**Juan Carlos Zuñiga Torres**

# Composição de Serviços em Ambientes Pervasivos: Um Modelo de Referência

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Doutor em Ciências.

São Paulo
2013

**Juan Carlos Zuñiga Torres**

# Composição de Serviços em Ambientes Pervasivos: Um Modelo de Referência

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Doutor em Ciências.

Sistemas Eletrônicos

Orientador:
Prof. Dr. Silvio Ernesto Barbin

São Paulo
2013

**Ficha Catalográfica**

Zuñiga Torres, Juan Carlos
Composição de Serviços em Ambientes Pervasivos: Um Modelo de Referência / J.C. Zuñiga Torres. - São Paulo, 2013.
112 p.

Tese (Doutorado) – Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Telecomunicações e Controle (PTC).

1. WEB semântica 2. Arquitetura orientada a serviços 3. Interação homem-máquina 4. Edifícios inteligentes 5. WEB 2.0 6. Geração de planos em inteligência artificial I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Telecomunicações e Controle (PTC). II. t.

# Resumo

Ambientes Pervasivos são ambientes povoados por diversos dispositivos (sensores, atuadores, etc.) e aplicações de software (locais ou distribuídas) incorporadas nestes ambientes físicos de forma transparente para o usuário. Ambientes deste tipo devem ser capazes de interagir e satisfazer as requisições do usuário de forma autônoma e transparente. Nesse sentido, um dos maiores desafios de pesquisa em ambientes pervasivos é a de estabelecer mecanismos automáticos para compor, de forma dinâmica, funcionalidades que satisfaçam as requisições dos usuários.

Nesse sentido, nós partimos da hipótese que mecanismos automáticos de interação entre ambientes e usuários podem ser abordados como um problema de composição automática de serviços em ambientes pervasivos. Portanto, nossa proposta é desenvolver um modelo referência, a partir do qual podem ser implementados sistemas que permitam ao ambiente pervasivo interagir com o usuário de forma natural, automática e dinâmica. Desta forma, o sistema de composição de serviços gerência e automatiza o processo de resolução de requisições feitas pelo usuário (de forma implícita e/ou explicita) através das funcionalidades (serviços) disponíveis no ambiente ou através de novas funcionalidades criadas pelo processo de composição. Para tal fim, este trabalho apresenta um "Modelo de Referência" que permita projetar, implementar e avaliar sistemas de composição de serviços que gerenciem e automatizem o processo de interação em diversos tipos de ambientes pervasivos.

Nossa proposta traz vantagens como: o baixo acoplamento e a interoperabilidade, isto porque é possível selecionar, integrar e reutilizar de forma eficiente e efetiva serviços heterogêneos provenientes de diversos tipos de dispositivos e/ou aplicações. Além disso, o modelo ontológico WSMO (*Web Services Modelling Ontology*) nós permite descrever semanticamente as capacidades dos serviços como também as informações contextuais presentes no ambiente, o que torna nosso sistema mais perto de um ambiente pervasivo real como o idealizado por Mark Weiser.

**Palavras Chave:** Composição de Serviços, Ambientes Pervasivos, Web Semântica, Serviços Web Semânticos, Planejamento, Composição Automática de Serviços Web, Informações Contextuais, WSMO.

# Abstract

Pervasive environments are populated by several devices (sensors, actuators, etc.) and software applications (local or distributed) incorporated these physical environments transparently to the user. Environments of this type should be able to interact and process user requests autonomously and transparently. In this sense, one of the biggest research challenges in pervasive environments is to establish automatic mechanisms to compose dynamically, features that meet the user requirements.

In this sense, we set the hypothesis that automatic mechanisms of interaction between users and environments can be addressed as a problem of automatic composition of services in pervasive environments. Therefore, our proposal is to develop a reference model, from which systems can be implemented to enable the pervasive environment interact with the user in a natural, automatic and dynamic. Thus, the system service composition management and automates the process of resolving requests made by the user (implicitly and / or explicitly) through the functionality (services) available in the environment or through new features created by the process of composition. To this end, this paper presents a "Reference Model" that allows to design, implement and evaluate systems of composition of services that manage and automate the interaction process in different types of pervasive environments.

Our proposal brings benefits such as loose coupling and interoperability, because it is possible to select, integrate and reuse in an efficient and effective heterogeneous services from different types of devices and / or applications. Furthermore, the ontological model WSMO (Web Services Modeling Ontology) allows us to semantically describe the capabilities of the services as well as contextual information in the environment, which makes our system closer to a real pervasive environment as conceived by Mark Weiser.

**Keywords:** Service Composition, Pervasive Environments, Semantic Web, Semantic Web Services, Automated Planning Service Composition, Context-Aware, WSMO.

# Lista de Abreviaturas

| | |
|---|---|
| **AAL** | Ambient Assisted Living |
| **AmI** | Ambient Intelligence |
| **API** | Application Programming Interface |
| **CPS** | Cyber-Physical System |
| **GPS** | Global Positioning System |
| **HTN** | Hierarchical Task Network |
| **HTTP** | Hypertext Transfer Protocol |
| **IA** | Inteligência Artificial |
| **IDE** | Integrated Development Environment |
| **IEEE** | Institute of Electrical and Electronics Engineers |
| **IoT** | Internet of Things |
| **JSHOP2** | Java Implementation of SHOP2 |
| **OWL** | Web Ontology Language |
| **OWL-S** | Web Ontology Language for Services |
| **PDDL** | Planning Domain Definition Language |
| **RFID** | Radio-Frequency IDentification |
| **RSSF** | Redes de Sensores Sem Fio |
| **SHOP2** | Simple Hierarchical Ordered Planner |
| **SOA** | Service Oriented Architecture |
| **SOAP** | Simple Object Access Protocol |
| **SOC** | Service Oriented Computing |
| **URI** | Uniform Resource Identifier |
| **W3C** | The World Wide Web Consortium |
| **WDSL** | Web Services Description Language |
| **WLAN** | Wireless Local Area Network |
| **WSDL-S** | Web Service Description Language for Services |
| **WSMO** | Web Services Modelling Ontology |
| **WSML** | Web Service Modeling Language |
| **WSMX** | Web Service Execution Environment |
| **XML** | eXtensible Markup Language |

# Lista de Figuras


2.1 Taxonomia das características da Computação Pervasiva (adaptado de [1]) . . . 14

2.2 *The Gator Tech Smart House* (adaptado de [2]) . . . . . 16

2.3 Arquitetura do projeto INHOME - [3] . . . . . 17

2.4 Sete dimensões contextuais - [4] . . . . . 18

2.5 A infra-estrutura em camadas da Web Semântica - [5] . . . . . 21

2.6 Principais conceitos do WSMO - [6] . . . . . 24

2.7 Orquestração e Coreografia - [7] . . . . . 32

2.8 Sistema automático de composição de serviços - [8] . . . . . 33

2.9 Arquitetura SPOC (*Semantic based Planning for Optimal web services Composition*) - [9] . . . . . 33

2.10 Arquitetura de Composição Automática de Serviços Web - [10] . . . . . 34

2.11 Grounding WSMO - Adaptado de [11] . . . . . 37

2.12 Interface do OWLSXPlan - [12] . . . . . 38

2.13 Interface do WEBRPLAN - [13] . . . . . 39

2.14 Interface do Transplan - [14] . . . . . 40

2.15 Interface do *Automated Web Service Composer* - [15] . . . . . 41

3.1 Arquitetura do Sistema de Composição GoalMorph - [16] . . . . . 54

3.2 Sistema de Composição SeGSeC-CoSMoS-CoRE - [17] . . . . . 55

3.3 Sistema de Composição iCAS - [18] . . . . . 56

3.4 Níveis de interação em MEDUSA - [19] . . . . . 57

3.5 Arquitetura de MEDUSA - [19] . . . . . 57

3.6 Modelo Referência para Serviços (RSM): Relações entre conceitos - [20] . . . 59

3.7 Revisão da Especificação WSMO - [21] . . . . . 59

3.8 *PSC-RM: Reference Model for Pervasive Service Composition* - [22] . . . . . 60

3.9 Cafeteira conceito da Qualcomm . . . . . 62

4.1 Abstração Arquitetural do Modelo de Referência Proposto . . . . . 66

4.2 Abstração Arquitetural observada em Níveis . . . . . 67

4.3 SECOM - *Semantic Context Model* - [23] . . . . . 68

4.4 OMC - Ontologias para Modelagem de Contexto - [24] . . . . . 69

5.1 Arquitetura derivada para um sistema de composição para comercio eletrônico . 73

5.2 Arquitetura para o Sistema de Composição para Sala de Aula . . . . . 76

5.3 Esquema que Descreve a Modelagem de um Domínio de Aplicação . . . . . 77

5.4 Especificação dos Serviços do Domínio *Bravo Air* . . . . . 78

5.5 Arquitetura para o sistema de composição de serviços para ambiente domótico . 79

5.6 Ontologia Bravo Air - Definição de Conceitos . . . . . 80

5.7 Ontologia Bravo Air - Definição de Instâncias . . . . . 81

5.8 Serviço Web Semântico: *Get Desired Flight Details* . . . . . 82

5.9 Definição dos Predicados no Arquivo PDDL Traduzido . . . . . 83

5.10 Serviço Web *Get Desired Flight Details* Anotado Semanticametne (SAWSDL) 84

5.11 Interface Gráfica de Gerenciamento . . . . . 86

5.12 Interface Gráfica de Requisições de Metas . . . . . 87

5.13 Interface Gráfica de Resultados . . . . . 88

5.14 Sequência Funcional do Módulo de Interface do Usuário Final . . . . . 89

5.15 Fluxo de processamento para carregar uma ontologia no repositório . . . . . 90

5.16 Modelo entidade-relacionamento do SCSAD . . . . . 91

5.17 Mecanismo de tradução do módulo tradutor . . . . . 92

5.18 Plano gerado para o SCSAD pelo JSHOP2 . . . . . 93

5.19 Workflow de execução da aplicação-exemplo . . . . . 95

# Lista de Tabelas

3.1 Dimensão de Especificação Geral . . . . . 47

3.2 Dimensão de Especificação do Usuário . . . . . 48

3.3 Dimensão de Especificação do Ambiente . . . . . 49

3.4 Dimensão de Especificação dos Dispositivos . . . . . 50

3.5 Dimensão de Composição . . . . . 51

3.6 Dimensão de Especificação do Usuário . . . . . 52

# Sumário

**1 Introdução** **1**

1.1 Contextualização . . . . . 1

1.2 Motivação . . . . . 5

1.3 Descrição do Problema . . . . . 5

1.4 Solução Proposta . . . . . 6

1.5 Objetivos . . . . . 7

1.6 Justificativas . . . . . 9

1.7 Metodologia . . . . . 10

1.8 Organização da Tese . . . . . 11

**2 Referencial Teórico** **13**

2.1 Computação Pervasiva . . . . . 13

2.2 Ambientes Pervasivos . . . . . 15

2.3 Computação Ciente de Contexto . . . . . 17

2.4 Serviços Web . . . . . 19

2.5 Web Semântica . . . . . 20

2.6 Serviços Web Semânticos . . . . . 22

2.6.1 WSMO . . . . . 22

2.7 Mecanismos de Inferência . . . . . 26

2.7.1 Planejamento . . . . . 27

2.7.2 Planejamento Hierárquico . . . . . 27

2.8 Composição de Serviços Web . . . . . 28

2.8.1 Composição de Serviços Web: Classificação . . . . . 29

2.8.2 Composição de Serviços Web: Arquiteturas . . . . . 32

2.8.3 Composição de Serviços Web: *Grounding* . . . . . 36

2.8.4 Sistemas de Composição de Serviços Web . . . . . 38

**3 Estado da Arte e Trabalhos Relacionados 42**

3.1 Estado da Arte dos Sistemas de Composição de Serviços para Ambientes Pervasivos . . . . . 42

3.2 Trabalhos Relacionados . . . . . 58

3.3 Além do Estado da Arte . . . . . 61

**4 Modelo de Referência 63**

4.1 Definição do Escopo . . . . . 63

4.2 Fundamentação . . . . . 64

4.3 Cenário Geral de Aplicação . . . . . 65

4.3.1 Abstração Arquitetural . . . . . 65

4.3.2 Modelagem Contextual Semântica . . . . . 68

**5 Aplicação do Modelo de Referência 71**

5.1 Sistema de Composição para Comercio Eletrônico . . . . . 71

5.1.1 Contextualização . . . . . 71

5.1.2 Especificação Arquitetural . . . . . 72

5.2 Sistema de Composição para Sala de Aula . . . . . 74

5.2.1 Contextualização . . . . . 74

5.2.2 Especificação Arquitetural . . . . . 75

5.3 Sistema de Composição para Ambiente Domótico (SCSAD) . . . . . 76

5.3.1 Contextualização . . . . . 76

5.3.2 Especificação Arquitetural . . . . . 78

5.3.3 Implementação da Aplicação-Exemplo . . . . . 79

**6 Considerações Finais** **96**

6.1 Contribuições . . . . . 96

6.2 Limitações do Trabalho . . . . . 98

6.3 Conclusões . . . . . 98

6.4 Trabalhos Futuros . . . . . 100

**Referências Bibliográficas** **102**

# 1 Introdução

## 1.1 Contextualização

Define-se um "Ambiente Pervasivo" como um ambiente físico povoado por diversos dispositivos (sensores, atuadores, etc.) e aplicações de software (locais ou distribuídas) incorporados no ambiente de forma que interajam natural e transparentemente com o usuário [1]. Neste tipo de ambientes, qualquer dispositivo e/ou aplicação é um potencial cliente ou provedor de funcionalidades. Além disso, dispositivos e aplicações devem interconectar-se uns aos outros com o objetivo de prover funcionalidades capazes de auxiliar nas atividades do usuário [25].

Porém, nem sempre os ambientes pervasivos são capazes de interagir com o usuário e nem satisfazer suas necessidades de forma automática e transparente devido à heterogeneidade dos dispositivos, à complexidade das aplicações, e ao dinamismo (mudança constante) das necessidades dos usuários e das características do ambiente.

Desta forma, um dos maiores desafios de pesquisa em ambientes pervasivos é a de desenvolver e implementar mecanismos automáticos para interagir e satisfazer as necessidades do usuário [26]. Isto é, os dispositivos e aplicações (heterogêneos e complexos) devem ser capazes de se comunicar e estabelecer alianças (combinar-se) de forma autônoma e com a mínima intervenção humana, para executar um conjunto de tarefas simples ou complexas com a finalidade de interagir e satisfazer as necessidades do usuário [27].

Nesse contexto, o paradigma da Computação Orientada a Serviços (*Service Oriented Computing* - SOC) [28] apresenta-se como o paradigma mais aceito no desenvolvimento e implementação de ambientes pervasivos [29], [26]. Este paradigma permite selecionar e integrar de forma eficiente e efetiva serviços heterogêneos para tratar as requisições do usuário [30].

Nesse sentido, nós partimos da hipótese que em um ambiente pervasivo as funcionalidades de dispositivos e aplicações podem ser disponibilizadas como serviços Web, de modo que o problema de implementar mecanismos automáticos para satisfazer necessidades do usuário

pode ser abordado como um processo de "Composição Automática de Serviços Web". Isto é, o mecanismo de composição procura por uma funcionalidade no ambiente pervasivo, que individualmente, satisfaça a requisição do usuário, caso não a encontre, o mecanismo de composição deverá criar uma nova funcionalidade (combinando os serviços que representam as funcionalidades existentes no ambiente pervasivo) que satisfaça a requisição especificada pelo usuário.

Define-se a composição automática de serviços Web como o processo computacional que tem a habilidade de procurar, selecionar e encadear, em ordem especifica, os mais adequados e relevantes serviços disponíveis para criar uma novo serviço (de valor agregado) que satisfaça a requisição do usuário em um tempo adequado [31]. Já para implementar o processo de composição automática de serviços Web, são necessárias tecnologias como: (i) Web Semântica, para representar os serviços de forma estruturada [32]; e (ii) Mecanismos de Inferência, para raciocinar de forma automática sobre estes serviços [33].

Para utilizar um serviço Web no âmbito da Web Semântica precisa-se da descrição semântica do serviço. Esta descrição semântica requer de uma infraestrutura clara que especifique e compartilhe (sem ambiguidade) as funcionalidades, os parâmetros de entrada/saída, e a forma como os serviços podem ser acessados e executados. Nesse sentido, uma solução proposta para a estruturação de descrições semânticas é o uso de ontologias. Uma ontologia é uma descrição de conceitos e relações entre conceitos [34]. O uso de serviços Web em conjunto com as ontologias têm sido uma área de intensa pesquisa, com objetivo de desenvolver modelos ontológicos que permitam a implementação dos chamados "Serviços Web Semânticos" [6].

Mas, a descrição semântica dos serviços Web, por si só, não resolve o problema de composição automática de serviços Web. Para tal fim, precisa-se também de mecanismos de inferência que raciocinem sobre as descrições semânticas dos serviços Web. Dentre esses mecanismos, existem abordagens predominante aceitas para resolver o problema de composição automática de serviços Web [8]: composição de "*Workflows*" [35], composição baseada em "Algoritmos de Planejamento" [33] e composição baseada em "Agentes" [36].

A abordagem baseada em *Workflows*, aproveita a similaridade entre um *Workflow* e um serviço composto para aplicar o processo de composição [35]. Já o Planejamento, que é uma sub-área da Inteligência Artificial (IA), estuda o processo de deliberação que escolhe e organiza ações, buscando atingir da melhor forma possível os objetivos pré-definidos [33]. Resumindo, planejamento é o processo, nada trivial, de compor um plano solução com as ações a serem seguidas para atingir uma meta ou objetivo. Neste caso, as ações são representadas pelos

serviços, e a meta a ser atingida representa a requisição especificado pelo usuário.

No entanto, compor serviços Web de forma automática no âmbito dos ambientes pervasivos precisa considerar características como [37]:

1. Ter a capacidade de recuperação a falhas devido à indisponibilidade temporal das funcionalidades do ambiente por diversos motivos imprevisíveis [38] [39];
2. Utilizar a relevância das informações contextuais presentes na interação com os usuários, como uma forma de otimizar e até personalizar o processo de composição [40];
3. Permitir agregar e retirar (do ambiente) de forma simples (*plug and play*) novos dispositivos (por exemplo: geladeiras, cafeteiras, fornos, etc.) com as mais diversas funcionalidades [41];
4. Gerenciar a heterogeneidade e complexidade dos distintos tipos de dispositivos e aplicações, e adaptar-se aos distintos cenários presentes nos ambiente pervasivo [42] [43];
5. Garantir a segurança, o controle de acesso e a confiabilidade das funcionalidades providas pelo ambiente [44] [45].

Nos últimos anos, diversos trabalhos têm explorado o processo de composição automática de serviços Web em ambientes pervasivos. Nos artigos [26], [37] e [46] são mapeadas e apresentadas uma serie de propostas sobre sistemas, *middlewares* e *frameworks* que, em algum grau, implementam um processo de composição de serviços para ambientes pervasivos. Nestes artigos, em alguns casos os ambientes pervasivos também são chamados de *Smart Spaces* e/ou Ambientes Inteligentes (*Ambient Intelligence* - AmI). Em base a estes mapeamentos e à pesquisa bibliográfica realizada, podemos dizer:

- Que nem todas as propostas abordam o processo de composição propriamente dito, isto é,em alguns casos só é abordado o processo de descoberta e disponibilização de serviços atômicos;
- Que poucas propostas abordam todas as características que devem estar presentes em um sistema de composição de serviços para ambientes pervasivos, em especial o aspecto de segurança;
- Que a grande maioria das propostas são construídas para um domínio de aplicação e/ou um tipo de ambiente especifico, isto é, a arquitetura e infraestrutura implementadas dificilmente podem ser reaproveitadas ou adequadas para novos ambientes pervasivos;

- Que poucas propostas apresentam uma implementação para um cenário real, isto é, a maioria das propostas avalia seu processo de composição sobre um cenário exemplo, o qual tem uma serie de restrições;

- Que nenhuma proposta apresenta uma especificação formal, completa e genérica o suficiente como para poder ser re-utilizada em distintos domínios de aplicação e/ou tipos de ambientes pervasivos, de forma que permita uma fácil e pratica adoção deste tipo de sistemas para ambientes pervasivos.

Neste contexto, o presente trabalho estuda o problema de composição automática de serviços em ambientes pervasivos, tendo como objetivo principal especificar formalmente um "Modelo de Referência" que permita projetar, implementar e avaliar sistemas de composição de serviços que gerenciem e automatizem o processo de interação em diversos tipos de ambientes pervasivos. Um modelo de referência descreve os elementos abstratos funcionais de um sistema, isto é, um modelo de referência não define uma arquitetura. No entanto, um modelo de referência pode conduzir à implementação de múltiplas arquiteturas [47].

A especificação do nosso modelo de referência permite descrever as funcionalidades a serem implementadas pelo sistema de composição como componentes de alto nível utilizando a linguagem OMG SysML (*OMG Systems Modeling Language*) [48]. Estes componentes representam os módulos de: interface, descoberta, tradução, composição, avaliação, armazenamento, execução e re-composição de serviços; além dos módulos complementares (necessários pelas características dos ambientes pervasivos) de: aquisição e gestão de informações contextuais e preferências do usuário, segurança e controle de acesso de serviços, e especificação do grau de automatização do sistema de composição.

Para avaliar o modelo proposto, foram construídas quatro arquiteturas (derivadas do modelo de referência) para ambientes pervasivos distintos: comercio eletrônico, plataforma petroleira, sala de aula e cidades inteligentes, de modo que foi demonstrada a capacidade de re-utilização e baixo-acoplamento do modelo proposto. Além disso, a implementação de uma aplicação-exemplo em base a um ambiente pervasivo domótico, permitiu identificar aspectos práticos como: a forma como o usuário especifica suas requisições e a estrategia de como armazenar os serviços e ontologias; os quais devem ser considerados no desenvolvimento e implementação de um sistema de composição de serviços para ambientes pervasivos.

## 1.2 Motivação

Entre as principais motivações deste trabalho podemos destacar:

- O nosso interesse em desenvolver pesquisa em ambientes pervasivos de forma real e concreta, analisando a problemática existente e aproveitando os conceitos da Computação Ciente de Contexto, da Web Semântica e do paradigma da Computação Orientada a Serviços como uma forma de solucionar as problemáticas existentes;
- O interesse social em explorar e superar os desafios relacionados à forma como os usuários especificam suas requisições e interagem com um ambiente pervasivo, em especial, usuários como pessoas idosas ou com algum tipo de deficiência os quais podem ser auxiliados em suas atividades diárias pelos ambientes pervasivos;
- O interesse cada vez maior por parte da indústria por soluções (implementações) reais, simples e práticas de sistemas pervasivos que possam gerenciar as novas gerações de dispositivos conectados (*Internet of Things* - IoT).

## 1.3 Descrição do Problema

Um dos maiores desafios de pesquisa em ambientes pervasivos é a de desenvolver e implementar mecanismos automáticos para interagir e satisfazer as necessidades do usuário [26]. Isto porque estes mecanismos precisam considerar as problemáticas [37]: da grande variedade (heterogeneidade) e quantidade (escalabilidade) de dispositivos e aplicações que podem ser embarcados no ambiente; de adaptação à mudança constante (dinamismo) de tipos, quantidade e necessidades de usuários e das características do ambiente; da necessidade por otimização e personalização do processo de composição (utilizando informações de contexto); de prover suporte para a recuperação a falhas em tempo de execução (re-composição) devido à indisponibilidade temporal das funcionalidades do ambiente por diversos motivos imprevisíveis; e de implementar mecanismos que garantam a privacidade e segurança das informações e funcionalidades do ambiente.

Na literatura há diversas propostas que abordam o processo de composição de serviços em ambientes pervasivos para resolver o problema de implementar mecanismos automáticos que interajam e satisfaçam as necessidades do usuário em um ambiente pervasivo, como mapeado nos artigos [26] [37] [46]. Porém, nenhuma destas propostas aborda o problema de composição automática de serviços em ambientes pervasivos considerando as problemáticas descritas de

forma conjunta e completa, e nem se apresenta uma especificação formal, completa e genérica o suficiente como para poder ser re-utilizada por distintos domínios de aplicação e/ou tipos de ambiente pervasivos.

Além disso, aspectos práticos como: a forma como o usuário especifica suas requisições (desejos ou necessidades); a complexidade das requisições especificadas; a granularidade das funcionalidades oferecidas pelo ambiente pervasivo; e a necessidade por satisfazer os requisitos do usuário de forma eficaz em termos de tempo, e de adaptabilidade a situações do mundo real; devem ser considerados na composição automática de serviços em ambientes pervasivos.

Obviamente estas problemáticas e aspectos práticos repercutem diretamente na complexidade do desenvolvimento e implementação de um sistema que gerencie a interação em um ambiente pervasivo, motivo pelo qual muitas das propostas de implementação são só direcionadas para domínios de aplicação e/ou tipos de ambiente pervasivos específicos como uma forma de diminuir a complexidade de implementação.

## 1.4 Solução Proposta

Neste trabalho nós não só pretendemos desenvolver um sistema isolado que corresponda às necessidades de automatizar e gerenciar as interações de um ambiente pervasivo especifico, se não, nós descrevemos uma especificação formal, completa e genérica o suficiente para poder ser re-utilizada por distintos domínios de aplicação e/ou tipos ambientes pervasivos, a qual permita projetar, implementar e avaliar sistemas de composição de serviços de forma simples e pratica.

Nesse sentido, nossa proposta visa desenvolver um Modelo de Referência a partir do qual possam ser projetados e implementados sistemas de composição de serviços que gerenciem a interação de um ambiente pervasivo com seus usuários de forma natural, automática e dinâmica. A especificação do nosso modelo de referência permite descrever as funcionalidades a serem implementadas por sistemas de composição como componentes de alto nível utilizando a linguagem OMG SysML [48]. Estes componentes representam os módulos de: interface, descoberta, tradução, composição, avaliação, armazenamento, execução e re-composição de serviços; além dos módulos complementares (impostos pelas características dos ambientes pervasivos) de: aquisição e gestão de informações contextuais e preferências do usuário, segurança e controle de acesso de serviços, e especificação do grau de automatização do sistema de composição. Além disso, o nosso modelo de referência também descreve os elementos abstratos funcionais de como: são feitas as requisições do usuário (de forma implícita e/ou explicita) ao sistema de composição, e como o sistema de composição responde às requisições

do usuário de forma eficaz em termos de tempo, de adaptabilidade e automatização.

A implementação pratica dos componentes do nosso modelo de referência requerem do uso de tecnologias: da Web Semântica, como ontologias para representar os serviços de forma estruturada [32]; e de Mecanismos de Inferência, como algoritmos de planejamento para raciocinar de forma automática sobre estes serviços [33]. Para tal fim, neste trabalho nós utilizamos e estendemos modelo ontológico WSMO (*Web Services Modelling Ontology*) [49] para suportar a descrição semântica dos serviços e das informações contextuais (contexto) associadas; e também, utilizamos como mecanismo de inferência o algoritmos de planejamento (planejador) JSHOP2 [50], que é a implementação JAVA do algoritmo SHOP2 (*Simple Hierarchical Ordered Planner*) baseado na técnica de planejamento hierárquico HTN (*Hierarquical Task Network*) [51].

Já a nossa proposta apresenta as vantagens de re-utilização e baixo acoplamento, isto porque, com a descrição formal dos componentes do modelo de referência através de relações (aspectos de associação), interações (semântica de operações) e restrições (regras de validação) é possível derivar arquiteturas de sistemas de composição para diversos tipos de ambientes pervasivos, e para sistemas cujo foco não seja a composição, como por exemplo, a implementação do componente de interface para especificação de requisições de usuário pode ser utilizado isoladamente para descrever desejos ou interesses do usuário por um produto, de forma que está especificação possa ser utilizada por sistemas de recomendação [52] ou por sistemas de busca e compra de produtos eletronicamente (Shopbots) [53].

## 1.5 Objetivos

### Objetivo Geral

- Estudar o problema de composição automática de serviços em ambientes pervasivos, e desta forma ter as bases conceituais e experimentais para especificar formalmente um “Modelo de Referência” que permita projetar, implementar e avaliar sistemas de composição de serviços que gerenciem e automatizem o processo de interação ambiente-usuário para diversos tipos de ambientes pervasivos.

### Objetivos Específicos

- Especificar os requisitos que os ambientes pervasivos impõem ao processo de composição automática de serviços;

- Identificar os elementos abstratos funcionais de um sistema de composição automática de serviços em ambientes pervasivos;
- Especificar as funcionalidades de um sistema de composição automática de serviços em ambientes pervasivos formalmente em um modelo de referência como componentes de alto nível;
- Especificar as relações (aspectos de associação), interações (semântica de operações) e restrições (regras de validação) dos componentes do modelo de referência;
- Derivar uma arquitetura em base ao modelo de referência, cumprindo os formalismo especificados;
- Projetar um sistema de composição automática de serviços em ambientes pervasivos a partir do modelo de referência e da arquitetura derivada como uma aplicação-exemplo, para tal fim será necessário:
  - Modelar as funcionalidades oferecidas pelos dispositivos e aplicações como serviços enriquecidos semanticamente utilizando o modelo ontológico WSMO;
  - Estender o modelo ontológico WSMO para prover suporte à modelagem de informações contextuais relevantes;
  - Modelar informações contextuais sobre o ambiente, usuário, dispositivos e aplicações;
  - Integrar o planejador JSHOP2 como mecanismo de inferência para implementar o processo de composição.
- Identificar as problemáticas e características decorrentes da implementação pratica da aplicação-exemplo;
- Avaliar a eficiência e eficacia da aplicação-exemplo, em termos de tempo, de adaptabilidade e considerando as informações contextuais definidas e o dinamismo presente no processo de interação com o ambiente;
- Analisar os resultados da avaliação da aplicação-exemplo para validar o modelo de referência proposto e implementar futuras melhorias;
- Discutir as conclusões atingidas e propor diretrizes a seguir para atingir uma evolução no desenvolvimento dos sistemas de composição para ambientes pervasivos.

## 1.6 Justificativas

A escolha de abordar o problema de interação em um ambiente pervasivo como um processo de composição automática de serviços, justifica-se pela maturidade de paradigmas como Computação Orientada a Serviços [30] e Web Semântica [32].

Já formalizar um modelo de referência, justifica-se pela necessidade de criar um modelo que especifique formalmente os elementos abstratos funcionais do sistema de composição, de modo que pode este modelo possa conduzir ao desenho de múltiplas arquiteturas e à implementação de sistemas de composição para distintos tipos de ambientes pervasivos.

A escolhida de utilizar a linguagem OMG SysML [48] para a descrição formal do modelo de referência, justifica-se porque esta linguagem permite: modelar diferentes tipos de requisitos com suporte para funções contínuas e estruturas de decomposição; representar e verificar processos do fluxo de controle com alto nível de detalhe; a representação e formalização de eventos, especificação de dependências, comportamento de classes, representação de valores de entrada e saída, e modelagem de subsistemas como uma visão integrada do sistema, incluindo hardware, software e partes eletro-mecânicas.

A escolha do modelo ontológico WSMO [49] para a descrição semântica dos serviços, justifica-se pelo seu poder de expressividade na descrição semântica dos serviços, isto é, os conceitos definidos no WSMO permitem expressar as requisições do usuário como metas (*goals*); já as funcionalidades dos serviços e o estado do mundo real podem ser especificados pelas pré-condições, suposições, pós-condições e efeitos [6]. Além disso, o WSMO provê um framework mais completo e claro que inclui uma linguagem formal WSML (*Web Service Modeling Language*) [54];

Já escolha de modelar informações contextuais utilizando modelos ontológicos, justifica-se porque a interação dos usuários com o ambiente produz uma grande quantidade de informação que deve ser aproveitada para otimizar o processo de interação [18].

A estudo de técnicas de planejamento Hierárquico ou Rede Hierárquica de Tarefas (*Hierarchical Task Network* - HTN) [51] e a utilização do planejador JSHOP2 para a implementação da aplicação-exemplo, justifica-se pela sua melhor adequação no processo de criação de planos solução por métodos de decomposição de tarefas e regras de controle de busca, aproveitando a natureza hierárquica inerente dos serviços Web [55] e das funcionalidades providas pelo ambiente pervasivo.

Finalmente, a escolha do tipo de ambiente pervasivo (domótico) para a aplicação-exemplo,

justifica-se pelos recursos (sensores, computadores, ambiente) disponíveis. Já a derivação de arquiteturas para os tipos de aplicação de comercio eletrônico, plataforma petroleira, sala de aula e cidades inteligentes, justifica-se pelas oportunidades de desenvolver projetos reais com distintas características e que por sua vez possam ser utilizados em futuros projetos de pesquisa.

## 1.7 Metodologia

Para atingir nossos objetivos, como etapa inicial desta pesquisa foi realizada uma ampla revisão bibliográfica dos principais conceitos, técnicas, ferramentas e áreas de aplicação da computação pervasiva, computação ciente de contexto, ambientes pervasivos, Web Semântica, Serviços Web Semânticos e do estado da arte da Composição Automática de Serviços Web, com ênfase em ambientes pervasivos. Para esta etapa foram considerados principalmente livros, artigos científicos, dissertações, teses, reportes técnicos e sites de Internet, os quais permitiram conhecer os fundamentos teóricos necessários para o desenvolvimento deste trabalho. A análise deste levantamento bibliográfico permitiu definir os objetivos do trabalho e estabelecer as bases conceituais e requisitos funcionais para o desenvolvimento do modelo de referência proposto.

Para dar uma maior robustez ao modelo, algumas ferramentas que permitem a composição de serviços Web ( [12], [13] e [14]) disponíveis na Web foram avaliadas de forma prática com o intuito de identificar requisitos funcionais do modelo a partir da base experimental. Estas ferramentas não tem uma aplicação prática para ambientes pervasivos, porém, sua experimentação ajuda a entender e especificar detalhes práticos de implementação sobre a interface de usuário e as formas como se representam formalmente as requisições do usuário.

A segunda etapa compreende a concepção e especificação do modelo de referência utilizando a linguagem OMG SysML [48]. Este processo inclui a especificação das relações (aspectos de associação), interações (semântica de operações) e restrições (regras de validação) dos módulos de: interface, descoberta, tradução, composição, avaliação, armazenamento, execução e re-composição de serviços; além dos módulos complementares (necessários pelas características dos ambientes pervasivos) de: aquisição e gestão de informações contextuais e preferências do usuário, segurança e controle de acesso de serviços, e especificação do grau de automatização do sistema de composição.

Como terceira etapa temos a construção de arquiteturas derivadas do modelo de referencia e a implementação de um aplicação-exemplo. O processo de implementação da aplicação-exemplo utiliza o modelo em espiral para desenvolvimento de software [56], visando que o protótipo a ser criado tenha uma complexidade crescente, visa-se a utilização de

ferramentas de software livre no desenvolvimento da prototipagem da aplicação-exemplo (servidor Web Apache Tomcat, sistema gerenciador de banco de dados PostgreSQL, linguagem de programação Java, modelo ontológico WSMO e planejador JSHOP2).

Finalmente a quarta etapa compreende o processo de avaliação do modelo de referência e da aplicação-exemplo. Nesse sentido, serão avaliados os seguintes critérios [57]:

- Referente ao modelo de referência:
  - Usabilidade/Complexidade, refere-se à quantidade de esforço que é necessário para entender e utilizar o modelo de referência para derivar e projetar sistemas de composição de serviços para ambientes pervasivos;
  - Baixo Acoplamento/Interoperabilidade, refere-se à possibilidade de poder implementar sistemas que não tenham como foco a composição de serviços;
- Referente à aplicação-exemplo:
  - Desempenho/Escalabilidade, refere-se ao desempenho de execução (p. ex. tempo de processamento para resolver a requisição do usuário) e as características de escalabilidade do sistema;
  - Correção, refere-se à capacidade da aplicação de se recompor, isto é, o sistema deve garantir que se a solução proposta falha, uma nova solução deve ser provida ao usuário;
  - Automação, refere-se à influência exercida pelo usuário, isto é, o sistema pode precisar da permissão e/ou ação do usuário para executar o processo de composição (p. ex. automático ou semi-automático).

## 1.8 Organização da Tese

O presente texto está organizado em seis capítulos, descritos a seguir:

- **Capítulo 1:** Apresenta a introdução, motivação, descrição do problema, solução proposta, objetivos, justificativas, metodologia e organização da tese;
- **Capítulo 2:** Apresenta o fundamento teórico da pesquisa, isto inclui conceitos sobre: Computação Pervasiva e Ciente de Contexto; Serviços Web e Web Semântica; e Composição de Serviços;

- **Capítulo 3:** Apresenta o estado da arte e os trabalhos relacionados ao presente trabalho;
- **Capítulo 4:** Apresenta a conceitualização do trabalho de doutorado, isto é a definição do escopo e a formalização do modelo de referência proposto;
- **Capítulo 5:** Apresenta a especificação das arquiteturas derivadas do modelo de referência para quatro cenários de aplicação e a validação do modelo através da implementação de uma aplicação-exemplo;
- **Capítulo 6:** Apresenta as conclusões, considerações adicionais sobre aspectos de implementação, trabalhos futuros e as contribuições atingidas no presente trabalho.

# 2 Referencial Teórico

Este capítulo descreve conceitos referentes à Computação Pervasiva, Ambientes Pervasivos, Computação Ciente de Contexto, Serviços Web, Web semântica, Serviços Web Semânticos e Composição de Serviços Web.

## 2.1 Computação Pervasiva

Define-se a computação ubíqua como a "tecnologia integrada ao ambiente, porém, de forma não intrusiva". O objetivo principal da computação ubíqua é integrar diversas tecnologias (dispositivos, aplicações de software locais ou distribuídas, etc.) ao ambiente de forma flexível e transparente para o usuário [58]. Assim, estas tecnologias interconectam-se umas as outras provendo uma variedade de funcionalidades que, eventualmente, são utilizadas para auxiliar as atividades do usuário [25].

