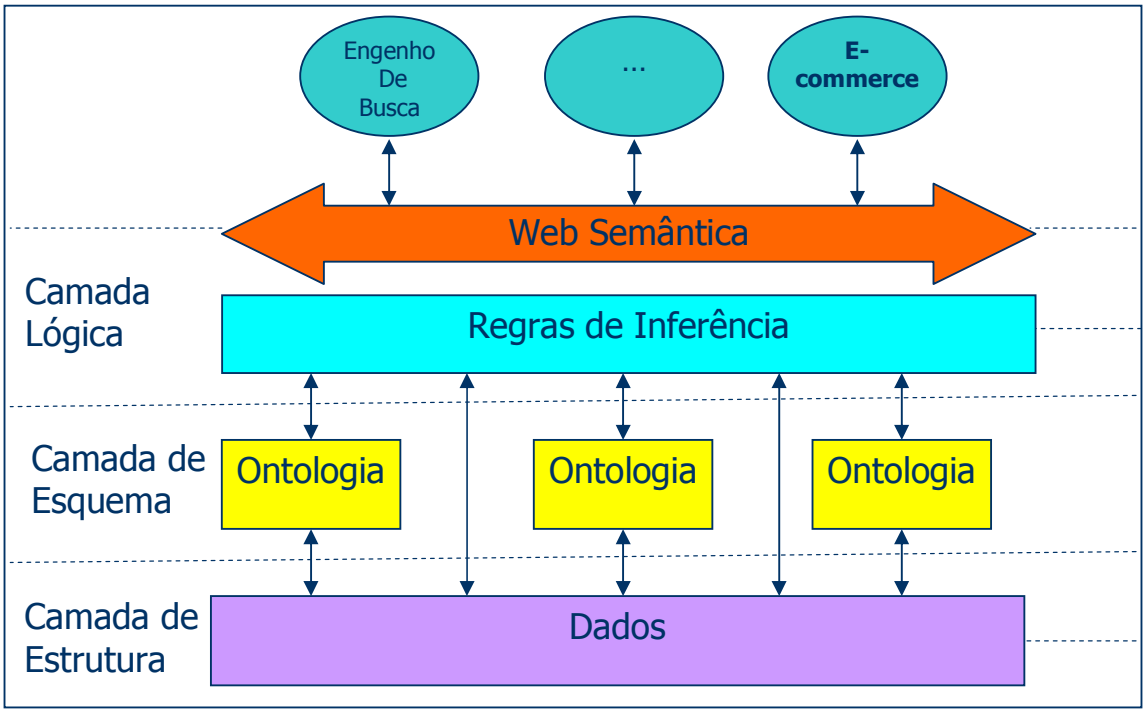

Figura 3: Arquitetura da Web Semântica (Berners – Lee et al., 2000).

- ***Camada de Estrutura*:** a representação do conhecimento é o alicerce da Web Semântica. Nesta camada são definidas as estruturas de dados. Várias linguagens de marcação de conteúdo derivadas de XML (e**X**tensible **M**arkup **L**anguage) (Pitts et al., 2000) tem sido criadas para este fim. RDF e RDF Schema (Abitebul et al., 2000) são atualmente as mais relevantes.

- ***Camada de Esquema:*** também chamada de camada ontológica. O uso de ontologias permite a resolução de problemas de identificação ou conflito de terminologia. O desenvolvedor de uma página XML pode ligá-la a uma ontologia que defina o significado de cada uma das tags utilizadas. A partir deste momento, sua integração com outras páginas XML, que possuam também sua própria ontologia definida, passa a ser possível através da

aplicação de regras de inferência definidas na camada lógica. Algumas linguagens que suportam a construção de ontologias são: OIL(Horrocks et al., 2000), DAML-OIL (Horroks, 2001) e OWL (Bechhofer et al., 2003).

- ***Camada Lógica:*** nesta camada são empregadas linguagens de representação de conhecimento formal, com primitivas comumente empregadas em linguagens de Descrição Lógica *(Description Logics)* (Nardi et al., 2002), de modo a proporcionar suporte ao *"raciocínio"* automático. Também são definidos os mecanismos de inferência sobre os dados. As regras de inferência fornecem aos agentes a possibilidade de *"raciocinar"* sobre os termos e seus significados definidos nas camadas inferiores. A linguagem RuleML *(Rule Markup Language)* (URL 2), em desenvolvimento pelo grupo de pesquisa *Rule Markup Initiative,* tem como objetivo expressar regras de inferência baseadas em linguagens como Prolog e Lisp em marcações XML. No nosso trabalho usamos a linguagem Lisp para testar um algoritmo de composição como mostramos na nossa contribuição.

2.2.2 Aplicações da Web Semântica

A Web Semântica proporciona uma evolução significativa em muitas áreas, dentre as quais destacam-se: Gestão do Conhecimento (URL 3), Assistentes Pessoais (URL 4), Recuperação de Informação (Stanley, 1999) e o Comércio Eletrônico.

A evolução das ferramentas de Comércio Eletrônico, mais especificamente do Business to Business (B2B), é um dos propósitos de nosso trabalho. Tradicionalmente, o B2B vem sendo realizado através de acordos bilaterais entre as

empresas, por exemplo, o padrão EDI *(Eletronic Data Interchange, Troca Eletrônica de Dados)*, requer que cada parceiro comercial negocie individualmente com o outro em um modelo *"um-para-um"*. A introdução de tecnologias decorrentes do desenvolvimento da Web Semântica torna possível a evolução para um modelo de negociação *"muitos-para-muitos"*, onde cada cliente tem a possibilidade de localizar e negociar com vários fornecedores e cada fornecedor tem a possibilidade de oferecer seus produtos ou serviços para vários clientes. É esta a finalidade do sistema ICS que estamos propondo para suportar o comércio eletrônico na categoria B2B. O ICS promove um modelo de negociação *"muitos-para-muitos"* entre clientes e fornecedores com baixo custo e boa performance.

As arquiteturas baseadas em serviços representam um novo paradigma para modelagem de sistemas. Na próxima seção apresentamos uma visão dos Web Services, aplicações fracamente acopladas que se comunicam através da WEB e que estão tendo um papel crescente nas interações entre as empresas através da Internet, principalmente nas cadeias de fornecimento entre empresas, constituindo redes de negócios do tipo Business-to-Business (B2B).

2.2.3 Web Services Semânticos

A World Wide Web deixou de ser um repositório de textos e imagens para se transformar em um provedor de serviços. Na última década, muitos sites disponibilizaram para os seus usuários serviços de compras, reservas, consultas, notícias, dentre outros. Entretanto, essas aplicações são, em sua maioria, baseadas em bancos de dados centralizados, pertencentes a um prestador de serviço específico, e, por isso, fornecem informações sobre uma única organização.

A definição de padrões de integração de sistemas de múltiplas organizações que atuam em domínios de negócios comuns, com o objetivo de compilar e reunir informações para seus clientes é ainda uma questão aberta. Muito esforço tem sido despendido para alcançar este objetivo, principalmente em função do crescimento exponencial das fontes de dados heterogêneas espalhadas pela Internet.

Os Web Services são um novo paradigma para implementação de arquiteturas orientadas a serviços na Internet, onde os sites trocam informações dinamicamente e sob demanda.

Nas arquiteturas baseadas em Web Services, os componentes são vistos como prestadores de serviços que realizam uma tarefa específica. Uma aplicação é vista como uma composição de serviços que juntos resolvem um problema para um determinado usuário.

Os Web Services são suportados pela SOA - Service Oriented Architecture. Na SOA, uma aplicação possui três entidades: o Provedor de Serviço *(Service Provider)*, o Solicitador de Serviço *(Service Requester)* e o Corretor de Serviços *(Service Broker)*.

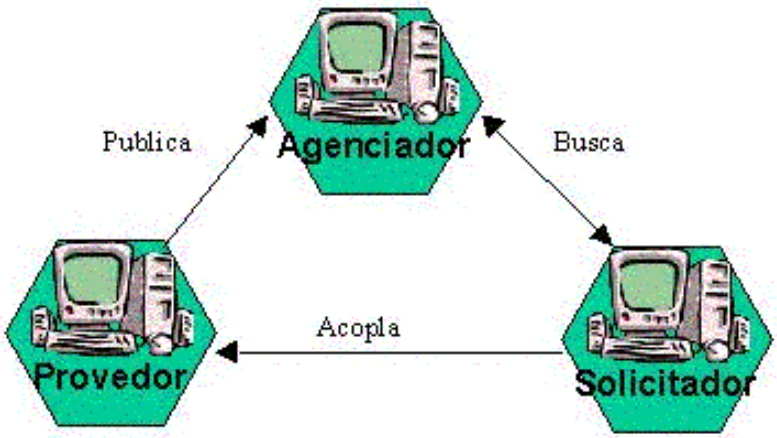

Figura 4: SOA – Service Oriented Architecture

Um Provedor de Serviço fornece a realização de algum serviço. Para tanto, deve anunciar suas funcionalidades. Um Solicitador de Serviços é uma entidade que irá utilizar-se de um serviço prestado por um Provedor de Serviço (um Web Service), logo, ele deve descobrir o Web Service que melhor atenda às suas necessidades. O Service Broker desempenha o papel de localizar o Provedor de Serviço que melhor se adeque às necessidades do Solicitador de Serviços.

Padrões de descoberta como UDDI *(Universal Description, Discovery and Integration)* e linguagens de descrição de serviços como WSDL *(Web Service Description Language)* são empregadas com este fim (Daum, 2002). Entretanto, eles não fornecem suporte a uma busca semântica. No padrão UDDI, atualmente não há mecanismo para descrever Metadados de serviços. Assim, a busca por partes que podem prover um produto ou serviço específico por um dado preço não é possível em um registro UDDI. Usualmente, o usuário pode encontrar um serviço a partir de seu código baseado em uma taxonomia pré-definida como UNSPSC (Universal Standard Products and Services Classification) e NAICS (National American Industry Classification System).

A SOA ainda necessita de alguns melhoramentos. Especificamente, o UDDI ainda tem algumas carências que necessitam ser corrigidas para automatizar a seleção, composição e interoperação de Web Services para permitir a construção de serviços que satisfaçam as reais necessidades do comércio eletrônico. A seguir enumeramos as principais carências:

- falta de semântica em WSDL e UDDI – um modelo semântico é necessário para interpretar as consultas e raciocinar sobre o conhecimento, por exemplo, a

requisição por serviços de “locação de automóveis” não descobre “locação de veículos”.

- falta de suporte para ligação entre conceitos inexatos e suporte multilingual - por exemplo, uma requisição por serviços de “locação de carros”, não descobre “rent of cars”;

- falta de especificação de Workflow – a linguagem WSDL não permite a especificação da ordem correta de invocação das operações, assim a composição do fluxo (plano) é ainda obtida manualmente.

A perspectiva da Web Semântica permite superar as limitações supramencionadas. A aplicação dos padrões e ferramentas decorrentes do desenvolvimento da Web Semântica para superar as limitações atuais da tecnologia de Web Services origina o que a comunidade acadêmica convencionou chamar de Web Services Semânticos (Semantic Web Services).

2.2.4 Adequação da Web Semântica e dos Web Services para o desenvolvimento de Ferramentas B2B

As arquiteturas baseadas em serviços proporcionam um modelo de negócio bastante flexível, onde as empresas oferecem e compram bens ou serviços de forma automática através da utilização da infra-estrutura da Internet. Segundo (Burbeck, 2000), existem três cenários para colaboração de serviços B2B:

- **Acoplamento rígido, estabelecido em tempo de compilação:** A aplicação precisa conhecer os detalhes do serviço com o qual vai colaborar porque o acoplamento é feito durante a concepção da aplicação. Deste modo, a aplicação sabe exatamente como interagir com o serviço.

- **Acoplamento dinâmico a colaborador pré-estabelecido:** A aplicação faz uso de um agenciador *(Broker)* para localizar um serviço específico. Neste cenário a aplicação sabe exatamente como fazer a solicitação para o agenciador, isto porque o programador codificou uma consulta específica a ser feita ao agenciador.

- **Acoplamento dinâmico e escolha dinâmica do colaborador:** A aplicação sabe a semântica e as chamadas da API do serviço a ser usado. Entretanto, consulta um agenciador com um padrão de busca que permite o retorno de uma série de alternativas. A aplicação escolhe um serviço da lista que melhor lhe atenda em tempo de execução.

A Web Semântica representa um grande passo para a elevação das aplicações de comércio eletrônico em direção ao terceiro cenário descrito por Burbeck (Burbeck, 2000). A descoberta de um serviço por uma aplicação é fortemente dependente de sua descrição. Linguagens de descrição semanticamente expressivas e formalmente definidas que possibilitam o processamento automático, por software, das anotações sobre os serviços que têm um papel central no sucesso das aplicações baseadas em Web Services. Burbeck (Burbeck 2000) ainda aponta os requisitos de alto nível destas descrições:

- Os serviços devem ser descritos em XML. Essas descrições precisam ser ao mesmo tempo legíveis por humanos e processadas por programas. Precisam ser extensíveis, uma vez que a evolução dos serviços Web não pode ser prevista em todos os seus detalhes;

- As descrições de serviços devem prover toda a informação semântica necessária aos solicitadores para que estes verifiquem se os requisitos serão satisfeitos e aos agenciadores para estes decidirem sobre a categorização precisa do serviço em sua taxonomia;

- As descrições devem ainda possibilitar a especificação de requisitos não-funcionais, como segurança, autenticação e privacidade, importantes para garantir a troca de informações necessárias para o consumo do serviço e questões contratuais.

O desenvolvimento da Web Semântica fez surgir algumas linguagens e padrões para anotação de Web Services como LARKS (*Language for Advertisement and Request for Knowledge)* (Sycara, 2002), *DAML-S (DARPA*[1] *Agent Markup Language for Services) e mais recentemente OWL-S (Web Ontology Language for Services)* (Martin et al).

Atualmente as aplicações de comércio eletrônico enquadram-se nos dois primeiros cenários descritos por Burbeck: acoplamento rígido e acoplamento dinâmico a colaborador pré-estabelecido. O ICS enquadra-se no terceiro cenário: acoplamento dinâmico e escolha de colaborador dinâmica. No ICS, o agente Matchamaker descrito em (Tomaz, 2003), desempenha o papel do agenciador descrito por Burbeck e o nosso trabalho estende a capacidade de colaboração dinâmica para serviços compostos automaticamente usando técnicas de planejadores automáticos.

[1] Defense Advanced Research Project Agency. Agência de Projetos de Pesquisa Avançada de Defesa dos EUA.

2.2.5 Conclusão

Os paradigmas apresentados neste capítulo são fundamentais para o desenvolvimento de nosso trabalho. O ICS é baseado em uma arquitetura SMA *(Sistema Multiagente)*. Resolvemos o problema da descoberta entre os agentes no ICS com o uso dos Web Services como blocos de montagem de nossa arquitetura e aliamos a visão da Web Semântica como forma de proporcionar um mecanismo de descoberta e composição automática dos Provedores de Serviços *(Services Providers)* pelos Solicitadores de Serviços *(Services Requesters).*

Na próxima seção apresentamos uma visão detalhada de *Ontologias* para entendimento de todas as fases do ciclo de vida no ICS.

2.3 Ontologias

A seguir discorremos um pouco sobre as técnicas de representação de conhecimento com o objetivo de proporcionar uma visão mais ampla do entendimento das ontologias, componente fundamental no contexto da Web Semântica e um detalhamento dos conceitos e aplicações da ontologia no ICS.

Em IA, o termo ontologia é usado para indicar um domínio de conhecimento ou o domínio semântico para um agente.

Os formalismos de representação de conhecimento possibilitam a estruturação de conhecimentos em ontologias, com conceitos e termos pertencentes a determinado domínio.

Ontologias são acordos a respeito de conceitualizações compartilhadas (Gruber, 1993). Usam-se ontologias comuns para descrever a representação de

uma base de conhecimento para um conjunto de agentes. Desta forma, os agentes podem passar informações sobre o domínio do discurso, sem necessariamente operar sobre uma teoria globalmente compartilhada (Bonifácio, 2002).

Até meados de 90 o conhecimento de um sistema especialista não podia ser reusado ou compartilhado, organizando-se em bases de conhecimentos monolíticas e isoladas, sem interface de acoplamento, e, portanto, sem interoperabilidade (Freitas, 2002).

As ontologias proporcionam um avanço significativo na representação de conhecimento, na medida em que são reusáveis, extensíveis, especializáveis e interoperáveis. Ou seja, as ontologias possibilitam a representação de conhecimento de uma maneira evolutiva. Podemos dizer que as ontologias representam para a abordagem declarativa de representação de conhecimento o que o paradigma da orientação à objetos representa para a abordagem procedural.

Quando as linguagens de alto nível foram desenvolvidas, duas abordagens antagônicas de projetos de sistemas emergiram: abordagem procedural e abordagem declarativa.

O paradigma procedural é mais adequado ao uso da arquitetura de Von Neumann[2] como modelo para representação da solução de um problema a ser resolvido pela máquina. Representar a solução de um problema para ser resolvido pelo computador no paradigma procedural consiste em escrever uma série de ações (procedimentos) que, se executadas seqüencialmente, conduzem à solução, ou

[2] Arquitetura proposta por John Von Neumann onde a execução de programas é baseada nos fluxos de instruções. Programa e dados encontram-se armazenados em uma memória. As instruções são buscadas na memória e então executadas. Instruções especiais, de desvios condicionais e incondicionais, são utilizadas para mudar o fluxo de controle.

seja, o programa representa a descrição da solução para o problema. A abordagem procedural, por ser mais próxima do modelo de Von Neumann, é mais eficiente em termos de performance e, atualmente, é o paradigma de programação mais utilizado.

Já na abordagem declarativa, o significado de um programa não é mais dado por uma sucessão de operações elementares que o computador realiza, mas por uma base de conhecimento a respeito de certo domínio. Por trás da base de conhecimento há uma máquina de inferência, em princípio *"escondida"* do programador, responsável por *"encontrar soluções"* para o problema descrito.

Guarino (Guarino,1998) classifica as ontologias em quatro tipos, de acordo com sua dependência em relação a uma tarefa específica ou a um ponto de vista:

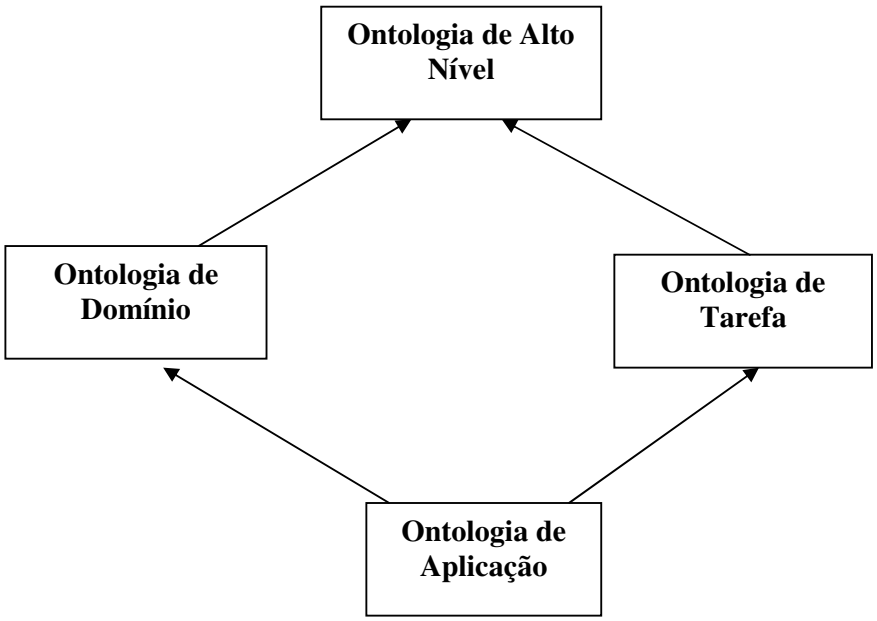

Figura 5: Tipos de Ontologias e seus relacionamentos (Guarino,1998).

- ***Ontologiam de Alto Nível:*** também chamada de ontologia de nível superior, descreve conceitos gerais como espaço, tempo, objeto, etc. Estes conceitos são dependentes de um problema específico.

- ***Ontologiam de Domínio:*** descrevem um vocabulário relacionado a um domínio genérico, como automóveis, medicamentos, livros, etc.

- ***Ontologias de Tarefa:*** descrevem uma tarefa ou atividade dentro de um domínio, como venda de carros, compra de livros, vendas de medicamentos, etc.

- ***Ontologias de Aplicação:*** descreve conceitos que dependem tanto de um domínio específico como de uma tarefa específica.

Uma ontologia contém, tipicamente, um vocabulário de conceitos ou classes, relacionamento entre conceitos, atributos de conceitos e um conjunto de axiomas que definem as regras do negócio (Erdmann, 2000), servindo para recuperação semântica de informações em ambientes heterogêneos de fontes de dados semi-estruturadas como ocorre na Web.

Os dados disponíveis na Web são ditos semi-estruturados, pois possuem metadados implicitamente armazenados juntamente com os dados, sendo também chamados de auto-descritivos. Pode-se ainda dizer que estes dados não são nem totalmente não-estruturados nem estritamente tipados (Abitebul et al., 2000).

2.3.1 Ontologias e a Web Semântica

A Web pode ser vista como um enorme banco de dados de documentos disponibilizados para leitura. É natural que a vejamos desta forma, no entanto, a

aplicação das técnicas de projeto e a utilização das ferramentas de gerenciamento de banco de dados convencionais para gerenciar e integrar dados na Web não é trivial. O rigor dos Sistemas Gerenciadores de Bancos de Dados *(SGBD's)* tradicionais não é aplicável ao formato dos dados encontrados na Web. Na Web os dados são semi-estruturados.

Uma ontologia pode ser entendida como um modelo conceitual cujo objetivo é permitir o acesso aos dados semi-estruturados, exatamente como acontece nos bancos de dados convencionais. Existe, entretanto, uma sutil diferença entre uma ontologia e um modelo conceitual. Um modelo conceitual descreve, dentre outras coisas, a estrutura dos dados em um banco de dados em alto nível de abstração. Uma ontologia não representa a estrutura das fontes de dados associadas a ela, apenas propõe uma estrutura de consenso para conceitos e relações que são úteis para grupos de especialistas (Arruda, 2002).

A utilização de ontologias como suporte para o desenvolvimento de soluções sob a perspectiva da Web Semântica tem dois propósitos:

i) Garantir um entendimento conceitual entre os agentes acerca de um determinado domínio a partir da definição de um vocabulário comum e compartilhado por todos os agentes de uma comunidade;

ii) Atribuir significado aos recursos disponíveis na Web para possibilitar que os agentes possam fazer uma recuperação semântica dos conteúdos publicados

A linguagem HTML *(HyperText Markup Language)* se preocupa apenas com a apresentação de conteúdo e não atribui qualquer significado aos documentos, o que

limita os agentes à recuperação sintática das informações contidas nos documentos publicados atualmente na Web.