Na literatura, diversos autores discutem as similaridades e diferenças entre os paradigmas da computação ubíqua e da computação pervasiva. Já neste trabalho essa discussão não é abordada, assumindo a definição dada em [1], na qual ele chama a computação ubíqua de computação pervasiva, e descreve algumas características deste paradigma:

- Espaços Inteligentes (*Smart Space*), refere-se aos espaços físicos (por exemplo: casas, escritórios, etc.) embarcados com infraestrutura computacional para captura de video e áudio;
- Invisibilidade (*Invisibility*), refere-se à capacidade da infraestrutura computacional embarcada no ambiente de interagir com os usuários de forma natural, transparente e até quase inconsciente;
- Escalabilidade localizada (*Localized Scalability*), refere-se à capacidade de poder integrar uma grande quantidade de nova infraestrutura computacional ao ambiente físico;

- Estrategia de Adaptação (*Adaptation Strategy*), refere-se à capacidade da infraestrutura computacional de adaptar-se às mudanças físicas do ambiente em função das requisições, condições e preferências do usuário;

A Figura 2.1 [1] apresenta um taxonomia que descreve a relação entre os sistemas distribuídos, a computação móvel e a computação pervasiva desde o ponto de vista dos conceitos e características abordadas por cada um destes paradigmas.

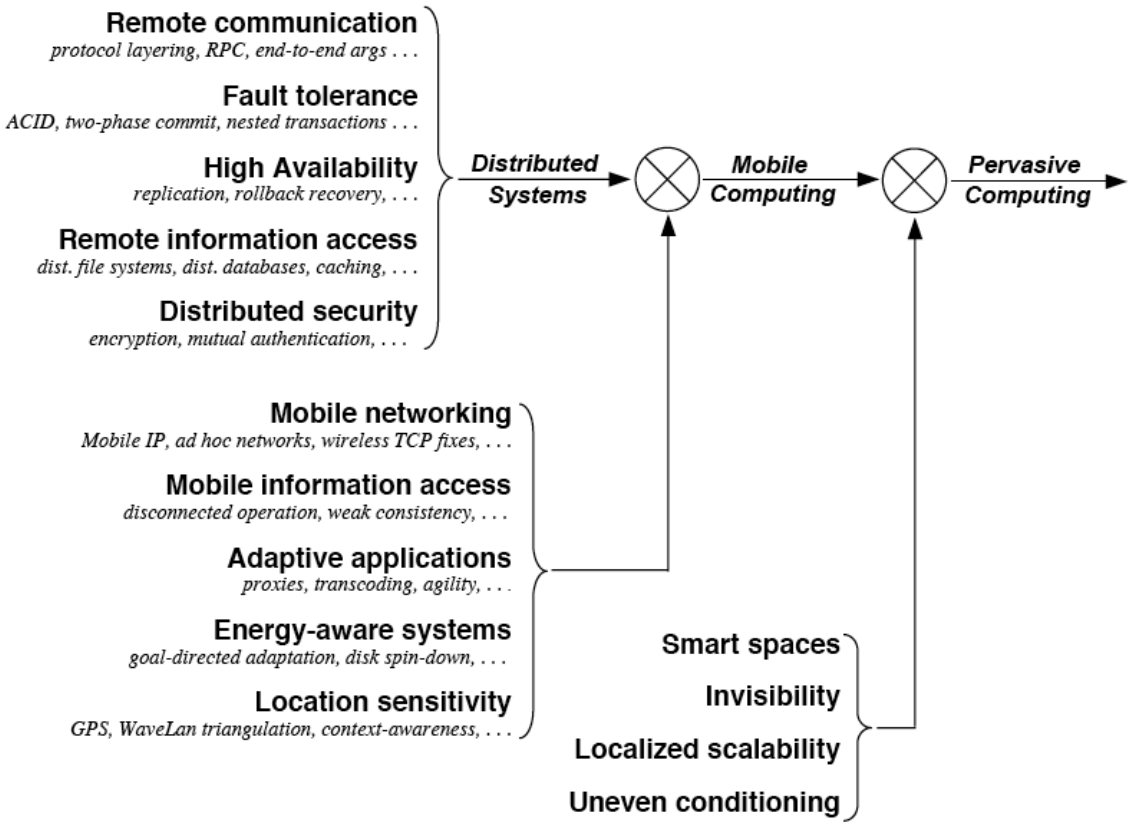


**Figura 2.1:** Taxonomia das características da Computação Pervasiva (adaptado de [1])

A aplicação prática da computação pervasiva pode ser encontrada em diversos ambientes como em: residências, ambientes de trabalho, meios de transporte, entre outros. Aplicações pervasivas oferecem diversas funcionalidades para estes ambientes, como: segurança, comodidade, informação, entretenimento, etc. [25]. Já infraestrutura computacional embarcada no ambiente e utilizada pelas aplicações pervasivas pode incluir: sensores e atuadores para residências e automóveis, eletrodomésticos (p. ex. ar-condicionado, aquecedor, TVs, etc.), dispositivos eletrônicos de uso pessoal (p. ex. relógios, telefones celulares, *Smartphones*, PDAs, consoles de jogos, etc.) e outros dispositivos com fines específicos tales como etiquetas inteligentes (*RFID tags and Readers*), Redes de Sensores Sem Fio (RSSF), dispositivos de GPS (*Global Positioning System*), entre outros [59].

## 2.2 Ambientes Pervasivos

"Ambientes Pervasivos" são ambientes complexos e heterogêneos povoados por diversos dispositivos (sensores, atuadores, etc.) e aplicações de software (locais ou distribuídas) incorporados nestes ambientes físicos, para interagir de forma natural e transparente com o usuário [1]. Estes dispositivos e aplicações interconectam-se uns aos outros com o objetivo de prover uma variedade de funcionalidades que, eventualmente, são utilizadas para auxiliar as atividades do usuário [25].

A interação natural e transparente em ambientes pervasivos refere-se a ao processo de interação homem-computador (*Human Computer Interaction* - HCI), neste caso o computador representa toda infraestrutura computacional embarcada no ambiente, onde o usuário aproveita os recursos e funcionalidades do ambiente sem quase modificar seu comportamento cotidiano. A interação do usuário com o ambiente acontece através da interação entre as mãos, os gestos, as palavras e as interfaces do ambiente como o mouse, o teclado, um sistema de câmeras para reconhecimento de gestos, um sistemas de reconhecimento de fala, entre outros [24].

Ambientes pervasivos também são chamados de: (i) ambientes inteligentes, quando técnicas de IA (Inteligência Artificial) para raciocínio e inferência são utilizadas no processo de interação usuário-ambiente [60]; e de (ii) *smart spaces*, quando são integrados dispositivos e aplicações de captura e monitoração de áudio e vídeo ao ambiente [61]. Na Figura 2.2 [2] apresenta-se o *Gator Tech Smart House* que é um ambiente pervasivo residencial embarcado com uma serie de dispositivos (sensores, atuadores, etc.) e aplicações. Esta figura caracteriza bem o que seria um ambiente pervasivo.

Cenários de aplicação de ambientes pervasivos incluem: residências, fabricas, museus, zoológicos, centros comerciais, veículos, entre outros [42]. No entanto, nos últimos anos um dos cenários mais explorados foi o de residencias que prestam assistência nas atividades das pessoas, em especial a pessoas idosas. Os principais serviços que são oferecidos para as pessoas em um ambiente pervasivo são: monitoração de seus sinais vitais, monitoração de seu comportamento (atividades e movimentos), conforto (automatizando tarefas e dando apoio em atividades da vida diária) e assistência na sua saúde (informações de medicamentos que podem ou não ser utilizados).

Entre os principais projetos em andamento sobre ambientes pervasivos que prestam assistência a idosos em ambientes domésticos destacam-se o projeto SOPRANO [62] e INHOME [3], que fazem parte do programa *Ambient Assisted Living* (ALL) [63] da comunidade Europeia. SOPRANO é uma plataforma *open source* baseada em uma combinação de

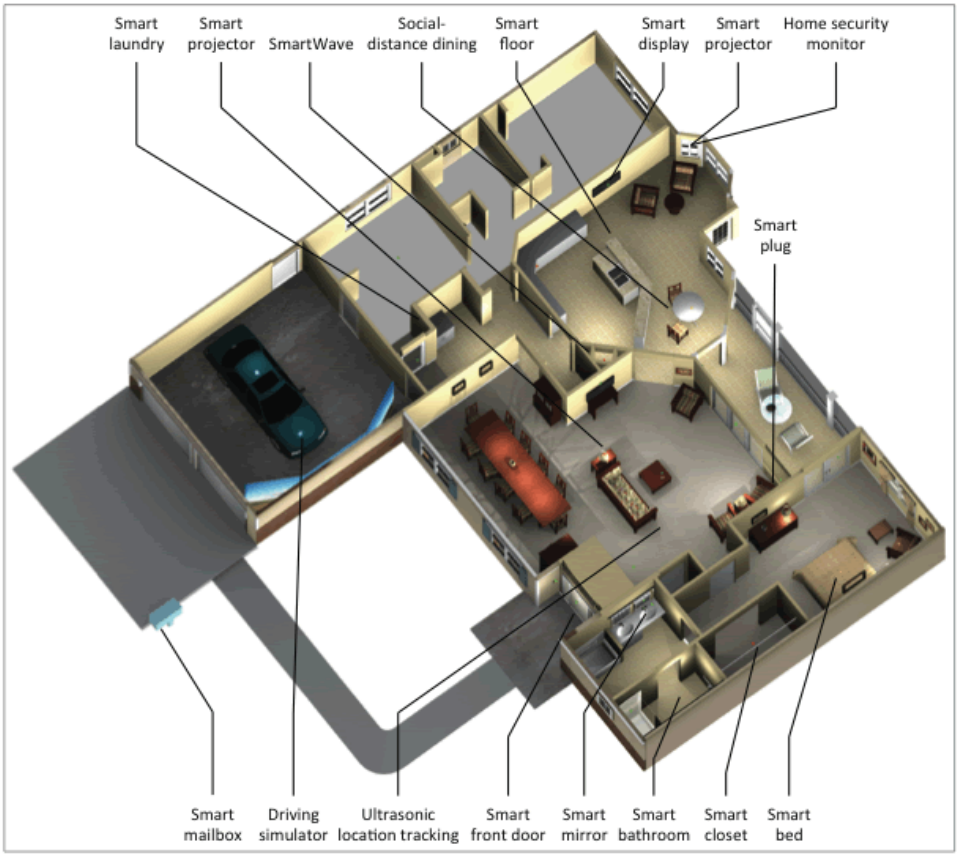


**Figura 2.2:** *The Gator Tech Smart House* (adaptado de [2])

tecnologias semânticas e orientadas a serviços. A infra-estrutura do SOPRANO oferece interfaces estritamente projetadas para usuários previamente definidos, isto é, cada um dos usuários e suas necessidades são modelados em uma ontologia formal. As descrições semânticas dadas pela ontologia permitem enriquecer os dados do contexto, os quais podem ser acessados e compartilhados em função das preferências e políticas de segurança do usuário. Já o projeto INHOME implementa uma arquitetura que é representada na Figura 2.3 a qual caracteriza a agrupação em categorias dos distintos tipos de dispositivos embarcados no ambiente, mostra os protocolos de comunicação e fluxo de dados que acontece no ambiente e apresenta o INHOME Terminal que é uma central de controle e automação que gerencia o ambiente.

No entanto, para que um ambiente pervasivo interaja de forma adequada com seus usuários, este precisa conhecer e entender o contexto (informações contextuais) sobre o próprio ambiente, sobre os dispositivos e sobre os usuários. Por exemplo, no processo de monitoração de sinais vitais, uma alerta ao serviço de pronto socorro deve acontecer quando os dados adquiridos são avaliados como anormais, isto é, fora do padrão definido para o usuário. Este padrão pode ter variações dependendo do tipo de usuário [64].

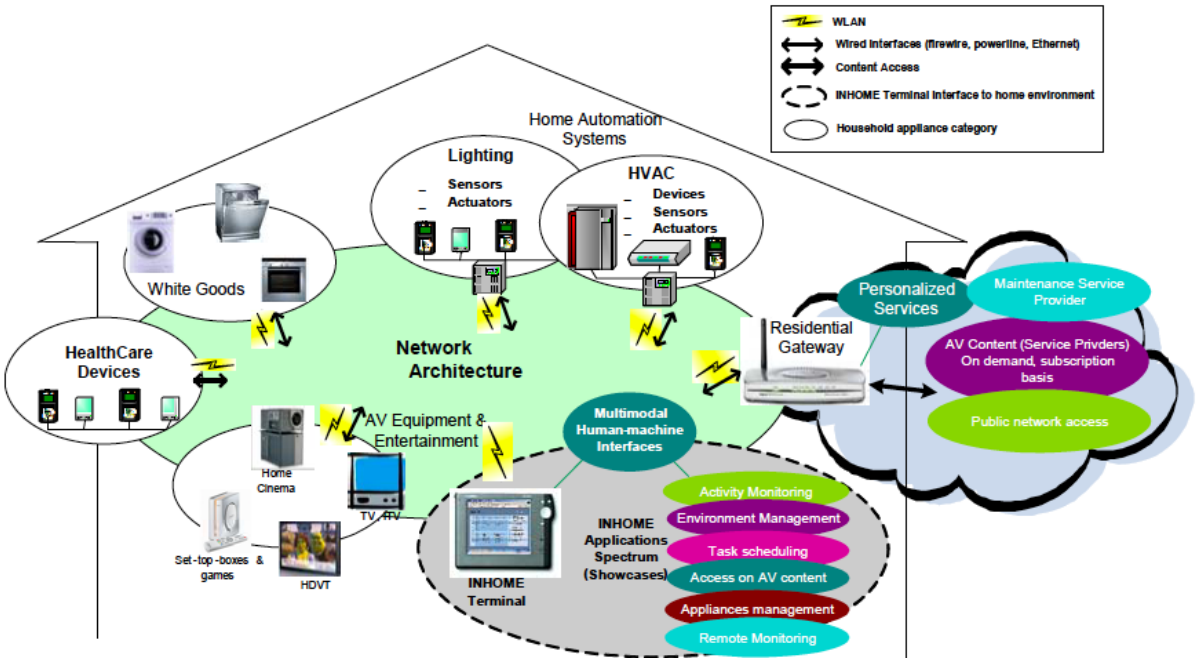


**Figura 2.3:** Arquitetura do projeto INHOME - [3]

## 2.3 Computação Ciente de Contexto

"Contexto é qualquer informação que pode ser utilizada por um sistema computacional para caracterizar a situação de uma entidade que é considerada relevante entre o usuário e o sistema". "Um sistema é ciente de contexto quando o sistema utiliza o contexto para fornecer informações relevantes e/ou serviços ao usuário, onde a relevância depende das tarefas do usuário" [65]. Já em [66] afirma-se que informações de contexto podem ser descritas em cinco dimensões:

- Quem (*Who*), são informações referentes à entidade que realiza uma ação. Uma entidade pode ser uma pessoa, um dispositivo, um agente software (p. ex. um serviço), um ambiente ou um sistema;
- Onde (*Where*), são informações referentes à localização das entidades e as relações espaciais entre entidades, isto é, relações espaciais como a distância entre duas pessoas que se encontram no mesmo ambiente;
- Quando (*When*), são informações referentes às atividades temporais ou eventos que realizam as entidades, isto é, duração d atividades, o tempo que um determinado evento inicia e termina, etc.;
- Que (*What*), são informações referentes às atividades físicas ou ações feitas pelas entidades. Esta dimensão de contexto é mais complexa, isto porque ela deve ser inferida em base as outras dimensões;

- Por que (*Why*), são informações referentes ao motivo pelo qual é feita uma determinada ação. As motivações das entidades podem ser inferidas em base as outras dimensões, especialmente, à dimensão Quem (*Who*), isto porque ela geralmente descreve os perfis de preferências das entidades.

Outras duas dimensões podem ser agregadas ao modelo de descrição de informações de contexto:

- Como capturar (*How capture*), define as regras de como as informações contextuais serão capturadas [67];
- Como apresentar (*How represented*), define as regras de como as informações contextuais serão apresentadas. Estas regras estão diretamente relacionadas ao tipo de representação dada à informação contextual (p. ex. como texto, como gráfico, como mapas, etc.) [4];

As sete dimensões contextuais são sumarizadas e representadas pela Figura 2.4.

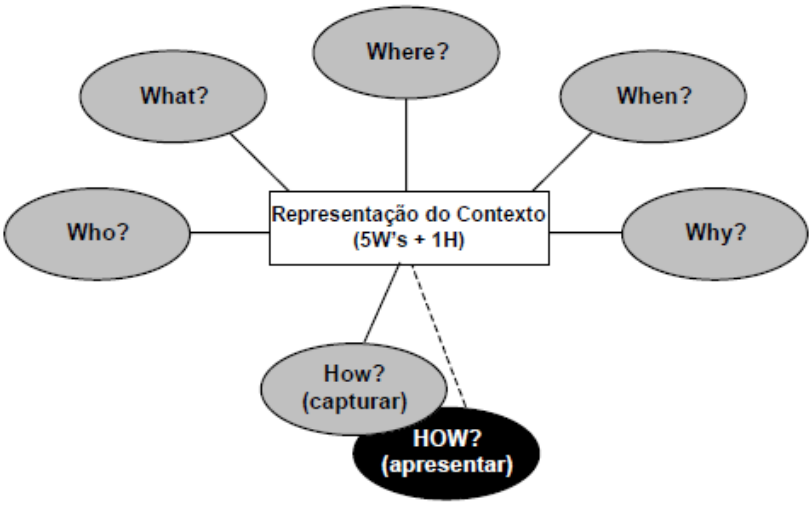


**Figura 2.4:** Sete dimensões contextuais - [4]

Já o desenvolvimento e implementação de aplicações cientes de contexto precisam se preocupar com desafios para [68]:

- Sensoriamento: Escolha e inclusão dinâmica dos contextos mais apropriados a cada entidades; Técnicas para coleta de contextos físicos, lógicos e virtuais; Atribuição de semântica uniforme aos contextos utilizados; Identificação e escolha de fontes de contextos;

- Modelagem: Definição de arquiteturas para sistemas cientes de contexto; Modelos de representação uniforme da sintaxe dos dados de contexto coletados; Modelo de armazenamento de dados contextuais; Modelo de comunicação adotado entre diversas entidades (usuários e/ou aplicações);

- Qualidade: Qualidade de contexto (QoC), granularidade das informações contextuais; Qualidade de serviço (QoS), propriedades não funcionais associadas às informações contextuais; Qualidade das fontes de contexto; Gerenciamento de aplicações cientes de contexto; Tratamento de falhas; Automatização de tarefas; Utilização de algoritmos de aprendizado; Identificação e tratamento de contextos individuais conflitantes; Identificação e tratamento de contextos coletivos conflitantes;

- Segurança: Segurança para troca de dados entre usuários e aplicações; Confiabilidade das fontes de contextos; Segurança da Informação de contexto;

Nesse sentido, sistemas pervasivos precisam considerar as informações contextuais presentes na interação com os usuários, como uma forma de otimizar e até personalizar o processo de interação [40].

## 2.4 Serviços Web

Define-se como "Serviço", no contexto de SOA (*Service Oriented Architecture*), à função de um sistema computacional construído de forma que possa ser facilmente vinculado a outros componentes de software para permitir o acesso a um conjunto de regras de negócio. Este acesso é provido através de uma interface descrita com restrições e políticas especificadas na descrição de serviço. Além disso, um serviço pode ser do tipo fornecedor ou consumidor, e deve permitir ser: reutilizável, independente da implementação, fracamente acoplado e capaz de encapsular sua lógica de negócios, o que possibilita que estes serviços podem ser compostos e orquestrados em processos de negócios [47].

Já um "serviço Web" ou *Web Service* é a implementação, mais comum, de um serviço no contexto de SOA. Note-se que os serviços Web não são a única forma de implementar serviços. A W3C (*World Wide Web Consortium*) [5] define um serviço Web como uma aplicação identificada por uma URI (*Uniform Resource Identifier*), cujas interfaces e ligações são definidas, descritas e descobertas utilizando-se a linguagem XML (*Extensible Markup Language*). Além disso, as descrições de interfaces dos serviços Web são descritas utilizando a linguagem "WDSL (Web Services Description Language) [69]. Já o protocolo SOAP (*Simple*

*Object Access Protocol*) tem como função encapsular as chamadas a métodos remotamente distribuídos, que serão transportados, por sua vez, por algum protocolo de transporte tal como o HTTP (*Hyper Text Transport Protocol*) utilizando a comunicação entre um navegador (*browser*) e um servidor Web [70].

Nesse sentido, um serviço Web pode ser conceituado como uma forma de expor uma funcionalidade de um sistema de informação e torná-la disponível através de padrões da tecnologia Web, o que faz com que serviços Web sejam independentes de linguagens e de plataformas, e permitam a fácil integração de sistemas heterogêneos [71]. Devido a estas características e facilidades, os serviços Web têm sido amplamente utilizados nos últimos anos, principalmente em aplicações Web. Porém, os serviços Web têm sido desenvolvidos, principalmente, para a interpretação e uso dos seres humanos e não das máquinas.

## 2.5 Web Semântica

A "Web Semântica" define-se como uma extensão da Web atual na qual o significado das informações está bem definida, permitindo um melhor trabalho cooperativo entre computadores e pessoas [32].

A Web semântica tem como finalidade fornecer um modelo capaz de atribuir um significado (sentido) aos conteúdos gerados por diferentes fontes de dados como páginas Web, banco de dados, sensores e até mesmo informações da nossa vida diária, de modo que possam ser interpretados, tanto, por seres humanos como por máquinas (computadores) através de agentes de software de forma automática [32]. Nesse sentido, a Web Semântica tem como objetivo associar descrições semânticas às informações da Web atual sob uma representação que habilite aos computadores a interpretarem e realizarem inferências sobre essas informações. A implementação da Web semântica gera serviços de recuperação, intercâmbio e integração de dados, garantindo a interoperabilidade das informações [72].

A infra-estrutura da Web Semântica está construída com base nas especificações [5]: do XML [73] para a estruturação de documentos; do *XML Schema* [74] para tipagem de dados de elementos XML; do RDF *Resource Description Framework* [75] que fornece um modelo genérico de representação de metadados em uma sintaxe XML; do *RDF Schema* [76] que fornece uma linguagem baseada em hierarquia de classes para descrever relacionamentos semânticos RDF por meio de classes, propriedades e relacionamentos entre classes e restrições; e de Ontologias [34] são os blocos principais da Web Semântica para a representação de conhecimento. A Figura 2.5 representa a infra-estrutura da Web Semântica em camadas, como

definido em [32].

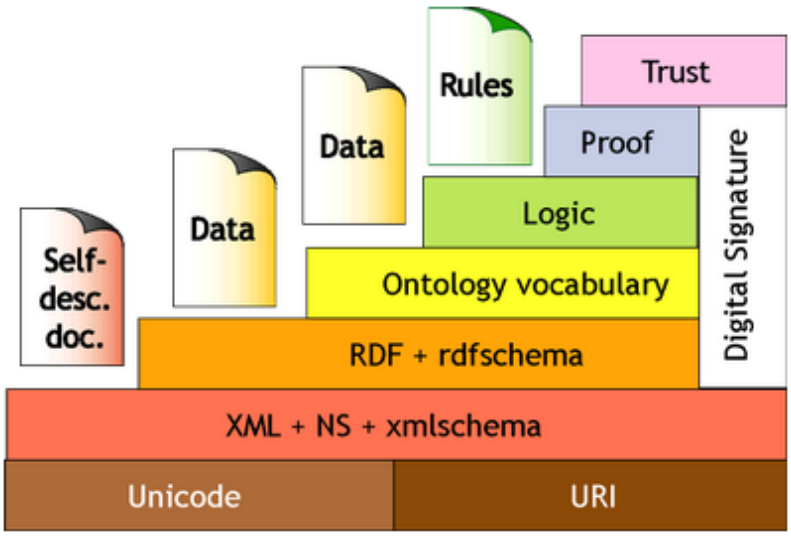


**Figura 2.5:** A infra-estrutura em camadas da Web Semântica - [5]

Para utilizar um serviço Web no contexto da Web Semântica, um agente de software precisa de uma descrição semântica do serviço que defina a forma como este serviço pode ser acessado e executado. Nesse sentido, é necessário que exista uma infraestrutura clara para especificar e compartilhar essas descrições semânticas. Uma solução proposta para a estruturação de descrições semânticas é o uso de ontologias.

O termo "ontologia" é originário da filosofia e é utilizado para representar uma visão do mundo em um sistema de categorias. Já uma das definições mais citadas na literatura de ontologia no contexto da Web semântica é a que define "Uma ontologia como uma especificação formal, explicita e compartilhada de uma conceitualização" [34]. Posteriormente em [77] analisa-se cada um dos termos desta definição:

- Conceitualização: refere-se a um modelo abstrato de algum fenômeno no mundo, o qual é identificado pelos seus conceitos relevantes;
- Explicita: significa que os conceitos descritos e suas restrições estão explicitamente definidos;
- Formal: refere-se ao fato de que os conceitos descritos seguem um formalismo matemático que evita ambiguidades, motivo pelo qual estes conceitos podem ser compreendidos pelas máquinas;
- Compartilhada: refere-se à noção de que os conceitos descritos em uma ontologia representam um conhecimento aceito por um grupo de pessoas e não de forma individual.

Nesse sentido, podemos dizer que uma ontologia é uma descrição de conceitos, relações e restrições entre conceitos que podem ser utilizados por um agente de software para processar informações. O uso de serviços Web em conjunto com as ontologias têm sido uma área de intensa pesquisa por parte da comunidade científica, com objetivo de desenvolver modelos ontológicos que permitam a implementação dos chamados "Serviços Web Semânticos".

## 2.6 Serviços Web Semânticos

Um serviço Web semântico não é outra coisa que um serviço Web que se auto-descreve semanticamente, isto é, descreve: suas funcionalidades através da especificação de restrições (pré-condições e pós-condições), e a forma como pode ser automaticamente descoberto e invocado [13]. Dentre os modelos ontológicos que permitem implementar esta descrição semântica nos serviços Web destacam-se: OWL-S (*Web Ontology Language for Services*) [78], WSMO (*Web Services Modelling Ontology*) [49] e WSDL-S (*Web Service Description Language for Services*) [79].

Já neste trabalho será utilizado o modelo ontológico WSMO, isto porque, este modelo provê um *framework* mais completo e claro para a descrição de serviços Web, o que possibilita a automatização (total ou parcial) de tarefas como: descoberta, seleção, composição, execução e monitoração da execução de serviços. Além disso, o WSMO oferece uma maior grau de expressividade na descrição dos serviços Web semânticos. Fatos que justificam a escolha deste modelo ontológico para nossa implementação prática. Nesse sentido, a continuação detalha-se os principais conceitos deste modelo ontológico.

### 2.6.1 WSMO

WSMO é um *framework* que implementa uma ontologia formal e um conjunto de linguagens que permitem descrever todos os aspectos relevantes para acessar um serviço Web de forma automática (total ou parcial) [6]. Nesse sentido, WSMO está baseado nos seguintes princípios [6] [80]:

- **Conformidade com a Web:** WSMO herda o conceito do identificador único de recursos URI (*Universal Resource Identifier*) como princípio essencial de projeto para a *World Wide Web*. Além disso, WSMO adota o conceito de espaço de nomes para denotar espaços de informação consistentes e suporta XML e outras recomendações da W3C (*World Wide Web Consortium*), bem como a descentralização de recursos;

- **Baseado em ontologias:** Ontologias são usadas como modelo de dados através de WSMO, significando que toda descrição de recursos, bem como todos os dados trocados durante o uso do serviço são baseados em ontologias. O uso extensivo de ontologias apoiá a interoperabilidade;

- **Desacoplamento estrito:** Desacoplamento denota que recursos de WSMO são definidos isoladamente, significando que cada recursos é especificado independentemente;

- **Mediação centralizada:** Como um princípio complementar do principio anterior, mediação manipula heterogeneidades que naturalmente surgem em ambientes abertos. Heterogeneidade pode ocorrer em termos de dados, ontologia subjacente, protocolos e processos. WSMO reconhece a importância de mediação para o sucesso do desenvolvimento dos serviços Web;

- **Separação do papel ontológico:** Usuários/clientes existem em contextos específicos, diferentes dos contextos dos serviços Web. WSMO diferencia desejos de usuários como metas (*goals*) dos serviços Web disponíveis;

- **Descrição versus Implementação:** WSMO diferencia a descrição dos elementos dos serviços Web semânticos das tecnologias executáveis (implementação do serviço - WSDL). Enquanto o primeiro requer uma descrição precisa, baseada em um formalismo apropriado, o segundo se preocupa com o suporte de tecnologias existentes para a execução de serviços Web semânticos;

- **Semântica de execução:** Para verificar a especificação de WSMO existem semânticas de execução formal de implementações de referência como WSMX e outros sistemas baseados em WSMO que fornecem a percepção técnica deste modelo ontológico;

- **Serviços versus serviços Web:** Um serviço Web é uma entidade computacional com a qual é possível alcançar uma meta do usuário por invocação. Um serviço, ao contrário, é o valor real fornecido por esta invocação. WSMO fornece um meio para descrever serviços Web que fornecem acesso a serviços (busca, compra, etc.).

Baseados nesses princípios, WSMO implementa quatro conceitos principais, os quais devem ser utilizados para descrever um serviço Web semântico. Estes conceitos são representados pela Figura 2.6 e descritos a seguir [6] [80]:

- **Ontologia:** O conceito Ontologia em WSMO define a semântica formal para as terminologias usadas dentro de todos os componentes de WSMO sejam claramente

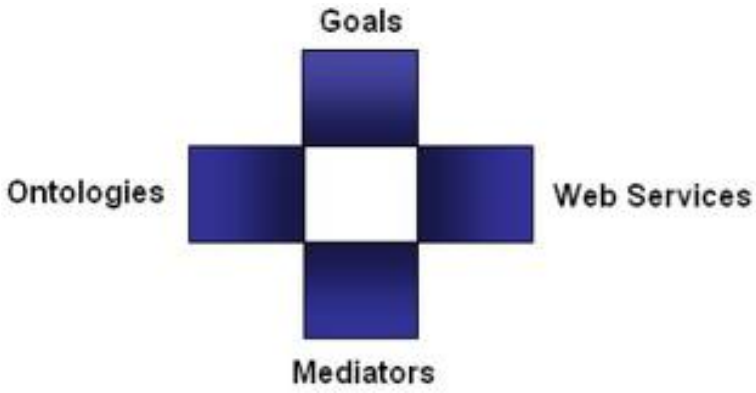


**Figura 2.6:** Principais conceitos do WSMO - [6]

especificados. A utilização de ontologias permite definir sem ambiguidades o vocabulário a ser compartilhado, o que permite a interoperabilidade e automatização em processo de integração de dados;

- **Serviços Web:** O conceito sobre serviços Web em WSMO fornece um modelo conceitual para descrever de maneira explícita todos os aspectos de um serviço Web: funcionalidades ou *capabilities*, propriedades não-funcionais e interface.
  - As funcionalidades ou *capabilities* utilizam os seguintes conceitos para expressar restrições e relações entre os serviços:
    - **Pré-Condições:** As pré-condições especificam os estados requeridos do espaço de informações antes da execução do serviço. Isto é, especificar as informações que espera o serviço Web a fim de prestar o serviço. As pré-condições condicionam o conjunto de estados do espaço de informações, de tal forma que qualquer estado que satisfaça essas condições pode servir como um estado inicial valido (no espaço de informações) para a execução do serviço de uma determinada maneira [6]. Por exemplo, para um domínio especifico, o espaço de informações está composto por todas as ações descritas no domínio. A ação que satisfaça as pré-condições (conjunto de estados requeridos) se converterá em parte do plano solução;
    - **Suposições:** As suposições descrevem o estado do mundo suposto antes da execução do serviço. Caso contrário, o êxito da prestação do serviço não é garantido. Diferente das pré-condições, suposições não são necessariamente verificadas pelo serviço. É feita esta distinção a fim de permitir a ideia explícita de condições sobre o estado do mundo que estão fora do espaço de informações. [6]. Por exemplo, para um domínio especifico, o estado do mundo é representado inicialmente pelo estado inicial especificado no domínio;

    * **Pós-condições:** As pós-condições descrevem o estado do espaço de informações que é atingido após a execução bem sucedida do serviço, mas também descrevem a relação entre as informações que são fornecidas ao serviço e os resultados gerados [6];
    * **Efeitos:** Os efeitos descrevem o estado do mundo que é atingido após o sucesso da execução do serviço, isto é, se as pré-condições e suposições do serviço foram satisfeitos [6].

  - Propriedades não-funcionais compreende itens como custos de invocação de serviço e parâmetros relacionados à qualidade de serviço (QoS);
  - A interface descreve: como interagir com um serviço do ponto de vista do solicitante por meio de sua coreografia; e descreve como o serviço faz uso de outros serviços e metas a fim de fornecer sua funcionalidade, por meio de sua orquestração. O WSMO através da linguagem WSML oferece suporte para a descrição de coreografias, porém nenhuma especificação é feita de como as coreografias devem ser interpretadas. Já a descrição de orquestrações deve ser feita por linguagens externas ao WSML.

- **Metas (*Goals*):** As Metas são usadas em WSMO para descrever desejos dos usuários/clientes. Elas especificam os objetivos do lado do requerente quando consultam um serviço Web, descrevendo, em alto nível, uma tarefa concreta a ser conseguida;

- **Mediadores:** Os Mediadores são manipuladores de heterogeneidade, isto é, resolvem as combinações possíveis entre recursos que deveriam ser interoperáveis. Heterogeneidade surge naturalmente em ambientes abertos e distribuídos, como é o caso da Web. Nesse sentido, WSMO define o conceito de mediador como um elemento dos seguintes quatro tipos:

  - **ggMediator:** Mediador que liga duas metas. Esta ligação representa a refinação de uma meta origem em uma meta alvo;
  - **ooMediator:** Mediador que importa ontologias e resolve possíveis combinações entre estas;
  - **wgMediator:** Mediador que liga serviços Web a metas, significando que um serviço Web satisfaz (totalmente ou parcialmente) a meta com a qual ele é ligado;
  - **wwMediator:** Mediador que liga dois serviços Web.

Para que WSMO possa implementar de forma real todos estes conceitos, ele faz uso de uma linguagem formal chamada WSML [54]. Na verdade, WSML é um conjunto de linguagens que foram projetadas especificamente para expressar as descrições semânticas do modelo ontológico WSMO. Entre estas linguagens destacam-se [54]:

- **WSML-Core:** É uma linguagem definida pela interseção da Lógica de Descrições (*Description Logic*) e da Lógica de Horn (*Horn Logic*). Esta linguagem é a menos expressiva e permite a modelagem de classes, atributos, relações binárias, instâncias, e as hierarquias de classes e de relações;
- **WSML-Rule:** É uma linguagem que estende WSML-Core e suporta Horn Logic baseada em semânticas de modelos mínimos. Além disso, WSML-Rule captura diversas extensões desenvolvidas na área de programação lógica;
- **WSML-Full:** Unifica todas as variantes de WSML. Esta linguagem é a mais próxima dos conceitos de WSMO. A sintaxe de WSML-Full é definida como base na sintaxe de WSMO.

No entanto, a descrição semântica dos serviços Web, por si só, não resolvem o problema da composição automática de serviços Web. Para compor serviços Web de forma automática, precisa-se também de mecanismos ou técnicas de inferência que raciocinem sobre as descrições semânticas dos serviços Web.

## 2.7 Mecanismos de Inferência

Entre os mecanismos de inferência predominante utilizados para a composição de serviços Web destacam-se as técnicas de planejamento em inteligência artificial, de *Workflows*, de agentes de software e soluções proprietárias (algoritmos específicos) [81] [36]. A abordagem baseada em *Workflows*, aproveita a similaridade entre um *Workflow* e um serviço composto para aplicar o processo de composição [35]. Já o Planejamento, que é uma sub-área da Inteligência Artificial (IA), estuda o processo de deliberação que escolhe e organiza ações, buscando atingir da melhor forma possível os objetivos pré-definidos [33].

Para a implementação de nossa aplicação-exemplo, nós utilizamos como mecanismo de inferência o algoritmos de planejamento (planejador) JSHOP2 [50], que é a implementação JAVA do algoritmo SHOP2 (*Simple Hierarchical Ordered Planner*) baseado na técnica de planejamento hierárquico HTN (*Hierarquical Task Network*) [51]. Nesse sentido, as seguintes

subseções apresentam o fundamento teórico referente ao planejamento, especificamente ao planejamento hierárquico.

### 2.7.1 Planejamento

Planejamento é o processo de compor um plano de ações a serem seguidas para atingir uma meta ou objetivo. Este processo não é trivial, já que no mundo real podemos ter casos nos quais o processo de planejamento pode se mostrar complexo e envolver dezenas, centos ou milhares de variáveis diferentes no processo de escolha de ações [82]. Resumindo, planejamento pode ser definido como uma série de eventos executados para atingir um objetivo através de um processo de escolha de ações prevendo os efeitos da execução dessas ações [13].

Já o planejamento automático é uma área da IA que estuda o processo de planejamento de forma computacional. Uma motivação para que esse estudo seja realizado é puramente prático, já que existe a necessidade de construir ferramentas capazes de fazer as escolhas avaliando milhares de ações para atingir o objetivo de forma automática [83]. A forma que estas escolhas são realizadas descrevem uma técnica de planejamento. Já um planejador é uma ferramenta de software que implementa uma técnica de planejamento. Para construir um planejador que seja capaz de resolver um problema de planejamento, é necessária uma linguagem formal para a representação: das ações (domínio de planejamento); e do estado inicial e meta (problema de planejamento). O domínio de planejamento descreve através de uma sintaxe formal a representação dos estados do mundo em ações dentro de um contexto específico. Já um problema de planejamento descreve um estado inicial do mundo, que é ponto de partida para chegar à meta especificada [83].

Entre as técnicas de planejamento existentes na literatura, nós aprofundamos os nossos estudos sobre: planejamento hierárquico [51] e planejamento hierárquico sob incerteza [84].

### 2.7.2 Planejamento Hierárquico

A técnica de planejamento hierárquico (HTN - *Hierarquical Task Network*) [51] consiste na criação de planos por decomposição de tarefas. Nesse sentido, no planejamento hierárquico o domínio é modelado por tarefas primitivas, que podem ser executas por um agente de software, e tarefas complexas, que não podem ser executadas diretamente pelo agente de software mas que especificam métodos que as decompõem em tarefas primitivas executáveis [13].

No planejamento hierárquico, o conhecimento sobre como sequenciar ações está na especificação de métodos de decomposição de tarefas. Este conhecimento (chamado de

conhecimento de controle dependente de domínio) é fornecido pelo especialista que projeta as tarefas na forma de métodos que especificam como uma tarefa composta pode ser decomposta em uma sequência de tarefas de baixo nível, isto é, ações que podem ser executadas pelo agente de software, sem o risco de existir conflitos entre elas [13].

Uma forma de resolver um problema de planejamento hierárquico é produzir todas as combinações de decomposições hierárquicas possíveis até encontrar alguma que gere uma rede de tarefas primitivas que seja livre de conflitos (método de busca exaustiva). Entretanto, levando-se em conta o tamanho do espaço de busca seria mais apropriado aproveitar-se da estrutura do problema e podar grandes ramos do espaço de busca eliminando métodos, restrições de variáveis e de ordem que levarão a becos sem saída. Tal técnica é chamada de crítica [51], e pode aproveitar características particulares de um domínio ou características gerais sobre interação de ações [13].

Já algoritmo de planejamento (planejador) JSHOP2 [50] é a implementação JAVA do algoritmo SHOP2 (*Simple Hierarchical Ordered Planner*) baseado na técnica de planejamento hierárquico. JSHOP2 utiliza uma notação baseada em LISP para representar o domínio e o problema de planejamento. O domínio é composto de um conjunto de operadores, métodos e axiomas. Os problemas de planejamento são compostos de um conjunto de átomos lógicos para representar o estado inicial e uma lista de tarefas a serem atingidas. De forma geral os componentes de um domínio e um problema de planejamento são expressões lógicas combinadas com átomos lógicos utilizados em forma de símbolos predicados e listas de termos.

## 2.8 Composição de Serviços Web

Os serviços Web têm atraído a atenção da indústria da computação a nível mundial devido a sua capacidade de solucionar problemas de reuso e interoperabilidade. Já a combinação de diversos serviços Web para atingir uma determinada funcionalidade é denominada Composição de Serviços Web e pode ser feita por um programador de forma manual ou por algoritmos computacionais de forma automática. Fazer esta combinação de forma manual gera um trabalho complexo e propenso a erros para qualquer pessoa, devido ao dinamismo, flexibilidade e escalabilidade da Web. Fato que tem levado à aparição de um considerável número de esforços de pesquisa, tanto no nível da indústria quanto da academia, para a "Composição Automática de Serviços Web".

Define-se a composição automática de serviços Web como o processo computacional que tem a habilidade de procurar, selecionar e encadear em uma ordem especifica os mais adequados

e mais relevantes serviços disponíveis para criar uma novo serviço (de valor agregado) que satisfaça a requisição do usuário em um tempo adequado [31]. Nesse sentido, existem várias abordagens tecnologias utilizadas para implementar esse processo computacional, entre elas destacam-se algoritmos de planejamento e *Workflows*.

Para entender e exemplificar o processo de composição de serviços Web, suponha que um usuário queira obter na Web o preço de um determinado livro em reais. Assuma que não existe nenhum serviço que seja capaz de devolver o preço deste livro em reais, mas que existe um serviço que pode devolver o preço em uma outra moeda (p. ex. em dólares) e outro serviço que faça conversão dessa moeda intermediária para o valor em reais. Invocando esses serviços na sequência correta, o usuário pode obter o resultado desejado independente da ausência de um serviço único que execute essa tarefa diretamente. Dessa maneira, um consumidor pode usufruir da combinação de serviços dependendo da disponibilidade, qualidade, preço e outros fatores [13]. No contexto da Web, o exemplo descrito pode ser caracterizado como um processo que envolve três etapas [13]:

1. Descoberta automática: significa localizar serviços que ofereçam o livro especificado. Localizar estes serviços pode ser um processo demorado desde que o espaço de busca destes serviços signifique procurar em toda a Web. Para evitar isso, o mecanismo de busca se deve restringir o domínio de aplicação, neste caso, poderíamos especificar como domínio de aplicação só os serviços de compra de livros de sites de livrarias;

2. Composição automática: seleciona e encadeia em uma que sequência especifica os serviços são uteis para atingir o objetivo (meta);

3. Execução automática: invocação (execução) dos serviços Web definidos no processo de composição.

### 2.8.1 Composição de Serviços Web: Classificação

Já na literatura existe uma grande discussão sobre a classificação que pode ser atribuída a um sistema de composição, isto é, alguns autores discutem sobre se a composição é automática ou manual, dinâmica ou estática. Nesse sentido, em [85] classifica-se a composição de serviços Web em cinco categorias:

- Composição automática e manual (*automatic and manual composition*), acontece em tempo de projeto, isto é, quando os serviços são escolhidos para compor o plano solução. Esta escolha de forma manual é feita por uma pessoa (usuário e/ou desenvolvedor), a qual

é propensa a erros e útil só para ambientes com pouca quantidade de serviços disponíveis. Já a composição automática específica um processo de escolha de serviços para o plano solução de forma programática utilizando algoritmos computacionais e está diretamente relacionada ao uso de ontologias, para a descrição semântica dos serviços e à utilização de mecanismos de inferência para a implementação programática da escolha dos serviços que compõem o plano solução;

- Composição estática e dinâmica (*static and dynamic service composition*), corresponde à capacidade do sistema, em tempo de execução, de poder substituir ou recompor o plano solução quando algum serviço não estiver disponível (p.ex. URL inválida), isto é, o sistema é dinâmico quando serviços podem re-selecionados ou re-descobertos para recompor o plano solução em base às mudanças no ambiente e com a mínima intervenção do usuário. Caso o sistema não possua esta capacidade considera-se este como estático.

- Composição orientada por modelos (model-driven service composition), baseia-se na composição dinâmica de serviços e corresponde à geração de modelos para descrever requisições do usuário, regras de negócios e fluxo de processos a nível abstrato. Regras de negócio podem ser utilizadas para estruturar e programar processo de composição, e para descrever a seleção de serviços e sea ligação (*binding*) a nível de execução. Estes modelos são gerados utilizando a linguagem UML (*Unified Modelling Language*) e são geralmente implementados utilizando *Workflows*;

- Composição declarativa (declarative service composition), nesta abordagem as requisições do usuário são expressas de forma declarativa utilizando linguagens formais. A abordagem declarativa consiste em duas fases: a primeira fase descreve um estado inicial e a meta desejada como ponto de partida, e constrói planos genéricos para atingir dita meta. A segunda fase escolhe um plano genérico, descobre os serviços adequados, constrói o fluxo de trabalho (*Workflows*). A primeira fase é implementada utilizando a linguagem PDDL (*Planning Domain Definition Language*). A segunda fase pode ser implementada utilizando linguagens de modelação de processo, tais como BPEL (*Business Process Execution Language*);

- Composição baseada em Contexto (*context-based service composition*), baseia-se no principio que um mesmo serviço pode ser disponibilizado através de diferentes canais e consumidos por diferentes dispositivos (p.ex. PCs, telefones celulares ou aparelhos de televisão). Nesse sentido, informações lidam com questões de qualidade de serviço (*Quality of Service* - QoS) são utilizadas na negociação da seleção de serviços

considerando as exigências de usuário, as características dos canais, e as restrições do provedor.

Já do ponto de vista de padrão arquitetural de execução a composição de serviços Web pode ser classificada em orquestração e coreografia de serviços, estes termos descrevem dois aspectos sobre como o processo composição é executado:

- Orquestração (*Orchestration*), é um modelo de processos executáveis que descreve o fluxo de execução (ordem e condições em que os serviços Web são invocados) em base a um coordenador (centralizado ou distribuído) que pode interagir com os serviços Web internos e externos. Modelos de orquestração podem ser implementados utilizando diagramas de atividades em UML, redes de Petri, orquestração baseada em regras, hierarquias de atividade, e cálculo-Pi. Dados e modelos de acesso a dados definem como os dados são especificados e trocados entre os serviços. O modelo de seleção de serviços lida com ligação (*binding*) estática e dinâmica (capacidade de substituir um serviço em tempo de execução). O modelo para tratamento de excepções lida com estados de excepção durante a execução de um serviço [85];

- Coreografia (*Choreography*), é um modelo de interação que descreve a colaboração que cada participante (serviço) terá no processo de composição. Esta colaboração é implementada através da troca de informações (mensagens) entre os serviços para definir as regras (sequência e condições) de interação. A linguagem WS-CDL (*Web Services Choreography Description Language*) permite descrever esta colaboração de forma clara e sem ambiguidades [86].

Resumindo, o foco da coreografia é a interação entre participantes e não a execução de serviços. Já o modelo de orquestração é um processo executável focado no fluxo de controle e a ordem de execução. A coreografia também permite descrever a colaboração e interação entre modelos de orquestração como representado pela Figura 2.7. O modelo ontológico WSMO provê suporte para orquestração e coreografia através da interface dos serviços Web.

No entanto, em [87, 88] indica que as classificações descritas em [85] para sistemas de composição de serviços Web são incompletas e ambíguas. Nesse sentido, em [87] é apresentada uma matriz de duas dimensões: tecnologia e ambiente. A dimensão de ambiente descreve atributos sobre Padrão arquitetural (orquestração e coreografia), Semiótica (semântica e sintaxe), Mecanismo de protocolo (SOAP e REST), Composição em tempo de design (manual, semi-automático e automático) e Recomposição em tempo de execução (estático e dinâmico).

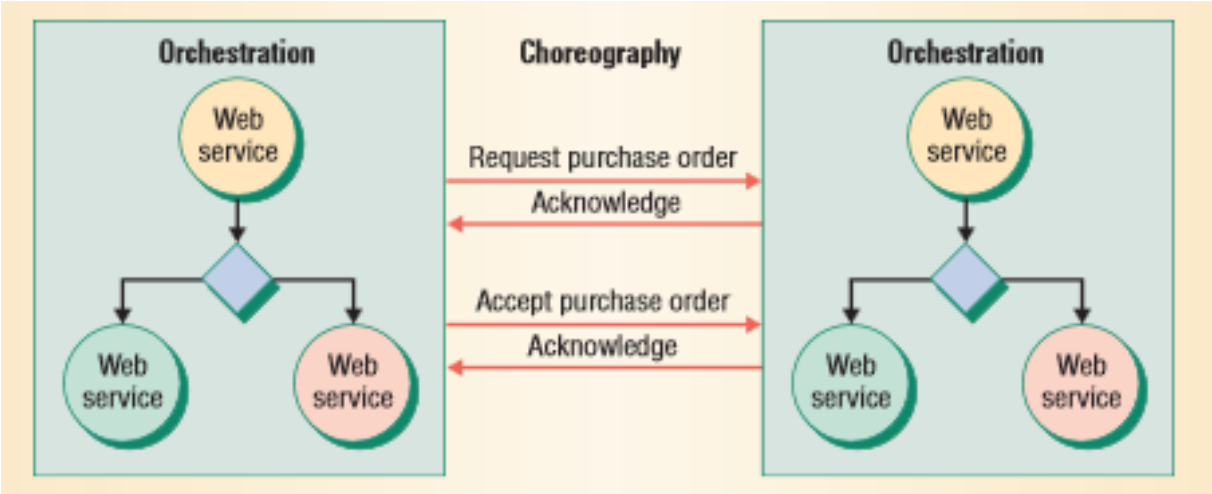


**Figura 2.7:** Orquestração e Coreografia - [7]

Já a dimensão sobre tecnologia específica os atributos de Processo (*One-Stop*, *Bridge* e *Double-Bridge*), Tipo de tecnologia (*Workflow*, orientado a modelos e algoritmos de planejamento) e Técnica da tecnologia (detalha o algoritmo ou linguagem de modelagem utilizada).

Sobre o atributo Processo da dimensão Tecnológica podemos dizer: *One-Stop* refere-se ao processo de composição que segue um único fluxo, isto é, a especificação da requisição, o processo de composição e a execução dos serviços são executados sem precisar nenhum processo externo, um exemplo deste tipo de processo é a composição de serviços baseados em *Workflows*; o processo de *Bridge* refere-se à necessidade de existir um processo de adequação do plano gerado pelo compositor para poder executar os serviços definidos no plano; e o processo de *Double-Bridge* refere-se à necessidade de executar um processo de tradução da especificação da requisição para o processo de composição. Em base a esta matriz podemos dizer que um sistema de composição de serviços Web pode ser automático e a sua vez dinâmico. Além disso, a matriz permite caracterizar um sistema de composição de serviços por cada um dos atributos definidos.

### 2.8.2 Composição de Serviços Web: Arquiteturas

Já em [8] apresenta-se um dos primeiros sistemas para composição automática de serviços Web. A estrutura geral do sistema de composição de serviços encontra-se ilustrado na Figura 2.8. Este sistema apresenta dois tipos de participantes: fornecedores e consumidores de serviços. Os fornecedores oferecem serviços na Web para sua livre utilização. Os consumidores que especificam uma requisição que deve ser satisfeita no consumo dos serviços oferecidos pelos fornecedores. O sistema também contém os seguintes componentes: tradutor, gerador de

processo, avaliador, executor e repositório de serviço. O tradutor traduz entre as línguas externas utilizadas pelos participantes e as línguas internas utilizadas por os componentes do sistema. Para cada requisição, o gerador de processos tenta gerar um plano composto por os serviços disponíveis no repositório de serviços para satisfazer a requisição. Se mais de um plano for gerado, o avaliador avalia todos os planos e define qual é o melhor para sua execução. O executor chama os serviços que compõem o plano e devolve o resultado para o fornecedor de serviços.

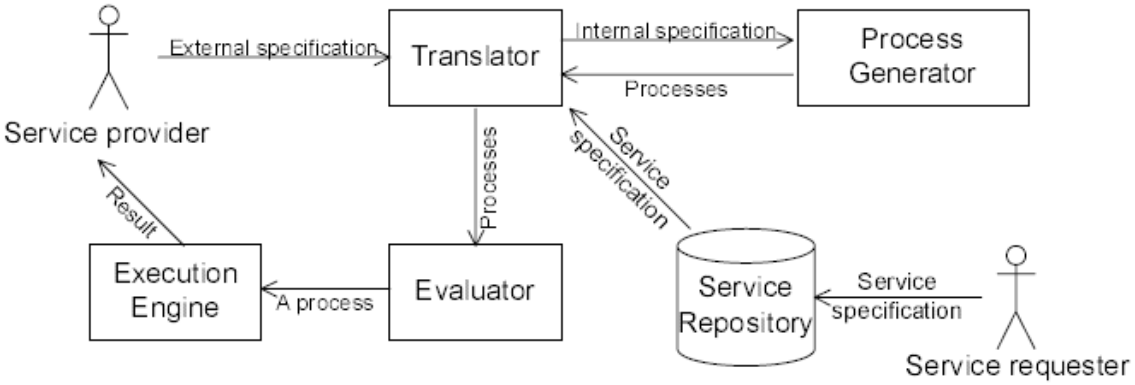


**Figura 2.8:** Sistema automático de composição de serviços - [8]

Em [9] apresenta-se a arquitetura SPOC (*Semantic based Planning for Optimal web services Composition*) para compor serviços Web, como mostrado na Figura 2.9.

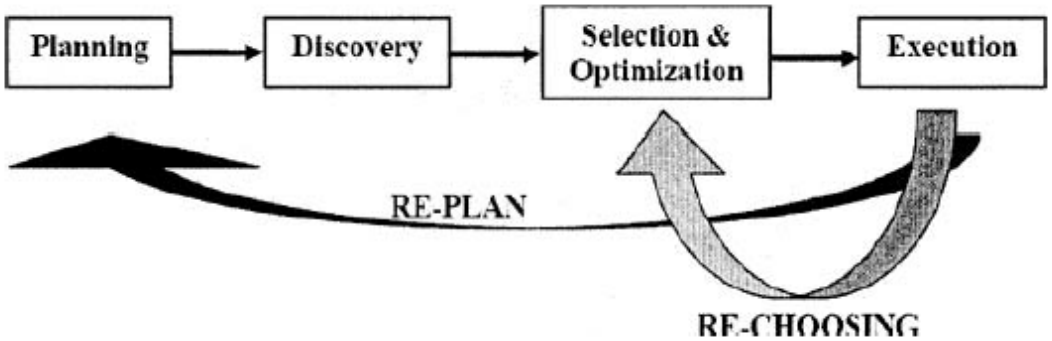


**Figura 2.9:** Arquitetura SPOC (*Semantic based Planning for Optimal web services Composition*) - [9]

Nesta arquitetura reduze-se o problema da composição de serviços Web para quatro fases fundamentais: a primeira é o Planejamento, que determina a ordem de execução das tarefas que compõem o plano que satisfaz a requisição, considera-se aqui uma tarefa como uma funcionalidade ou uma atividade do serviço. A segunda é a Descoberta, que visa encontrar serviços candidatos para cada tarefa no plano. A terceira fase visa a Seleção e Otimização do

conjunto de serviços candidatos para cada tarefa, esta seleção e otimização acontece em base a quatro critérios como parâmetros de qualidade: custo, tempo, disponibilidade e reputação. Cada um dos serviços candidatos receberão um valor para representar esses critérios de qualidade. A quarta fase de Execução é caracterizada como um problema, isto porque, durante o processo de execução se algum serviço não estiver disponível (p.ex. URL inválida) o sistema deve propor outra solução a ser executada o que corresponde à seta de re-escolha. Já se nenhuma outra solução for proposta, o sistema deve solicitar a construção de um outro plano (p.ex. reordenando as tarefas) o que corresponde à seta de re-planejamento.

Em base aos trabalhos [8], [9] e [89] nós projetamos e publicamos uma arquitetura para a composição automática de serviços web [10] a qual é apresentada na Figura 2.10 e que é composta pelos módulos de interface, de descoberta, de tradução, de composição, de avaliação e de execução. De forma geral, nossa arquitetura segue uma sequência de eventos que começa quando:

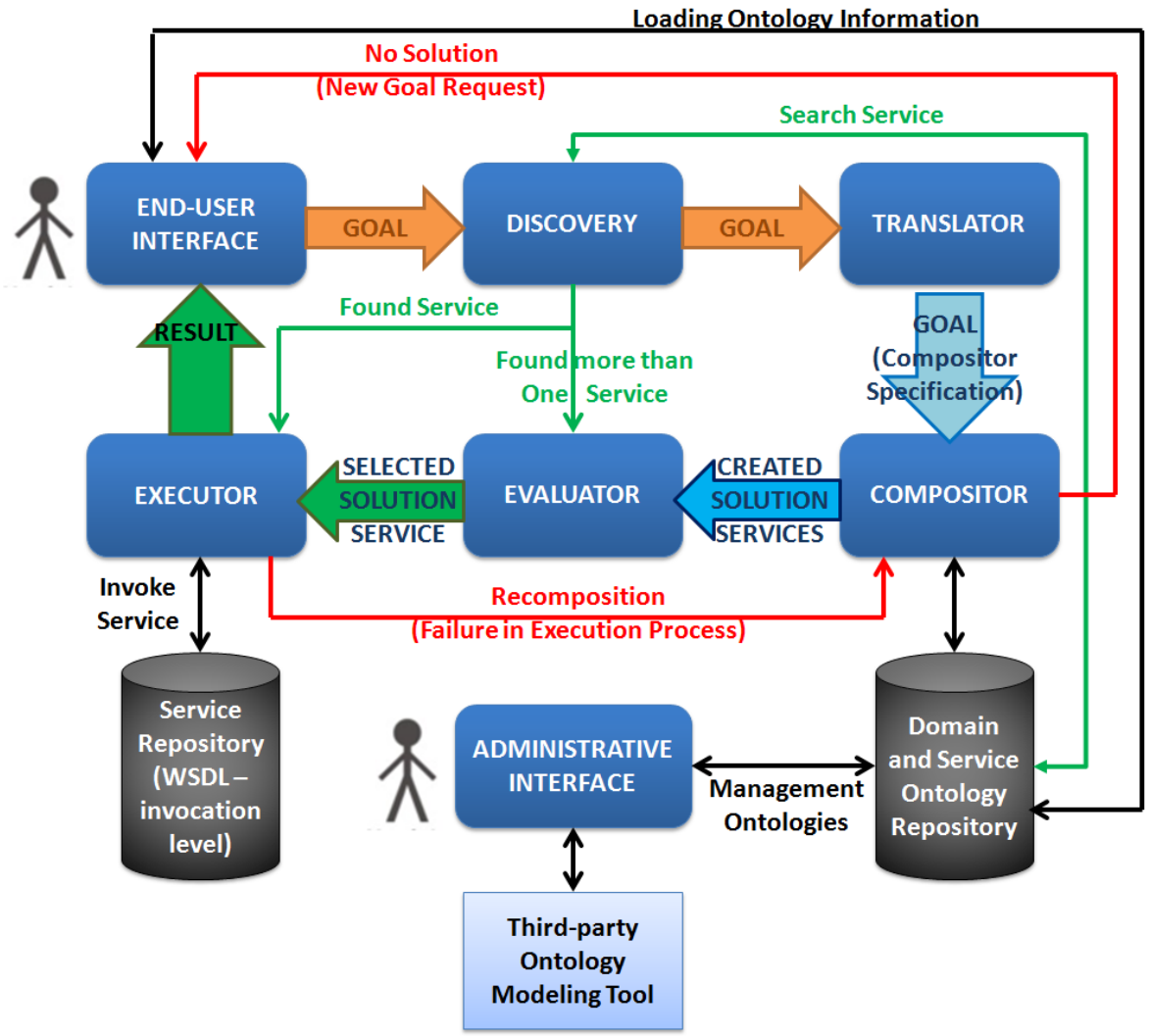


**Figura 2.10:** Arquitetura de Composição Automática de Serviços Web - [10]

- Na interface de usuário são descritas as requisições do usuário como uma meta (objetivo a ser atingido). Dependendo da meta especificada precisa-se preencher algumas informações relevantes para a criação do estado inicial, isto é, refinamento da meta através da especificação do domínio de aplicação e algumas restrições ou condições desejadas. Este módulo utiliza as ontologias do repositório para especificar o domínio de aplicação;

- O módulo de descoberta interage com a nossa base de conhecimento composta pelas ontologias de domínio e ontologias de serviços em busca de um serviço que satisfaça a meta especificada pelo usuário. Quando nenhum serviço é encontrado, o processo de composição é acionado;

- Como geralmente a linguagem em que as metas, serviços e ontologias são especificados é distinta à linguagem utilizada pelos mecanismos de inferência, o módulo de tradução permite modelar o domínio de planejamento (serviços Web associados ao domínio escolhido) e problema de planejamento (meta especificada e informações preenchidas para o estado inicial) na linguagem utilizada pelo compositor;

- Neste caso, o módulo compositor implementa um algoritmo de planejamento, o que lhe permite raciocinar sobre o domínio e o problema de planejamento, gerando um ou vários planos solução. Pode acontecer que nenhum plano solução seja gerado, nesse caso é solicitada uma nova especificação da meta;

- O módulo de avaliação analisa as propriedades não funcionais associadas aos serviços Web semânticos descritos no plano, e em função a essa análise determina qual é o melhor plano a ser executado;

- Já o módulo executor lê (interpreta) o plano e inicia uma sequência de execução dos serviços Web. Pode acontecer que em tempo de execução, algum dos serviços especificados no plano fique indisponível, nesse caso um novo processo de composição ou a utilização de um plano alternativo é solicitado;

- Finalmente, o módulo de interface administrativa permite gerenciar (importar ou retirar) ontologias, serviços, algoritmos de planejamento e definição de propriedades não funcionais. A criação de ontologias e serviços é feita utilizando ferramentas externas como o *WSMO Studio* [90].

Em base à matriz apresentada em [87, 88] e descrita anteriormente, nós podemos classificar a arquitetura proposta como:

- Orquestrada, isto porque o módulo executor centraliza o processo de como devem ser executados os serviços;
- Semântica, já que a especificação das requisições, a busca e composição de serviços utilizam descrições semânticas para implementação de ditos processos;
- Baseada em SOAP, devido a que a implementação do serviços a nível de execução utilizam este protocolo;
- Automática, isto porque a busca, seleção, combinação e execução de serviços é feita por mecanismos computacionais;
- Dinâmica, já que a arquitetura proposta prevê a necessidade de ter suporte para recompor planos em tempo de execução;
- *Double-Bridge*, devido à necessidade de fazer transformações: entre a especificação do usuário e o módulo de composição e entre o plano gerado e mecanismo de execução de serviços;
- Baseada em Planejamento, isto porque o módulo de tradução foi projetado para gerar um domínio e problema de planejamento;
- Na implementação do protótipo da arquitetura proposta foi utilizado o planejador ND-HSHOP2, que é uma implementação na linguagem Haskell [91] do planejador SHOP2 não-determinístico proposto em [92] e implementado em [55].

### 2.8.3 Composição de Serviços Web: *Grounding*

O *Grounding* é definido como o mecanismo que detalha o mapeamento de acesso de um serviço Web semântico (descrito utilizando alguma linguagem ontológica como WSMO ou OWL-S) para um serviço Web a nível de invocação (descrito utilizando alguma linguagem de serviços como WSDL ou WADL) [93]. Este mapeamento é necessário porque sistemas e aplicações Web, atualmente, só implementam a nível de invocação serviços Web e não serviços Web Semânticos. Já o mecanismo de *Grounding* em WSMO permite conectar o serviço descrito semanticamente com o serviço descrito sintaticamente de duas formas [11]:

- *Data Grounding*: A nível sintáctico, serviços Web se comunicam com seus clientes trocando mensagens XML descritos em XML Schema [74]. Já a nível semântico, as entradas e saídas do serviço são descritos utilizando ontologias. Nesse sentido, ao trocar

informações entre estes níveis precisa-se a transformação dos dados contidos na ontologia para um formato XML (*lowering mapping*) e vice-versa (*lifting mapping*). Como a semântica descrita em um arquivo XML é apenas implícita (nome das tags descritas em texto simples), precisa-se de designer humano para especificar a transformação de dados do tipo *lifting*;

- *Choreography Grounding*: A coreografia permite ao cliente saber que mensagens podem ser enviadas e/ou recebidas em qualquer ponto específico durante a interação com um serviço. Em WSMO, a coreografia é modelada como uma máquina de estado abstrata, onde os estados são descritos com uma ontologia, e o cliente obtêm permissão para ler e escrever instâncias de certos conceitos que foram marcados como "*in*", "*out*" ou "*shared*" (que representa tanto *in* como *out*). Nesse sentido, quando criada a coreografia os conceitos marcados como "*in*" e "*shared*" são associados com os mensagens de entrada do serviço e os conceitos marcados como "*out*" e "*shared*" são associados com os mensagens de saída do serviço;

A Figura 2.11 descreve as conexões entre as descrições semânticas em WSMO e as descrições sintáticas em WSDL de Serviços Web. Especificamente o atributo interface do serviço Web semântico associa-se às operações do serviço Web a nível sintático. Já a ontologia que descreve os conceitos utilizados pelo serviço Web semântico também é utilizada pelo XML Schema para fazer a transformação de dados para mensagens em formato XML.

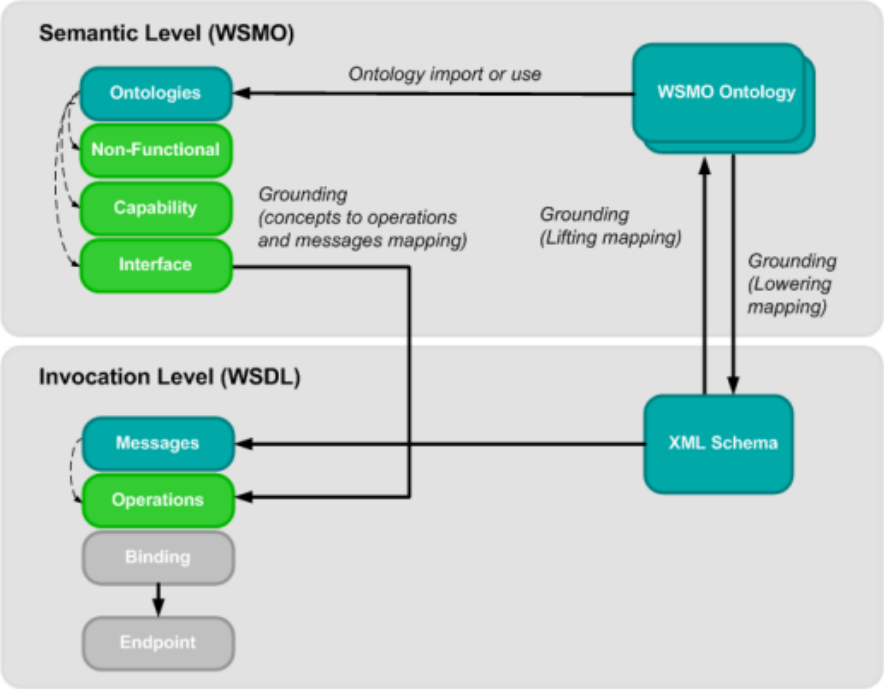


**Figura 2.11:** Grounding WSMO - Adaptado de [11]

### 2.8.4 Sistemas de Composição de Serviços Web

Com o intuito de adquirir conhecimentos práticos sobre a composição de serviços Web para especificar nosso modelo de referência, foram estudados e testados (na medida do possível) três sistemas de composição automática de serviços Web descritos a seguir:

OWLSXPlan [12] é um sistema de composição de serviços Web que pode ser considerado de planejamento, de processo *Bridge* e dinâmico, isto porque, foi construído em base a um planejador do tipo *Fast Forward* e um componente HTN. Além disso, este sistema inclui um componente chamado OWSL2PDDL, que faz o mapeamento entre a linguagem OWL-S e o PDDL (linguagem compreendia pelos planejadores). Quando um plano solução é interrompido na sua execução devido a algum tipo de falha, o sistema permite gerir um novo plano, alternando desta forma o processo de composição e execução. Porém, OWLSXPlan não funciona bem em domínios parcialmente observáveis e não é uma solução eficiente em termos de tempo. Também não trata não-determinismo. Nesse sentido, serão aproveitados os conceitos sobre o OWSL2PDDL, para nós podermos construir um componente que faça o mapeamento entre o WSMO (serviços) e a sintaxe do planejador HTN a ser implementado. Já a Figura 2.12 apresenta a interface deste sistema, onde pode-se observar que a definição do problema ou requisição do usuário requer de conhecimentos técnicos.

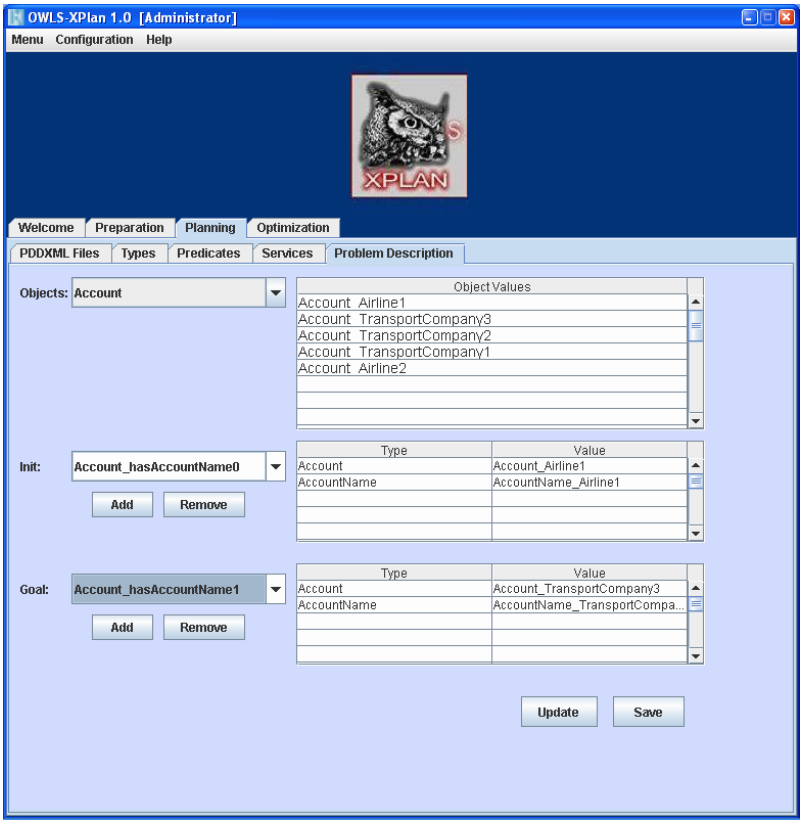


**Figura 2.12:** Interface do OWLSXPlan - [12]

Já em [13] e [14] apresentam-se duas ferramentas para a composição automática de serviços Web. Na Figura 2.13 apresenta-se a interface da ferramenta desenvolvida em [13] chamada de WEBRPLAN. Esta ferramenta não se encontra disponível para sua utilização. Nesse sentido, a análise desta ferramenta foi feita da dissertação de mestrado [13]. Desta análise pode-se deduzir que a interface inicial permite carregar os serviços Web semânticos descritos na linguagem OWL-S que determinam o domínio de aplicação a ser utilizado. Logo, a interface permite a tradução destes serviços Web para um domínio de planejamento na sintaxe do planejador JShop2, para finalmente, apresentar uma interface onde o usuário pode escolher a tarefa desejada e ingressar certas informações que representam o estado inicial e final.

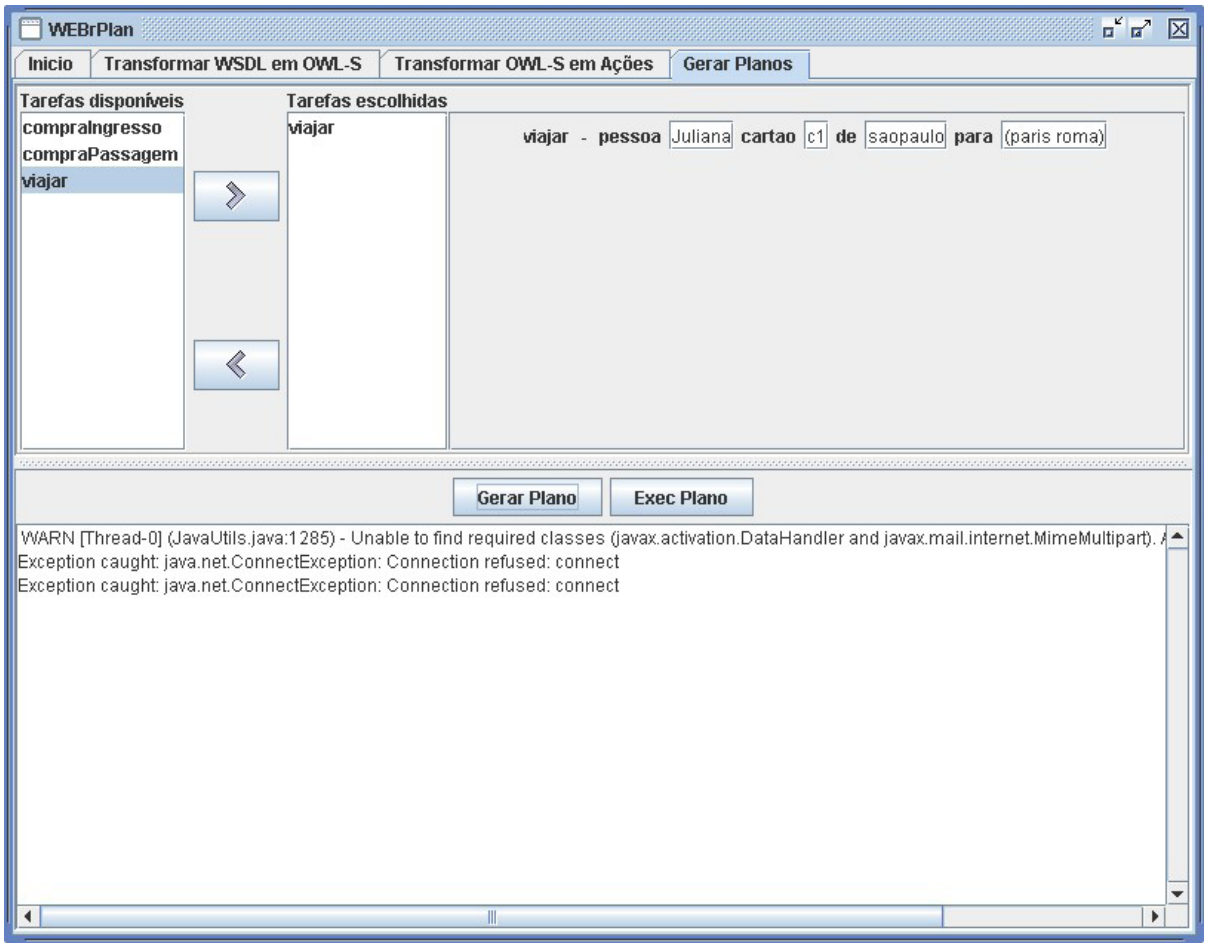


**Figura 2.13:** Interface do WEBRPLAN - [13]

Já no trabalho apresentado em [14] descreve-se a ferramenta TRANSPLAN. Esta ferramenta encontra-se disponível para sua experimentação[1]. Desta experimentação observou-se que a interface do TRANSPLAN, apresentada na Figura 2.14, segue o seguinte fluxo: Carregam-se os serviços Web semânticos descritos em OWL-S. Este carregamento, aparentemente, faz a tradução de OWL-S para a sintaxe do planejador JShop2. Estes serviços

[1] SourceForge - http://pt.sourceforge.jp/projects/sfnettransplan/

são apresentados como operações disponíveis na interface. Logo, a interface permite definir o estado inicial e a meta desejada para começar o processo de planejamento.