A linguagem XML, padrão para publicação, combinação e intercâmbio de documentos, desenvolvido pelo consórcio W3C *(World Wide Web Consortium)*, tem servido como plataforma para definição de algumas linguagens para representação de ontologias adequadas para uso no âmbito da Internet.

Na próxima seção, algumas das principais linguagens desenvolvidas sobre a plataforma XML para representação de ontologias.

2.3.2 Linguagens para Representação de Ontologias

A seguir apresentamos as principais linguagens utilizadas para a representação de conhecimento no domínio da Web Semântica

Muitas são as linguagens de representação definidas. Para representar uma ontologia é necessária uma linguagem específica. Entretanto, para aplicações Web é necessário que a linguagem de anotação ontológica possua uma sintaxe no padrão XML. As linguagens para representação de ontologias que nos interessam são as utilizadas para construir ontologias compartilhadas por agentes de software no âmbito da Web.

RDF e RDF Schema (RDFS)

RDF *(Resource Description Framework)* é uma linguagem para representação de metadados baseada em XML.

RDF capacita a modelagem semântica, pois, sobre a sua especificação foram definidas outras linguagens com primitivas suficientes para a completa modelagem e raciocínio ontológico.

A linguagem RDF permite fazer declarações a respeito de recursos Web, não apenas páginas, mas também a qualquer componente de software ou de hardware acessível pela Web e, também a recursos off-line, que não podem ser acessados diretamente pela Web.

As declarações RDF possuem a forma de uma tripla, que consiste de um predicado, um sujeito e um objeto. O sujeito é o recurso que se quer descrever, o predicado é cada propriedade do recurso e, o objeto os valores das propriedades.

```
<rdf:RDF>
    <rdf:Description about=”Aorta”>
        <localizacao>tronco</localização>
    </rdf:Description>
</rdf:RDF>
```

Figura 6: Representação RDF

O modelo RDF descreve recursos sempre no nível de instância de recursos. RDF Schema (RDFS) permite definir uma taxonomia de recursos em termos de classes, subclasses e superclasses de recursos.

O uso de RDFS é opcional e provê informações sobre a interpretação das declarações formuladas em um modelo de dados RDF. A expressividade das primitivas de RDF e RDFS não é suficiente para a completa modelagem e raciocínio ontológico. RDF e RDFS possuem capacidade de representação de restrições e de raciocínio sobre suas declarações bastante limitadas. Entretanto, RDF e RDFS

serviram como um ponto de partida para construção de linguagens que permitem expressar o completo raciocínio ontológico.

RDF pode ser vista como uma tecnologia de capacitação para a modelagem semântica, como uma linguagem montadora genérica, sobre a qual podem ser criadas linguagens específicas do domínio e da tarefa (Daum, 2002).

DAML-OIL

O Joint Commitee US/EU foi criado em outubro de 2000 por Jim Hendler do US Defense Advanced Research Projects Agency *(DARPA)* e Hans-Georg Stork da European Union Informations Society *(IST)* com o objetivo de desenvolver uma linguagem semanticamente expressiva para representar o completo raciocínio ontológico.

Inicialmente uma linguagem chamada DAML-ONT *(DARPA Agent Markup Language)* foi desenvolvida pelo DARPA para que entendesse RDF com construtores das linguagens de representação de conhecimento baseada em Frame. Como RDFS, DAML-ONT tinha uma especificação semântica fraca.

Paralelamente aos esforços do DARPA, na construção de DAML-ONT, um grupo de pesquisa europeu ligado ao *IST* desenvolveu uma linguagem de descrição de ontologias voltada para a Web chamada OIL *(Ontology Inference Layer)*. Do mesmo modo que DAML-ONT, OIL possuía uma sintaxe baseada em RDFS e um conjunto de construtores baseados em Frames. Entretanto, um forte rigor formal foi estabelecido e a linguagem foi explicitamente definida. OIL possui uma semântica formal e suporte a raciocínio provido pelas linguagens de lógica de descrição.

Os grupos de pesquisa resolveram unir seus esforços e unificaram DAML-ONT a OIL, produzindo deste modo, DAML-OIL.

Um conjunto de declarações DAML-OIL pode permitir a conclusão de uma outra declaração DAML-OIL. Por exemplo, podemos expressar o fato de que "*paternidade*" é um relacionamento mais genérico que "*maternidade*" e que "Maria é mãe de William" juntas, as definições permitem a conclusão de que "Maria é um dos pais de William". Desta forma, se um usuário perguntar "*quem são os pais de William?*", o sistema pode responder que "*Maria é um dos pais de William*", mesmo que o fato não tenha sido declarado em nenhum lugar.

A seguir, apresentamos DAML-S uma ontologia de alto nível utilizada para descrever Web Services. A ontologia DAML-S representa uma padronização na forma de descrever Web Services utilizando notação DAML-OIL. No ICS, em (Tomaz, 2003), a DAML-S é usada para fazer as anotações das propagandas e dos requisitos de compras no ICS. No contexto de planejamento automático, a DAML-S é traduzida para um planejador (SHOP2) para fazer o plano de composição de serviços Web descoberto pelo agente Matchmaker aumentando assim o número de negócios realizados no ICS.

DAML-S

Para fazer uso de um Web Service um agente de software necessita de uma descrição "*machine understandable*" do serviço que ele realiza e da forma como ele pode ser acessado. Com o objetivo de criar uma linguagem semanticamente expressiva para descrever as capacidades de um Web Service está sendo desenvolvida a linguagem de marcação DAML-S *(DARPA Agent Markup Language*

for Services), uma ontologia DAML-OIL de alto nível para descrição de propriedades e capacidades de Web Services.

DAML-S provê uma padronização de como descrever as funcionalidades de um Web Service *(ServiceProfile)*, como o Web service realiza sua tarefa *(ServiceModel)* e como podemos acessá-lo *(ServiceGrouding)*. Dentre os benefícios proporcionados por DAML-S destacamos a automação de Web Services por meio da utilização de agentes e, a possibilidade de raciocínio *(reasoning)* sobre as propriedades e capacidades dos Web Services. Como podemos observar na figura 8, o modelo DAML-S possui três componentes: *ServiceProfile, ServiceModel* e *ServiceGrouding.*

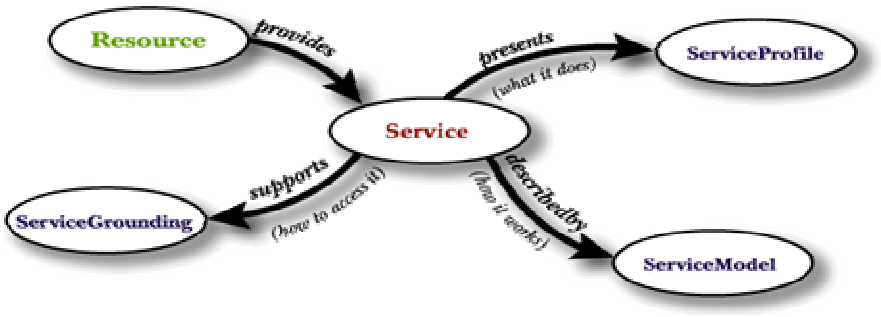

Figura 7: Ontologia de Serviço DAML-S (Anupriya, 2002).

Cada seção da ontologia DAML-S possui objetivos e funcionalidades particulares:

- ServiceProfile
  - Objetivo: proporcionar uma descrição de alto nível do Web Service.
  - Utilidade: povoar serviços de registro, descoberta automática de serviços e *Matchmaking***.**

- ServiceModel
  - Objetivo: descrever como o Web Service trabalha.
  - Utilidade: composição, interoperação e monitoramento de Web Services.
- ServiceGrouding
  - Objetivo: especificação da forma de acesso para o Web Service.
  - Utilidade: descrição de protocolos de comunicação e mecanismos de transporte. Ex. CORBA, http, RMI, etc.

Nós temos particular interesse pelas informações contidas na *ServiceProfile*. Faremos, portanto, maior detalhamento desta seção.

DAML-S ServiceProfile

O objetivo do *ServiceProfile* é descrever as funcionalidades do Web Service, ou seja, o que o Web Service provê.

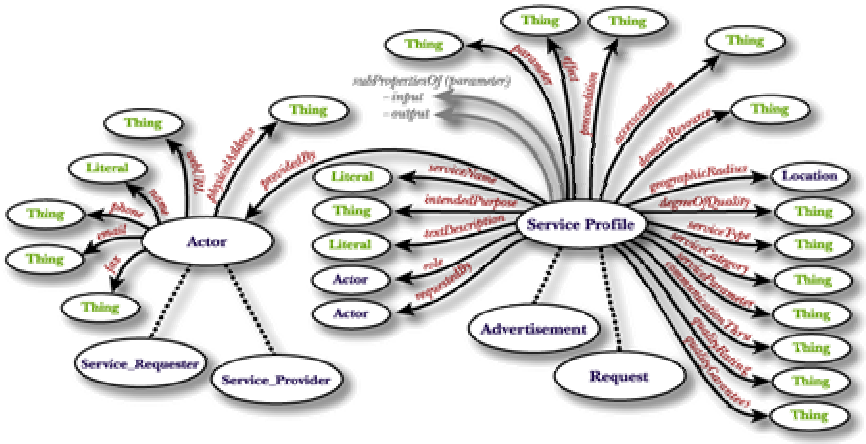

Figura 8: Service Profile (Anupriya, 2002).

Como podemos observar na figura 8, a ServiceProfile provê uma superclasse para descrição de alto nível dos serviços. No contexto ICS a ServiceProfile pode descrever tanto os requisitos de um comprador quanto a propaganda de um fornecedor. A propriedade *presents*, como podemos observar na figura 5, relaciona a classe *ServiceProfile* a classe *service*. A propriedade *presentedBy* é o inverso de presents e especifica que um dado profile descreve um serviço. ServiceProfile fornece também informações *"human-readable"* como: *serviceName*, *textDescription* e *contactInformation*. A classe *Actor* provê informações sobre o cliente ou fornecedor do serviço. A classe *ServiceProfile* ainda fornece a descrição funcional do serviço que é especificada em termos de *entradas (inputs)*, *saídas (outputs)*, pré-*condições (pre-conditions)* e *efeitos (effects)*.

- Uma entrada é o que o Web Service requer para produzir a saída desejada.

- Uma saída é o que o Web Service fornece como resposta a uma solicitação, é a confirmação de que uma solicitação foi recebida e devidamente processada.

- Pré-condições representam as condições do mundo real que devem ser verdadeiras para que o serviço seja executado. Os efeitos são as mudanças ou ações efetivadas pelo Web Service, por exemplo, a venda de um produto.

A motivação para a escolha dessa linguagem é que a DAML-S é uma ontologia de alto nível que utiliza a notação DAML-OIL. Utilizamos conceitos e técnicas da DAML-S para descoberta e composição dos serviços, pois ela fornece a

mesma descrição funcional dos serviços (entrada, saída e pré-condições) da dos planejadores mais eficientes, sendo assim a tradução da DAML-S para SHOP2 para viabilizar a construção de planos de composição de serviços no ambiente ICS.

DAML Service Model

O Service Model descreve o fluxo de controle e dados envolvidos na utilização dos serviços. Ele foi planejado para permitir a composição e execução automática dos serviços.