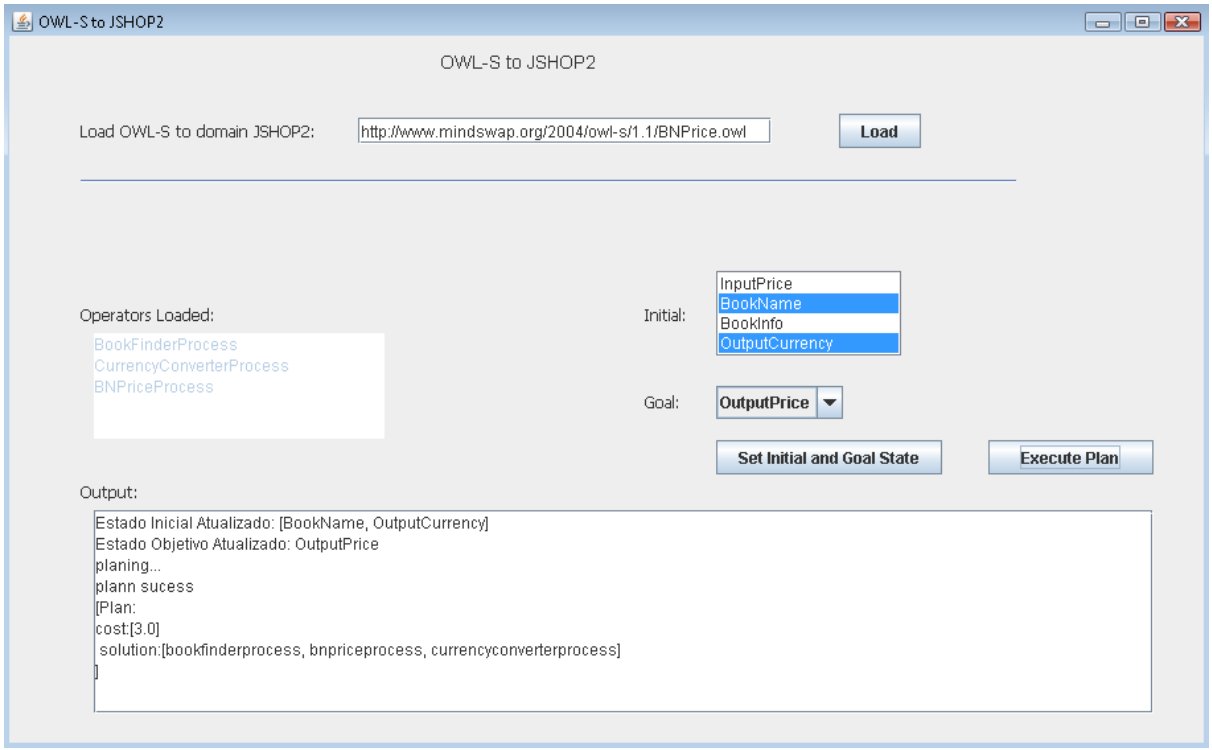


**Figura 2.14:** Interface do Transplan - [14]

O sistema AWSC - *Automated Web Service Composer* [15] é um sistema projetado para operar com problemas de composição do mundo real, tais como observação parcial do ambiente, efeitos não-determinísticos e recuperação de falhas. O algoritmo de planejamento deste sistema baseia-se no planejador Simplanner [94], o qual utiliza decomposição de metas e busca com heurística obtidas do *Relaxed Plan Graph* (RPG). Uma das características mais importantes do AWSC é sua capacidade de resposta em tempo real. Além disso, este sistema utiliza o modelo ontológico OWL-S para modelar os serviços. Esta ferramenta não foi encontrada para sua experimentação.

Já Figura 2.15 representa a interface do sistema AWSC, da qual podemos deduzir que o resultado da execução do processo de composição é apresentado do lado esquerdo da figura 2.15, já do lado superior direito é apresentada uma representação de grafos para o plano solução gerado e na parte inferior direita apresenta-se o estado detalhado dos serviços que fizera parte do plano assim como uma janela com os dados da requisição do usuário. A descrição da requisição do usuário é feita de forma manual utilizando a linguagem OWL e que é convertida pelo sistema para a linguagem PDDL entendida pelo planejador.

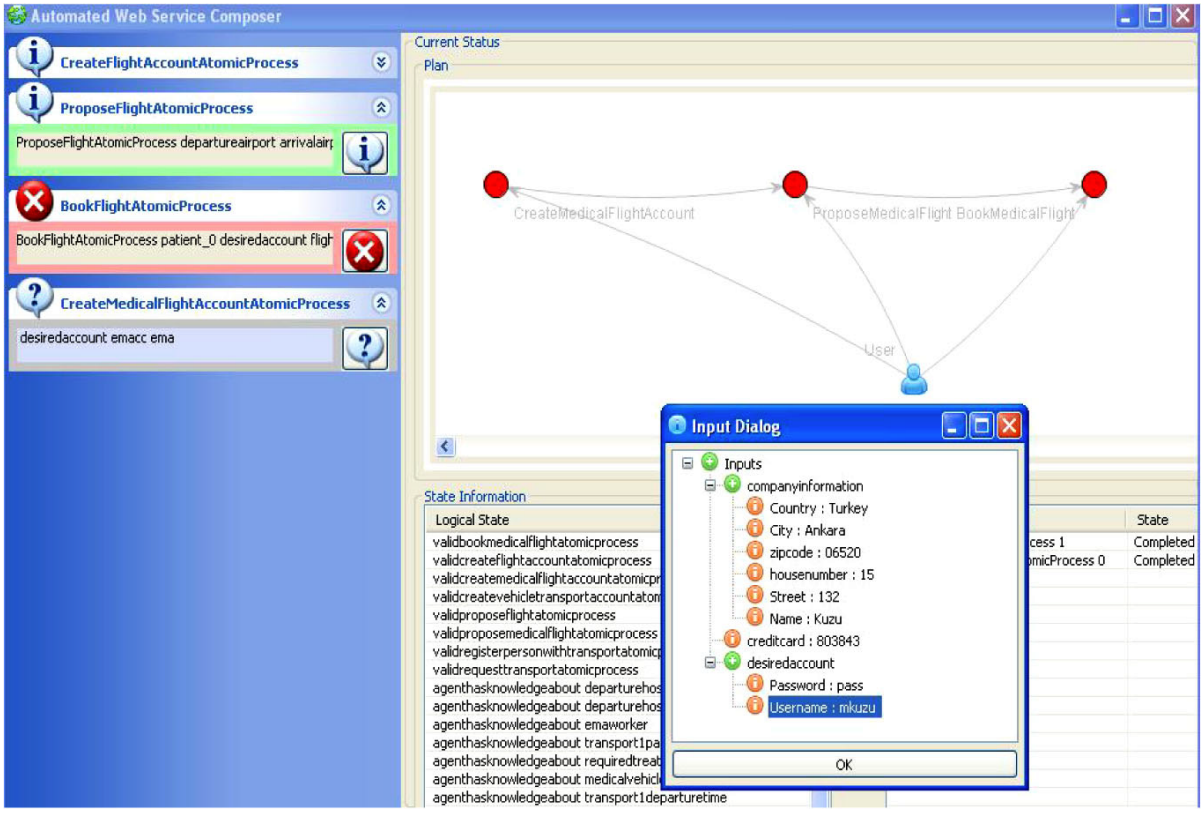


**Figura 2.15:** Interface do *Automated Web Service Composer* - [15]

Note-se que nos sistemas apresentados, as interfaces projetadas seguem a sequência de carregar os serviços Web semânticos e as ontologias (que representam o domínio da aplicação), para logo escolher uma meta e preencher algumas informações que componham o estado inicial do problema. Mesmo que estes sistemas não tem uma aplicação prática para ambientes pervasivos, nós podemos concluir que existe a necessidade que sistemas de composição de serviços Web projetem interfaces que permitam especificar requisições em tempo real e que estas interfaces sejam muito menos técnicas e mais simples e intuitivas.

# 3 Estado da Arte e Trabalhos Relacionados

Neste capítulo apresenta-se o estado da arte sobre composição automática de serviços em ambientes pervasivos e principais trabalhos relacionados sobre especificação formal de sistemas de composição de serviços para ambientes pervasivos.

## 3.1 Estado da Arte dos Sistemas de Composição de Serviços para Ambientes Pervasivos

Como compor serviços Web de forma automática no âmbito dos ambientes pervasivos precisa considerar características como [37]:

1. Ter a capacidade de recuperação a falhas devido à indisponibilidade temporal das funcionalidades do ambiente por diversos motivos imprevisíveis [38] [39];
2. Utilizar a relevância das informações contextuais presentes na interação com os usuários, como uma forma de otimizar e até personalizar o processo de composição [40];
3. Permitir agregar e retirar (do ambiente) de forma simples (*plug and play*) novos dispositivos (por exemplo: geladeiras, cafeteiras, fornos, etc.) com as mais diversas funcionalidades [41];
4. Gerenciar a heterogeneidade e complexidade dos distintos tipos de dispositivos e aplicações, e adaptar-se aos distintos cenários presentes nos ambiente pervasivo [42] [43];
5. Garantir a segurança, o controle de acesso e a confiabilidade das funcionalidades providas pelo ambiente [44] [45].

Nos últimos anos, diversos trabalhos têm explorado o processo de composição automática de serviços Web em ambientes pervasivos. Nos artigos [26], [37] e [46] são mapeadas e

apresentadas uma serie de propostas sobre sistemas, *middlewares* e *frameworks* que, em algum grau, abordam parcial ou totalmente as características anteriormente descritas para processo de composição de serviços para ambientes pervasivos. Nestes artigos, em alguns casos os ambientes pervasivos também são chamados de *Smart Spaces* e/ou Ambientes Inteligentes (*Ambient Intelligence* - AmI).

Em base à classificação feita em [87, 88] a qual foi descrita na Subseção 2.8.1, e aos critérios de classificação utilizados nos artigos [26], [37] e [46], foi desenvolvida por nós uma nova matriz de classificação de cinco dimensões: Dimensão de Especificação Geral, Dimensão de Especificação do Usuário, Dimensão de Especificação do Ambiente, Dimensão de Especificação dos Dispositivos, Dimensão de Composição, e Dimensão de Execução. Cada uma destas fases possui as seguintes dimensões:

- Dimensão de Especificação Geral: Agrupa os atributos referentes à especificação geral do sistema de composição.
  - Descrição dos Serviços: Este atributo especifica se os atributos funcionais e não-funcionais dos serviços foram descritos sintática ou semanticamente. A descrição semântica pode ser utilizada posteriormente nos processos de descoberta e composição;
  - Linguagem de Descrição dos Serviços: Este atributo detalha a linguagem utilizada na descrição dos serviços, por exemplo: WSDL, DAML OIL, OWL-S, WSML (WSMO);
  - Contexto: Este atributo indica se o processo de composição leva em conta informações de contexto, por exemplo, localização, preferencias de usuário, capacidades dos dispositivos, etc;
  - Qualidade de Serviço (QoS): Este atributo especifica se o processo de composição leva em conta (durante a criação do plano) informações de qualidade, por exemplo, latência, uso de memória, etc;
  - Segurança: Este atributo indica se o sistema de composição considera regras de segurança, por exemplo, identificação de usuário, controle de acesso, etc;
- Dimensão de Especificação do Usuário: Agrupa os atributos referentes à especificação do usurário e da meta ou requisição definida pelo usuário.
  - Tipo de Usuário: Este atributo indica se o sistema de composição é orientado para a utilização de usuários finais sem conhecimento técnico ou para usuário do tipo

desenvolvedor que tem conhecimento técnico sobre o ambiente e sobre o domínio de aplicação;

  - Criação da Meta: Este atributo indica como o usuário especifica sua requisição (meta), por exemplo, interagindo com algum tipo de interface, especificando um arquivo de configuração, etc;
  - Granularidade da Meta: Este atributo indica se a descrição da meta é implícita ou explicita, por exemplo, encontrar a impressora mais próxima (implícita), encontrar uma impressora da marca Hp (explicita);
  - Formalização da Meta: Este atributo especifica o tipo de estrutura utilizada para a descrição formal da meta, por exemplo, Goal (WSMO), serviço atômico (OWL-S), Workflow, etc;

- Dimensão de Especificação do Ambiente: Agrupa os atributos referentes ao ambiente onde se executa o processo de composição.
  - Tipo de Ambiente: Este atributo indica a classificação do tipo de ambiente físico no qual se executa o processo de composição, por exemplo, ambiente hospitalar, fabrica, casa, etc.
  - Topologia do Ambiente: Este atributo especifica se o ambiente permite a inclusão ou exclusão de dispositivos de forma dinâmica na topologia do ambiente, isto é, considera-se fixa quando não é permitido este dinamismo, e ad-hoc quando novos dispositivos podem ser agregados ao ambiente;
  - Domínio da Aplicação: Este atributo indica os domínios de aplicação que agrupam os serviços disponíveis para um ambiente, por exemplo, entretenimento, comunicação, compras, cuidados com a saúde, etc;
  - Rol dos Serviços: Este atributo detalha o tipo de operações que os serviços implementam em função ao domínio ao qual pertencem, por exemplo, busca de restaurantes, localização por GPS, exames médicos, etc.

- Dimensão de Especificação dos Dispositivos: Agrupa os atributos referentes aos dispositivos presentes no ambiente onde se executa o processo de composição e a implementação feita para validar o sistema proposto.
  - Dispositivo de Interação com o Usuário: Este atributo indica o dispositivo que permite ao usuário especificar suas requisições, por exemplo, dispositivo móvel (*smartphone*), central de monitoramento (*gateway*), dispositivos de entretenimento, etc;

  - Dispositivos presentes no Ambiente: Este atributo permite descrever os tipos de dispositivos presentes nos ambientes que disponibilizam suas funcionalidades como serviços, por exemplo, sensores, atuadores, eletro-eletrônicos, RFIDs, etc;
  - Mínimo Dispositivo que Executa o Processo de Composição: Este atributo indica o requirimento de hardware mínimo para executar o sistema de composição, por exemplo, servidor, PC, PDA, etc;
  - Avaliação da Proposta: Este atributo especifica como o sistema de composição foi avaliado, isto é, com a implementação de uma aplicação a nível de simulação, implementação pratica com uso de dispositivos, etc.

- Dimensão de Composição: Agrupa os atributos referentes ao processo de composição propriamente dito.
  - Método de Descoberta de Serviços: Este atributo indica o mecanismo utilizado para a busca dos serviços disponíveis no repositório, por exemplo, UPnP, WS-discovery, Jini, UDDI, etc;
  - Mecanismo de Inferência: Este atributo especifica o tipo de mecanismo de inferência que raciocinem sobre as descrições dos serviços Web, por exemplo, planejamento, *Workflows*, *Model Driven*, agentes, etc;
  - Linguagem do Mecanismo de Inferência: Este atributo detalha a linguagem utilizada pelo tipo de mecanismo de inferência utilizado, por exemplo, PDDL, BPEL, etc;
  - Automação do Processo de Composição: Este atributo especifica se o processo de escolha e seleção de serviços para compor o plano solução é feita de forma manual ou semi-automática ou automática;
  - Representação do Plano Solução: Este atributo indica a linguagem utilizada para descrever o plano solução gerado pelo processo de composição, por exemplo, OWL-S, BPEL, XML, coreografia executável, representação interna, etc.

- Dimensão de Execução: Agrupa os atributos referentes ao processo de execução do plano gerado.
  - Fluxo de Processamento: Este atributo especifica se o sistema de composição segue um único fluxo (direto), isto é, a especificação da requisição, o processo de composição e a execução dos serviços utilizam uma mesma linguagem de representação. Caso se precise de um ou mais processos de tradução entre as linguagens que que representam a especificação da requisição, o processo de

composição e a execução dos serviços, estes são denominados de ponte e dupla ponte, respetivamente;

- Padrão de Execução: Este atributo especifica se o controle ou gestão do processo de execução é feita de forma centralizada (orquestração) ou distribuída (coreografia);
- Tipo de Execução: Este atributo especifica se a execução (chamada) dos serviços a nível de invocação é estática ou dinâmica (suporta replanejamento em caso de falha na execução);
- Mecanismo de Execução: Este atributo especifica a tecnologia utilizada para implementar os serviços a nível de invocação, por exemplo: SOAP, REST e OSGI;

Em base à matriz descrita e aos projetos selecionados apresentados nos artigos [26], [37] e [46], foram desenvolvidas as tabelas 3.1, 3.2, 3.3, 3.4, 3.5 e 3.6. A nomenclatura utilizada nestas tabelas é: ✓para indicar a presença do atributo, X para indicar que o atributo em questão não é suportado pela proposta avaliada. A Tabela 3.1 apresenta a dimensão de especificação geral, na qual se especifica e comprara os atributos: Descrição dos Serviços (DesServ), Linguagem de Descrição dos Serviços(LinDesServ), Contexto (Cxto), Qualidade de Serviço (QoS) e Segurança (Seg).

A Tabela 3.2 apresenta a dimensão de especificação do usuário, na qual se especifica e comprara os atributos: Tipo de Usuário (TipUsr), Criação da Meta (CriGoal), Granularidade da Meta (GrnGoal) e Formalização da Meta (FrmGoal).

A Tabela 3.3 apresenta a dimensão de especificação do ambiente, na qual se especifica e comprara os atributos: Tipo de Ambiente (TipAmb), Topologia do Ambiente (TplAmb), Domínio da Aplicação (DomApp) e Rol dos Serviços (RolServ).

A Tabela 3.4 apresenta a dimensão de especificação dos dispositivos, na qual se especifica e comprara os atributos: Dispositivo de Interação com o Usuário (DisIntUsr), Dispositivos presentes no Ambiente (DisAmb), Mínimo Dispositivo que Executa o Processo de Composição (MinDis) e Avaliação da Proposta (AvlPro).

A Tabela 3.5 apresenta a dimensão de composição, na qual se especifica e comprara os atributos: Método de Descoberta de Serviços (DisServ), Mecanismo de Inferência (Inf), Linguagem do Mecanismo de Inferência (LinInf), Automação do Processo de Composição (Comp) e Representação do Plano Solução (RepSolv). Finalmente, a Tabela 3.6 apresenta a dimensão de execução, na qual se especifica e comprara os atributos: Fluxo de Processamento (Fluxo), Padrão de Execução (PadExe), Tipo de Execução (TipExe) e Mecanismo de Execução (MecExe).

**Tabela 3.1:** Dimensão de Especificação Geral

| Nome do Projeto - Autor | DesServ | LinDesServ | Cxto | QoS | Seg |
|---|---|---|---|---|---|
| CASE - Hesselman et al. (2006) [95] | Semântica | OWL-S | ✓ | X | X |
| SIC - Mingkhwan et al. (2006) [96] | Semântica | OWL-S | X | X | X |
| Kaefer et al. (2006) [97] | Sintática | XML | X | X | X |
| Pourreza et al. (2006) [98] | Semântica | OWL-S | ✓ | ✓ | X |
| COSEP - Lee et al. (2006) [99] | Semântica | X | ✓ | X | X |
| Qiu et al. (2006) [40] | Semântica | OWL-SC | ✓ | X | X |
| Aura - Sousa et al. (2006) [100] | Sintática | XML | X | X | X |
| Daidalos - Yang et al. (2006) [101] | Sintática | X | ✓ | ✓ | X |
| iCap - Dey et al. (2006) [102] | Sintática | X | ✓ | X | X |
| PalCom - Svensson et al. (2006) [103] | Sintática | Script | X | X | X |
| COCOA - Mokhtar et. al. (2006) [104] | Semântica | OWL-S | ✓ | ✓ | ✓ |
| Lee et al. (2007) [105] | Sintática | Key-Value | ✓ | ✓ | X |
| Vukovic (2007) [16] | Sintática | WSDL | ✓ | X | X |
| Bottaro et al. (2007) [106] | Sintática | Key-Value | ✓ | X | X |
| Wisner et al. (2007) [107] | Sintática | WSDL | X | X | X |
| DSCiPC - Kalasapur et al. (2007) [108] | Semântica | X | ✓ | ✓ | X |
| DSD - Baresi et al. (2007) [109] | Sintática | BPEL | X | ✓ | X |
| AMIGO - Thomson et al. (2008) [110] | Semântica | OWL-S | ✓ | ✓ | X |
| DCC - Want et al. (2008) [111] | Sintática | X | ✓ | X | X |
| Mobile Pipes - Trevor (2008) [112] | Sintática | JSON | X | X | X |
| QuAMobile - Amundsen et al.(2008) [113] | Sintática | WSDL | ✓ | ✓ | X |
| Paluska et al. (2008) [114] | Sintática | Script | X | X | X |
| Beauche et al. (2008) [115] | Semântica | YAWL | X | X | X |
| SeGSeC - Fujii et al. (2009) [17] | Semântica | OWL-S | ✓ | X | X |
| Bertoli et al. (2009) [116] | Sintática | BPEL | X | X | X |
| MySIM - Ibrahim et al. (2009) [117] | Semântica | OWL-S | X | ✓ | X |
| ReSCo - Lagesse et al. (2010) [118] | Sintática | X | X | X | ✓ |
| Santofimia et al. (2011) [119] | Semântica | OWL-S | ✓ | X | X |

**Tabela 3.2:** Dimensão de Especificação do Usuário

| Nome do Projeto - Autor | TipUsr | CriGoal | GrnGoal | FrmGoal |
|---|---|---|---|---|
| CASE - Hesselman et al. (2006) [95] | Usuário Final | Interação | Explicita | Workflow |
| SIC - Mingkhwan et al. (2006) [96] | Usuário Final | Configuração | X | Workflow |
| Kaefer et al. (2006) [97] | Desenvolvedor | Programação | Explicita | Workflow |
| Pourreza et al. (2006) [98] | X | X | X | Workflow |
| COSEP - Lee et al. (2006) [99] | Desenvolvedor | Configuração | Explicita | Serviço Web |
| Qiu et al. (2006) [40] | Desenvolvedor | Programação | X | Serviço Web |
| Aura - Sousa et al. (2006) [100] | Usuário Final | Interação | Implícita | X |
| Daidalos - Yang et al. (2006) [101] | X | Configuração | Implícita | X |
| iCap - Dey et al. (2006) [102] | Usuário Final | Interação | Implícita, Explicita | X |
| PalCom - Svensson et al. (2006) [103] | Usuário Final | Configuração | Explicita | Script |
| COCOA - Mokhtar et. al. (2006) [104] | Usuário Final | X | Explicita | Workflow |
| Lee et al. (2007) [105] | Usuário Final | Interação | X | Workflow |
| Vukovic (2007) [16] | Usuário Final | X | X | Serviço Web |
| Bottaro et al. (2007) [106] | Desenvolvedor | Interação | Explicita | Workflow |
| Wisner et al. (2007) [107] | Usuário Final | Interação | Explicita | Workflow |
| DSCiPC - Kalasapur et al. (2007) [108] | Desenvolvedor | Configuração | Implícita | X |
| DSD - Baresi et al. (2007) [109] | Desenvolvedor | Configuração | Implícita, Explicita | Workflow |
| AMIGO - Thomson et al. (2008) [110] | Usuário Final | Interação | Implícita | Workflow |
| DCC - Want et al. (2008) [111] | Usuário Final | Interação | Explicita | X |
| Mobile Pipes - Trevor (2008) [112] | Desenvolvedor | Configuração | Explicita | X |
| QuAMobile - Amundsen et al.(2008) [113] | Desenvolvedor | Configuração | Implícita | X |
| Paluska et al. (2008) [114] | Usuário Final | Interação | Implícita, Explicita | Serviço Web |
| Beauche et al. (2008) [115] | Usuário Final | Interação | Explicita | Workflow |
| SeGSeC - Fujii et al. (2009) [17] | Usuário Final | Interação | Implícita | Workflow |
| Bertoli et al. (2009) [116] | Desenvolvedor | Programação | Explicita | Serviço Web |
| MySIM - Ibrahim et al. (2009) [117] | Usuário Final | Interação | Implícita, Explicita | Serviço Web |
| ReSCo - Lagesse et al. (2010) [118] | Desenvolvedor | Programação | Explicita | X |
| Santofimia et al. (2011) [119] | Usuário Final | Interação | Implícita, Explicita | JADEX Goal |

**Tabela 3.3:** Dimensão de Especificação do Ambiente

| Nome do Projeto - Autor | TipAmb | TplAmb | DomApp | RolServ |
|---|---|---|---|---|
| CASE - Hesselman et al. (2006) [95] | Veicular | Fixa | Entretenimento | Áudio e video |
| SIC - Mingkhwan et al. (2006) [96] | Casa | Fixa | Entretenimento | Video |
| Kaefer et al. (2006) [97] | Casa | Fixa | Entretenimento | Video |
| Pourreza et al. (2006) [98] | Casa | Fixa | Serviços | Ligar dispositivos |
| COSEP - Lee et al. (2006) [99] | Casa | Fixa | Serviços | Ligar lampadas |
| Qiu et al. (2006) [40] | Web | X | Viagem | Busca, Passagem |
| Aura - Sousa et al. (2006) [100] | Casa | Ad hoc | Entretenimento | Video |
| Daidalos - Yang et al. (2006) [101] | Escritório | X | Reunião | Voz sobre IP |
| iCap - Dey et al. (2006) [102] | Publico | X | Entretenimento | Busca de locais |
| PalCom - Svensson et al. (2006) [103] | Escritório | Ad hoc | Serviços | Busca dispositivos |
| COCOA - Mokhtar et. al. (2006) [104] | Casa | Fixa | Entretenimento | Video |
| Lee et al. (2007) [105] | Shopping | Fixa | Compras | Produtos |
| Vukovic (2007) [16] | Publico | Fixa | Entretenimento | Restaurantes |
| Bottaro et al. (2007) [106] | Casa | Fixa | Serviços | Ligar lampadas |
| Wisner et al. (2007) [107] | Escritório | X | Serviços | Busca dispositivos |
| DSCiPC - Kalasapur et al. (2007) [108] | Escritório | Ad hoc | Serviços | Busca dispositivos |
| DSD - Baresi et al. (2007) [109] | Escritório | Ad hoc | Serviços | Busca dispositivos |
| AMIGO - Thomson et al. (2008) [110] | Casa | Fixa | Entretenimento | Controle dispositivos |
| DCC - Want et al. (2008) [111] | Casa | Ad hoc | Entretenimento | Busca dispositivos |
| Mobile Pipes - Trevor (2008) [112] | Publico | Fixa | Entretenimento | Busca de locais |
| QuAMobile - Amundsen et al.(2008) [113] | Casa | Ad hoc | Entretenimento | Video |
| Paluska et al. (2008) [114] | Casa | X | Entretenimento | Video |
| Beauche et al. (2008) [115] | Shopping | X | Compras | Pagamentos |
| SeGSeC - Fujii et al. (2009) [17] | Escritório | Fixa | Serviços | Busca dispositivos |
| Bertoli et al. (2009) [116] | Web | X | Viagens | Busca passagens |
| MySIM - Ibrahim et al. (2009) [117] | Casa | Fixa | Serviços | Busca dispositivos |
| ReSCo - Lagesse et al. (2010) [118] | Publico | Fixa | Controle de Transito | Video |
| Santofimia et al. (2011) [119] | Casa | Ad hoc | Serviços | Busca dispositivos |

**Tabela 3.4:** Dimensão de Especificação dos Dispositivos

| Nome do Projeto - Autor | DisIntUsr | DisAmb | MinDis | AvlPro |
|---|---|---|---|---|
| CASE - Hesselman et al. (2006) [95] | X | Sistema Veicular | X | X |
| SIC - Mingkhwan et al. (2006) [96] | Smartphone | Eletrônicos | Servidor | Protótipo |
| Kaefer et al. (2006) [97] | X | TVs | X | Aplicação-Exemplo |
| Pourreza et al. (2006) [98] | Gateway | Eletrônicos | Servidor | Protótipo |
| COSEP - Lee et al. (2006) [99] | PDA | X | PDA | Protótipo |
| Qiu et al. (2006) [40] | PC | X | Servidor | Aplicação-Exemplo |
| Aura - Sousa et al. (2006) [100] | PDA | TVs, DVD | Servidor | Aplicação-Exemplo |
| Daidalos - Yang et al. (2006) [101] | X | Dispositivos Comunicação | X | Aplicação-Exemplo |
| iCap - Dey et al. (2006) [102] | PDA | X | Servidor | Protótipo |
| PalCom - Svensson et al. (2006) [103] | PDA | Eletrônicos | PDA | Protótipo |
| COCOA - Mokhtar et. al. (2006) [104] | X | TV | Servidor | Protótipo |
| Lee et al. (2007) [105] | Gateway | Eletrônicos | Servidor | Aplicação-Exemplo |
| Vukovic (2007) [16] | X | X | Servidor | Protótipo |
| Bottaro et al. (2007) [106] | PDA | Atuadores | PC | Protótipo |
| Wisner et al. (2007) [107] | Smartphone | Eletrônicos | PC | Protótipo |
| DSCiPC - Kalasapur et al. (2007) [108] | Sensores | Eletro-eletrônicos | PDA | Aplicação-Exemplo |
| DSD - Baresi et al. (2007) [109] | PDA | Eletrônicos | Servidor | Aplicação-Exemplo |
| AMIGO - Thomson et al. (2008) [110] | PDA | Eletrônicos | Servidor | Protótipo |
| DCC - Want et al. (2008) [111] | PDA | Eletrônicos | PDA | Aplicação-Exemplo |
| Mobile Pipes - Trevor (2008) [112] | PDA | X | Servidor | Protótipo |
| QuAMobile - Amundsen et al.(2008) [113] | PDA | TV | Servidor | Protótipo |
| Paluska et al. (2008) [114] | PC | TV, Som | Servidor | Protótipo |
| Beauche et al. (2008) [115] | PC | X | Servidor | X |
| SeGSeC - Fujii et al. (2009) [17] | PDA | Eletrônicos | Servidor | Aplicação-Exemplo |
| Bertoli et al. (2009) [116] | X | X | Servidor | Aplicação-Exemplo |
| MySIM - Ibrahim et al. (2009) [117] | PDA | Eletrônicos | Servidor | Aplicação-Exemplo |
| ReSCo - Lagesse et al. (2010) [118] | PDA | TV, Câmera | Servidor | Protótipo |
| Santofimia et al. (2011) [119] | Sensores | Atuadores | Servidor | Protótipo |

**Tabela 3.5:** Dimensão de Composição

| Nome do Projeto - Autor | DisServ | Inf | LinInf | Comp | RepSolv |
|---|---|---|---|---|---|
| CASE - Hesselman et al. (2006) [95] | Jini | Workflow | X | Semi-Automática | Própria |
| SIC - Mingkhwan et al. (2006) [96] | X | Workflow | XML | Automática | XML |
| Kaefer et al. (2006) [97] | UPnP | Workflow | FTD | Manual | FTD |
| Pourreza et al. (2006) [98] | UPnP | Workflow | OWL-S | Automática | OWL-S |
| COSEP - Lee et al. (2006) [99] | X | Data-Mining | X | Automática | X |
| Qiu et al. (2006) [40] | WS-Discovery | Planejamento | HTN | Automática | DAG |
| Aura - Sousa et al. (2006) [100] | | | | | |
| Daidalos - Yang et al. (2006) [101] | | | | | |
| iCap - Dey et al. (2006) [102] | | | | | |
| PalCom - Svensson et al. (2006) [103] | | | | | |
| COCOA - Mokhtar et. al. (2006) [104] | X | Workflow | OWL-S | Manual | OWL-S |
| Lee et al. (2007) [105] | OSGI | Workflow | XML | Automática | XML |
| Vukovic (2007) [16] | UDDI | Planejamento | HTN SHOP2 | Automática | BPEL4WS |
| Bottaro et al. (2007) [106] | OSGI | Workflow | X | Automática | Interna |
| Wisner et al. (2007) [107] | UPnP | Planejamento | Scripts | Manual | X |
| DSCiPC - Kalasapur et al. (2007) [108] | | | | | |
| DSD - Baresi et al. (2007) [109] | | | | | |
| AMIGO - Thomson et al. (2008) [110] | | | | | |
| DCC - Want et al. (2008) [111] | | | | | |
| Mobile Pipes - Trevor (2008) [112] | | | | | |
| QuAMobile - Amundsen et al.(2008) [113] | | | | | |
| Paluska et al. (2008) [114] | X | Planejamento | HTN | Automática | Scripts |
| Beauche et al. (2008) [115] | X | Planejamento | HTN | Automática | YAWL |
| SeGSeC - Fujii et al. (2009) [17] | | | | | |
| Bertoli et al. (2009) [116] | | Planejamento | BDD | | Interna |
| MySIM - Ibrahim et al. (2009) [117] | | Workflow | Capabilities matching | | Interna |
| ReSCo - Lagesse et al. (2010) [118] | | Workflow | Constraint matching | | |
| Santofimia et al. (2011) [119] | | Planejamento | HTN | | Interna |

**Tabela 3.6:** Dimensão de Especificação do Usuário

| Nome do Projeto - Autor | Fluxo | PadExe | TipExe | MecExe |
|---|---|---|---|---|
| CASE - Hesselman et al. (2006) [95] | | Orquestração | Estática | SOAP |
| SIC - Mingkhwan et al. (2006) [96] | | Orquestração | Estática | SOAP |
| Kaefer et al. (2006) [97] | | Orquestração | Estática | SOAP |
| Pourreza et al. (2006) [98] | | Orquestração | Estática | OSGI |
| COSEP - Lee et al. (2006) [99] | | Orquestração | Estática | SOAP |
| Qiu et al. (2006) [40] | | Orquestração | Estática | SOAP |
| Aura - Sousa et al. (2006) [100] | | Orquestração | Dinâmica | SOAP |
| Daidalos - Yang et al. (2006) [101] | | Orquestração | Dinâmica | SOAP |
| iCap - Dey et al. (2006) [102] | | Orquestração | Estática | SOAP |
| PalCom - Svensson et al. (2006) [103] | | Orquestração | Dinâmica | SOAP |
| COCOA - Mokhtar et. al. (2006) [104] | | Orquestração | Estática | SOAP |
| Lee et al. (2007) [105] | | Orquestração | Dinâmica | OSGI |
| Vukovic (2007) [16] | | Orquestração | Estática | SOAP |
| Bottaro et al. (2007) [106] | | Orquestração | Estática | OSGI |
| Wisner et al. (2007) [107] | | Orquestração | Estática | SOAP |
| DSCiPC - Kalasapur et al. (2007) [108] | | Coreografia | Dinâmica | SOAP |
| DSD - Baresi et al. (2007) [109] | | Coreografia | Dinâmica | BPEL |
| AMIGO - Thomson et al. (2008) [110] | | Orquestração | Dinâmica | BPEL |
| DCC - Want et al. (2008) [111] | | Orquestração | Estática | SOAP |
| Mobile Pipes - Trevor (2008) [112] | | Orquestração | Estática | REST |
| QuAMobile - Amundsen et al.(2008) [113] | | X | Dinâmica | SOAP |
| Paluska et al. (2008) [114] | | Orquestração | Estática | SOAP |
| Beauche et al. (2008) [115] | | X | X | YAWL |
| SeGSeC - Fujii et al. (2009) [17] | | Coreografia | Estática | BPEL |
| Bertoli et al. (2009) [116] | | Orquestração | Estática | SOAP |
| MySIM - Ibrahim et al. (2009) [117] | | Orquestração | Estática | SOAP |
| ReSCo - Lagesse et al. (2010) [118] | | Orquestração | Estática | SOAP |
| Santofimia et al. (2011) [119] | | Orquestração | Dinâmica | SOAP |

Das tabelas apresentadas sobre os vinte e oito trabalhos avaliados, podemos dizer o seguinte:

- Que nem todas as propostas abordam o processo de composição propriamente dito, isto é,em alguns casos só é abordado o processo de descoberta e disponibilização de serviços atômicos;
- Que poucas propostas abordam todas as características que devem estar presentes em um sistema de composição de serviços para ambientes pervasivos, em especial o aspecto de segurança;
- Que a grande maioria das propostas são construídas para um domínio de aplicação e/ou um tipo de ambiente especifico, isto é, a arquitetura e infraestrutura implementadas dificilmente podem ser reaproveitadas ou adequadas para novos ambientes pervasivos;
- Que poucas propostas apresentam uma implementação para um cenário real, isto é, a maioria das propostas avalia seu processo de composição sobre um cenário exemplo, o qual tem uma serie de restrições;
- Que nenhuma proposta apresenta uma especificação formal, completa e genérica o suficiente como para poder ser re-utilizada em distintos domínios de aplicação e/ou tipos de ambientes pervasivos, de forma que permita uma fácil e pratica adoção deste tipo de sistemas para ambientes pervasivos.

Além dos trabalhos descritos nas tabelas anteriores, o detalhamento de alguns outros trabalhos que descrevem o estado da arte para os sistemas de composição de serviços em ambientes pervasivos e que apresentam características interessantes a serem aproveitadas na especificação do nosso modelo de referência são descritos a seguir:

- No tralho de doutorado [16] se propõe um sistema de composição chamado GoalMorph que considera informações sensíveis ao contexto e a chamada dinâmica de serviços. Este trabalho utiliza a técnica de planejamento baseada em orientação à meta (*goal-oriented*), avaliando informações de contexto como: recursos disponíveis, limitações de tempo e localização do usuário. Este trabalho também tem suporte para mudanças de contexto em tempo de execução, isto é, recuperação de falhas. A Figura 3.1 apresenta arquitetura do GoalMorph, da qual podemos destacar que é dividida em quatro fases, sendo a primeira a de especificação de requisição de usuário a qual é detalhada com informações de contexto. A segunda e terceira fases encarregam-se, respectivamente, da criação do domínio e

problema de planejamento e execução do composição de composição. A quarta fase refere-se à execução do plano gerado e sua monitoração para caso algum serviço falhe o processo de re-composição seja executado. Este trabalho foi projetado para sistemas Web e não para ambientes pervasivos, mas a capacidade do sistema de considerar uma falha na execução do serviço em base a informações de contexto, é uma característica a ser aproveitada e definida no nosso modelo de referência.

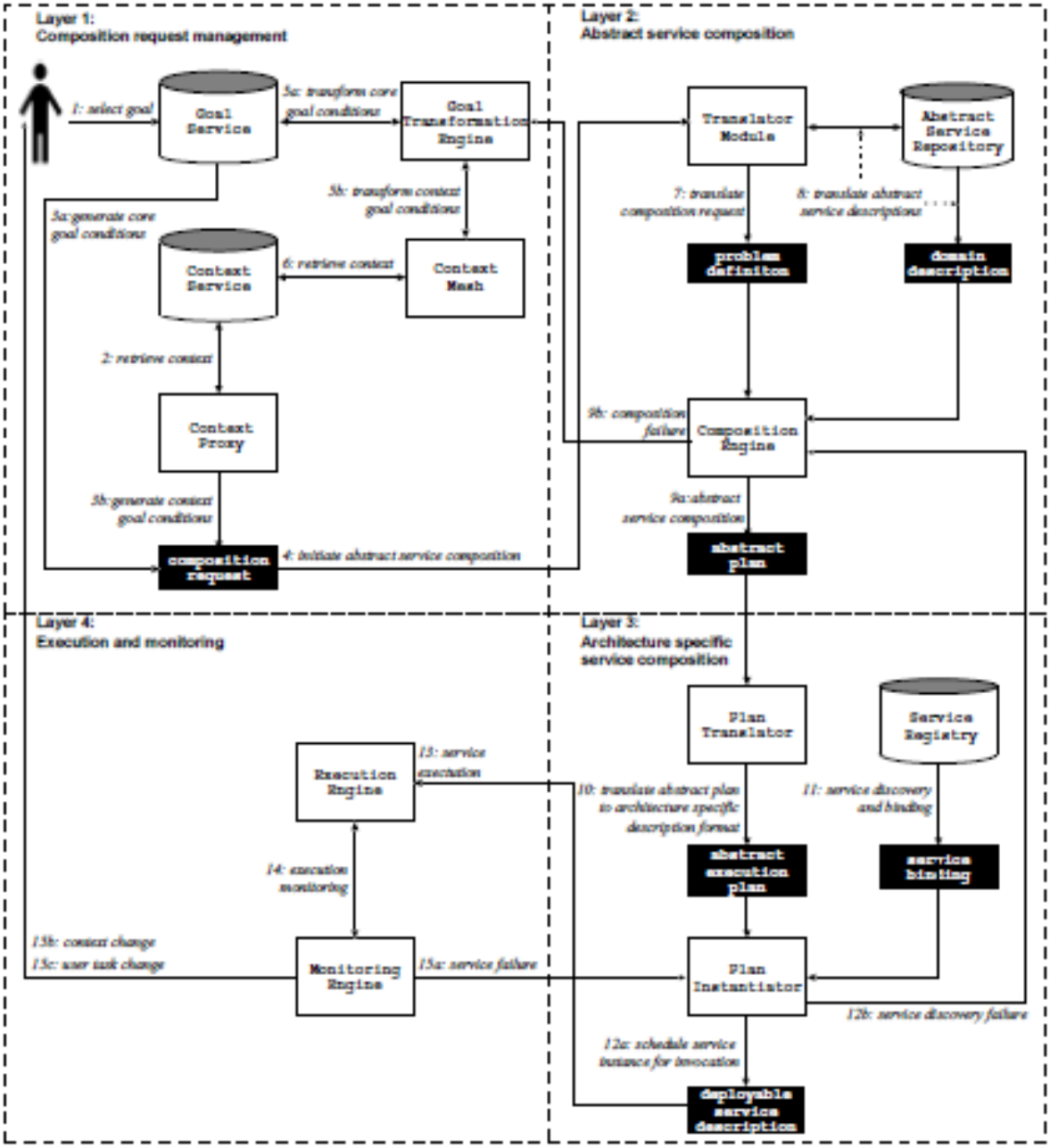


**Figura 3.1:** Arquitetura do Sistema de Composição GoalMorph - [16]

- Em [17] apresenta-se um sistema para composição de serviços baseados em informações contextuais. Este sistema é composto pelo *Component Service Model with Semantics*

(CoSMoS), *Component Runtime Environment* (CoRE), e *Semantic Graph based Service Composition* (SeGSeC). CoSMoS é um modelo semântico que permite modelar informações contextuais dos usuários em regras. SeGSeC permite especificar a requisição do usuário em uma linguagem natural, a qual é transformada pelo CoSMoS em uma especificação formal através do uso de ontologias e RDF Schemas. Com a especificação semanticamente descrita o SeGSeC executa o processo de composição utilizando a técnica de *Workflows*. Já o CoRE é um middleware que permite a implementação de diversos mecanismos de descoberta e execução de serviços. A Figura 3.2 apresenta os componentes deste sistema e descreve o processo de descrição de requisição em linguagem natural por parte do usuário. Este trabalho mesmo não sendo específico para ambientes pervasivos, tem uma característica interessante sobre como as requisições do usuário podem ser descritas utilizando linguagem natural. Nesse sentido, esta característica sera aproveitada no nosso modelo de referência.

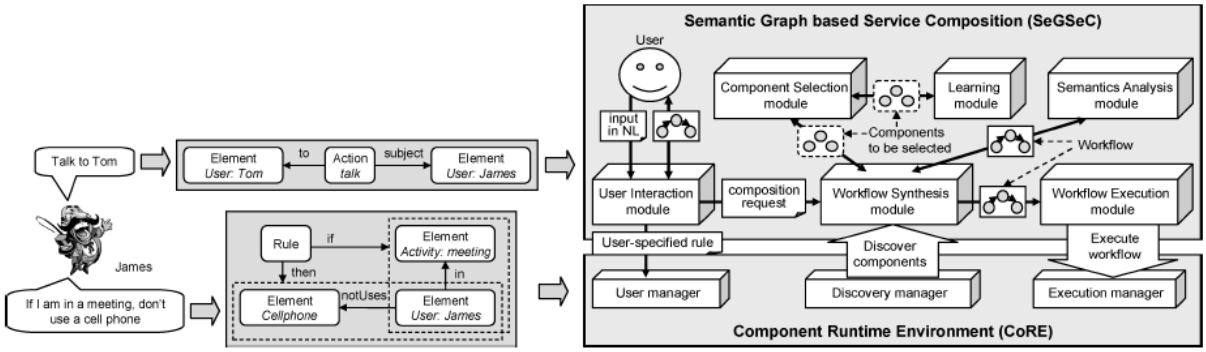


**Figura 3.2:** Sistema de Composição SeGSeC-CoSMoS-CoRE - [17]

- Já em [18] apresenta-se um sistema de composição de serviços móveis com informações contextuais. Neste trabalho utiliza-se o *Semantic Context Model* (SECOM) para modelar as informações contextuais e apresenta-se o uma arquitetura chamada iCAS, a qual da suporte ao processo de composição. A Figura 3.3 apresenta a arquitetura iCAS, da qual podemos destacar a utilização de diversos dispositivos para a aquisição de informações de contexto e a descrição do uso de uma interface gráfica de usuário para especificação de requisições, a qual utiliza informações de preferencias e perfil do usuário para otimizar a requisição. Já o foco deste trabalho é baseado em um campus universitário o qual caracteriza um ambiente pervasivo. Nesse sentido, o sistema deve fornecer recursos pedagógicos e auxiliar a interação sócio-pedagógica de vários tipos de usuários (alunos, professores, funcionarios, etc.). Estes recursos podem ser representados por materiais de aula como textos e livros assim como por ambientes físicos como aulas ou auditórios. Já este trabalho não suporta suporte a falhas em tempo de execução nem processo de

recomposição, porém o uso de informações do usuário como perfil e preferências, é uma característica a ser considerado pelo nosso modelo de referência.

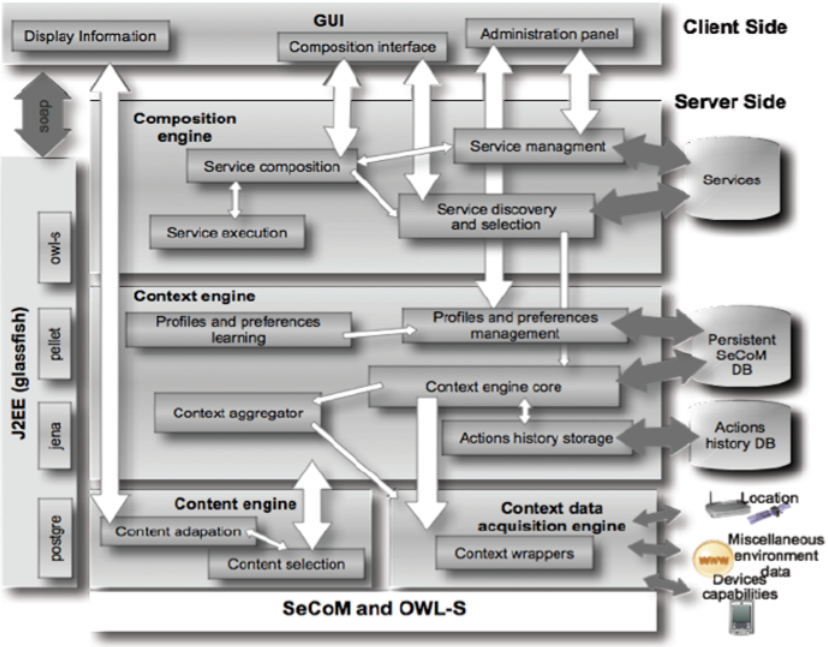


**Figura 3.3:** Sistema de Composição iCAS - [18]

- MEDUSA (*Middleware for End-User Composition of Ubiquitous Applications*) [19] é um middleware para a composição de aplicações de acordo com as necessidades do usuário. Em MEDUSA, o usuário utiliza um dispositivo móvel como uma unidade de controle remoto que lhe permite selecionar e criar listas (compor) de reprodução de áudio e vídeo em base à interação com objetos físicos como RFIDs, os quais estão associados aos arquivos multimídia que farão parte da lista de reprodução. Os níveis de interação entre o usuário e o sistema podem ser manual, semi-manual, iniciativa mista e autônomo, como representados pela Figura 3.4. O nível manual o usuário deve especificar todos os serviços a utilizar; no semi-manual uma vez feito o toque em algum dispositivo, o sistema começa o processo de composição, porém o usuário precisa definir alguns outros parâmetros; na iniciativa mista o sistema sugere para o usuário algum serviço disponível no ambiente; e no autônomo o sistema escolhe e executa algum serviço que espera-se seja do interesse do usuário.

  Outra questão importante abordada em MEDUSA é a interoperabilidade entre dispositivos heterogêneos, redes e plataformas. A Figura 3.5 apresenta a arquitetura de MEDUSA, da qual podemos destacas a camada de comunicação e interoperabilidade implementa a API ubiSOAP, a qual estabelece e gerencia a conexão dos serviços com os distintos protocolos de comunicação suportados. Nesse sentido, em MEDUSA a forma

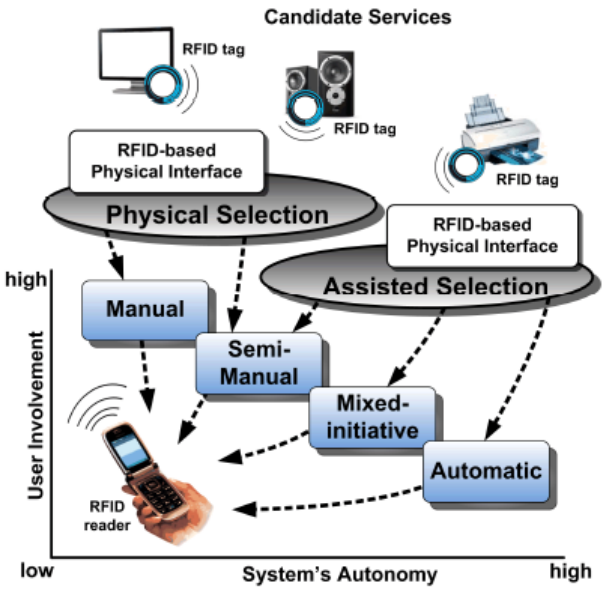


**Figura 3.4:** Níveis de interação em MEDUSA - [19]

como o usuário expressa suas requisições (interação através do toque com objetos físicos) e os níveis de interação (iniciativa mista e autônoma) são características interessantes a serem consideradas na especificação de nosso modelo de referência.

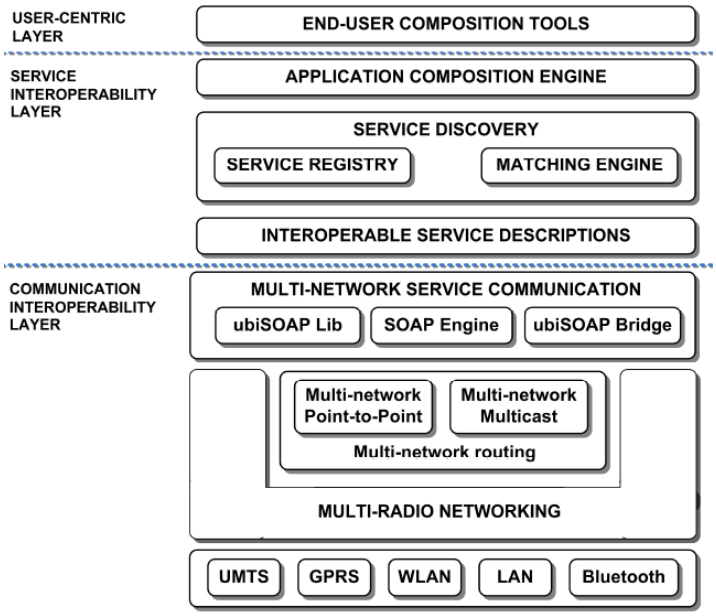


**Figura 3.5:** Arquitetura de MEDUSA - [19]

## 3.2 Trabalhos Relacionados

Como o objetivo principal do presente trabalho é especificar formalmente um "Modelo de Referência" que permita projetar, implementar e avaliar sistemas de composição de serviços em ambientes pervasivos, nós consideramos como trabalhos diretamente relacionados ao nosso as propostas que abordam, em algum grau, a especificação ou formalização de um modelo de referência para sistemas de composição de serviços. Nesse contexto, nós podemos destacar os seguintes trabalhos:

- *Towards a reference service model for the Web of Services* [20]: Este artigo apresenta um modelo referência para serviços (*Reference Service Model* - RSM), o qual tem como objetivo alinhar e facilitar a interoperabilidade semântica entre os serviços anotados utilizando diferentes modelos ontológicos (por exemplo, WSMO, OWL-S, etc) e os serviços anotados de forma sintática por diversos usuários de forma colaborativa. RSM identifica e abstraí os seguintes conceitos: serviço (*Service* - S), parâmetros de entrada do serviço (*Service Input* - $S_I$), parâmetros de saída de serviço (*Service Output* - $S_O$), parâmetros de contexto do serviço (*Service Context* - $S_C$), logica de negócios do serviço (*Service Logic* - $S_L$), provedor de serviços (*Service Provider* - $S_P$), cliente do serviço (*Service Client* - $S_{CL}$) e comentários sobre o serviço(*Service Feedback* - $S_F$). De forma que um serviço (S) pode ser expressado como:

  $$S = \{S_I, S_O, S_C, S_L, S_P, S_{CL}\}$$

  A Figura 3.6 do lado esquerdo apresenta as relações entre os conceitos definidos no RSM, já do lado direito apresenta-se os mesmos conceitos em forma de grafo. Como prova de conceito, neste trabalhos apresentam-se dois estudos de caso: um motor de busca de serviços semântico, focado na capacidade do RSM para encontrar interligação entre serviços descritos utilizando diferentes modelos de serviços semânticos; e um portal governamental, focando na capacidade do RSM para facilitar a anotação sintática social dos serviços.

- *Description of Web service composition model based on Z notation* [120]: Neste trabalho é proposta a descrição formal do processo de composição de serviços Web utilizando a notação Object Z [121] através do exemplo VTA (*Virtual Travel Agency*) descrito em [49]. Nesse sentido, é descrito formalmente o comportamento de uns serviços Web e suas operações, para logo descrever as comunicações entre estes serviços. Certamente, nenhum processo de composição é descrito.

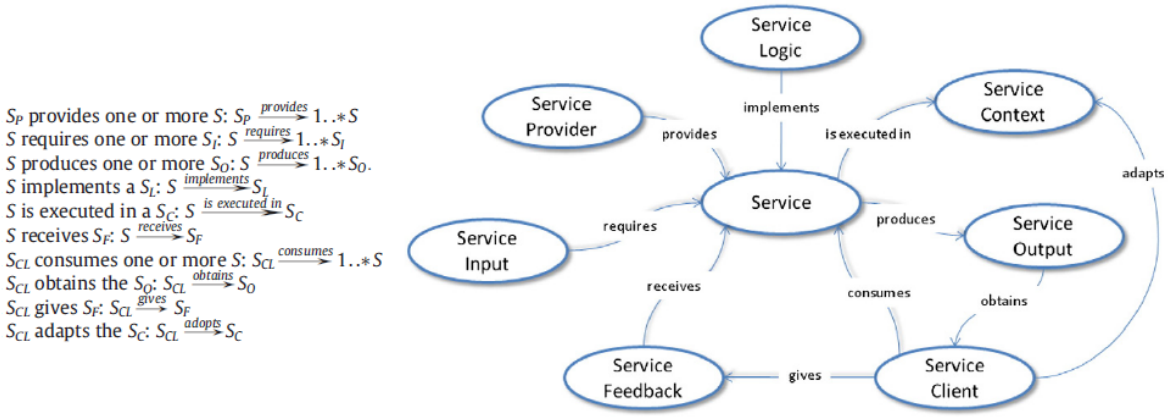


**Figura 3.6:** Modelo Referência para Serviços (RSM): Relações entre conceitos - [20]

- *A formal model of the Semantic Web Service Ontology (WSMO)* [21]: Neste trabalho os autores apresentam uma formalização dos elementos do modelo ontológico WSMO. Esta formalização é feita utilizando a linguagem Object Z. A formalização incluí os quatro elementos principais do WSMO (ontologias, metas, serviços Web e mediadores) e um modelo de execução dinâmica de serviços Web. A especificação da dinâmica do modelo de execução é descrito através de valores, variáveis, expressões e regras de transição, as quais permitem verificar a mudança de estado na execução de um serviço Web. A formalização permite verificar a consistência da linguagem através de ferramentas de verificação. Nesse sentido, os autores deste trabalho identificaram um erro na especificação de atributos de conceitos na linguagem WSMO. O erro acontece devido ao fato que segundo a especificação do WSMO, um atributo do tipo hasType definido para um conceito pode ter apenas um conceito como valor. Não entanto, no modelo ontológico WSMO o valor do conceito pode ser interpretado como uma String, o que gera uma violação na especificação. Uma solução proposta para dita violação é a agregação do *data Type* como valor possível para um atributo do tipo hasType definido para um conceito, como apresentado na Figura 3.7.

**Current WSMO specification:**
Class *attribute* sub-Class wsmoElement
  *hasType* type *concept* ... ....
**Revised WSMO specification:**
Class *attribute* sub-Class wsmoElement
  *hasType* type **concept or dataType** ... ....

**Figura 3.7:** Revisão da Especificação WSMO - [21]

- *PSC-RM: Reference Model for Pervasive Service Composition* [22]: Este artigo acunha o termo composição de serviços pervasivos (*Pervasive Service Composition* - PSC). Este termo define a junção da composição de serviços e da computação pervasiva para

gerenciar as atividades diárias do usuário. Nesse sentido, um modelo de referência genérico (PSC-RM) é proposto com o objetivo de orientar o projeto e implementação de arquiteturas para sistemas de composição de serviços pervasivos. Este modelo de referência baseia-se no estudo e analise do modelo de atividades genéricas dos usuários deste tipo de sistemas. Além disso, o modelo de referência classifica o processo de composição em cinco tipos (composição orientada a serviços, composição baseada em P2P, composição baseada em contexto, composição baseada em interfaces de usuário multimodal, e composição baseada em confiança) em base as características dos sistemas pervasivos. A Figura 3.8 apresenta o modelo de referência PSC-RM, a qual é composta pelas camadas de aplicação, de sistema e de suporte.

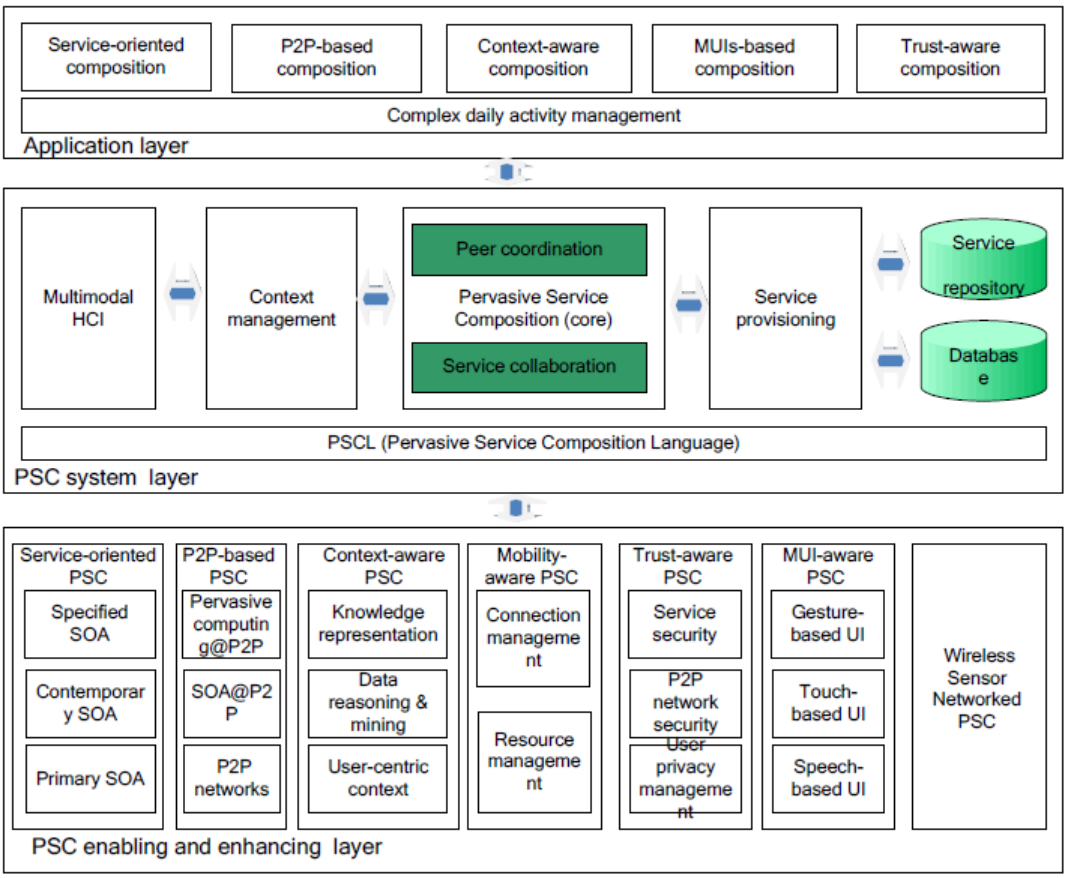


**Figura 3.8:** *PSC-RM: Reference Model for Pervasive Service Composition* - [22]

Dos trabalhos relacionados apresentados podemos dizer que nenhuma proposta apresenta uma especificação formal, completa e genérica o suficiente como para poder ser re-utilizada em distintos domínios de aplicação e/ou tipos de ambientes pervasivos. Isto porque em [20] só é apresentada uma formalização para a anotação semântica e sintática de serviços Web por distintas linguagens ontológicas. Já em [120] e [21] aborda-se a formalização do modelo ontológico WSMO e da dinâmica presente no processo de execução de serviços. Especialmente

em [21] demostra-se que o WSMO é ainda um modelo em construção, com falhas e com a necessidade de aprimoramento. Finalmente, em [22] um modelo de referência para a composição de serviços pervasivos é apresentada, porém, a classificação feita neste artigo não nós parece a mais adequada (isto em base ao apresentado na subseção 3.1), além de não apresentar nenhum formalismo. Portanto formalizar um modelo de referência, como proposto neste trabakih justifica-se pela necessidade de criar um modelo que especifique formalmente os elementos abstratos funcionais do sistema de composição, de modo que pode este modelo possa conduzir ao desenho de múltiplas arquiteturas e à implementação de sistemas de composição para distintos tipos de ambientes pervasivos.

## 3.3 Além do Estado da Arte

Nos últimos anos, novas tendencias e paradigmas computacionais tem-se destacado de forma convergente com os sistemas e ambientes pervasivos, entre estas tendencias destacam-se o paradigma da Internet das coisas ou *Internet of Things* - (IoT) [41] , da Computação em Nuvem (*Cloud Computing*) [122] e dos Sistemas Ciber-Físicos (*Cyber-Physical Systems - CPS*) [123].

**Internet das coisas** Nesse sentido, diversos fabricantes de dispositivos eletro-eletrônicos vem desenvolvendo dispositivos com capacidades de processamento e de comunicação a rede, por exemplo, a Qualcomm apresentou na Mobile World Congress de 2013 uma cafeteira conceito com estas capacidades. Uma foto desta cafeteira é apresentada na Figura 3.9. Assim como este dispositivo conceito, muitos outros vem sendo projetados com o proposito de oferecer diversas funcionalidade para os usuários. Note-se também que estes dispositivos além de oferecer funcionalidades, eles também vão gerar grandes quantidades de dados, como por exemplo, modos e vezes de utilização (hábitos de consumo). Estas grandes quantidades de dados podem ser consideradas como *Big Data* [124], a qual ao ser analisada e integrada pode oferecer informações relevantes e uteis nas atividades dos usuários.

Quando um conjunto deste tipo de dispositivos (IoT) encontram-se instalados em um ambiente pervasivo, um sistema de composição de serviços deve poder identificar as funcionalidades oferecidas de formas simples e automática. Nesse sentido, é necessário que os fabricantes disponibilizem junto aos dispositivos uma especificação semântica do perfil e funcionalidades do mesmo, assim como suas condições de funcionamento (pre e pós-condições), de modo que quando um usuário compra o dispositivo, só precisa levar-lo para casa (ambiente pervasivo) e conectar-lo à tomada, que o sistema de

**Figura 3.9:** Cafeteira conceito da Qualcomm

composição devera identificar o dispositivo e disponibilizar suas funcionalidades como novos serviços.

**Computação em Nuvem** Já a computação em nuvem não é apenas sinônimo de uso sob demanda de recursos de computacionais, mas este paradigma fornece escalabilidade ao processo de composição de serviços, isto porque, em cenários complexos com múltiplos usuários, grande quantidades de serviços e utilização de mecanismos de inferência com suporte a não-determinismo o sistema de composição precisa lidar com diversas demandas processamento. Além disso, a grande quantidade de dados que podem ser gerados pelos dispositivos, precisam ser armazenados e gerenciados de uma forma adequada. Informações contextuas do usuário, como perfis e preferencias podem ser puxados da nuvem e utilizados para personalizar o processo de composição quando o usuário se encontra em diversos ambientes, para tal fim precisa-se de um mecanismo que garanta a identificação e autenticação do usuário.

**Sistemas Ciber-Físicos** O recente surgimento do conceito de Sistemas Ciber-Físicos, propõe a integração do mundo físico com sistemas computacionais (hardware e software) levando em consideração regras de controle, fenômenos físicos, eventos aperiódicos e periódicos com diversos requisitos [125]. Nesse sentido, o sistema de composição de serviços em um ambiente pervasivo também precisa considerar estas características, por exemplo, o sistema de composição pode gerar um plano solução que não cumpra alguma regra de controle especificada pelo ambiente. Além disso, a integração do mundo físico com o sistema de composição, também precisa considerar a localização dos dispositivos, isto é, por regra de controle um dispositivo do tipo fogão não pode ser instalado em ambiente definido como banheiro.

# 4 Modelo de Referência

O objetivo deste trabalho é estudar o problema de composição automática de serviços em ambientes pervasivos, e desta forma ter as bases conceituais e experimentais para especificar, projetar, implementar e avaliar um sistema que automatize o processo de composição, de forma eficaz em termos de tempo e de adaptabilidade a situações do mundo real.

Nesse sentido, nossa proposta é desenvolver e implementar um sistema permita ao ambiente pervasivo interagir com o usuário de forma natural, automática e dinâmica. Desta forma, o sistema gerencia e automatiza o processo de resolução de requisições feitas pelo usuário (de forma implícita e/ou explicita) através das funcionalidades (serviços) disponíveis no ambiente ou através de novas funcionalidades criadas pelo processo de composição. Para tal fim, neste trabalho está sendo estendido o modelo ontológico WSMO (*Web Services Modelling Ontology*) [49] para suportar a descrição de informações contextuais (contexto).

## 4.1 Definição do Escopo

Fazem parte do escopo deste trabalho:

- Modelar as funcionalidades oferecidas pelo ambiente pervasivo, como serviços enriquecidos semanticamente. Dependendo da disponibilidade de dispositivos sensores reais, alguns serviços serão simulados via software para complementar as funcionalidades oferecidas em um ambiente pervasivo;

- Modelar informações contextuais sobre o ambiente, usuário, dispositivos e aplicações. O grau de detalhe das informações contextuais modeladas, dependera do nível interação necessário no ambiente pervasivo;

- Implementar algoritmos de planejamento que permitam raciocinar sobre incerteza e contexto. Inicialmente será implementado e estendido o planejador hierárquico não determinístico proposto por [92];

- Desenvolver os componentes de interface, descoberta, composição, avaliação, execução e re-composição serviços em ambientes pervasivos. Nesse sentido, o processo de descoberta e avaliação serão projetados de forma funcional mas simples, sem nenhuma sofisticação. Além disso, nos precisamos desenvolver um componente de tradução que faça o mapeamento entre os serviços e a sintaxe entendida pelo planejador.

Não faze parte do escopo deste trabalho o estudo e implementação de protocolos de comunicação para gerenciar as informações de sensores e atuadores heterogêneos, assim como, não é implementada no nosso sistema aspectos de segurança e privacidade das informações e serviços disponíveis no ambiente pervasivo.

## 4.2 Fundamentação

Nesse sentido, nossa proposta visa desenvolver um Modelo de Referência a partir do qual possam ser projetados e implementados sistemas de composição de serviços que gerenciem a interação de um ambiente pervasivo com seus usuários de forma natural, automática e dinâmica. A especificação do nosso modelo de referência permite descrever as funcionalidades a serem implementadas por sistemas de composição como componentes de alto nível utilizando a linguagem OMG SysML [48]. Estes componentes representam os módulos de: interface, descoberta, tradução, composição, avaliação, armazenamento, execução e re-composição de serviços; além dos módulos complementares (impostos pelas características dos ambientes pervasivos) de: aquisição e gestão de informações contextuais e preferências do usuário, segurança e controle de acesso de serviços, e especificação do grau de automatização do sistema de composição. Além disso, o nosso modelo de referência também descreve os elementos abstratos funcionais de como: são feitas as requisições do usuário (de forma implícita e/ou explicita) ao sistema de composição, e como o sistema de composição responde às requisições do usuário de forma eficaz em termos de tempo, de adaptabilidade e automatização.

Já formalizar um modelo de referência, justifica-se pela necessidade de criar um modelo que especifique formalmente os elementos abstratos funcionais do sistema de composição, de modo que pode este modelo possa conduzir ao desenho de múltiplas arquiteturas e à implementação de sistemas de composição para distintos tipos de ambientes pervasivos.

A escolhida de utilizar a linguagem OMG SysML [48] para a descrição formal do modelo de referência, justifica-se porque esta linguagem permite: modelar diferentes tipos de requisitos com suporte para funções contínuas e estruturas de decomposição; representar e verificar processos do fluxo de controle com alto nível de detalhe; a representação e formalização de

eventos, especificação de dependências, comportamento de classes, representação de valores de entrada e saída, e modelagem de subsistemas como uma visão integrada do sistema, incluindo hardware, software e partes eletro-mecânicas.

## 4.3 Cenário Geral de Aplicação

A especificação do nosso modelo de referência permite descrever as funcionalidades a serem implementadas pelo sistema de composição como componentes de alto nível utilizando a linguagem OMG SysML (*OMG Systems Modeling Language*) [48]. Estes componentes representam os módulos de: interface, descoberta, tradução, composição, avaliação, armazenamento, execução e re-composição de serviços; além dos módulos complementares (necessários pelas características dos ambientes pervasivos) de: aquisição e gestão de informações contextuais e preferências do usuário, segurança e controle de acesso de serviços, e especificação do grau de automatização do sistema de composição.

### 4.3.1 Abstração Arquitetural

O cenário geral de aplicação descreve o ambiente físico, os dispositivos (sensores, atuadores, eletro-eletrônicos) e os módulos que podem ser utilizados na implementação de um sistema de composição de serviços para ambientes pervasivos. Uma Abstração Arquitetural do cenário geral de aplicação é apresentado na Figura 4.1, o qual está dividido em três perspectivas: ambiente, segurança e composição.

A perspectiva de **Ambiente** está composta pelos dispositivos que são embarcados no ambiente pervasivo. Além disso, esta perspectiva também descreve se o ambiente é *Outdoor* ou *Indoor*. Já a perspectiva de segurança define o tipo de mecanismo de segurança que deve ser implementado para garantir as informações dos usuários, dos dispositivos e do processo de composição de forma geral. Finalmente, a perspectiva de composição descreve os módulos necessários para implementar um sistema de composição de serviços. Estas perspectivas também podem ser observada como níveis. Nesse sentido a Figura 4.2 apresenta quatro níveis. O primeiro descreve os dispositivos, o segundo descreve os serviços implementados em base dos dispositivos, a terceira descreve os módulos utilizados pelo sistema de composição e finalmente a quarte descreve os diversos tipos de usuários.