O Service Model descreve o funcionamento interno do serviço e é especializado como um modelo de processo (ProcessModel) na Ontologia de Processos. Na Ontologia de Processos DAML-S, cada serviço é modelado como um processo:

- que produz a transformação de dados de entrada em dados de saída, e;

- que produz a transição do mundo de um estado para outro. Estas transições são descritas pelas pré-condições e efeitos do processo.

O modelo de processo (ProcessModel) tem três tipos de processos: Atômico, Simples e Composto. Um processo atômico não pode ser decomposto e representa um Web Services diretamente executável. Um processo simples é utilizado como elemento de abstração para prover uma visão ou de um processo atômico ou uma representação simplificada de um processo composto.

O processo composto pode ser decomposto em outros processos compostos ou em processos atômicos e representa um Web Service Composto. A decomposição de um Web Service composto é especificada através de suas

estruturas de controle, que podem ser: Sequence, Split, Split+Join, Unordered, Choice, If-Then-Else, Iterate, Repeat-While and Repeat-Until.

Cada Web Services é visto como a generalização de uma tarefa que as pessoas querem realizar na Web. A definição DAML-S de um processo atômico provê toda informação de um Web Services que é executado diretamente para realizar a tarefa. A definição DAML-S de um processo composto especifica toda informação necessária para seleção e composição de Web Services para realizar a tarefa. A Composição Automática de Web Services consiste no desenvolvimento de Software capaz de manipular as definições DAML-S, encontrar automaticamente um conjunto de processos atômicos, cuja execução em uma determinada ordem realize a tarefa. Para gerar o plano de composição dos Web Services atômicos sugerimos o uso da tecnologia de AI Planning.

2.3.3 Conclusão

A visão da Web Semântica, em particular a utilização de uma ontologia compartilhada pelos agentes em um domínio de aplicação, proporciona uma solução para o problema da descoberta e emparceiramento automático de Web Services. Para tanto, a ontologia tem que ser descrita utilizando-se uma linguagem formal - para permitir o tratamento automático por agentes de software e, semanticamente expressiva - para descrever com precisão as especificações dos clientes e as propagandas dos fornecedores.

3 INTELLIGENT COMMERCE SYSTEM – ICS

Neste capítulo, descrevemos o ambiente ICS (Labidi et al., 2003) de suporte ao comércio eletrônico na categoria B2B. Destacamos os principais conceitos, características e funcionalidades de cada fase do ciclo de vida. O objetivo é fornecer uma visão clara do sistema, de modo que possamos delimitar e justificar nossa contribuição dentro do ICS.

3.1 Introdução

A automação do comércio para resolver problemas de tomada de decisão, pode ser solucionada por agentes de software (Jennings et al., 1998)

O ICS surgiu como uma alternativa proposta à modalidade B2B, onde agentes de software negociam a compra e venda de produtos ou serviços no ambiente da Web. O desenvolvimento e o funcionamento do ICS acontece seguindo um ciclo de vida composto por cinco fases.

A principal finalidade do ICS é automatizar as tarefas de encontrar parceiros comerciais, negociar e fechar contratos, atualmente realizadas por operadores humanos nos portais B2B.

O ICS é baseado em agentes inteligentes móveis e estacionários visando automatizar o processo de compra e venda entre as empresas, reduzindo seus custos operacionais e agilizando seus processos de negócios. Os agente móveis são responsáveis por representar os usuários do sistema, ou seja, empresas que desejam realizar negócios através da Internet. Os agentes estacionários realizam

funções internas que permitem implementar seus procedimentos. Os agentes fazem uso da infra-estrutura da Internet para se comunicar, negociar e fechar contratos.

Nas seções seguintes apresentamos em mais detalhes cada componente do ICS.

3.2 Ciclo de Vida do Comércio Eletrônico no ICS

O ICS utiliza uma extensão do modelo de ciclo de vida proposto por (Jennings et al., 1996) e (Bartolini, 2001). Foram introduzidos: a fase de modelagem do usuário, o conceito de feedback de informações e ontologias específicas para cada fase. Ficando o ciclo de vida do comércio eletrônico no ICS como ilustrado na figura 9.

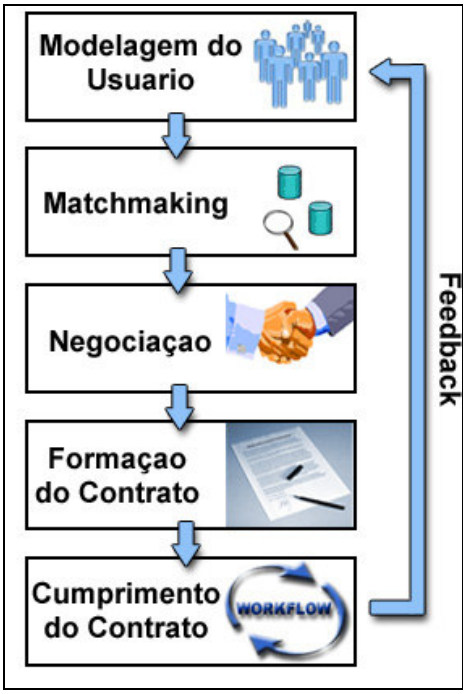

Figura 9. Ciclo de vida do comércio eletrônico no ICS.

- ***Modelagem do usuário:*** É uma característica importante do ICS que permite um tratamento personalizado a cada agente negociador dentro do mercado virtual com base em informações coletadas do próprio ambiente de negociação.

- ***Matchmaking:*** É um dos principais aspectos em ambientes de comércio eletrônico e consiste basicamente em aproximar fornecedores e compradores com interesses complementares. No ICS, o *matchmaking* é realizado fazendo-se a comparação sintática e semântica das características que são requeridas por uma das partes negociantes (o comprador) e oferecidas pelas outras partes negociantes (os vendedores).

- ***Negociação:*** Uma vez agrupados os possíveis parceiros de negócio, os agentes negociantes iniciam o processo de negociação para concordar ou discordar os temas mutuamente aceitáveis, ou seja, cada agente busca a compatibilidade dos interesses da empresa que representa com os interesses das demais empresas.

- ***Formação do Contrato:*** Consiste da formalização em um contrato eletrônico com validade jurídica dos termos que regem a negociação como: preço, prazo, forma de pagamento, meio de entrega etc. Para tanto, o contrato eletrônico deve ser assinado por cada uma das partes e certificado por uma entidade certificadora como *Verisign* a fim de garantir sua privacidade, autenticidade, integridade e o não repúdio por alguma das partes.

- ***Execução de Contratos:*** Os contratos trabalham com datas limites e sua execução segue um protocolo. Um contrato pode ser visto como

um protocolo de recompensa ou punição, onde os agentes são recompensados quando cumprem o contrato e punidos em caso contrário. Propomos a utilização da tecnologia de *Workflow* temporal (Labidi et al., 2000) para controlar o fluxo de execução dos contratos firmados dentro do ICS.

3.3 Arquitetura do ICS

O ICS foi idealizado para proporcionar uma arquitetura aberta, flexível e evolutiva. Com esta finalidade foram empregadas camadas de abstração de alto nível. O ICS é baseado na arquitetura de agentes. Utilizamos agentes móveis para representar os negociantes (vendedores e compradores). O objetivo desta abordagem é proporcionar maior robustez e agilidade no processo de negociação, pois deste modo, os agentes negociantes podem migrar através da rede para um lugar comum (um *host*) e somente depois de estarem fisicamente próximos, iniciarem um processo de negociação.

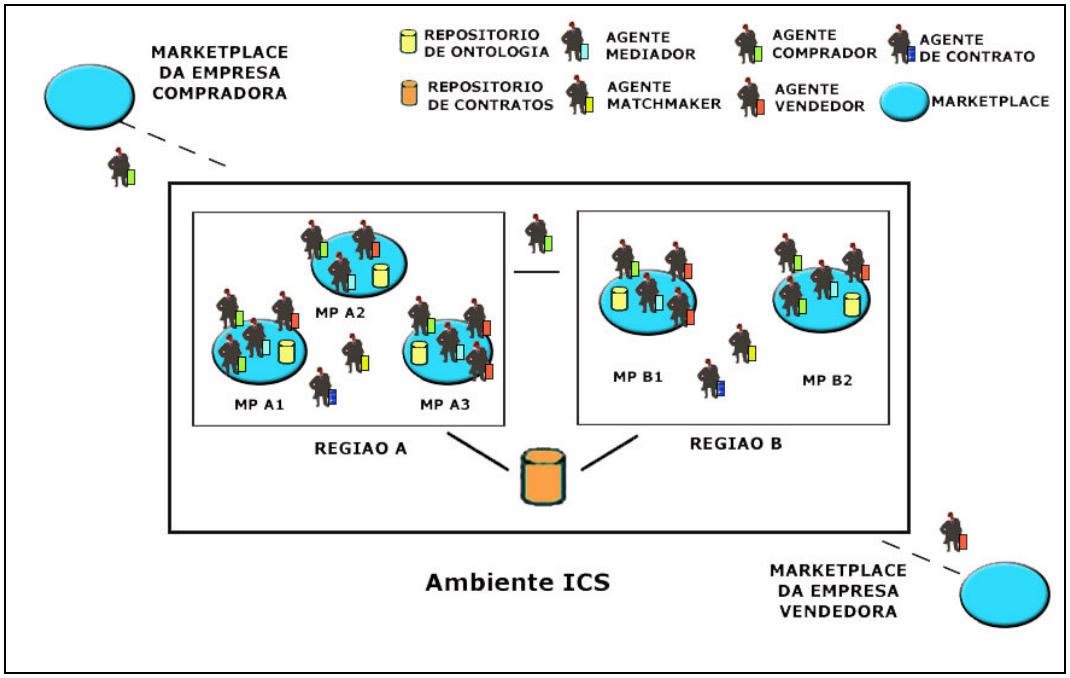

Figura 10: Arquitetura do ICS.

A arquitetura do ICS é baseada em duas abstrações: *Mercado Virtual e Região,* quatro classes de agentes: *Agente Matchmaker, Agente Mediador, Agente Negociante e Agente de Modelagem*, e três repositórios de dados semi-estruturados: *Repositório de Ontologias, Base de Estereótipos e Base de Propagandas* (Tomaz, 2003). A figura 7 ilustra uma visão de alto nível da arquitetura do ICS.

A seguir, detalhamos cada componente do ICS. Os componentes estão dispostos em uma seqüência didática que permite uma melhor compreensão de cada componente bem como as interações existentes entre eles.

3.3.1 Marketplace

Um marketplace é o local onde os agentes realizam suas funções - a negociação e o fechamento do contrato. Nele ocorrem todos os procedimentos internos do ICS que fornecem suporte ao processo de comércio eletrônico por ele provido. Isto é, detalhes sobre protocolos de comunicação, de negociação, ontologia, identificação de agentes negociadores em vendedores e compradores, armazenamento de anúncios em repositórios, formação de contratos, entre outros. Todos são resolvidos dentro de um marketplace.

Um marketplace é instanciado em uma máquina (*host*), ou seja, os agentes operam localmente. Entretanto, nada impede que em uma máquina seja instanciado mais de um mercado virtual.

3.3.2 Região

Uma região é o agrupamento de marketplaces que provê um alto nível de abstração para a comunicação entre agentes. Os marketplaces de uma região são

reunidos pela mesma área de negócios, modelada através de uma ontologia de domínio. Por isto, a comunicação entre os agentes se dá apenas entre marketplaces de uma mesma região, exceto para os móveis que podem, adicionalmente, trafegar entre diferentes regiões.