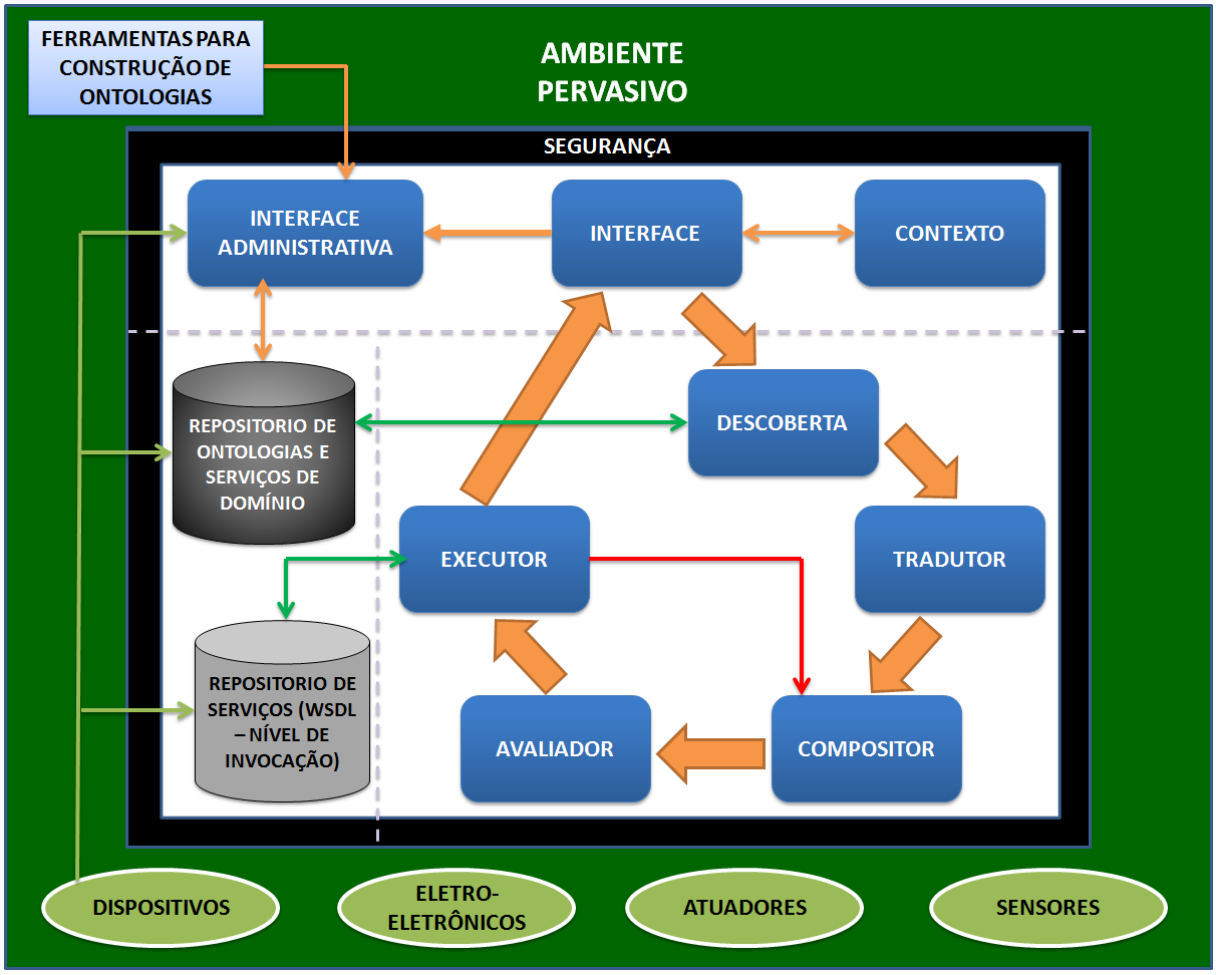


**Figura 4.1:** Abstração Arquitetural do Modelo de Referência Proposto

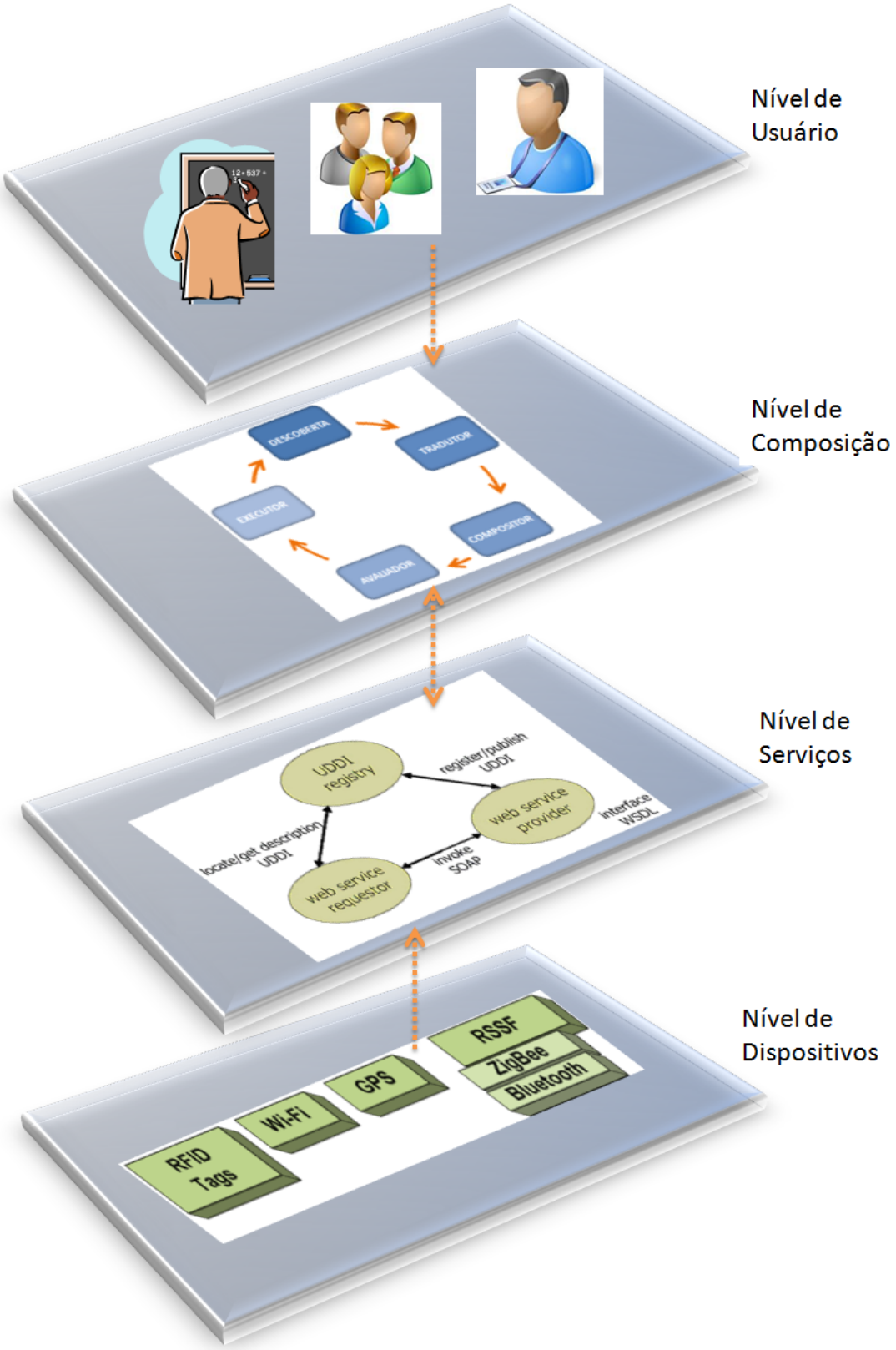


**Figura 4.2:** Abstração Arquitetural observada em Níveis

### 4.3.2 Modelagem Contextual Semântica

A modelagem de contexto utilizando ontologias permite a definição do comportamento do Ambiente Pervasivo em tempo de execução, mas não fornece o suporte necessário para lidar com o dinamismo do ambiente. Uma das alternativas para lidar com o dinamismo é o uso de contexto junto com os serviços Web semânticos [24]. Nesse sentido, no trabalho de [23] apresenta-se o SECOM (*Semantic Context Model*), um modelo ontológico independente de domínio para a representação de informações contextuais com semântica explícita e padronizado seguindo as especificações de padrão da Web semântica. O SECOM está composto por sete ontologias principais (*Actor, Activity, Space, Spatial Event, Temporal Event, Device, Time*) e mais seis ontologias de suporte (*Contact, Relationship, Role, Project, Document, Knowledge*), estas ontologias são apresentadas na Figura **??**.

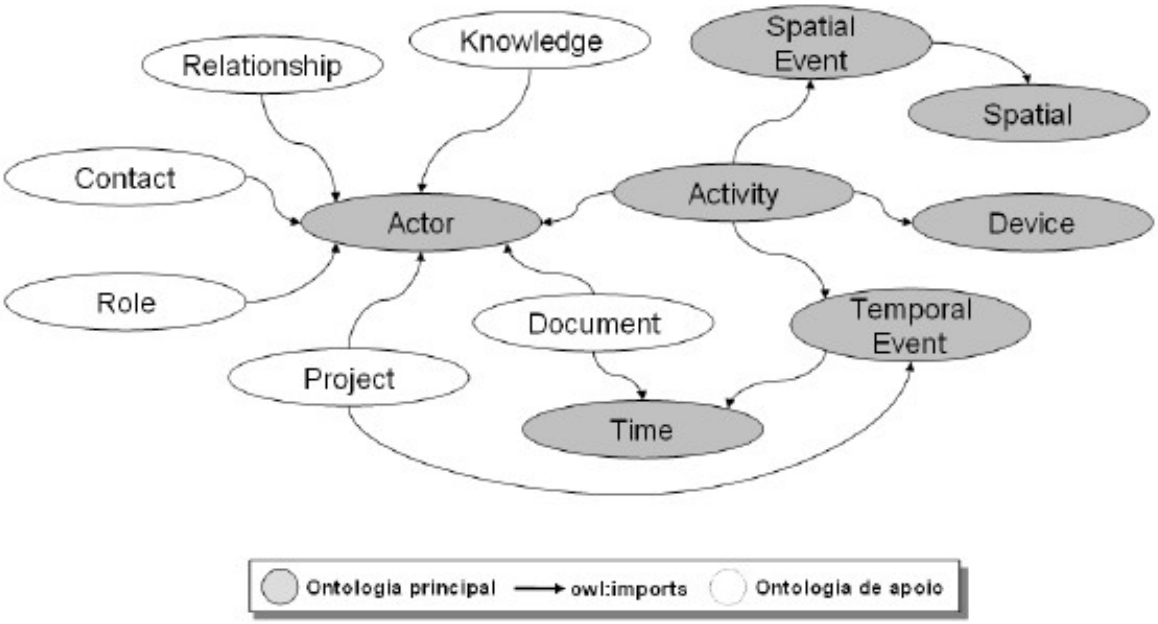


**Figura 4.3:** SECOM - *Semantic Context Model* - [23]

Já no trabalho de mestrado de [24] apresenta-se uma ontologia para modelagem de contexto (OMC). OMC foi projetado visando ambientes pervasivos e está composto por nove ontologias independentes de domínio: *Person, Space, Time, Device, ComputationalDevice, Activity, Preference, Policy* e *Service*. Este trabalho especifica que uma vez definido o tipo de ambiente pervasivo a ser utilizado, devem-se estender as ontologias necessárias. Isto é, para um ambiente pervasivo como uma casa, ontologias dependentes de domínio devem ser criadas, como por exemplo: *HomePerson HomeSpace, HomeDevice, HomeActivity* e *OpenDoorService*. Estas ontologias são apresentadas na Figura **??**.

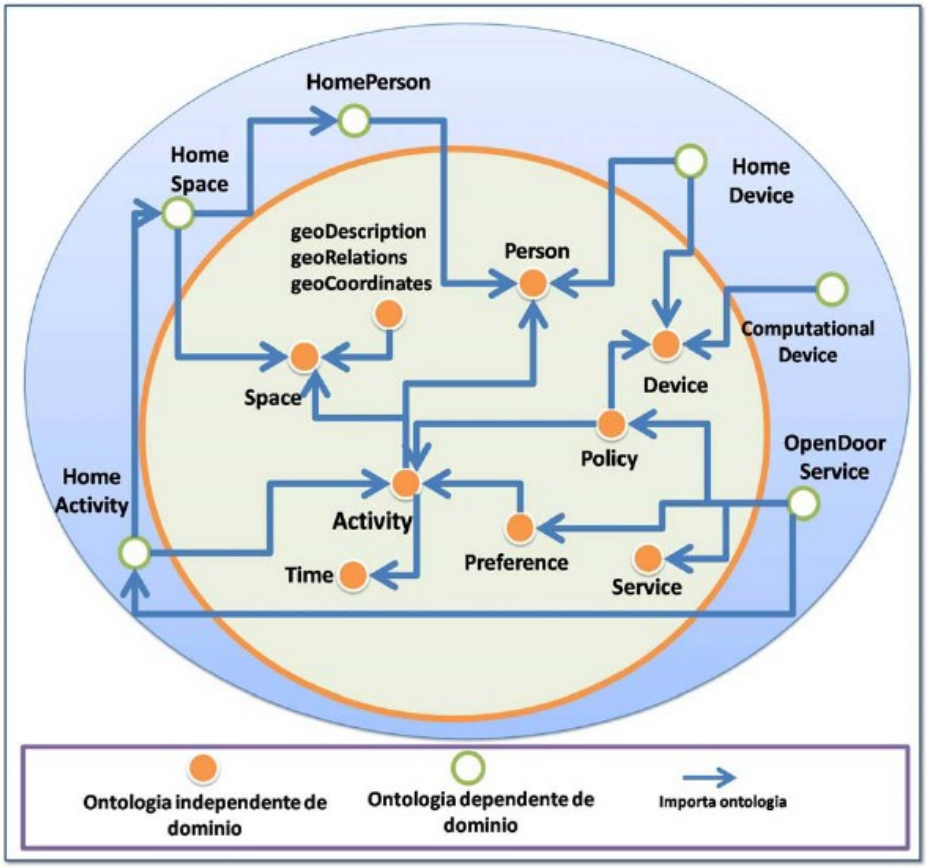


**Figura 4.4:** OMC - Ontologias para Modelagem de Contexto - [24]

Além disso, em [126] apresentam-se cinco métricas (*Precision, Freshness, Spatial Resolution, Temporal Resolution* e *Probability of correctness*) que permitem quantificar a veracidade das informações contextuais, isto é, a métrica de *Precision* descreve a granularidade das informações contextuais com a qual descreve-se uma situação do mundo real. Por exemplo, a informação de que a temperatura de um quarto é de 17,3 graus Celsius, é um nível de precisão superior à de 17 graus Celsius.

Nesse sentido, os trabalhos de [23], [24] e [126] foram estudados para estender o modelo ontológico WSMO para dar suporte de maneira formal as informações contextuais geradas no ambiente. Já as métricas serão utilizadas para otimizar o processo de composição.

- Reconhecer as requisições e desejos do usuário e trabalhar consequentemente com estes;
- Conhecer das informações relevantes ao contexto que caracterizam o estado de uma entidade (usuário, ambiente, dispositivos e aplicações);
- Em tempo real, deve se adaptar as mudanças físicas no ambiente e produzir novas funcionalidades (se for preciso);
- Permitir a escalabilidade e acessibilidade aos dispositivos e aplicações presentes no ambiente.

# 5 Aplicação do Modelo de Referência

Neste capítulo apresenta-se a aplicação do modelo de referência mediante a especificação arquitetural de quatro sistemas de composição de serviços para distintos ambientes pervasivos, além da descrição sobre os detalhes de implementação da aplicação-exemplo. Os sistemas de composição para Sala de Aula, Cidades Inteligentes e Plataforma Petroleira são descritos em inglês, isto porque, estes sistemas fazem parte de propostas de projeto de pesquisa feitas para a IBM, do programa Brasil União Europeia, e a ABB, respectivamente.

## 5.1 Sistema de Composição para Comercio Eletrônico

### 5.1.1 Contextualização

Com um ambiente empresarial cada vez mais competitivo, muitas organizações vem procurando por ferramentas e serviços de inteligência competitiva que lhes permitam obter uma vantagem competitiva sustentável frente à concorrência. Sistemas de Inteligência Competitiva (SIC) são ferramentas de gestão da informação, que permitem o monitoramento (coleta e análise) do ambiente interno e externo (produtos, concorrentes, fornecedores, reguladores, parceiros e clientes) de uma empresa para obter informações relevantes no seu processo de tomada de decisão [127] e melhorar a sua competitividade no seu nicho de mercado [128].

SICs voltadas para o comércio eletrônico, especificamente para a busca de produtos e comparação de preços, geralmente coletam os dados de fontes semi-estruturadas ou proporcionadas por fornecedores e parceiros para sua posterior análise, o que em alguns casos pode levar a tomar decisões sobre informações defasadas, isto é, as informações geradas podem já não ser validas em função do tempo transcorrido entre a coleta e a análise. Além disso, sistemas deste tipo não são totalmente automatizados, precisando que pessoas (treinadas) ajudem no processo de determinação e desambiguação dos produtos a serem coletados e analisados [129].

Propostas mais sofisticadas e recentes [130], [131], apresentam soluções de busca de

produtos e comparação de preços baseadas na utilização de semântica e ontologias para aprimorar e automatizar os processos de coleta e análise de dados. Porém, estas propostas não apresentam nenhum tipo de suporte para a tomada de decisões, focando-se no processo de identificação inequívoca de produtos. Em [128] se apresenta uma ontologia e uma arquitetura geral para um subsistema semântico de análise de dados de um SIC, entretanto nenhum tipo de resultado experimental é apresentado, de modo que permita a validação do subsistema proposto.

Neste contexto, nós propomos o desenvolvimento de um sistema de inteligência competitiva em base ao modelo de referência proposto no Capítulo 4. Nesse sentido, o sistema de composição proposto deve permitir que o usuário especifique uma requisição. Esta requisição descreve um produto o qual deve ser monitorado continuamente (coleta, organização, análise e compartilhamento) para obter informações sobre preços, disponibilidade, forma de pagamento, frete em diversos sites de comercio eletrônico que comercializem dito produto. O propósito do sistema é descobrir serviços que ofereçam o produto especificado ou serviços que proporcionem informações relevantes sobre o mesmo que possam ser utilizadas para gerar informações gerenciais confiáveis em forma de relatórios e *dashboards*, de forma rápida e atualizada (em tempo-real), visando proporcionar uma vantagem competitiva sustentável.

### 5.1.2 Especificação Arquitetural

Nós partimos da hipótese que sites de comercio eletrônico disponibilizam ontologias sobre os produtos que comercializam assim como serviços com anotação semântica, como por exemplo o projeto do *Best Buy* para anotação semântica com microformatos [132], o que permite que mecanismos de inferência possam raciocinar e construir relações sobre estes produtos. Junto às ontologias, mecanismos de agregação automática de informações podem ser desenvolvidos para anotar semanticamente os dados coletados, com o fim de automatizar o processo de de coleta e análise de produtos a serem monitorados gerando um ganho significativo na precisão e escalabilidade das informações gerenciais geradas.

A Figura 5.1 apresenta a arquitetura derivada para o sistema de composição para comercio eletrônico. O módulo de interface deve permitir especificar um produto de forma parametrizada, isto é, fazer a especificação do produto do seguinte modo: “todos os painéis solares que tem saída de 12A e custam menos de US$ 2000”. Este tipo de requisição requer que o sistema conheça a semântica implícita entre as palavras-chave descritas, de modo que consiga inferir que “12A” é uma característica que identifica um tipo de produto e que “US$ 2000” é o preço limite estipulado.

O módulo de descoberta funciona como um motor de busca semântico que procura

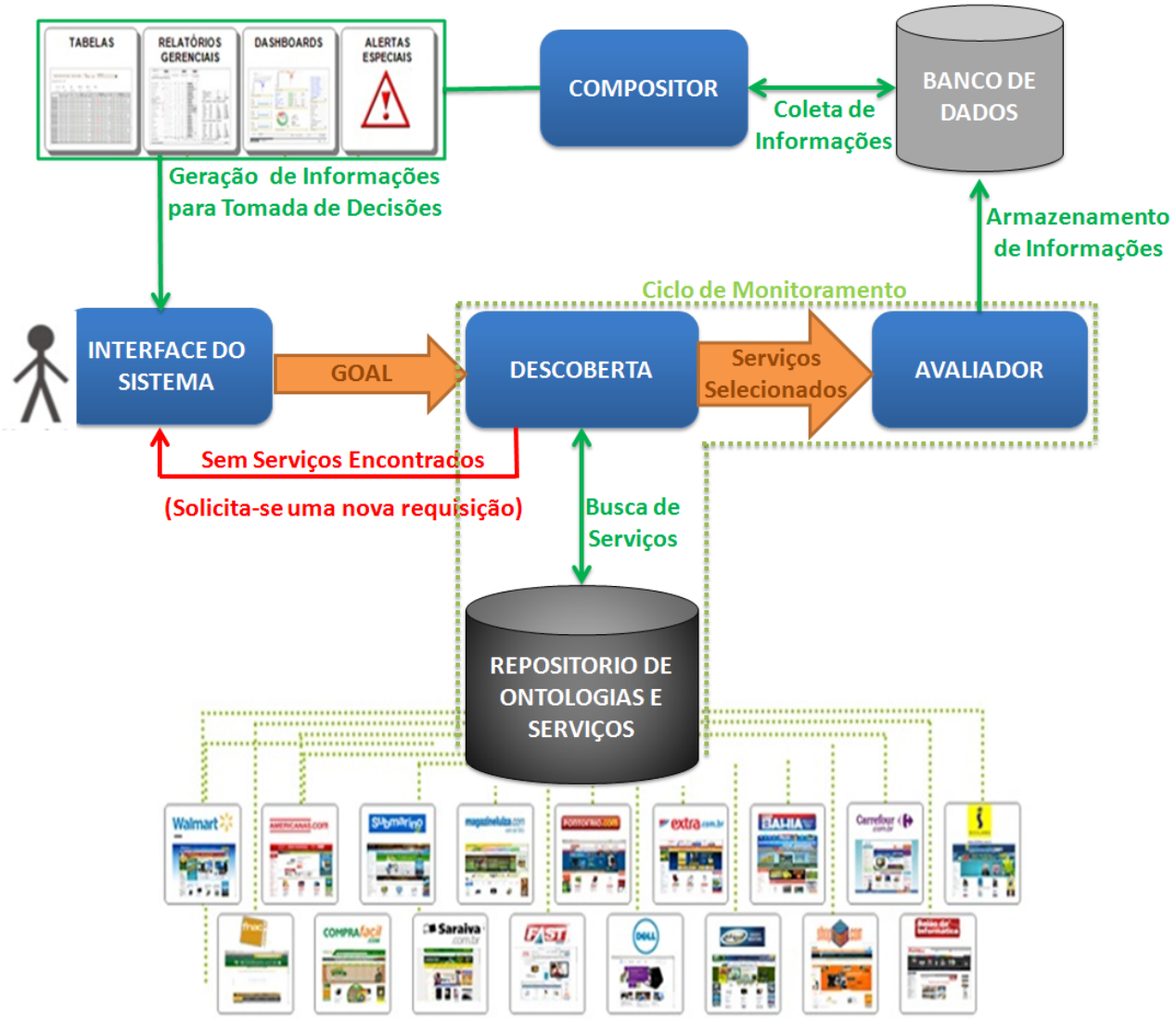


**Figura 5.1:** Arquitetura derivada para um sistema de composição para comercio eletrônico

nos diversos sites de comercio eletrônico serviços que ofereçam o produto especificado ou serviços que proporcionem informações relevantes sobre o mesmo. Os serviços e informações selecionadas são enviadas para o módulo de avaliação. O módulo de avaliação utiliza as informações parametrizadas na requisição para avaliar quais serviços e/ou informações são mais relevantes. Os serviços e informações considerados relevantes são armazenados no banco de dados, com a agregação da data e hora em que estas informações foram coletadas e analisadas. Os processos de descoberta (busca) e avaliação são executados de forma continua (ciclo de monitoramento) durante um período de tempo que pode ser estabelecido na requisição do produto no módulo de interface. Quando o ciclo de monitoramento chega a seu fim, o módulo compositor recupera as informações armazenadas no banco de dados e constrói um plano gerencial listando os serviços e informações relevantes ao produto monitorado, este plano segue uma linha de tempo o que permite descrever com maior clareza o estado do produto.

Vale destacar que o sistema de composição para comercio eletrônico apresentado é o conceito principal do projeto PIPE (Pesquisa Inovativa em Pequenas Empresas) fase 1, apresentado (protocolo 12/51035-6) à FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo).

## 5.2 Sistema de Composição para Sala de Aula

### 5.2.1 Contextualização

In the academic environment of Cloud is need to create solutions that meet the needs that arise as a result of the steps of analysis and development of the research work, steps that require the use of resources are not always available in experimental environments (laboratories). The proposal addresses use of techniques of “cloud composition service”, integrating environments IaaS to meet the needs of computing resources and software for students and teachers who make up the community of Cloud USP.

The maintenance of dozens of computers in the labs becomes a burden for the system administrator, and many times the academic community (students, professors and researchers) need different development environments in order to learning, teaching and researching. However, often these development environments are not available in laboratories or the complexity and bureaucracy of building and maintenance of dozens of different development environments leads to compromise the teaching and researching process. The difficulty in managing infrastructure for this kind of service composition is the need to use proprietary software to provide a model of composition (automatically configure, deploy applications in

the cloud etc.) generic [133].

Cloud application platform as a service (aPaaS) is a cloud service that offers development and deployment environments for application services. It's an extended application server "in the sky". An aPaaS is a general-purpose platform for building software-as-a-service (SaaS)-style applications. It implements such characteristics of cloud computing as multitenancy, elastic scalability, self-service and use tracking. It frees application designers and developers from having to reinvent this nontrivial "wheel" [134].

Our proposal describes the ability to provide the user (teacher, student) virtualized environments with a predefined architecture, exploring the use of open source tools such as OpenStack to build a cloud infrastructure as a service. In this sense, our proposal extends the concept of aPaaS for a service composition process. A cloud aPaaS composition can be seen as a process of understand a user request (professor requirements) to combine several SaaS (for example: development software tools) to build a PaaS (development environment). The PaaS provider is determined by the capabilities of the SaaS selected [135], [136].

### 5.2.2 Especificação Arquitetural

Our proposal describes the study and modeling of an environment based on Service Composition Cloud in order to create virtualized work environments managed through a service interface. Our proposal is a service composition system that offers development environments for application services. This system is a general-purpose platform for building software-as-a-service (SaaS)-style applications. It should be implements such characteristics of cloud computing as multitenancy, elastic scalability, self-service and use tracking. It frees application developers (professor or students) from having to reinvent this nontrivial software development environment. A picture **??** present the architecture developed for these system.

The principal objetives of this system are:

- Develop a service composition system for automatic creation of software development environments;
- Develop a interface for formal specification (goal) of software requirement;
- Modeling formal specification, using ontologies, about characteristics of software;
- Develop a automatic program (script) to create virtual machine with hardware specification;

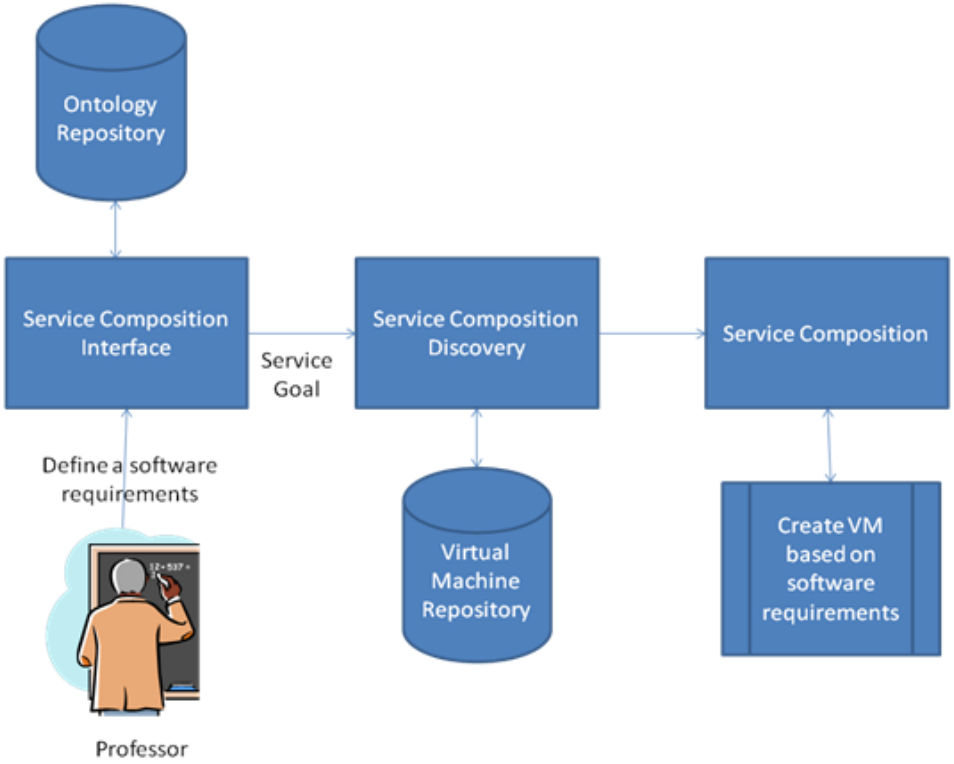


**Figura 5.2:** Arquitetura para o Sistema de Composição para Sala de Aula

- Develop a automatic program (script) to install defined software on the goal.

## 5.3 Sistema de Composição para Ambiente Domótico (SCSAD)

### 5.3.1 Contextualização

Para implementar uma aplicação-exemplo, primeiramente, nós precisamos definir e descrever o domínio da aplicação a ser implementado. Esta descrição deve especificar claramente a dinâmica das funcionalidades e características do domínio de aplicação escolhido. Dada a descrição do domínio de aplicação, nós podemos começar o processo de modelagem semântica deste domínio. A modelagem semântica do domínio envolve a criação da ontologia de domínio e dos serviços (semânticos e de invocação) correspondentes a este domínio, como se mostra na Figura 5.3.

O domínio de aplicação escolhido para nossa aplicação-exemplo foi do *Bravo Air* [137]. Este domínio representa uma companhia aérea fictícia citada na literatura da Web semântica. A dinâmica dos serviços oferecidos pelo *Bravo Air* permitem fazer a reserva de vôos. Para efetuar uma reserva, são requeridas algumas informações como: aeroporto de origem, aeroporto de

**Figura 5.3:** Esquema que Descreve a Modelagem de um Domínio de Aplicação

destino, data de ida, e se tiver retorno, a data de retorno. No caso do vôo pretendido estar disponível, uma lista de itinerários válidos para a rota escolhida é apresentada para o usuário. O usuário escolhe um itinerário com o fim de poder fazer a reserva de uma passagem. Más, para efetuar a reserva, primeiramente, o usuário precisa autenticar (validar) o usuário como um cliente da empresa. Uma vez validado, e em função das informações previamente descritas, uma reserva de passagem é efetuada no nome do usuário autenticado no sistema. Já uma descrição dos serviços oferecidos pelo *Bravo Air* foi definido em [137] e descrito a seguir:

- *Get Desired Flight Details:* este serviço recebe informações, como aeroportos (origem e destino) e datas da viagem (ida e retorno, se houver retorno), para procurar os itinerários que cumpram estes requisitos;
- *Select Available Flight:* este serviço apresenta uma lista de itinerários compatíveis com os requisitos do usuário e recebe informações sobre o itinerário escolhido;
- *Login:* este serviço valida a identidade do usuário com a base de dados do sistema;
- *Confirm Reservation:* este serviço gera uma reserva de passagem em nome do cliente em função das informações previamente informadas.

Já na Figura 5.4 é apresentada uma tabela onde são detalhadas informações sobre os parâmetros de entrada, de saída, pré-condições e pós-condições especificadas para cada um dos serviços Web descritos no domínio *Bravo Air*.

No intuito de demonstrar a composição de serviços no ambiente pervasivo, o domínio *Bravo Air* foi estendido para integrar serviços de um relojo que acorde o usuário no horário adequado para o dia da viagem, um serviço de impressão que gere o boleto eletrônico da passagem, um serviço de mapas e geolocalização que gere a rota para o aeroporto, e um serviço que ativa o sistema de alarme da casa no período da viagem.

| | *Get Desired Flight Details* | *Select Available Flight* | *Login* | *Confirm Reservation* |
|---|---|---|---|---|
| **Entradas** | AeroportoSaida, AeroportoChegada, dataPartida, dataChegada e idaVolta | itinerario | Usuario e Senha | IdReserva e Confirmação |
| **Saídas** | Não definidas | listaItinerario | Não definidas | IdReserva, ItinerarioEscolhido e Usuario |
| **Pre-condições** | **Escolher informações sobre o itinerário** | Escolher informações sobre o vôo desejado | Usuário e senha existam no sistema | Usuário esteja cadastrado no sistema (logado) |
| **Pós-condições** | **Solicitar seleção de Lista de Itinerários** | Solicitar autenticação do usuário | Usuário é cadastrado no sistema | É reservado um assento no vôo escolhido. |

**Figura 5.4:** Especificação dos Serviços do Domínio *Bravo Air*

### 5.3.2 Especificação Arquitetural

A arquitetura desenvolvida para o sistema de composição de serviços para ambiente domótico é apresentada na Figura 5.5. Cada caixa da Figura 5.5 representa um módulo do sistema e as setas correspondem a interações entre os módulos. O módulo Interface é responsável pela interação entre o usuário e o resto do sistema. Ele permite ao usuário especificar suas requisições. Essas requisições são modeladas na forma de metas ou *goals* utilizando a linguagem WSML.

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

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

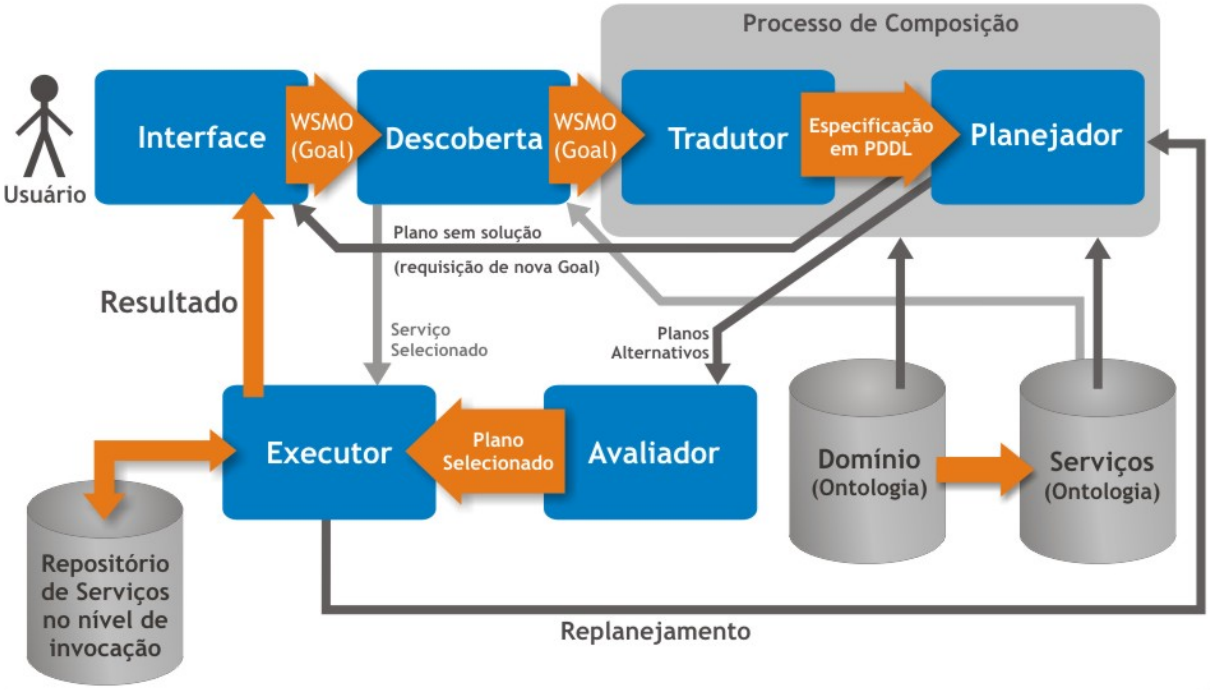


**Figura 5.5:** Arquitetura para o sistema de composição de serviços para ambiente domótico

determinar qual se adapta melhor à solicitação do usuário.

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

### 5.3.3 Implementação da Aplicação-Exemplo

#### Modelagem do Domínio *Bravo Air*

Uma das características das aplicações baseadas na Web semântica é que todos os conceitos e os relacionamentos entre conceitos devem estar claramente definidos. Portanto, nós especificamos uma ontologia de domínio chamada *Bravo Air* na qual são definidos conceitos como: aeroporto (*Airport*), itinerário (*Itinerary*), lista de itinerários (*ItineraryList*), etc., como se mostra na Figura 5.6.

Na ontologia de domínio também pode-se especificar instâncias (valores dos conceitos) como se apresenta na Figura 5.7. Estas instâncias descrevem valores específicos dados aos conceitos presentes na ontologia e que são utilizados para ajudar a construir o estado inicial do mundo, especificando por exemplo os aeroportos suportados pelo domínio.

```
Bravo_Air_Ontologia*
wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-full"
namespace { _"http://www.example.org/ontologies/Bravo_Air_Ontologia#",
            dc _"http://purl.org/dc/elements/1.1#",
            foaf _"http://xmlns.com/foaf/0.1/",
            wsml _"http://www.wsmo.org/wsml/wsml-syntax#",
            loc _"http://www.wsmo.org/ontologies/location#",
            oo _"http://example.org/ooMediator#" }
/******************************
* Ontologia
******************************/
ontology _"http://www.example.org/ontologies/Bravo_Air_Ontologia"
    nfp
        dc#title hasValue "WSML example ontology"
        dc#subject hasValue "Bravo Air - Reserva de vôos"
        dc#description hasValue "Define os conceitos do Serviço GetDiseredFlightDetails"
    endnfp

    usesMediator _"http://example.org/ooMediator"
    importsOntology { _"http://www.wsmo.org/ontologies/location",
                _"http://xmlns.com/foaf/0.1" }

concept Aeroporto
    nonFunctionalProperties
        dc#description hasValue "Define o conceito AEROPORTO"
    endNonFunctionalProperties
    nome ofType _string
    endereco ofType _string

concept ItinerarioVoo
    nonFunctionalProperties
        dc#description hasValue "Define o conceito Itinerario de Vôo"
    endNonFunctionalProperties
    nome ofType string
    id ofType integer
    aeroportoS ofType Aeroporto
```

**Figura 5.6:** Ontologia Bravo Air - Definição de Conceitos

Bravo_Air_Ontologia*

```
instance guarulhos memberOf Airport
    nome hasValue "Aeroporto Internacional de Guarulhos"
    endereco hasValue "Guarulhos"

instance jorge_chavez memberOf Airport
    nome hasValue "Aeroporto Internacional Jorge Chavez"
    endereco hasValue "Lima"

instance galeao memberOf Airport
    nome hasValue "Aeroporto do Galeao"
    endereco hasValue "Rio de Janeiro"

instance Congonhas memberOf Aeroporto
    nome hasValue "Aeroporto de Congonhas"
    endereco hasValue "São Paulo"
```

**Figura 5.7:** Ontologia Bravo Air - Definição de Instâncias

Já para a implementação desta ontologia de domínio foi utilizada a linguagem WSML [54] e a ferramenta WSMO Studio [138]. Esta ferramenta permite uma simples e rápida modelagem da ontologia de domínio. Além disso, ela oferece um visualizador (*WSMO Visualizer*) que permite visualizar de forma gráfica os conceitos inclusos na ontologia, bem como as relações entre estes conceitos, facilitando a descrição e documentação da ontologia.

Entretanto, além dos conceitos e relações modelados na ontologia de domínio, também precisa-se modelar a dinâmica de funcionamento do domínio *Bravo Air*. A modelagem desta dinâmica é feita através dos serviços Web semânticos (ontologia de serviços) associados ao domínio de aplicação. Os serviços Web semânticos modelam informações semânticas que incluem pré-condições, pós-condições, efeitos, informações sobre as propriedades não funcionais (por exemplo: qualidade do serviço) e informações sobre a coreografia, as quais dependem do modelo ontológico, neste caso o WSMO.

O processo de modelagem destes serviços Web semânticos é bastante complexo, isto porque, a forma como são modeladas as informações refletiram na forma como estes serviços são traduzidos para uma linguagem de planejamento e porque a modelagem também define qual abordagem de modelagem de domínio e problema de planejamento será implementada, isto é, "casamento de pré-condições e pós-condições entre serviços" ou "casamento de parâmetros entre serviços". Por exemplo, a Figura 5.8 apresenta a descrição do serviço Web semântico *Get*

*Desired Flight Details* associado ao domínio *Bravo Air*.

```
GetDiseredFlightDetails
wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-rule"
namespace { _"http://www.example.org/ontologies/GDFD#",
     dc _"http://purl.org/dc/elements/1.1#",
     foaf _"http://xmlns.com/foaf/0.1/",
     wsml _"http://www.wsmo.org/wsml/wsml-syntax#",
     loc _"http://www.wsmo.org/ontologies/location#",
     oo _"http://example.org/ooMediator#",
     wsmostudio _"http://www.wsmostudio.org#" }


webService _"http://example.org/Bravo_Air"
     nonFunctionalProperties
          wsmostudio#version hasValue "0.7.2"
          dc#title hasValue "Reservas de vôo."
          dc#creator hasValue "The OWL Services Coalition"
          dc#language hasValue "Inglês"
     endNonFunctionalProperties

     importsOntology
            _"http://www.example.org/ontologies/Bravo_Air_Ontologia"

capability _"http://www.example.org/ontologies/#GDFDcap_GDFD"

    precondition
         nonFunctionalProperties
              dc#description hasValue "Como entrada o Serviço Web o aeroporto de partida."
         endNonFunctionalProperties
         definedBy
              exists ?AeroportoSaida
      ( ?AeroportoSaida memberOf _"http://www.example.org/ontologies/#GDFDAeroporto" ).
```

**Figura 5.8:** Serviço Web Semântico: *Get Desired Flight Details*

Este serviço Web semântico descreve as pré-condições e pós-condições a serem satisfeitas para poder executar o serviço. Neste serviço também podem ser especificadas propriedades não funcionais que podem ser utilizadas pelo módulo avaliador para determinar a escolha do plano solução mais eficiente. Note-se que estes serviços Web semânticos, também, são especificados utilizando a linguagem WSML e ferramenta *WSMO Studio*.