3.3.3 Repositório de Ontologias (modelos de domínios)

É uma base de dados semi-estruturada que permite armazenar as ontologias de domínio. O ICS não se limita a operar em um domínio de negócio especifico, e pode evoluir para quantos domínios se queira. Para tanto, basta adicionar uma nova ontologia de domínio no repositório de ontologias. Cada Ontologia armazenada no repositório está associada a uma região específica.

3.3.4 Base de Estereótipos

É uma base de dados semi-estruturada que permite descrever o perfil de cada usuário. É um protótipo do modelo do usuário. É utilizada pelos Agentes negociantes para deliberarem sobre as preferências das empresas que eles estão representando. Por exemplo, determinada empresa pode preferir prazo ou qualidade a preço. Deste modo, podemos dizer que a Base de Estereótipos dá suporte às negociações entre os Agentes Compradores e Vendedores.

3.3.5 Base de Propagandas

É uma base de dados semi-estruturada que armazena todas as propagandas anotadas em DAML-S. Após um comprador especificar seus requisitos de compra o agente *Matchmaker* percorre este repositório em busca de propagandas que satisfaçam aos requisitos do comprador.

3.3.6 Agente Matchmaker

O agente *Matchmaker* tem por objetivo aproximar agentes negociantes com objetivos complementares, ou seja, ele é o responsável pela composição dos Mercados Virtuais.

Para realizar sua tarefa o Agente *Matchmaker* precisa descobrir os identificadores dos agentes com objetivos complementares, instanciar um Marketplace e instanciar um Agente Mediador que controlará o processo de negociação entre os agentes negociantes dentro do Marketplace.

3.3.7 Agente Mediador

O Agente Mediador opera como um árbitro dentro de um Marketplace. Ele acompanha cada transação realizada e intervém quando necessário com o objetivo de resolver problemas de negociação, formação e execução de contratos. Para cada Marketplace há uma instância do Agente Mediador.

Uma vez aproximados pelo *Matchmaker*, os agentes negociantes iniciam a negociação sob a supervisão do mediador, que pode desclassificar ou punir agentes que infringirem regras básicas de negociação. Por exemplo, um agente pode intencionalmente reduzir seu preço a valores inexeqüíveis para ganhar uma concorrência e depois não firmar o contrato de compra e venda.

Outra característica relevante do agente mediador é que ele registra todas as negociações realizadas dentro do Marketplace para fins de auditoria. Novos agentes negociantes, ao entrarem no mercado, podem fazer uso das trilhas de auditoria para se situarem. Por exemplo, verificar negociações anteriores à sua entrada no

Marketplace e assim aumentar ou diminuir o preço de seu produto para se tornar mais competitivo.

3.3.8 Agente Negociante

Um agente negociante pode ser um comprador ou um vendedor. Estes agentes são instanciados pelas empresas negociantes (compradoras ou vendedoras) usuárias do ICS a partir de sua interface Web ou através de uma chamada remota aos métodos do serviço de Match, já que o *Matchmaker* é ele próprio um Web Service. Ao Instanciar um agente negociante é necessário inicialmente saber se ele deseja comprar ou se ele deseja vender um produto ou serviço. O ICS fornece um formulário de cadastro de propaganda caso seja um agente vendedor ou um formulário de cadastro de consulta caso seja um agente comprador.

3.3.9 Agente de Modelagem

O agente de modelagem tem por finalidade informar ao agente mediador das preferências de cada agente negociante para que o agente mediador possa interagir de maneira personalizada com cada um dos agentes negociantes.

Para inferir o perfil de cada agente negociante o agente de modelagem reúne informações provenientes da interface, do modelo de domínio (Repositório de Ontologias) e da Base de estereótipos.

Para melhor explicar o papel do agente de modelagem, segue dois exemplos de atividades desempenhadas por este agente:

i) É o agente de modelagem quem informa ao agente mediador se um determinado agente vendedor está ou não interessado em fazer parcerias com outros agentes vendedores para fornecer determinado produto ou serviço para um agente comprador.

ii) É o agente de modelagem responsável por informar ao agente mediador se, para um agente comprador, o critério mais relevante na definição de um fornecedor é o preço, a qualidade, a forma de pagamento, o prazo de entrega, ou qualquer outra característica do produto ou serviço ofertados pelos agentes vendedores.

3.3.10 Conclusão

Apresentamos neste capítulo os detalhes funcionais de cada componente do modelo ICS de negociação, os dados necessários e providos por eles além de suas estruturas internas as quais cada um necessita para execução de suas atribuições no processo de negociação.

O ICS diferencia-se das aplicações B2B convencionais principalmente por não ser limitado a um único domínio de negócio, podendo evoluir para um número ilimitado de domínios.

Outro importante diferencial é a utilização da tecnologia de agentes móveis que possibilita um processo de negociação mais eficiente, uma vez que os agentes com interesses complementares interagem localmente em um *host*.

A abordagem da Web Semântica para resolver o problema de descoberta dos possíveis parceiros de negócios também é outro diferencial, que faz com que a

descoberta seja muito mais eficiente e flexível, pois a busca não se restringe a aspectos sintáticos, mas também a aspectos semânticos e contextualizados.

A arquitetura baseada em um ciclo de vida bem definido possibilita a componentização da implementação do ICS, proporcionando maior qualidade do produto final.

No próximo capítulo é abordado o enriquecimento do processo de matching – composição de serviços utilizando técnicas e ferramentas de planejadores automáticos.

4 COMPOSIÇÃO DE SERVIÇOS NO ICS

Neste capítulo, apresentamos a justificativa do uso da composição de serviços no ICS baseado em planejadores automático. Fazemos uso de um planejador SHOP2 para implementar um algoritmo no ambiente Allegro Commom Lisp.

4.1 Introdução

O objetivo do processo de composição é aumentar o número de respostas positivas para os compradores, isto se dá devido à soma das capacidades de serviços simples (composição) capazes de atender apenas parcialmente a requisição de um comprador, porém em um ambiente colaborativo podem atender completamente aos requisitos do comprador.

Em nossa abordagem a composição de serviços acontece em duas fases. A primeira fase é a do matching, como proposto por (Tomaz, 2003), cuja abordagem ignora a possibilidade de composição. Sendo assim, caso uma requisição necessite de um serviço composto para ser atendido, o matchmaker retorna um falso negativo. Ou seja, a requisição pode ser atendida através da soma das capacidades dos serviços simples, entretanto, atualmente, o matchmaker não realiza essa tarefa de composição e retorna um conjunto de serviços simples que realizam apenas parcialmente a requisição solicitada.

A nossa contribuição consiste em acrescentar uma segunda fase ao processo de matching. A segunda fase é a geração do plano de composição, ou seja, a computação do fluxo de composição. É nesta fase que propomos o uso das técnicas

de planejamento automático para gerar o fluxo de interação entre os Web Services descobertos na primeira fase.

A vantagem da realização de composição no ambiente ICS é que a partir de um conjunto restrito de serviços publicados no repositório de propaganda, um vasto número de novos serviços pode se formar. Isto enriquece o ambiente e aumenta a possibilidade de satisfação dos requisitos feitos pelos compradores, o que aumenta também o número de negócios realizados no ICS.

4.2 Cenário de Motivação

Para ilustrar este processo, consideremos o problema de composição descrito em (Sirin et al., 2003), que consiste em compor as capacidades de um serviço de tradução de idiomas on-line e um serviço de dicionário. O serviço de tradução traduz textos entre vários pares de idiomas, porém o serviço de dicionário retorna o significado de palavras somente em inglês. Se um usuário precisar do serviço de dicionário em francês, nenhum destes serviços sozinho satisfaz essa necessidade. Mas juntos os serviços são capazes de atender à requisição: o texto em francês é passado como entrada para o serviço de tradução, que retorna como saída o texto em inglês. Esta saída é passada como entrada para o serviço de dicionário em inglês para obter sinônimos que então são traduzidos de volta para o francês.

A figura 10 ilustra o processo de composição, onde pode ser observado que para realizar a composição dos serviços é necessário que se estabeleça uma comunicação entre eles. Os serviços se comunicam utilizando o protocolo padrão para troca de mensagens entre Web Services (SOAP – Simple Object Application

Protocol). O fluxo das mensagens é definido utilizando técnicas de planejamento automático.

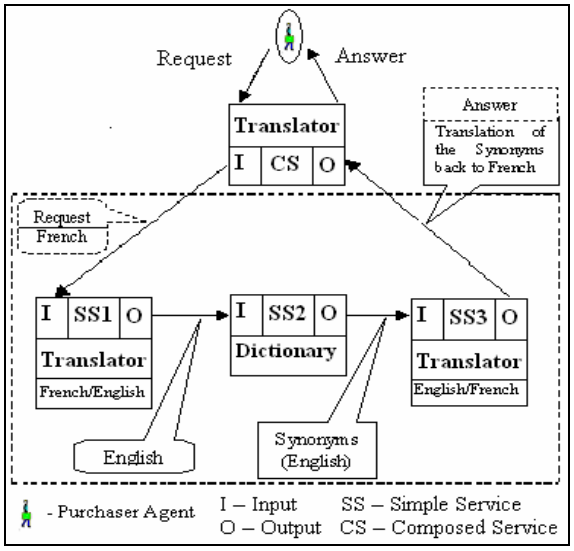

Figura 11: Plano de Composição (Baluz et al, 2004)

Com o objetivo de deixar mais claro o funcionamento da arquitetura proposta, apresentamos um exemplo prático de composição de serviços na segunda fase do matchmaking. Com este fim apresentamos a seguir uma descrição do objetivo do problema descrito na figura 11 e das ações necessárias para alcançá-lo usando a linguagem ADL (Action Description Language).

Iniciar(Em(Francês,texto))
Objetivo(Em(Francês, sinônimo))
Ação(Usar tradutor Francês/Inglês(texto))
Precond: Em(Francês,texto)
Efeito: Em(Inglês,texto)
Ação(usar dicionário inglês(texto,sinônimos))
Precond: Em(Inglês,texto)
Efeito: [Em(Inglês,[sinônimos])]
Ação(usar tradutor inglês/francês(sinônimo))

Precond: Em(Inglês,sinônimo)

Efeito: Em(Francês,sinônimo)

Figura 12: Problema da figura 11 descrito em ADL.

Baseado nesta descrição, um planejador segue a seguinte seqüência de ações: considerando o estado inicial, a única ação possível de ser executada é a primeira ação, que transforma o texto inicial em francês em um texto traduzido para o inglês. Com o novo texto em inglês, o planejador pode utilizar a segunda ou a terceira ação. A terceira ação (tradução do texto em inglês para o francês), entretanto, não pode ser utilizada, pois o efeito da primeira ação não corresponde as sua pré-condição, apenas levaria o texto de volta para o estado inicial. Aplicando a segunda ação, tem-se o texto transformado para um texto sinônimo. Com este sim, o planejador pode aplicar novamente a segunda ação ou a terceira. Pela utilização do operador [ ], o planejador fica impedido de utilizar novamente a segunda ação, o que somente transformaria o texto sinônimo em inglês em outro texto sinônimo em inglês. Aplicando a terceira ação, o planejador transforma o texto sinônimo em inglês em um texto sinônimo em francês, atingindo o estado objetivo.

4.3 SHOP2 (Simple Hierarchical Ordered Planner)

SHOP2 é um sistema de planejamento independente de domínio baseado na decomposição de tarefas ordenadas (ordered task decomposition), uma versão modificada do planejamento Rede Hierárquica de Tarefa (Hierarchical Task Network) que envolve o planejamento de tarefas na mesma ordem que eles são executados.

O SHOP2 foi avaliado entre os 4 melhores planejadores que competiram no AIPS (Artificial Intelligence Planning Systems) 2002.

As principais características de SHOP2 são:

- SHOP2 conhece o estado corrente do mundo a cada passo do processo de planejamento. Isto é um diferencial de SHOP2 em relação a maioria dos outros planejadores baseados em HTN, pois em SHOP2 as tarefas são executadas na mesma ordem em que são executadas, e torna possível o estado corrente no mundo no processo de planejamento, que por conseguinte torna possível para mecanismo de avaliação de pré-condições SHOP2 incorporar significante poder de inferência e raciocínio, incluindo a possibilidade para chamar programas externos. Isto torna SHOP2 ideal como base para planos de integração com fontes de informação externas, incluindo fontes de dados Web.

- Tem grande poder de expressividade. Por exemplo, nas pré-condições dos operadores e métodos é possível misturar computação simbólica/numérica e executar chamadas para programas externos.

- SHOP2 pode ser usado para criar algoritmos muito eficientes de domínio específico.

- SHOP2 incorpora muitas características para pddl, exemplo, suporte para quantificadores e efeitos condicionais em métodos e operadores.

- SHOP2 permite a combinação de tarefas parcialmente ordenadas e completamente ordenadas através do uso das palavras chave :*unordered* e *:ordered.*

- SHOP2 permite a otimização dos custos de planos. Para pequenos problemas a capacidade pode ser usada para encontrar o custo mínimo do plano. Para grandes

problemas esta capacidade pode ser usada com limitadores de tempo para fornecer menor custo de planejamento que é encontrado dentro de um dado tempo limite.

SHOP2 é escrito em Common Lisp. Para executar SHOP2 é necessário ter Common Lisp instalado. Em nossos experimentos utilizamos a implementação Allegro Common Lisp Versão 6.2.

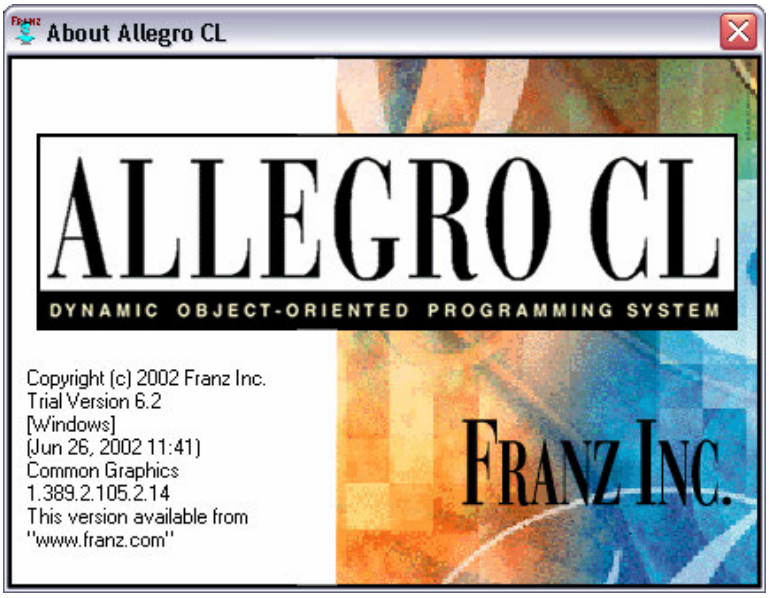

Figura 13: Ferramenta de reconhecimento do Common Lisp para execução do planejador SHOP2 – ALLEGRO COMMON LISP.

Um problema de planejamento em SHOP2 é uma tripla (*S, T, D*), onde *S* é o estado inicial, *T* é uma lista de tarefas, e *D* é uma descrição do domínio. Dada a tripla (*S, T, D*) como entrada, SHOP2 retornará um plano *P* = (*p1,p2,...,pn),* uma seqüência de operadores instanciados que executa *T* a partir de *S* em *D.*

Em shop2 a construção de um plano é baseada em operadores e métodos.

Definition 1 (Operador) - Um operador SHOP2 é uma expressão da forma *(h($\vec{v}$) Pré Del Add)* onde:

- *h($\vec{v}$)* representa uma tarefa primitiva com uma lista de parâmetros de entrada $\vec{v}$

- *Pré* representa as pré-condições dos operadores.

- *Del* representa a lista dos operadores negativos com a lista de coisas que se tornarão falsas após a execução do operador.

- *Add* representa a lista de operadores positivos com a lista de coisas que se tornarão verdadeiras após a execução do operador.

Definition 2 (método) – Um método SHOP2 é uma expressão da forma *(h($\vec{v}$) Pré T)* onde:

- *h($\vec{v}$)* representa uma tarefa composta com uma lista de parâmetros de entrada $\vec{v}$.

- *Pré* representa as pré-condições dos métodos.

- *T* representa uma lista parcialmente ordenadas de subtarefas que consiste na decomposição de *(h($\vec{v}$)*.

4.4 Algoritmo de tradução de DAML-S para SHOP2

A execução de um processo atômico é uma chamada para um programa para a Web com os parâmetros instanciados. A execução de um processo de composição consiste na coleção de processos atômicos específicos. Ao invés de executar diretamente o processo de composição com o programa dentro de um interpretador DAML-S, nós podemos tratar o processo de composição como uma especificação para como compor uma seqüência de execução de processos atômicos. A seguir

mostraremos como codificar o problema do processo de composição como um problema de planejamento em SHOP2, assim, SHOP2 pode ser usado com DAML-S para automaticamente gerar as chamadas para a composição de web services.

Apresentamos o algoritmo proposto por Dan Wu et. al. para traduzir uma coleção de modelos de processos DAML-S K em um domínio SHOP2 D. A abordagem proposta por (Dan Wu et. al., 2004), parte da seguinte hipótese:

Dada uma coleção de modelos de processos DAML-S *K = {k1, ...kn}*, assume-se que:

i) todos os processos atômicos definidos em K podem possuir efeitos(effects) ou saídas(outputs), mas nunca ambos. Um processo atômico com somente saída caracteriza um Web Service provedor de informação, enquanto um processo atômico com somente efeito caracteriza um Web Service que altera o ambiente. Se um Web Service possui as duas características assume-se que ele pode ser dividido.

ii) Não há processo de composição em K com as estruturas de controle Split e Split+Join, pois, atualmente SHOP2 não suporta concorrência. Não são considerados modelos de processos DAML-S que usem Split e Slit +Join.

iii) Os efeitos de todos os processos em K não são condicionais. SHOP2 não suporta efeito condicional.

As coleções de definições de processos DAML-S K em um domínio SHOP2 D são feitas como segue:

i) cada processo atômico com efeitos(effects) em K, é codificado como um operador SHOP2 que simula os efeitos de um Web Services que altera o ambiente.

ii) cada processo atômico com saída(outputs) em K, é codificada como um operador SHOP2 no qual a pré-condição é uma chamada para um Web Service provedor de informação.

iii) cada processo simples ou composto em K, é codificado como um ou mais métodos SHOP2.

Dadas as considerações acima, apresentamos os algoritmos propostos por (Dan Wu, 2004):

- para traduzir uma definição de DAML-S de um processo atômico apenas com efeitos em um operador SHOP2:

TRANSLATE-ATOMIC-PROCESS-EFFECT(Q)

Input: uma definição daml-s Q de um processo atômico A apenas com efeitos(effects).

Output: um operador SHOP2 *O*.

Procedimento:

1. $\vec{v}$: a lista de parâmetros de entrada definida para A em Q.
2. *Pré* = conjunto de todas as pré-condições de A, como definida em Q.
3. *Add* = uma coleção de todos os efeitos positivos de A, como definido em Q.
4. *Del* = uma coleção de todos os efeitos negativos em A, como definido em Q.

5. *Return O = (A($\vec{v}$) Pré Del Add)*

O algoritmo acima traduz cada definição DAML-S atômica em um operador SHOP2 que simulará os efeitos de um Web Service alterador de domínio através da mudança dos seus estados locais através de um operador.

- para traduzir uma definição DAML-S de um processo atômico que possui apenas saídas (outputs) em um operador SHOP2.

TRANSLATE-ATOMIC-PROCESS-OUTPUT(Q)

Entrada: uma definição DAML-S Q de um processo atômico A apenas com saídas (outputs)

Saída: um operador SHOP2 *O*

Procedimento:

1. $\vec{v}$: a lista de parâmetros definida para A como em Q.

2. *Pré* = uma conjunção de todas pré-condições de A, como definida em Q, acrescido de mais uma pré-condições da forma (assign y (call Monitor A $\vec{v}$)), onde Monitor é um procedimento que tratará as chamadas SHOP2 para web services.

3. *Add* = y

4. *Del* = Ø

5. *Return O* = (A($\vec{v}$) *Pré Del Add*)

O algoritmo acima traduz cada definição de processos DAML-S atômica em um operador SHOP2 que chamará os Web Services provedores de informação

em suas pré-condições. Deste modo, os Web Services provedores de informação é executado durante o processo de planejamento.

- para traduzir uma definição DAML-S de um processo simples em um método SHOP2.

TRANSLATE-SIMPLE-PROCESS(Q)

Entrada: uma definição DAML-S Q de um processo simples S

Saída: um método SHOP2 M

Procedimento:

1. $\vec{v}$ = lista de parâmetros definidos para S como em Q.
2. *Pré* = conjunto de todas as pré-condições de S como definida em Q
3. *T* = o processo atômico que realiza S ou o processo composto representado por S, como definido em Q.
4. Retorna *M=S($\vec{v}$) Pré T)*

- para traduzir uma definição DAML-S de um processo composto com estrutura de controle **Sequence** em um método SHOP2.

TRANSLATE-Sequence-PROCESS(Q)

Entrada: uma definiçao DAML-S Q de um processo composto C com estrutura de controle **Sequence**.

Saída: um método SHOP2 M.

Procedimento:

1: $\vec{v}$ = a lista de parâmetros de entrada definida para C como em Q.

2. *Pré* = uma conjunção de todas pré-condições de C como definida em Q.

3. *B* = Estrutura de Controle **Sequence** de C como definido em Q.

4.*(b1,...,bm)* = a seqüência de processos componentes de B como definida em Q.

5. *T* = lista de tarefas ordenada de *(b1,...,bm)*.

6. Retorna *M = (C($\vec{v}$) Pré T)*

- para traduzir uma definição DAML-S de um processo composto com estrutura de controle **if-then-else** em um método SHOP2.

TRANSLATE-If-then-else-PROCESS(Q)

Entrada: uma definição DAML-S Q de um processo composto C com estrutura de controle **If-then-else**.

Saída: um método SHOP2 M.

Procedimento:

1: $\vec{v}$ = a lista de parâmetros de entrada definida para C como em Q.

2. π_{if} = condições para o ***if*** como definidos em Q.

3. Pré 1 = conjunção de todas as pré-condições em C como definidas em Q e π_{if} .

4. Pré 2 = conjunção de todas as pré-condições em C como definidas em Q.

5. b1 = é o processo para cláusula Then como definida em Q.

6. b2 = é o processo para cláusula Else como definida em Q.

7. Retorna *M = (C($\vec{v}$) Pre1 b1 Pre2 b2)*

- para traduzir uma definição DAML-S de um processo composto com estrutura de controle **Repeat-While** em métodos SHOP2.

TRANSLATE-Repeat-While-PROCESS(Q)

Entrada: uma definição DAML-S Q de um processo composto C com estrutura de controle **Repeat-While**.

Saída: uma coleção de métodos SHOP2 M.

Procedimento:

1: $\vec{v}$ = a lista de parâmetros de entrada definida para C como em Q.