Já todos os serviços Web semânticos associados ao domínio *Bravo Air* precisam ser traduzidos pelo módulo tradutor para uma linguagem de planejamento (por exemplo, PDDL) e para uma linguagem que permita a execução dos serviços Web (por exemplo, WSDL). Nesse sentido, e como parte da especificação da nossa ferramenta para este primeiro estudo de caso, nós modelamos o domínio e problema de planejamento e os serviços Web executáveis de forma que se refletisse, o mais adequadamente possível, o serviço Web semântico previamente modelado para o domínio de aplicação. Isto é, em base à modelagem do domínio de aplicação, nós construímos de forma manual os arquivos equivalentes para um linguagem de planejamento e para uma linguagem de execução de serviços. O qual nós permitiu definir qual deveria ser o formato ou o mecanismo (algoritmo) que nosso módulo tradutor deve implementar para criar automaticamente os domínios e problemas de planejamento, assim como os serviços web

executáveis.

Já para definir o modelo padrão a ser traduzido para o domínio e problema de planejamento equivalente ao serviço Web semântico, nós criamos uma tabela que expressasse a equivalência entre os termos em WSML e PDDL (como descrito no capítulo 3 deste relatório). Nesse sentido, o nome do domínio é traduzido em base ao nome definido na ontologia de domínio. Todos os serviços Web semânticos associados a esta ontologia de domínio são traduzidos como ações deste domínio de planejamento. Por exemplo, os nomes dos serviços Web são equivalentes aos nomes das ações, isto é, todo serviço Web é traduzido em uma ação. Os *types* (tipos) são definidos em função dos conceitos definidos na ontologia de domínio, por exemplo: Aeroporto. Já os predicados, do domínio de planejamento, são criados pelas pré-condições, pós-condições e efeitos definidos nos serviços Web. Por motivos de descrição da implementação, coloca-se como prefixo PreCondition, PostCondition e Effect mais o nome do serviço Web, por exemplo: PreConditionGetDesiredFlightDetails, PostConditionLogIn, EffectConfirmReservation, etc., como se apresenta na Figura 5.9. Porém, este formatação não é obrigatória.

```
(:predicates    (PreConditionGetDesiredFlightDetails    ?DepartureAirport ?ArrivalAirport - Airport
                                                        ?OutboundDate ?InboundDate - FlightDate
                                                        ?RoundTrip - RoundTrip
                )
                (PostConditionGetDesiredFlightDetails   ?OutGetDesiredFlightDetails - OutGetDesiredFlightDetails)

                (PreConditionSelectAvailableFlight      ?PreferredFlightItinerary - FlightItinerary)
                (PostConditionSelectAvailableFlight     ?AvailableFlightItineraryList - FlightItineraryList)

                (PreConditionLogIn      ?AcctName - AcctName ?Password - Password)
                (PostConditionLogIn     ?OutLogIn - OutLogIn)

                (PreConditionConfirmReservation ?ReservationID - ReservationNumber ?Confirm - Confirmation)
                (PostConditionConfirmReservation        ?PreferredFlightItinerary - FlightItinerary
                                        ?AcctName - AcctName ?ReservationID - ReservationNumber
                )
                (EffectConfirmReservation       ?HaveFlightSeat - HaveSeat)
)
```

**Figura 5.9:** Definição dos Predicados no Arquivo PDDL Traduzido

Entretanto, para definir o modelo padrão a ser traduzido para implementar os serviços Web executáveis (WSDL) equivalente ao serviço Web semântico, nós utilizamos o algoritmo desenvolvido em colaboração do projeto de TCC (Trabalho de Conclusão de Curso) da aluna Keith de Souza [139]. Este algoritmo mapeia automaticamente serviços Web semânticos descritos em WSML para serviços Web descritos em WSDL. Este mapeamento é chamado de mecanismo *Grounding* [11].

Entenda-se o conceito de *Grounding* como o mapeamento entre descrições semânticas e sintáticas. O *Grounding* (ou mapeamento) de um serviço especifica os detalhes de como acessar o serviço, o tipo de protocolo a ser utilizado, formatos de mensagem, etc. Ele realiza o mapeamento entre a descrição dos serviços semânticos (descrição abstrata do serviço ou nível semântico) com um serviço Web concreto (WSDL ou nível de invocação), ao relacionar as

entradas e saídas dos serviços semânticos com os formatos de mensagens concretas do serviço Web [11].

Em resumo, o *Grounding* deve mapear a interface do serviço em WSML para as definições de mensagens em WSDL e detalhes de comunicação, como por exemplo: tipo de protocolo (SOAP, HTTP), como os dados XML são encapsulados, etc. Este tipo de grounding pode ser especificado com enlaces (links) desde a descrição de WSML, ou usando anotações semânticas em WSDL (por exemplo: SAWSDL) [139]. Nesse sentido, no presente trabalho foram utilizadas as anotações semânticas SAWSDL para especificar o mecanismo *Grounding*, isto porque, a ferramenta *WSMO Studio* provê esta funcionalidade. Portanto, nos criamos os serviços Web (WSDL), e os anotamos semanticamente utilizando a ferramenta *WSMO Studio* [138], que permite editar os arquivos WSDL, agregando notações semânticas, como se apresenta na Figura 5.10. Logo, utilizando a ferramenta Netbeans e o plugin Axis, nós importamos estes arquivos WSDL para empacota-los no servidor Web que nos permitira executar posteriormente através do módulo executor.

```
WSDL_GetDiseredFlightDetails*
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://DefaultNamespace" xmlns:apachesoap="http://xml.apache.org/xml-soap"
                  xmlns:impl="http://DefaultNamespace" xmlns:intf="http://DefaultNamespace"
                  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:sawsdl="http://www.w3.org/ns/sawsdl">
<!--WSDL created by Apache Axis version: 1.4
Built on Sep 22, 2009 (03:55:48 PDT)-->

<wsdl:types>
  <schema elementFormDefault="qualified" targetNamespace="http://DefaultNamespace"
          xmlns="http://www.w3.org/2001/XMLSchema">
   <element name="getFlightInformation"/>
    <complexType>
     <sequence>
      <element name="aeroportoSaida" type="xsd:string"
        sawsdl:modelReference="http://www.example.org/ontologies/Bravo_Air_Ontologia#Aeroporto"/>
      <element name="aeroportoChegada" type="xsd:string"
        sawsdl:modelReference="http://www.example.org/ontologies/Bravo_Air_Ontologia#Aeroporto"/>
      <element name="dataSaida" type="xsd:dateTime"/>
      <element name="dataChegada" type="xsd:dateTime"/>
      <element name="idaVolta" type="xsd:boolean"
        sawsdl:modelReference="http://www.example.org/ontologies/Bravo_Air_Ontologia#IdaVolta"/>
     </sequence>
    </complexType>
   </element>
   <element name="getFlightInformationResponse">
    <complexType>
     <sequence>
      <element name="getFlightInformationReturn" type="xsd:boolean"
        sawsdl:modelReference="http://www.example.org/ontologies/Bravo_Air_Ontologia#Resposta"/>
     </sequence>
    </complexType>
   </element>
  </schema>
```

**Figura 5.10:** Serviço Web *Get Desired Flight Details* Anotado Semanticametne (SAWSDL)

Note-se que, o conceito de mecanismo *Grounding* é um assunto bastante complexo, ao ponto que a única especificação formal existente para o mecanismo *Grounding* é proprietária e não disponível para sua utilização. Portanto, o trabalho desenvolvido pela aluna Keith de Souza, pode ser entendido como o ponto de partida para uma definição formal do mecanismo *Grounding*, porém, este precisa de uma discussão bem mais aprofundada, algo que esta sendo

planejado como trabalhos futuros a serem desenvolvidos, em especial utilizando o conceito de compiladores como feito no módulo tradutor.

Agora, com todos os elementos disponíveis (serviços Web, serviços Web semânticos e domínio e problema de planejamento modelados), o seguinte passo é implementar o módulo de interface.

### Módulo Interface

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

### Interface Gráfica de Gerenciamento

Esta interface gráfica permite que um usuário autorizado (p. ex. administrador) faça o *upload* e armazenamento de documentos WSML (ontologias, serviços web e mediadores) nos repositórios (banco de dados) do SCSAD. Estes documentos WSML são associados a um domínio de aplicação (p. ex. viagens). Um domínio de aplicação é como um tema (categoria) de aplicações que tratam informações e funcionalidades similares. Para facilitar o uso do SCSAD por usuários não técnicos, as ontologias submetidas devem ser associadas a ícones representativos, desta forma a interface gráfica de requisição de metas torna-se graficamente mais intuitiva. A Figura 5.11 representa a interface gráfica gerenciamento.

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

### Interface Gráfica de Requisição de Metas

A interface gráfica de requisição de metas permite que qualquer usuário especifique uma requisição ao SCSAD. Esta interface gráfica deve ser: dinâmica, isto é, o formulário apresentado deve corresponder ao domínio de aplicação escolhido; e com um alto grau de

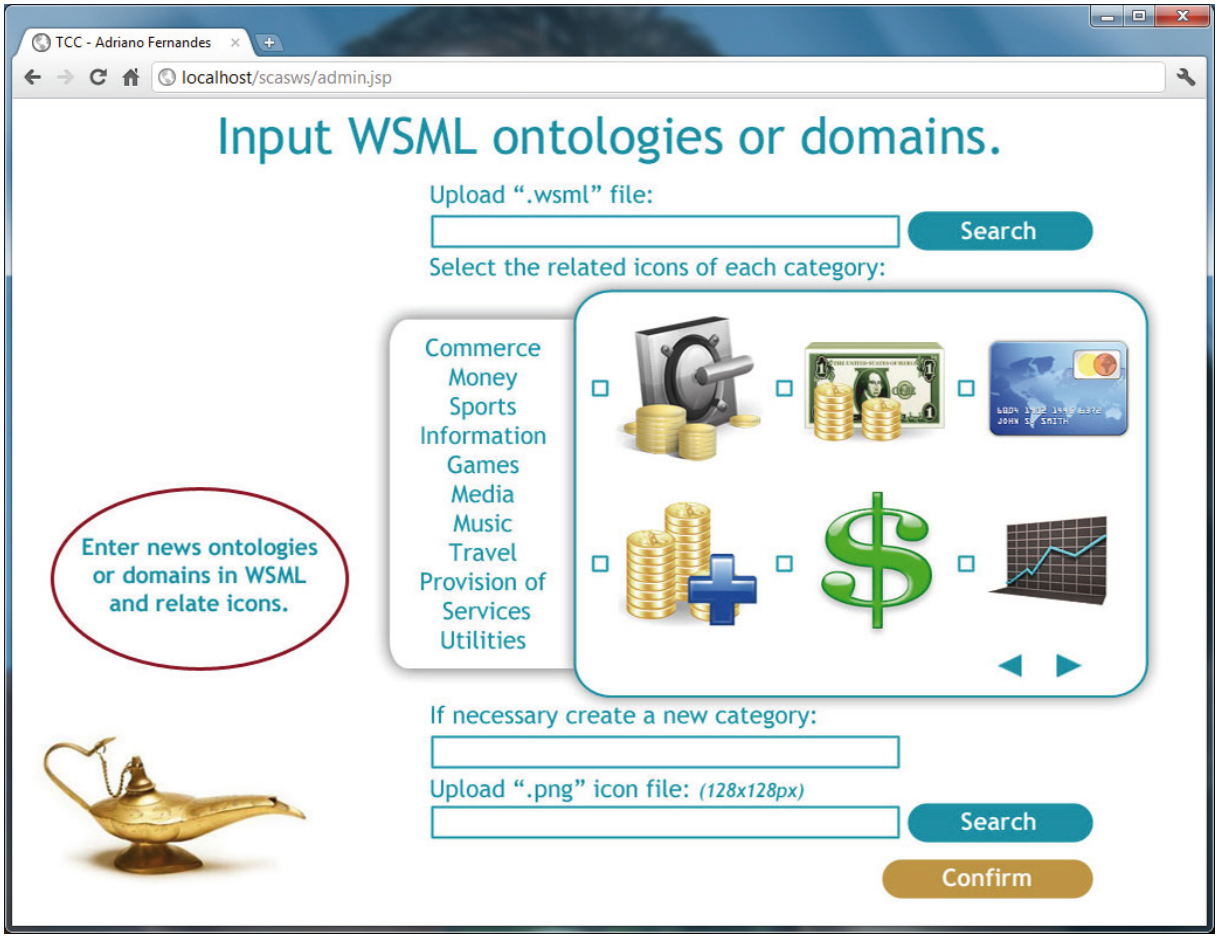


**Figura 5.11:** Interface Gráfica de Gerenciamento

abstração, isto é, que usuários não técnicos possam especificar suas requisições de forma simples e clara.

Nesse sentido, a primeira tela (formulário) desta interface gráfica permite a escolha de uma categoria (domínio de aplicação). A escolha da categoria gera uma lista de ícones representativos das ontologias e serviços web associados a esta categoria, como representado na Figura 5.12A. A seleção destes ícones representa o refinamento da requisição do usuário. Por exemplo, para descrever que o usuário quer viajar, ele escolhe a categoria correspondente e entre os ícones associados ele pode refinar seu desejo indicando que quer pagar com cartão de credito, que o médio de transporte seja de trem e que as passagens sejam enviadas pelo serviços de postagem local, como pode ser observado na parte superior central da Figura 5.12A.

Ao ser selecionado o botão "Next Step", o módulo Interface gera uma segunda tela (formulário) a qual permite especificar (escolher em alguns casos) os valores (instâncias das ontologias) dos tipos de dados associados aos ícones escolhidos. Por exemplo, para o domínio de aplicação "Viajar", e a seleção dos ícones viagem de trem e modo de entrega das passagens, o formulário gerado contem informações como a cidade origem, cidade destino, horário da viagem e serviço de entrega das passagem, como pode ser observado na Figura 5.12B.

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

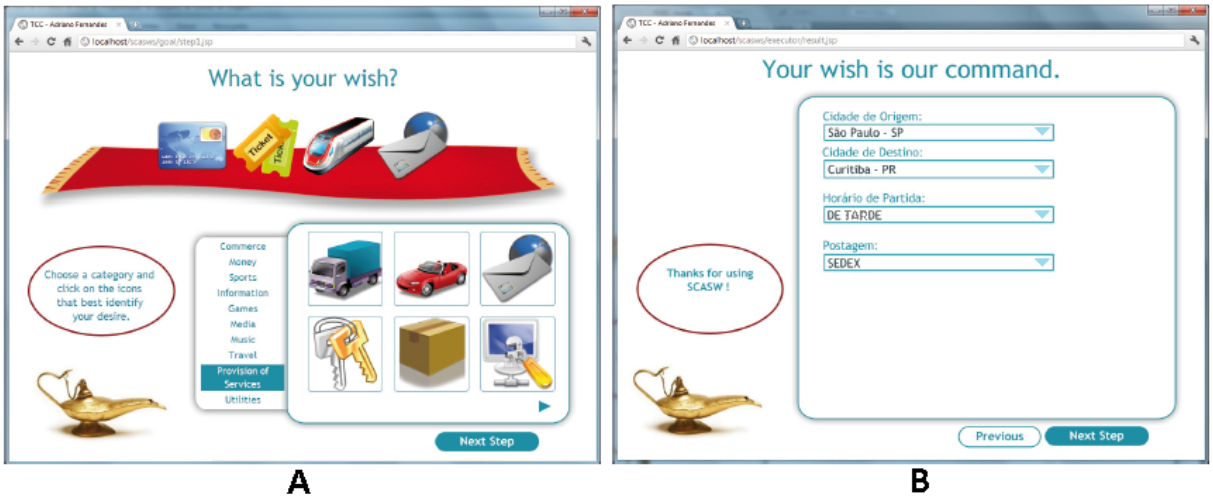


**Figura 5.12:** Interface Gráfica de Requisições de Metas

### Interface Gráfica de Resultados

Em base à arquitetura do SCSAD, a interface gráfica de resultados é responsável por apresentar ao usuário o resultado gerado pelo módulo Executor. A construção da tela (formulário) que contem as informações dos serviços web executados, segue o mesmo conceito da interface gráfica de requisição de metas, isto é, são utilizados os objetos Java associados às pós-condições e efeitos dos serviços web inclusos no plano solução e executados pelo módulo Executor.

Para o exemplo da subseção anterior, o resultado gerado informa as cidades de origem e destino, o horário da viagem, o valor e forma de pagamento da passagem, e o valor e o forma de entrega das passagem, como pode ser observado na Figura 5.13. Caso o módulo Planejador ou Executor não consigam gerar ou executar o plano, respectivamente, o fluxo do processo é devolvido à interface gráfica de requisição de metas.

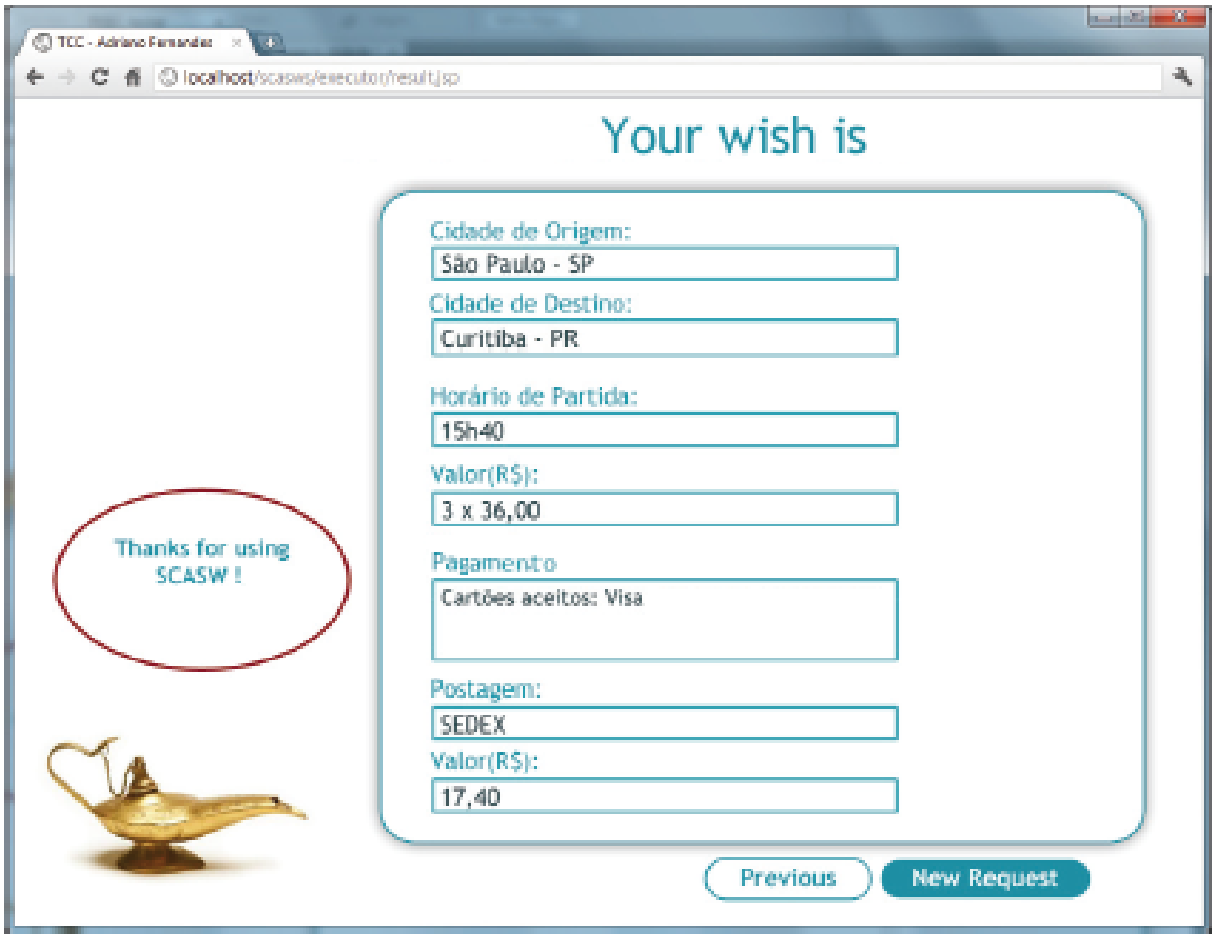


**Figura 5.13:** Interface Gráfica de Resultados

De forma geral, o módulo Interface para o usuário final do SCSAD apresenta a seguinte sequência funcional, como mostrado na Figura 5.14.

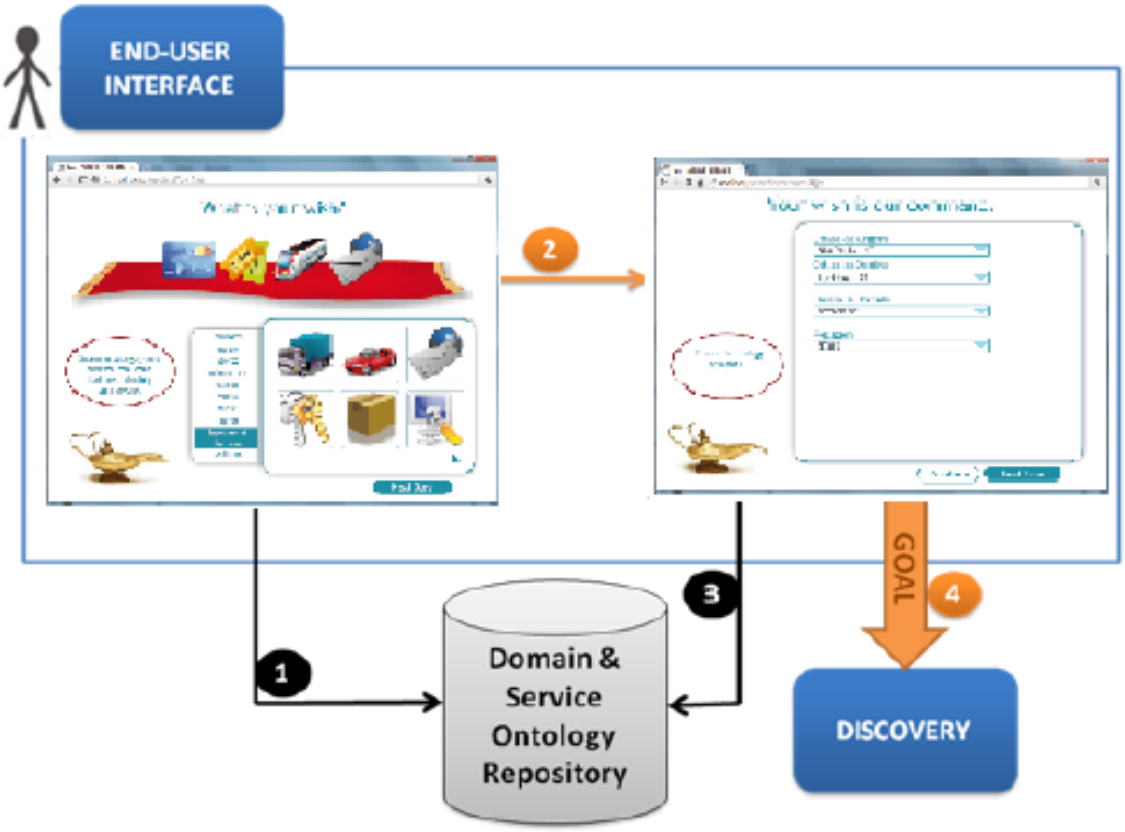


**Figura 5.14:** Sequência Funcional do Módulo de Interface do Usuário Final

### Repositório de Ontologias

Cada ícone da interface de usuário final apresentado na Figura 5.12A representa um conceito de uma ontologia. Os ícones podem corresponder a conceitos de diferentes ontologias sendo que estas estão armazenadas no repositório de ontologias. Para criar a interface de usuário final de forma dinâmica o sistema utiliza o repositório de ontologias para carregar as imagens gráficas (ícones) associadas às ontologia de domínio que fazem parde do domínio da aplicação. As ontologias utilizadas para nossa aplicação-exemplo foram Localização (locationOntology), Pagamento (paymentOntology), Reserva de viagem (tripReservationOntology) e Veículos (vehicleOntology).

As ontologias criadas foram armazenadas em um banco de dados para que pudessem ser recuperadas quando necessário. Para apoiar este processo, foi utilizada a API WSMO4j. Esta API atua como um conversor que permite criar objetos Java do tipo Ontologia, Conceito, etc., os quais são instanciados a medida que o arquivo WSML é lido. A partir destes objetos Java, as informações são armazenadas no banco de dados. Porém, esta API apresenta algumas deficiências que dificultaram o processo de armazenamento de ontologias. Já o fluxo de armazenamento de ontologias á apresentado na Figura 5.15.

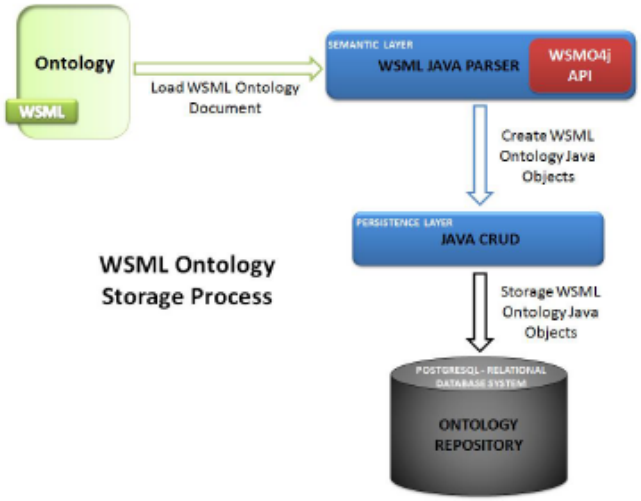


**Figura 5.15:** Fluxo de processamento para carregar uma ontologia no repositório

Já o modelo entidade-relacionamento do banco de dados do SCSAD é apresentado na Figura 5.16.

A interface de gerenciamento do sistema é responsável por alimentar o repositório de ontologias e domínios do SCSAD. Através dela, o administrador relaciona o conteúdo semântico contido nas ontologias com ícones que representam metaforicamente os conceitos que as ontologias expressam. Como mecanismo de segurança a interface de gerenciamento requer realizar o login no sistema.

### Módulo Tradutor

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

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

A gramática de representação do mecanismo de tradução mapeia a equivalência de termos

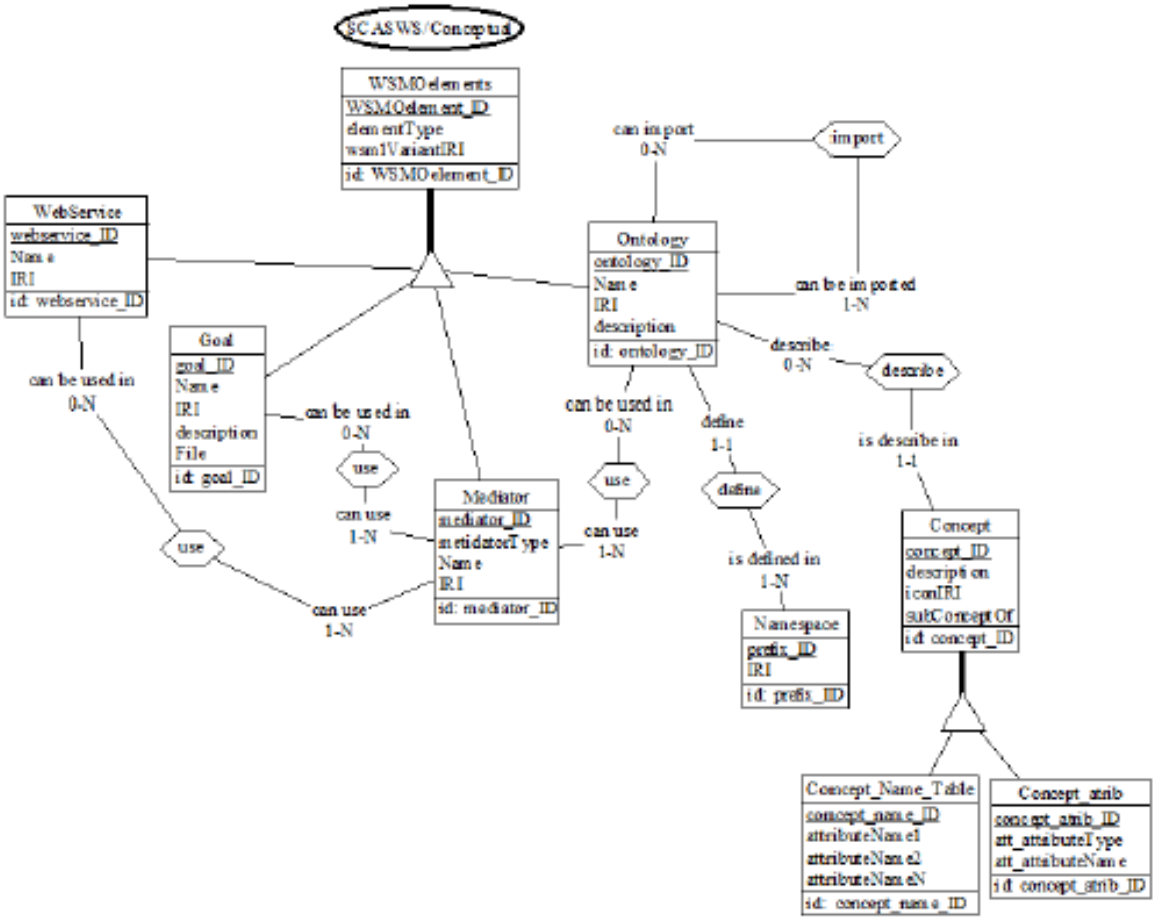


**Figura 5.16:** Modelo entidade-relacionamento do SCSAD

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

Para validar os arquivos PDDL gerados, foi projetada uma gramática que representa a linguagem PDDL. Assim, os arquivos PDDL gerados são submetidos à ferramenta Antlr IDE, a qual produz uma árvore sintática em conformidade aos arquivos submetidos, validando a tradução realizada. Todo este fluxo de processamento é representado na Figura 5.17.

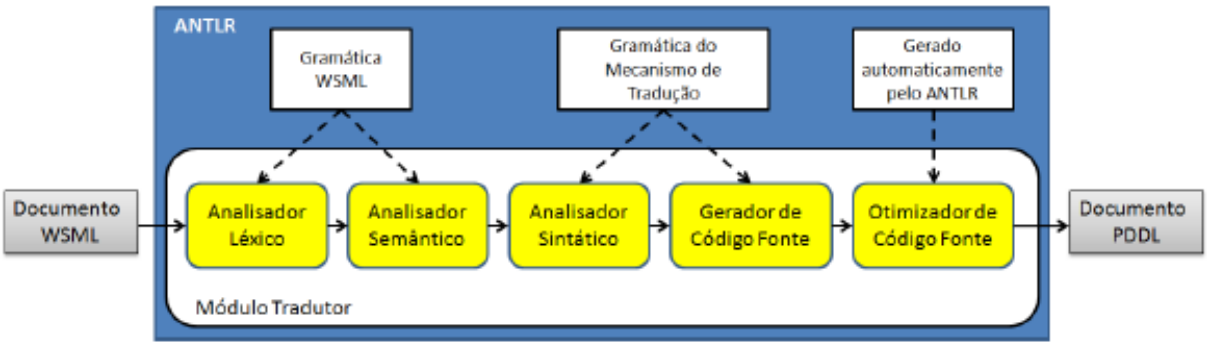


**Figura 5.17:** Mecanismo de tradução do módulo tradutor

### Módulo Compositor

O SCSAD como foi definido na subseção 5.3.1 oferece quatro serviços que descrevem a funcionalidade de uma companhia aérea fictícia, e mais outros quatro serviços que descrevem a funcionalidade de um ambiente pervasivo. Nesse sentido, no decorrer dos seguintes parágrafos é especificado um problema, baseado no *Bravo Air*, que deve ser resolvido e executado pela SCSAD.

O problema proposto deve solicitar ao SCSAD que reserve uma passagem avião e faça o planejamento necessário para executar a viagem. Nesse sentido, para atingir esta requisição, no módulo de interface de usuário final foi escolhido o domínio *Bravo Air*, e logo a meta *Have Flight Seat*. Dadas estas escolhas, apresenta-se uma interface (*Bravo Air User Interface*) que permite ingressar informações como cidade origem, cidade destino, datas da viagem, como se apresentou na Figura 5.12.

Já para reservar uma passagem, o sistema precisa verificar a existência do vôo e do itinerário desejado, assim como verificar o cadastro do usuário no banco de dados (autenticação do usuário), o qual é descrito na especificação da meta (*goal*) na figura anterior. Nesse sentido,

com o domínio e problema de planejamento modelados pelo módulo tradutor, o planejador Jshop2 utilizado, gera um plano solução correspondente como apresentado na Figura **??**.

```
(!GET_DESIRED_FLIGHT_DETAILS JUAN_CARLOS SAO_PAULO GUARULHOS)
(!SELECT_AVAILABLE_FLIGHT GUARULHOS JORGE_CHAVEZ )
(!GET_DESIRED_FLIGHT_DETAILS JUAN_CARLOS LIMA JORGE_CHAVEZ)
(!SELECT_AVAILABLE_FLIGHT JORGE_CHAVEZ RODRIGUEZ_BALLON)
(!GET_DESIRED_FLIGHT_DETAILS JUAN_CARLOS AREQUIPA RODRIGUEZ_BALLON)
(!LOG_IN JUAN_CARLOS)
(!CONFIRM_RESERVATION JUAN_CARLOS GUARULHOS JORGE_CHAVEZ RODRIGUEZ_BALLON)
```

**Figura 5.18:** Plano gerado para o SCSAD pelo JSHOP2

Os planos gerados devem ser interpretados e executados pelo módulo executor. Nesse sentido, os oito serviços oferecidos pelo domínio *Bravo Air* adaptado, foram implementados como serviços Web utilizando a linguagem WSDL. Então, quando o módulo executor lê um plano e encontra o nome de uma ação, ele associa este nome ao nome do serviço Web e repassa os parâmetros da ação como parâmetros de entrada para o serviço Web.

### Executor

O Módulo Executor é responsável por receber um fluxo de trabalho (Workflow) e executá-lo. Uma nova *thread* é criada para invocar e executar, de forma simultânea, cada uma das atividades do Workflow. A versão atual do mecanismo executor permite a execução de três tipos de atividades: serviços web, métodos escritos em Java e aplicações locais. Além da execução de fluxos de trabalho, este módulo permite ao usuário criar, importar, editar e exportar um Workflow.

O Módulo Executor é composto de atividades, cada atividade contém um conjunto de conectores (entradas e saídas). Existem dois tipos possíveis de fluxos entre atividades: fluxo de dados, quando a entrada de uma atividade recebe a saída de outra atividade; e controle de fluxo, quando uma atividade deve ser executada somente após a outra atividade terminou sua execução. Uma interface chamada *Activity Executor* fornece a interface para a execução de cada tipo de atividade permitida no sistema.

O Módulo Executor tem uma função de verificação, que é responsável por verificar três aspectos do Workflow para garantir que está pronto para ser executado. Os aspectos são: (i) a compatibilidade sintática e semântica dos fluxos de dados, que verifica se cada fluxo de dados é sintaticamente (ou seja, eles usam tipos de dados compatíveis) e semântica (isto é, eles usam conceitos que são ontologicamente compatíveis) compatíveis; (ii) ausência de dependências circulares entre as atividades, a fim de evitar ciclos infinitos, e (iii) o enchimento de todos os dados de entrada de atividades (uma entrada de dados pode ser enchido com a produção de

outra atividade, com a informação de entrada recebida a partir do módulo de interface ou por do utilizador).

O Módulo Executor recebe um Workflow e executa cada atividade em uma nova *thread*. As atividades são executadas simultaneamente, respeitando as dependências (dados e fluxos de controle). Todos os tipos de atividades são invocados e/ou executadas de forma dinâmica, a fim de fazer isso, o mecanismo de workflow utiliza o conceito Reflexão.

Outra funcionalidade do Módulo Executor é receber um plano (um conjunto ordenado de tarefas produzidas pelo planejador) e criar um Workflow baseado neste plano. A criação do Workflow usa as seguintes etapas: cada tarefa do plano é convertido para uma atividade no Workflow usando a mesma ordem em que aparecem no plano. Para cada entrada da última atividade criada, o sistema tenta encontrar uma base de dados compatíveis na produção das atividades anteriores. Após esta criação automática o usuário é convidado a preencher as entradas restantes e para verificar se os fluxos foram criados corretamente. Este mecanismo de execução de Workflows foi desenvolvido pelo professor Luciano Digiampietri, e adaptado para a execução do sistema proposto.

Figura **??**A apresenta o Workflow resultante de exemplo o *Bravo Air*. O Módulo compositor produziu um plano com a seguinte sequência de serviços web: *getFlightInformation*, *selectFlight*, login e confirmação. Este plano responde à solicitação do usuário para um serviço composto com que ele pode visualizar voos disponíveis, selecionar uma e confirmar o voo.

O plano é lido pelo Executor e o Workflow apresentado na Figura **??**A é produzido. Nesta figura, cada retângulo é uma atividade (neste exemplo, todas as atividades são serviços web), as setas cinza são fluxos de controle e as setas pretas são fluxos de dados. O usuário é convidado a preencher as entradas de fluxos de trabalho, a fim de definir alguns parâmetros, como aeroporto de partida e aeroporto de chegada. A Figura **??**B apresenta a atividade *getFlightInformation*.