2. π_{while} = condições para o ***While*** como definidos em Q.

3. Pré = conjunção de todas as pré-condições de C como definidas em Q.

4. b1 = é o processo para ***Repeat*** como definida em Q.

5. M1 = (C($\vec{v}$) pre C1($\vec{v}$)

6. M2 = (C1($\vec{v}$) π_{while} b1 {} {})

7. Retorna *M = {M1,M2}*

- para traduzir uma definição DAML-S de um processo composto com estrutura de controle **Repeat-Until** em métodos SHOP2.

TRANSLATE-Repeat-Until-PROCESS(Q)

Entrada: uma definição DAML-S Q de um processo composto C com estrutura de controle **Repeat-Until**.

Saída: uma coleção de métodos SHOP2 M.

Procedimento:

1: $\vec{v}$ = a lista de parâmetros de entrada definida para C como em Q.

2. π_{Until} = condições para o **Until** e como definidos em Q.

3. *Pré* = conjunção de todas as pré-condições de C como definidas em Q.

4. b1 = é o processo para **Repeat** como definida em Q.

5. M1 = (C($\vec{v}$) pre C1($\vec{v}$)

6. M2 = (C1($\vec{v}$) (not (π_{Until})) b1 Ø Ø)

7. Retorna M = {M1,M2}

- para traduzir uma definição DAML-S de um processo composto com estrutura de controle **Choice** em métodos SHOP2.

TRANSLATE-Choice-PROCESS(Q)

Entrada: uma definição DAML-S Q de um processo composto C com estrutura de controle **Choice**.

Saída: uma coleção de métodos SHOP2 M.

Procedimento:

1: $\vec{v}$ = a lista de parâmetros de entrada definida para C como em Q.

2. *Pré* = conjunção de todas as pré-condições de C como definidas em Q.

3. *B* = estrutura de controle **Choice** de C como definida em Q.

4. *(b1,…, bm)* = conjunto de processos componentes de B como definidos em Q.

5. For I = 1,..,m *Mi = (C($\vec{v}$) Pré b1)*

6. Retorna *M = {M1,...,Mm}*

- para traduzir uma definição DAML-S de um processo composto com estrutura de controle **Unordered** em um método SHOP2.

TRANSLATE-Unordered-PROCESS(Q)

Entrada: uma definição DAML-S Q de um processo composto C com estrutura de controle **Unordered**.

Saída: um método SHOP2 M.

Procedimento:

1: $\vec{v}$ = a lista de parâmetros de entrada definida para C como em Q.

2. *Pré* = uma conjunção de todas pré-condições de C como definida em Q.

3. *B* = Estrutura de Controle **Unordered** de C como definido em Q.

4. *(b1,...,bm)* = conjunto de processos componentes de B como definida em Q.

5. *T* = lista de tarefas não ordenada de *(b1,...,bm)*.

6. Retorna *M = (C($\vec{v}$) Pré T)*

E finalmente para traduzir uma coleção de modelos processo DAML-S em um domínio SHOP2

TRANSLATE-PROCESS-MODEL(K)

Entrada: uma coleção de modelos de processo DAML-S K

Saída: um método SHOP2 D.

Procedimento:

1: D = Ø

2. Para cada definição de processo atômico Q em K

Se o processo atômico tem apenas outputs.

O = TRANSLATE-ATOMIC-PROCESS-OUTPUT(Q)

Se o processo atômico tem apenas efeito

O = TRANSLATE-ATOMIC-PROCESS-EFFECT(Q)

Adicione *O* a *D*.

3. Para cada definição de processo simples Q em K

M = TRANSLATE-SIMPLE-PROCESS(Q)

Adicione M a D

4. Para cada definição de processo composto Q em K

Se o processo tem uma estrutura de controle **Sequence**

M = TRANSLATE-Sequence-PROCESS(Q)

Se o processo tem uma estrutura de controle **If-Then-Else**

M = TRANSLATE-If-Then-Else-PROCESS(Q)

Se o processo tem uma estrutura de controle **Choice**

M = TRANSLATE-Choice-PROCESS(Q)

Se o processo tem uma estrutura de controle **Repeat-While**

M = TRANSLATE-Repeat-While-PROCESS(Q)

Se o processo tem uma estrutura de controle **Repeat-Until**

M = TRANSLATE-Repeat-Until-PROCESS(Q)

Se o processo tem uma estrutura de controle **Unordered**

M = TRANSLATE-Unordered-PROCESS(Q)

Adicione M a D

5. Retorne D

Para justificar o é apresentado a seguir a execução de experimento em SHOP2 utilizando o ambiente Allegro Commom Lisp.

O Allegro Commom Lisp é instalado após o recebimento da licença conseguida no próprio site do SHOP2 para certificar o software. O SHOP2 precisa ser carregado primeiro, gerando uma extensão *.fasl* para executar qualquer exemplo. O exemplo é aberto sendo compilado e carregado logo após, gerando assim os planos de composição do domínio do problema.

Experimento(1): Um domínio de empréstimo. O problema é conseguir um empréstimo gerando planos para conseguir este empréstimo. Os Métodos para conseguir o dinheiro só pode ser usando cartão de crédito ou hipoteca da casa.

```
(defdomain loan (
  (:operator (!borrow-mortgage ?ammt ?acct)
         () () ((debt ?ammt ?acct)) 15.0)
  (:operator (!borrow-credit-card ?ammt ?acct)
         () () ((debt ?ammt ?acct)) 0.0)
  (:operator (!procrastinate)
         () () () 0.0)
  (:operator (!pay-mortgage-interest ?acct)
         ((debt ?ammt ?acct)) () () (* .1 ?ammt))
```

```
  (:operator (!pay-credit-card-interest ?acct)
         ((debt ?ammt ?acct)) () () (* .2 ?ammt))
credit-card
  (:method (get-money ?ammt ?acct)
       ((have-credit-card ?acct))
       (:ordered (!borrow-credit-card ?ammt ?acct)
            (!pay-credit-card-interest ?acct)))
  (:method (get-money ?ammt ?acct)
       ((have-house ?acct))
       (:ordered (!borrow-mortgage ?ammt ?acct)
            (!pay-mortgage-interest ?acct)))
  (:method (get-money ?ammt ?acct)
       ((have-house ?acct))
       (:ordered (!procrastinate)
            (!borrow-mortgage ?ammt ?acct)
            (!pay-mortgage-interest ?acct)))))
(defproblem p-150-both loan
  ((have-credit-card 1001) (have-house 1001)) ((get-money 150 1001)))
 (find-plans 'p-150-both :verbose :long-plans :which :all :optimize-cost t)
```

Experimento (2): O problema do tradutor de sinônimos apresentado na figura 8 e explicado logo em seguida. Colocamos o problemas na sintaxe PDDL para ser executado no planejador SHOP2 no ambiente do Allegro Commom Lisp.

Código do Algoritmo:

```
(defdomain traducao
      (:operator (!tradutor-fr-en ?fenetre) () () ((Emingles window)))
      (:operator (!tradutor-en-en ?window) () () ((Emingles window2)))
```

```
      (:operator (!tradutor-en-fr ?window2) () () ((Emfrances fenetre2)))

(:method (traduzir-fr-en ?fenetre)

      ((Emfrances fenetre)) ((!tradutor-fr-en ?fenetre)))

(:method (traduzir-en-fr ?window2)

      ((Emingles window2)) ((!tradutor-en-fr ?window2)) )

(:method (usar-dic ?window)

      ((Emingles window)) () ((!dicionario ?window)))))

(defproblem traduz traducao

      ((Emfrances fenetre)) ((traduzir-fr-fr fenetre fenetre2)))

      (find-plans 'traduz :verbose :long-plans :which :all)
```

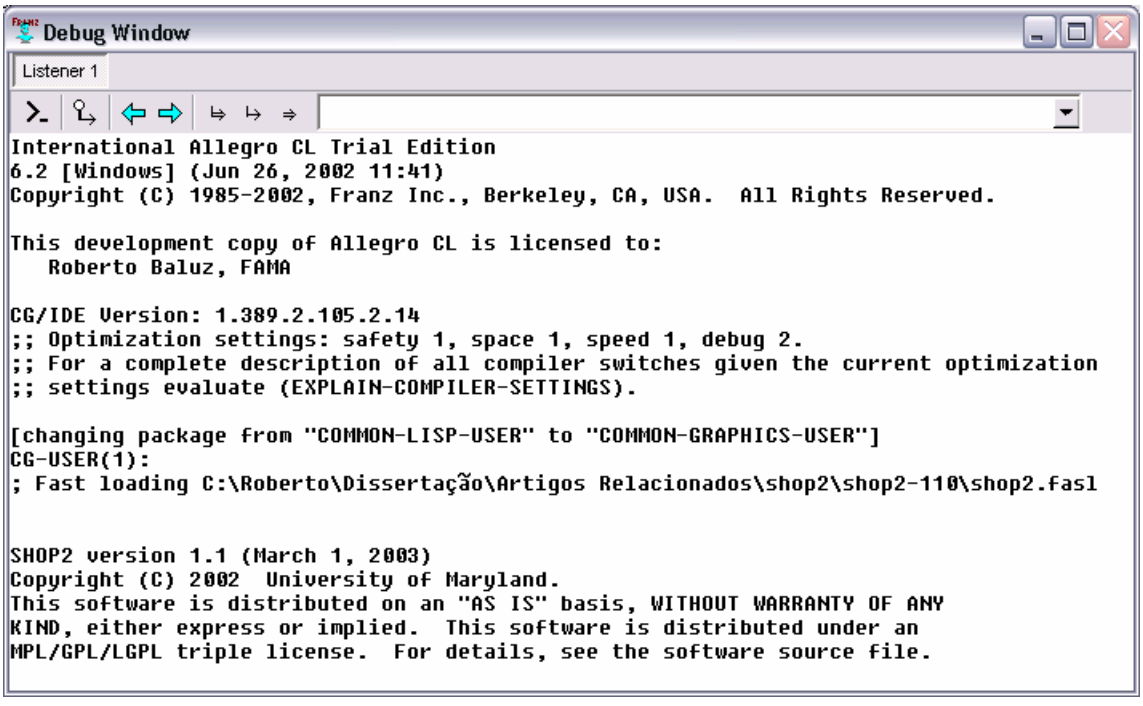

Figura 14: Janela do Allegro Commom Lisp enquanto o SHOP2 está no ar

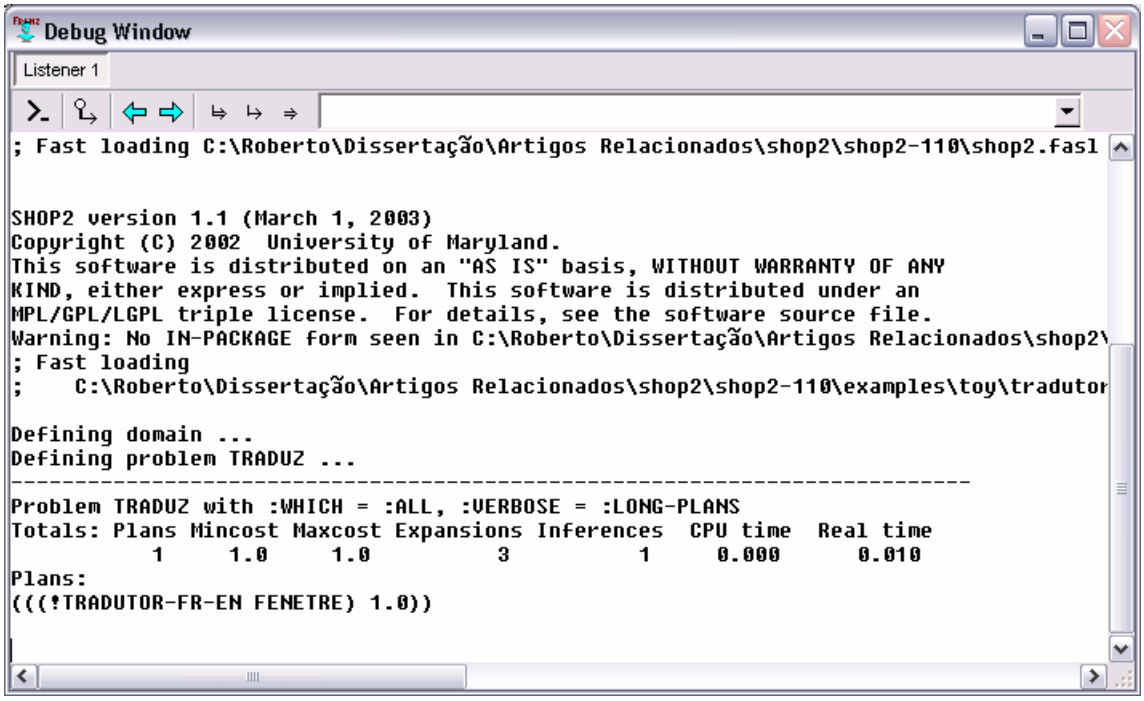

Figura 15: Resposta do Tradutor, uma palavra sinônima em francês com 1 plano.

CONCLUSÃO

Nesta dissertação, propusemos o uso das técnicas de planejamento automático para permitir a composição de serviços Web no ambiente ICS com a finalidade de aumentar o número de possibilidades de negociação entre possíveis parceiros de negócios a partir de um incremento de funcionalidade do agente Matchmaker.

Propomos a inclusão de uma etapa adicional ao processo de Matching atualmente definidos no ICS. Esta etapa adicional faz uso das técnicas de planejamento automático para permitir a tradução das descrições DAML-S dos serviços Web para um domínio SHOP2. Uma vez definido o plano, o problema é submetido ao planejador que gera automaticamente a seqüência de interações entre serviços.

Nossa abordagem é bastante inovadora se comparada ao estado corrente do processo de composição de serviços, pois não requer nenhum tipo de intervenção humana.

Este trabalho nos proporcionou uma experimentação das teorias de importantes áreas de pesquisa: Planejamento Automático e Web Services Semânticos, e foi objeto de uma publicação internacional:

> C. Roberto Baluz, S. Labidi, R. F. Tomaz, B. Wanghon and Nathália R. S. Oliveira. ***Composition of Web Services in the ICS Architecture*****.** 6th International Conference on Enterprise Information Systems. Universidade Portucalense, Porto – Portugal, 14-17, April 2004.

Como proposta para trabalhos futuros a curto e médio prazo, pode-se enumerar:

Avaliação de outras ferramentas de planejamento automático como alternativa para composição de serviços no ICS;

Desenvolver um protótipo para validar as idéias aqui apresentadas implementando chamadas a Serviços Web reais;

Efetuar testes em larga escala deste protótipo;

Aperfeiçoar o algoritmo de tradução de DAML-S para SHOP2 de modo a permitir a tradução das estruturas de controle SPLIT e SPLIT+JOIN que suportam o processamento concorrente;

Integrar funcionalidades ao código do agente Matchmaker desenvolvido em (Tomaz, 2003). Os resultados obtidos nos experimentos realizados demonstram viabilidade do desenvolvimento de um protótipo para validar as idéias aqui apresentadas e nos encorajam a aprofundar nossos estudos nessa desafiadora linha de pesquisa.

REFERÊNCIA BIBLIOGRÁFICA

(Abitebul et al., 2000) Abitebul, S.; Buneman, P.; Suciu, D. Data on the Web: from Relations to Semistructured Data and XML. San Francisco: Morgan Kaufmann. 2000.

(Arruda, 2002) Arruda, Ladjane.MEDIWEB: Um Integrador Semântico de Dados na Web Baseado em Mediador. Dissertação de Mestrado. UFPB, Campina Grande. 2002.

(Anupriya, 2002) Anupriya A. et al. DAML-S: Semantic Markup for Web Services. DAML services Coalition. 2002.

(Baluz et al., 2004) C. Roberto Baluz, S. Labidi, R. F. Tomaz, B. Wanghon and Nathália R. S. Oliveira. Composition of Web Services in the ICS Architecture. ICEIS 2004.

(Bechhofer et al., 2003) Bechhofer, S. et al. OWL Web Ontology Language Reference. Draft Report, 2003. Acessado em Agosto de 2003. Disponível na Internet por www em: http://www.w3.org/TR/owl-ref.

(Berners – Lee et al., 2000) BERNERS – LEE, Tim et al. Semantic Web Development Proposal. Acessado em Janeiro de 2002. Disponível na Internet por www em: http://www.w3c.org/2001/sw/. 2000.

(Bonifácio, 2002) Bonifácio, A. Ontologias e Consultas Semânticas: Uma Aplicação ao Caso Lattes. Dissertação de Mestrado. UFRGS, Porto Alegre. 2002.

(Burbeck, 2000) Burbeck, S. The Tao of e-vusiness services – The evolution of Web applications into service-oriented components with Web Services. IBM Software Group. 2000.

(Daum, 2002) Daum, Bertold. Arquitetura de Sistemas com XML: conteúdo, processo e apresentação. Rio de Janeiro. Ed. Campus. 2002.

(Dan Wu et. al., 2003) Dan Wu, Evren Sirin, James Hendler, Dana Nau, Bijan Parsia. Automatic Web Services Composition Using SHOP. The Twelfth Internacional World Wide Web Conference 20-24 May 2003, Budapest, HUNGARY.

(Erdmann apud Arruda, 2000) Erdmann, M.; Decker, S. Ontology-aware XML-Queries. Submission for WebDB. 2000.

(Freitas et al., 2002) Freitas, F.; Bittencourt, G. Comunicação entre Agentes em Ambientes Distribuídos Abertos: o Modelo “peer-to-peer”.Revista Eletrônica de Iniciação Científica. Ano II. Vol II. Número II. Junho de 2002. ISSN 1519-8219. 2002

(Gruber, 1993) Gruber, T. Towards principles for the design of ontologies used for knowledgesharing. Starford University, Knowledge Systems Laboratory Technical Report KSL93-04. 1993.

(Guarino, 1998) Guarino, N. Formal Ontology and Information Systems. in: N. Guarino, (Ed.) Formal Ontology in Information Systems. pp. 3-15, IOS Press, Amsterdam, Netherlands. 1998.

(Horrocks et al., 2000) Horrocks, I. et al. OIL in a Nutshell, Proc. ECAI' 00 Workshop on Application of Ontologies and PSMs, Berlim, Germany, 2000, pp. 4.4-4.12. 2000.

(Horrocks, 2001) Horrocks, I; Harmelen, F. V. Reference Description of the DAML+OIL Ontology Markup Language. Draft Report, 2001. Acessado em Janeiro de 2002. Disponível na Internet por www em: http://www.daml.org/2000/12/reference.html. 2001.

(Horrocks, 2002) Horrocks, I. Tessaris, Sergio. Querying the Semantic Web: a Formal Approch. Presented at 1ª. International Semantic Web Conferece (ISWC-2002). 2002.

(Jennings et al., 1996) Jennings, N. R., Faratin, P., Johnson, M. J., O'Brien, P. O., and Wiegand, M. E. 1996. Using intelligent agents to manage business processes. In First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM-96) (April 1996), pp. 345–360. 1996.

(Jennings et al., 2000) Jennings, N., and Wooldridge, M. Agent-oriented software engineering. In Handbook of Agent Technology (ed. J. Bradshaw) AAAI/MIT Press. 2000.

(Labidi et al., 2003) Sofiane Labidi, Luis C. Fonseca, Othon B. Filho and Edson Nascimento, Intelligent B2B Commerce System. In the textbook Techno-Legal Aspects of Information Society and New Economy: an Overview, Formatex. Spain. 2003.

(Lecheta, 2004) Edson Martins Lecheta. Algoritmos Genéticos para Planejamento em Inteligência Artificial. Dissertação de Mestrado, Universidade Federal do Paraná. Curitiba PR, 2004.

(Martin et al., 2003) David Martin Et Al. OWL-S 1.0 Release, W3C Web Ontology Workgroup, http://www.daml.org/services/owl-s/1.0/

(Minsky, 1975) Minsky, M. Frame System Theory. In P.N. Johnson-Laird and P.C. Wason (eds.), Thinking: Readings in Cognitive Science. Cambridge: Cambridge University Press. 355-376. 1975.

(Morales, 1999) Morales, Eduardo. Representación de Conocimiento. Acessado em Março de 2002. Disponível na Internet por www em: http://www.mor.itesm.mx/~rdec/principal.html.

(Nardi et al., 2002) D. Nardi, R. J. Brachman. An Introduction to Description Logics. In the Description Logic Handbook, edited by F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, P.F. Patel-Schneider, Cambridge University Press, 2002, pages 5-44. 2002.

(Pitts et al., 2000) Pitts -Moultis, N. XML BLACK BOOK, São Paulo: MAKRON BOOKS. 2000.

(Rich, 1993) Rich, E.; Knight, K. Inyeligência Artificial. Segunda Edição. Ed. Makon Books, São Paulo.1993.

(Silva, 2003) Silva, Felipe Vieira. Planejamento automático aplicado a problemas dependentes de recursos**.** Dissertação de Mestrado (Mestrado em Ciência da Computação), Universidade Federal do Ceará. 2003.

(Sirin et al., 2003) Sirin, E., Hendler, J. and Parsia, B. Interactive Composition of Semantic Web Services. University of Maryland, USA. 2003.

(Sparck, 1997) Sparck-Jones, Karen; Willet, Peter. Readings in Information Retrieval. San Francisco. Morgan Kaufmann. 1997.

(Stanley, 1999) L., Stanley. Descoberta de Conhecimento em Textos. Exame de Qualificação CPGCC-UFRGS, Porto Alegre, 1999.

(Studer, 1998) Studer, Rudi et al. Knowledge Engineering: principles and methods. Data & Knowledge Engineering, v.25, n.1/2, Março de 1998.

(Stuart, et. al, 1995) Stuart Russell, Peter Norvig. Inteligência Artificial. Editora Prentice Hall, 1995.

(Sycara, 2002) Katia Sycara, Seth Widoff, Matthias Klusch and Jianguo Lu, "LARKS: Dynamic Matchmaking Among Heterogeneous Software Agents in Cyberspace." Autonomous Agents and Multi-Agent Systems, 5, 173–203, 2002.

(Tomaz, 2003) Ricardo Ferraz Tomaz. Semantic Matching dos Agentes Negociantes no Ambiente ICS de Comércio Eletrônico. Dissertação de Mestrado (Mestrado em Ciência da Computação), Universidade Federal do Maranhão. 2003.

URLS

(URL 1) Página do W3C "*World Wide Web Consotioum*",. 2003. URL: http://www.w3.org/

(URL 2) Página Rule Markup Language RuleML. http://www.dfki.uni-kl.de/ruleml/

(URL 3) Página da Sociedade Brasilerira de Gestão do Conhecimento. http://www.sbgc.org.br/cgi/cgilua.exe/sys/start.htm?tpl=home.

(URL 4) Foudation For Intelligent Physical Agents FIPA Personal AssistantSpecification. http://www.fipa.org/specs/fipa00083/XC00083B.html.