O Workflow é executado como se segue: (i) a atividade *getFlightInformation* é executada com os dados introduzidos pelo utilizador preencheram no módulo de interface. Em seguida, o usuário seleciona o voo desejado entre os disponíveis (esta seleção é a entrada da atividade *selectFlight*). O resultado da execução desta última atividade é utilizada como entrada da atividade confirmação, mas esta atividade pode ser executada somente após o usuário estar logado no sistema, isto é a atividade login.

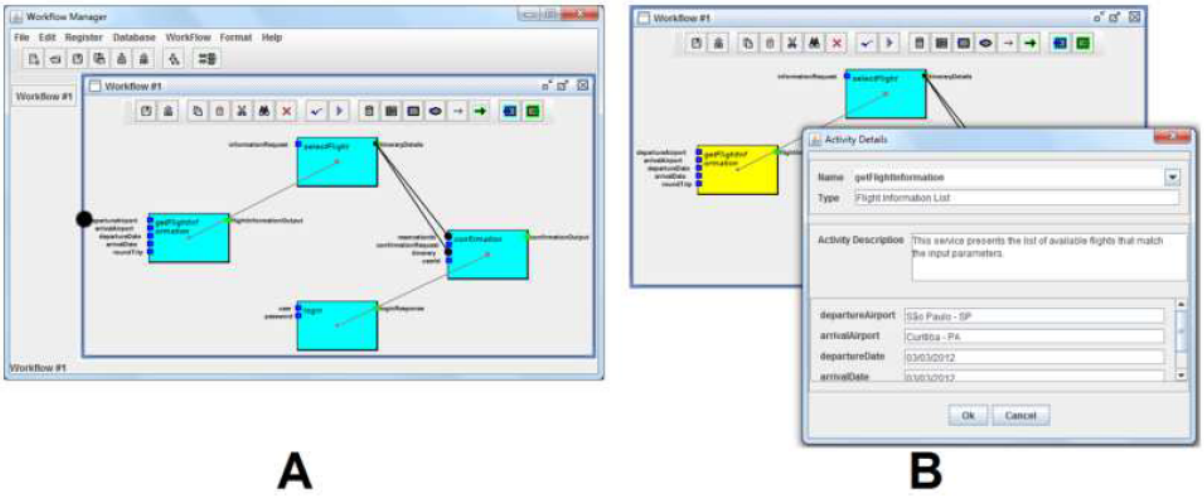


**Figura 5.19:** Workflow de execução da aplicação-exemplo

# 6 Considerações Finais

Este capítulo apresenta os resultados e considerações gerados a partir dos aspectos práticos de implementação da aplicação-exemplo. Além disso, neste capítulo também são apresentadas as contribuições atingidas, as conclusões inferidas no presente trabalho. Já no final deste capítulo são apresentados alguns tópicos a serem desenvolvidos como trabalhos futuros.

## 6.1 Contribuições

De forma geral este trabalho procura incentivar e facilitar o desenvolvimento de sistema de composição de serviços em ambientes pervasivos, sendo a principal contribuição os elementos teóricos e conceituais descritos no modelo de referência, fato que justifica a extensão do levantamento bibliográfico e escrita do presente trabalho. Outras contribuições atingidas pelo presente trabalho são sumarizadas a seguir:

- Como resultado do levantamento bibliográfico e estudo feito sobre a composição de serviços Web, no Capítulo 2 foi desenvolvida, apresentada e publicada uma arquitetura para a composição automática de serviços web baseada em planejamento [10]. Está arquitetura foi o ponto inicial para o desenvolvimento do modelo de referência proposto;

- Em base ao estudo e análise do estado da arte sobre a composição de serviços Web em ambientes pervasivos, no Capítulo 3 foi desenvolvida e apresentada uma matriz de classificação que identificar as características das diversas proposta e facilita a comparação entre elas. Esta matriz também auxilia na especificação de requisitos na hora de projetar um novo sistema de composição de serviços para ambientes pervasivos;

- Em base ao estudo e análise dos trabalhos relacionados sobre a formalização de sistemas composição de serviços Web em ambientes pervasivos feito no Capítulo 3, detectou-se a falta de um modelo formal e genérico o suficiente que facilite o desenvolvimento deste tipo de sistemas, nesse sentido, o modelo de referência apresentado no Capítulo 4 vem a

suprir está carência, facilitando e orientando o desenvolvimento de sistemas composição de serviços Web para diversos tipos de ambientes pervasivos;

- Já o modelo de referência proposto apresenta as vantagens de re-utilização e baixo acoplamento, isto porque, com a descrição formal dos componentes através de relações (aspectos de associação), interações (semântica de operações) e restrições (regras de validação) é possível derivar arquiteturas de sistemas de composição para diversos tipos de ambientes pervasivos, como apresentado no Capítulo 5;

- A linguagem OMG SysML [48] é uma linguagem de modelagem de propósito geral para aplicação em engenharia de sistemas, especialmente utilizada no projeto de sistemas embarcados, nesse sentido, a especificação do nosso modelo de referência utilizando está linguagem traz a contribuição de apresentar a utilização desta linguagem para projetos integrados de hardware, software e interação de usuários em ambientes pervasivos;

- Em base aos aspectos práticos de implementação da aplicação-exemplo, as seguintes contribuições técnicas foram atingidas:

  – Como apresentado no Capítulo 5, foi desenvolvida uma extensão do modelo ontológico WSMO para dar, formalmente, suporte às informações contextuais geradas pelo ambiente pervasivo. Esta ontologia pode ser utilizada no desenvolvimento de outros tipos de aplicações na área de pesquisa da computação ciente de contexto;

  – Como apresentado no Capítulo 5, foi utilizado o conceito de compiladores e a ferramenta Antlr para implementar o processo de tradução entra a linguagem de especificação de metas e serviços e a linguagem utilizada pelo mecanismo de inferência. A construção da gramatica que permite traduzir a linguagem WMSL para PDDL demonstra e motiva a utilização deste conceito e ferramenta para o processo de tradução para outras linguagens. Além disso, esta abordagem difere da comumente implementada por outros trabalhos que utilizam tabelas de equivalências ou algoritmos específicos para fazer a tradução, o que torna está tarefa mais complexa e pouco reutilizável;

  – Ao compararmos as interfaces propostas nos trabalhos descritos no estado da arte (Capítulo 3) com a interface de usuário final desenvolvida no Capítulo 5, fica evidente a contribuição atingida pela implementação do nosso módulo Interface, isto porque a interface gráfica desenvolvida é mais intuitiva e facilita o uso por parte de usuários sem conhecimentos técnicos;

- Já o modelo entidade-relação e conceitual do banco de dados desenvolvido para o armazenamento das ontologias e serviços Web semânticos apresentado no Capítulo 5, junto à API WSMO4j permitiram a implementação dinâmica de interfaces gráficas, o que facilita o reuso de uma mesma interface por distintos tipos de ambientes pervasivos. Este modelo de armazenamento também visa a possibilidade de fácil compartilhamento de informações para outros tipos de aplicações.

## 6.2 Limitações do Trabalho

- Mesmo que no cenário geral de aplicação apresentado no Capítulo 4 mostre a utilização de diversos dispositivos sensores, atuadores, eletro-eletrônicos e dispositivos móveis, a implementação da aplicação-exemplo é restrita a uns poucos dispositivos pelo fato da indisponibilidade dos mesmos;

- Embora o mecanismo de inferência influencie diretamente na eficiência de um sistemas de composição de serviços, a implementação ou extensão de um planejador ou outro mecanismo de inferência, fica fora do espoco do presente projeto, de modo que é utilizado um planejador disponível na literatura.

## 6.3 Conclusões

- Conforme a análise feita no levantamento bibliográfico, resolver a problemática de mecanismos automáticos que interajam e satisfaçam as necessidades do usuário de forma natural e transparentes em diversos ambientes pervasivos através de sistemas de composição automática de serviços em ambientes pervasivos apresenta-se como uma solução viável, pratica e adequada em função da maturidade e avanços das tecnologias envolvidas (computação pervasiva, informações de contexto, Web semântica, planejamento, etc.);

- Já em base à análise feita no levantamento bibliográfico, o modelo ontológico WSMO apresentou-se como o modelo mais consistente para a especiação de serviços e especialmente para especificação de requisições do usuário como metas (*goals*), isto pela sua especificação formal e pelo sua maior expressividade semântica, como apresentado em [80];

- Conforme a análise feita no levantamento bibliográfico, verificou-se que a utilização de ontologias para a modelagem de informações de contexto é proposta mais adequada, em função da inferência semântica que pode ser aplicada e as vantagens de interoperabilidade e reuso por outro tipo de aplicações;

- Utiliza-se o armazenamento das ontologias e serviços baixo uma estrutura de banco de dados, para permitir a usabilidade destas informações por diversos tipos de aplicações, como em trabalhos de gerenciamento de dados, repositórios de dados e fundamentalmente para facilitar a construção e definição das interfaces de usuário final;

- Demostrou-se com a utilização do conceito de compiladores e da ferramenta ANTLR tornou mais simples a criação de analisadores léxicos, sintáticos, semânticos e a geração de código fonte para implementar o processo de tradução entre as linguagens a serem utilizadas pelo sistema de composição de serviços. Além disso, estas gramaticas podem ser reutilizadas por outros projetos, demostrando o reuso desta implementação;

- Estabelece-se que o uso de perfis de usuário (informações de contexto) para descrever preferências, permite facilitar e até certo grau automatizar (implícita ou explicitamente) o processo de especificação das suas requisições. Além disso, dispositivos como RFIDs podem ser utilizados na identificação de usuários, de modo que mecanismos de segurança possam ser atribuídos na especificação de requisições;

- Demostrou-se que especificar as interfaces através de um representação iconográfica facilita a utilização deste tipo de sistemas por usuários sem conhecimento técnico, isto se comparado a interfaces de outras propostas encontradas na literatura;

- Demonstra-se que a aplicação-exemplo, mesmo que restrita na quantidade de serviços implementados, permitiu avaliar os conceitos definidos no presente trabalho como uma prova de conceito valida;

- Verificou-se que as taxas de processamento estão diretamente relacionadas à complexidade da reaquisição feita pelo usuário e ao mecanismo de inferência implementado;

- Contatou-se que a especificação e desenvolvimento de um sistema de composição automática de serviços para ambientes pervasivos é uma tarefa nada trivial e que inclui o uso de diversas linguagens, ferramentas e tecnologias que, ainda, estão em processo de desenvolvimento e formalização, nesse sentido o modelo de referência proposto auxilia e facilita o desenvolvimento deste tipo de sistemas;

- Recomenda-se que a especificação e implementação das ontologias associadas a um sistema de composição de serviços siga padrões e normas definidos para o tipo de ambiente ou domínio de aplicação em questão, como apresentado no sistema de composição para plataforma petroleira no Capítulo 5;

- Finalmente, conclui-se que o objetivo principal deste trabalho foi atingido satisfatoriamente por permitir a complementação de informações não disponíveis na literatura através do modelo de referência especificado, objetivando instruir e auxiliar aos interessados no desenvolvimento de sistemas de composição de serviços em ambientes pervasivos para resolver a problemática de mecanismos automáticos que interajam e satisfaçam as necessidades do usuário de forma natural e transparentes em diversos ambientes pervasivos como o idealizado por Mark Weiser.

## 6.4 Trabalhos Futuros

Como trabalhos futuros nós pretendemos:

- Estender e otimizar as implementações dos módulos descritos no modelo de referência, em especial implementar e estender o planejador proposto em [92], para que de suporte ao não-determinismo da disponibilidade dos serviços e ao raciocínio das informações contextuais presente no processo de composição de forma mais adequada;

- Implementar, testar e comparar o funcionamento do sistema de composição de serviços em ambientes pervasivos com distintos mecanismos de inferência, por exemplo, agentes e logica fuzzy;

- Estudar e agregar a utilização de objetos semânticos complexos [140] no processo de descrição semântica de serviços e no processo de inferência;

- Propor uma metologia de desenvolvimento para sistemas de composição de serviços em base ao modelo de referência proposto, de modo que todo o ciclo de desenvolvimento (criação de ontologias e serviços semânticos; design de interfaces de interação; informações contextuais sobre usuários, dispositivos e ambientes; politicas de segurança e risco dos ambientes, etc.) seja formalmente descrito;

- Aprimorar a aplicação-exemplo apresentada no Capítulo 5, em especial focando a residência como ambiente pervasivo que preste assistência ao desenvolvimento de atividades de pessoas idosas.

- Por em implementação pratica os sistemas de composição descritos no Capítulo 5.

# Referências Bibliográficas


[1] SATYANARAYANAN, M. Pervasive computing: vision and challenges. **Personal Communications, IEEE**, v. 8, n. 4, p. 10–17, 2001.

[2] HELAL, S. et al. The gator tech smart house: a programmable pervasive space. **Computer**, v. 38, n. 3, p. 50–60, 2005. ISSN 0018-9162.

[3] VERGADOS, D.; ALEVIZOS, A.; MARIOLIS, A.; CARAGIOZIDIS, M. Intelligent services for assisting independent living of elderly people at home. In: **Proceedings of the 1st international conference on PErvasive Technologies Related to Assistive Environments**. New York, NY, USA: ACM, 2008. p. 79:1–79:4. ISBN 978-1-60558-067-8.

[4] CALVETTI, P.; SILVEIRA, M. Considerações sobre apresentação de informações de contexto em interfaces com o usuário. In: **Proceedings of the VIII Brazilian Symposium on Human Factors in Computing Systems**. Porto Alegre, POA, Brazil: Sociedade Brasileira de Computação, 2008. p. 80–87. ISBN 978-85-7669-203-4.

[5] W3C. World wide web consortium website: Disponível em: <http://www.w3.org/>. **Acesso em Agosto, 07**, 2010.

[6] FENSEL, D. et al. **Enabling Semantic Web Services: The Web Service Modeling Ontology**. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007. 193 p. ISBN 978-3-540-34519-0.

[7] PELTZ, C. Web services orchestration and choreography. **Computer**, v. 36, n. 10, p. 46–52, 2003. ISSN 0018-9162.

[8] RAO, J.; SU, X. A survey of automated web service composition methods. In: **Proceedings of The 1st International Workshop on Semantic Web Services and Web Process Composition (SWSWPC)**. San Diego, USA: Springer, 2004.

[9] CLARO, D. B.; ALBERS, P.; HAO, J.-K. Web services composition. In: CARDOSO, J.; SHETH, A. P. (Ed.). **Semantic Web Services, Processes and Applications**. [S.l.]: Springer, 2006, (Semantic Web And Beyond Computing for Human Experience, v. 3). p. 195–225. ISBN 978-0-387-30239-3.

[10] ZUñIGA, J. C.; PéREZ-ALCáZAR, J. J.; DIGIAMPIETRI, L. Implementation issues for automatic composition of web services. In: **Proceedings of The 9th International Workshop on Web Semantics (WebS), co-located with Dexa 2010**. Bilbao, Spain: IEEE Computer Society, 2010.

[11] KOPECKY, J.; MORAN, M.; VITAR, T.; ROMAN, D.; MOCAN, A. **WSMO Grounding**. [S.l.], 2007.

[12] KLUSCH, M.; GERBER, A.; SCHMIDT, M. Semantic web service composition planning with owls-xplan. In: **Proceedings of the 1st Intl. AAAI Fall Symposium on Agents and the Semantic Web, Arlington VA, USA, AAAI Press**. [S.l.: s.n.], 2005.

[13] CHAHOUD, J. J. **Planejamento para Serviços Web Semânticos**. Dissertação (Mestrado) — Instituto de Matemática e Estatística da Universidade de São Paulo, 2006.

[14] SILVA, M. V. A. **TRANSPLAN: UMA SOLUÇÃO PARA MAPEAR E PLANEJAR SERVIÇOS WEB SEMÂNTICOS**. Dissertação (Mestrado) — UNIVERSIDADE FEDERAL DA BAHIA, LABORATÓRIO DE SISTEMAS DISTRIBUÍDOS, 2008.

[15] KUZU, M.; CICEKLI, N. K. Dynamic planning approach to automated web service composition. **Applied Intelligence**, v. 36, n. 1, p. 1–28, 2012. ISSN 1573-7497.

[16] VUKOVIC, M. **Context aware service composition**. Tese (Doutorado) — University of Cambridge - Computer Laboratory, 2007.

[17] FUJII, K.; SUDA, T. Semantics-based context-aware dynamic service composition. **ACM Trans. Auton. Adapt. Syst.**, v. 4, n. 2, 2009.

[18] SOUSA, J. P.; FONSECA, B.; CARRAPATOSO, E.; PAREDES, H. An evolutionary platform for the collaborative contextual composition of services. In: 5784, L. (Ed.). **Proceedings of the 15th international conference on Groupware (CRIWG)**. Berlin, Heidelberg: Springer-Verlag, 2009. p. 182–189.

[19] DAVIDYUK, O.; GEORGANTAS, N.; ISSARNY, V.; RIEKKI, J. J. R. Medusa: Middleware for end-user composition of ubiquitous applications. In: MASTROGIOVANNI, F.; CHONG, N. (Ed.). **Handbook of Research on Ambient Intelligence and Smart Environments: Trends and Perspectives**. [S.l.]: IGI Global, 2010.

[20] LOUTAS, N.; PERISTERAS, V.; TARABANIS, K. Towards a reference service model for the web of services. **Data & Knowledge Engineering**, v. 70, n. 9, p. 753–774, 2011.

[21] WANG, H. H.; GIBBINS, N.; PAYNE, T. R.; REDAVID, D. A formal model of the semantic web service ontology (wsmo). **Information Systems**, Elsevier Science Ltd., v. 37, n. 1, p. 33–60, 2012. ISSN 0306-4379.

[22] ZHOU, J. et al. Psc-rm: Reference model for pervasive service composition. In: **Frontier of Computer Science and Technology, 2009. FCST '09. Fourth International Conference on**. [S.l.: s.n.], 2009. p. 705 –709.

[23] NETO, R. de F. B. **Um processo de software e um modelo ontológico para apoio ao desenvolvimento de aplicações sensíveis a contexto**. Tese (Doutorado) — Instituto de Ciências Matemáticas e de Computação (ICMC) Universidade de São Paulo, 2006.

[24] PONCE, E. **MODELAGEM DE CONTEXTO UTILIZANDO ONTOLOGIAS**. Dissertação (Mestrado) — Escola Politécnica da Universidade de São Paulo, 2008.

[25] WEISER, M. Some computer science issues in ubiquitous computing. **Communications of the ACM**, v. 36, n. 7, p. 75–84, 1993.

[26] URBIETA, A.; BARRUTIETA, G.; PARRA, J.; URIBARREN, A. A survey of dynamic service composition approaches for ambient systems. In: **Proceedings of the 1st ICST Int. Conf. on Ambient Media and Systems (Ambi-sys'08 )**. [S.l.]: ACM, 2008.

[27] RANGANATHAN, A.; CAMPBELL, R. Autonomic pervasive computing based on planning. In: **Proceedings of The International Conference on Autonomic Computing**. [S.l.: s.n.], 2004. p. 80–87.

[28] PAPAZOGLOU, M. P.; GEORGAKOPOULOS, D. Service-oriented computing: Introduction. **Communications of the ACM**, v. 46, n. 10, p. 25–28, 2003.

[29] KALASAPUR, S.; KUMAR, M.; SHIRAZI, B. Evaluating service oriented architectures (soa) in pervasive computing. In: **Pervasive Computing and Communications, 2006. PerCom 2006. Fourth Annual IEEE International Conference on**. [S.l.: s.n.], 2006. p. 10 pp. –285.

[30] PAPAZOGLOU, M. P.; TRAVERSO, P.; DUSTDAR, S.; LEYMANN, F. Service-oriented computing: State of the art and research challenges. **IEEE Computer**, v. 40, n. 11, p. 38–45, 2007. ISSN 0018-9162.

[31] MEDJAHED, B.; BOUGUETTAYA, A. **Service Composition for the Semantic Web**. [S.l.]: Springer New York Dordrecht Heidelberg London, 2011. ISBN 978-1-4419-8464-7.

[32] BERNERS-LEE, T.; HENDLER, J.; LASSILA, O. The semantic web. **Scientific American**, v. 284, n. 5, p. 35–43, 2001.

[33] GHALLAB, M.; NAU, D.; TRAVERSO, P. **Automated Planning, Theory and Practice**. [S.l.]: Morgan Kaufmann Publishers, Elsevier, 2004. ISBN 1-55860-856-7.

[34] GRUBER, T. R. **A Translation Approach to Portable Ontology Specifications**. [S.l.], April 1993.

[35] CASATI, F.; SAYAL, M.; SHAN, M. Developing e-services for composing e-services. In: **Proceedings of The 13th International Conference on Advanced Information Systems Engineering (CaiSE)**. [S.l.]: Springer Verlag, 2001.

[36] WAHL, T.; SINDRE, G. A survey of development methods for semantic web service systems. **International Journal of Information Systems in the Service Sector (IJISSS)**, v. 1, n. 2, p. 1–16, 2009.

[37] BRONSTED, J.; HANSEN, K. M.; INGSTRUP, M. Service composition issues in pervasive computing. **IEEE Pervasive Computing**, IEEE Computer Society, Los Alamitos, CA, USA, v. 9, p. 62–70, 2010. ISSN 1536-1268.

[38] FILHO, E. C. L. **Um Middleware Extensível para Disponibilização de Serviços em Ambientes Pervasivos**. Dissertação (Mestrado) — UNIVERSIDADE FEDERAL DE CAMPINA GRANDE, 2006.

[39] YANG, H.-I.; BOSE, R.; HELAL, A.; XIA, J.; CHANG, C. Fault-resilient pervasive service composition. In: KAMEAS, A. D.; CALLAGAN, V.; HAGRAS, H.; WEBER, M.; MINKER, W. (Ed.). **Advanced Intelligent Environments**. [S.l.]: Springer US, 2009. p. 195–223.

[40] QIU, L.; SHI, Z.; LIN, F. Context optimization of ai planning for services composition. In: **IEEE International Conference on e-Business Engineering**. [S.l.: s.n.], 2006. p. 610–617.

[41] PFISTERER, D. et al. Spitfire: Toward a semantic web of things. **IEEE Communications Magazine**, 2011.

[42] BOGDANOWICZ, K. D. ans M.; SCAPOLO, F.; LEIJTEN, J.; BURGELMAN, J. **Scenarios for Ambient Intelligence in 2010**. [S.l.], 2001.

[43] ZHOU, J.; RIEKKI, J.; YLIANTTILA, M. Modeling service composition and exploring its characteristics. In: **Services - I, 2009 World Conference on**. [S.l.: s.n.], 2009. p. 446 –451.

[44] HANSEN, K. M.; BRONSTED, J. **Modeling Service Composition Reliability in Pervasive Computing**. [S.l.], 2010.

[45] FARAVELON, A.; CHOLLET, S.; VERDIER, C.; FRONT, A. Enforcing Privacy as Access Control in a Pervasive Context. In: IEEE. **2012 IEEE CONSUMER COMMUNICATIONS AND NETWORKING CONFERENCE (CCNC)**. [S.l.], 2012. (IEEE Consumer Communications and Networking Conference), p. 380–384. ISBN 978-1-4577-2071-0. IEEE Consumer Communications and Networking Conference (CCNC), Las Vegas, NV, JAN 14-17, 2012.

[46] STAVROPOULOS, T.; VRAKAS, D.; VLAHAVAS, I. A survey of service composition in ambient intelligence environments. **Artificial Intelligence Review**, Springer Netherlands, p. 1–24, 2011. ISSN 0269-2821.

[47] MACKENZIE, M.; LASKEY, K.; MCCABE, F.; BROWN, P. F.; METZ, R. **Reference Model for Service Oriented Architecture**. [S.l.], 2006.

[48] OMG. **OMG Systems Modeling Language (OMG SysML) Version 1.3**. [S.l.], 2012.

[49] LAUSEN, H.; POLLERES, A.; ROMAN, D. **WSMO: Web Service Modelling Ontology**. [S.l.], 2005.

[50] ILGHAMI, O. **Documentation for JSHOP2**. [S.l.], 2006.

[51] KUTLUHAN, E. **Hierarchical Task Network Planning: Formalization, Analysis, and Implementation**. Tese (Doutorado) — University of Maryland, 1995.

[52] PEIS, E.; CASTILLO, J. M. del; DELGADO-LÓPEZ, J. Semantic recommender systems. analysis of the state of the topic. **Hipertext. net**, v. 6, p. 1–5, 2008.

[53] FASLI, M. Shopbots: A syntactic present, a semantic future. **Internet Computing, IEEE**, v. 10, n. 6, p. 69–75, 2006. ISSN 1089-7801.

[54] BRUJIN, J. de et al. **Modeling Semantic Web Services**. Berlin, Heidelberg: Springer-Verlag, 2008. 191 p. ISBN 978-3-540-68169-4.

[55] HERRMANN, R. G. **Planejamento Hierárquico sob incerteza Knightiana**. Dissertação (Mestrado) — Instituto de Matemática e Estatística da Universidade de São Paulo, 2008.

[56] BOEHM, B. W. A spiral model of software development and enhancement. **Computer**, v. 21, n. 5, p. 61–72, 1988.

[57] KUSTER, U.; KONIG-RIES, B.; PETRIE, C.; KLUSCH, M. On the evaluation of semantic web service frameworks. **International Journal On Semantic Web and Information Systems**, v. 4, n. 4, 2008.

[58] WEISER, M. The computer for the 21st century. **Scientific American Special Issue on Communications, Computers, and Networks**, p. 94–104, September 1991.

[59] MONTEIRO, J. D. A. **Desenvolvimento de aplicações multi-plataformas para dispositivos móveis**. Dissertação (Mestrado) — Instituto de Ciências Matemáticas e de Computação - ICMC-USP, 2006.

[60] COOK, D. J.; AUGUSTO, J. C.; JAKKULA, V. R. Ambient intelligence: Technologies, applications, and opportunities. **PERVASIVE AND MOBILE COMPUTING**, 5, n. 4, p. 277–298, AUG 2009. ISSN 1574-1192.

[61] SINGH, R.; BHARGAVA, P.; KAIN, S. State of the art smart spaces: application models and software infrastructure. **Ubiquity**, ACM, New York, NY, USA, p. 7:2–7:9, 2006. ISSN 1530-2180.

[62] WOLF, P.; SCHMIDT, A.; KLEIN, M. Soprano - an extensible, open aal platform for elderly people based on semantical contracts. In: **Proceedings of the 3rd Workshop om Artificial Intelligence Techniques for Ambient Intelligence (AITAmi)**. [S.l.: s.n.], 2008.

[63] AAL. The ambient assisted living (aal) joint programme, website: Disponível em: <http://www.aal-europe.eu/>. **Acesso em Novembro, 08**, 2010.

[64] XAVIER, A.; RAABE, A.; SIGULEM, M. S. nd D. Desafios de interação e acessibilidade para o usuário idoso. In: **Proceedings of the IX Congresso Brasileiro de Informática em Saúde (CBIS)**. [S.l.: s.n.], 2004.

[65] DEY, A. K. Understanding and using context. **Personal Ubiquitous Computing**, v. 5, n. 1, p. 4–7, 2001.

[66] ABOWD, G. D.; MYNATT, E. D. Charting past, present, and future research in ubiquitous computing. **ACM Trans. Comput.-Hum. Interact.**, v. 7, n. 1, p. 29–58, 2000.

[67] TRUONG, K. N.; ABOWD, G. D.; BROTHERTON, J. A. Who, what, when, where, how: Design issues of capture & access applications. In: **Proceedings of the 3rd international conference on Ubiquitous Computing**. [S.l.]: Springer-Verlag, 2001. p. 209–224. ISBN 3-540-42614-0.

[68] COMPUTAçãO Ubíqua Ciente de Contexto: Desafios e Tendências. In: . Recife, Brazil: Sociedade Brasileira de Computação, 2009. cap. 3, p. 99–149.

[69] POTT, S.; KOPACK, M. **Teach Yourself Web Services in 24 hours**. [S.l.]: SAMS, 2003. ISBN 0-672-32515-2.

[70] COMER, D. E. **Internetworking with TCP/IP: Principles, Protocols, and Architectures**. 4. ed. [S.l.]: Prentice Hall, 2000.

[71] CLARO, D. B.; MACêDO, R. J. de A. Serviços web e sua relação com sistemas de informação: Uma abordagem dos conceitos básicos às composições semânticas. In: **IV Simpósio Brasileiro de Sistemas de Informação**. [S.l.: s.n.], 2008.

[72] NETO, R. de F. B.; PIMENTEL, M. da G. C. Interoperabilidade semântica entre aplicações cientes de contexto. **impósio Brasileiro de Sistemas Multimídia e Web - Webmidia 03**, 2003.

[73] QUIN, L. Extensible markup language (xml). **Disponível em: <http://www.w3.org/XML/> Acesso em Junho, 05**, 2009.

[74] SPERBERG-MCQUEEN, C.; THOMPSON, H. Xml schema. **Disponível em: <http://www.w3.org/XML/Schema> Acesso em Junho, 05**, 2009.

[75] KLYNE, G.; CARROLL, J.; MCBRIDE, B. Resource description framework (rdf): Concepts and abstract syntax. **Disponível em: <http://www.w3.org/TR/rdf-concepts/> Acesso em Junho, 05**, 2009.

[76] BRICKLEY, D.; GUHA, R.; MCBRIDE, B. Rdf vocabulary description language 1.0: Rdf schema w3c recommendation. **Disponível em: <http://www.w3.org/TR/rdf-schema/> Acesso em Junho, 05**, 2009.

[77] STUDER, R.; BENJAMINS, V.; FENSEL, D. Knowledge engineering: Principles and methods. **Data and Knowledge Engineering**, v. 25, n. 1:2, p. 161 – 197, 1998. ISSN 0169-023X.

[78] MARTIN, D. et al. **OWL-S: Semantic Markup for Web Services**. [S.l.], 2004.

[79] AKKIRAJU, R. et al. **WSDL-S: Web Service Semantics**. [S.l.], 2005.

[80] SOUZA, K. S. de; PéREZ-ALCáZAR, J. J.; DIGIAMPIETRI, L. A.; ZUñIGA, J. C. Um estudo comparativo de frameworks para desenvolvimento de aplicações baseadas em serviços web semânticos. **Revista Eletrônica de Iniciação Científica (REIC)**, v. 9, n. 2, junho 2009.

[81] CHAN, M.; BISHOP, J.; BARESI, L. **Survey and Comparison of Planning Techniques for Web Services Composition**. [S.l.], 2007.

[82] SIRIN, E.; PARSIA, B. Planning for semantic web services. In: **In Semantic Web Services Workshop at 3rd International Semantic Web Conference**. [S.l.: s.n.], 2004.

[83] GHALLAB, M.; NAU, D.; TRAVERSO, P. Planning based on markov decision processes. In: **Automated Planning theory and practice**. [S.l.: s.n.], 2004. p. 379–403.

[84] KUTER, U.; NAU, D. S. Forward-chaining planning in nondeterministic domains. In: **Proceedings of the 19th national conference on Artifical intelligence (AAAI)**. [S.l.: s.n.], 2004.

[85] DUSTDAR, S.; SCHREINER, W. A survey on web services composition. **Int. J. Web Grid Serv.**, Inderscience Publishers, Inderscience Publishers, Geneva, SWITZERLAND, v. 1, n. 1, p. 1–30, 2005. ISSN 1741-1106.

[86] PAHL, C.; ZHU, Y. A semantical framework for the orchestration and choreography of web services. **Electron. Notes Theor. Comput. Sci.**, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 151, n. 2, p. 3–18, maio 2006. ISSN 1571-0661.

[87] LI, Z.; O'BRIEN, L.; KEUNG, J.; XU, X. Effort-oriented classification matrix of web service composition. **Internet and Web Applications and Services, International Conference on**, IEEE Computer Society, Los Alamitos, CA, USA, v. 0, p. 357–362, 2010.

[88] LI, Z.; O'BRIEN, L. A qualitative approach to effort judgment for web service composition based soa implementations. In: **Advanced Information Networking and Applications (AINA), 2011 IEEE International Conference on**. [S.l.: s.n.], 2011. p. 586–593. ISSN 1550-445X.

[89] DIGIAMPIETRI, L. A.; ALCáZAR, J. J. P.; MEDEIROS, C. B. Ai planning in web services composition: a review of current approaches and a new solution. In: **In: Proceedings of the XXVII Brazilian computer society conference (CSBC)**. [S.l.: s.n.], 2008.

[90] DIMITROV, M.; SIMOV, A.; MOMTCHEV, V.; KONSTANTINOV, M. Wsmo studio - a semantic web services modelling environment for wsmo. **In LNCS - The Semantic Web: Research and Applications**, v. 4519, p. 749–758, 2007.

[91] JONES, S. P.; HUGES, J. **Haskell 98 language and libraries**. [S.l.], 2008.

[92] KUTER, U. **Planning Under Uncertainty: Moving Forward**. Tese (Doutorado) — University of Maryland, 2006.

[93] ROMAN, D. et al. Semantic web services ? approaches and perspectives. In: ____. **Semantic Web Technologies**. John Wiley & Sons, Ltd, 2006. p. 191–236. ISBN 9780470030332. Disponível em: <http://dx.doi.org/10.1002/047003033X.ch10>.

[94] SAPENA, O.; ONAINDíA, E. Planning in highly dynamic environments: an anytime approach for planning under time constraints. **Applied Intelligence**, Springer US, v. 29, n. 1, p. 90–109, 2008. ISSN 0924-669X. Disponível em: <http://dx.doi.org/10.1007/s10489-007-0083-x>.

[95] HESSELMAN, C.; TOKMAKOFF, A.; PAWAR, P.; IACOB, S. Discovery and composition of services for context-aware systems. In: **Proceedings of the First European conference on Smart Sensing and Context**. [S.l.: s.n.], 2006. p. 67–81. ISSN 1380-7501.

[96] MINGKHWAN, A. et al. Dynamic service composition in home appliance networks. **Multimedia Tools Appl.**, v. 29, n. 3, p. 257–284, 2006. ISSN 1380-7501.

[97] KAEFER, G.; SCHMID, R.; PROCHART, G.; WEISS, R. Framework for dynamic resource-constrained service composition for mobile ad hoc networks. In: **Workshop on System Support for Ubiquitous Computing (UBICOMP)**. [S.l.: s.n.], 2006.

[98] POURREZA, H.; GRAHAM, P. On the fly service composition for local interaction environments. In: **Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom)**. [S.l.: s.n.], 2006. p. 393–399.

[99] LEE, S. Y.; LEE, J. Y.; LEE, B. I. Service composition techniques using data mining for ubiquitous computing environments. **International Journal of Computer Science and Network Security**, v. 6, n. 9, p. 110–117, 2006.

[100] SOUSA, J.; POLADIAN, V.; GARLAN, D.; SCHMERL, B.; SHAW, M. Task-based adaptation for ubiquitous computing. **IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews**, v. 36, n. 3, p. 328–340, 2006.

[101] YANG, Y.; MAHON, F.; WILLIAMS, M.; PFEIFER, T. Context-aware dynamic personalised service re-composition in a pervasive service environment. In: MA, J.; JIN, H.; YANG, L.; TSAI, J.-P. (Ed.). **Ubiquitous Intelligence and Computing**. [S.l.]: Springer Berlin Heidelberg, 2006, (Lecture Notes in Computer Science, v. 4159). p. 724–735.

[102] DEY, A. K.; SOHN, T.; STRENG, S.; KODAMA, J. icap: Interactive prototyping of context-aware applications. In: **in Proceedings of Pervasive 2006**. [S.l.: s.n.], 2006. p. 254–271.

[103] SVENSSON, D.; HEDIN, G.; MAGNUSSON, B. Pervasive applications through scripted assemblies of services. In: **Pervasive Services, IEEE International Conference on**. [S.l.: s.n.], 2007. p. 301–307.

[104] MOKHTAR, S.; GEORGANTAS, N.; ISSARNY, V. Cocoa : Conversationbased service composition for pervasive computing environments. In: **Pervasive Services, 2006 ACS/IEEE International Conference on**. [S.l.: s.n.], 2006. p. 29–38.

[105] LEE, C.; KO, S.; LEE, S.; LEE, W.; HELAL, S. Context-aware service composition for mobile network environments. In: INDULSKA, J.; MA, J.; YANG, L.; UNGERER, T.; CAO, J. (Ed.). **Ubiquitous Intelligence and Computing**. [S.l.]: Springer Berlin Heidelberg, 2007, (Lecture Notes in Computer Science, v. 4611). p. 941–952. ISBN 978-3-540-73548-9.

[106] BOTTARO, A.; BOURCIER, J.; ESCOFFIER, C. Autonomic context-aware service composition. In: **2nd IEEE International Conference on Pervasive Services**. [S.l.: s.n.], 2007.

[107] KALOFONOS, D.; WISNER, P. A framework for end-user programming of smart homes using mobile devices. In: **Consumer Communications and Networking Conference, 2007. CCNC 2007. 4th IEEE**. [S.l.: s.n.], 2007. p. 716–721.

[108] KALASAPUR, S.; KUMAR, M.; SHIRAZI, B. Dynamic service composition in pervasive computing. **Parallel and Distributed Systems, IEEE Transactions on**, v. 18, n. 7, p. 907–918, 2007. ISSN 1045-9219.

[109] BARESI, L.; NITTO, E.; GHEZZI, C.; GUINEA, S. A framework for the deployment of adaptable web service compositions. **Service Oriented Computing and Applications**, Springer-Verlag, v. 1, n. 1, p. 75–91, 2007.

[110] THOMSON, G.; BIANCO, S.; MOKHTAR, S.; GEORGANTAS, N.; ISSARNY, V. Amigo aware services. In: MüHLHAUSER, M.; FERSCHA, A.; AITENBICHLER, E. (Ed.). **Constructing Ambient Intelligence**. [S.l.: s.n.], 2008, (Communications in Computer and Information Science, v. 11). p. 385–390.

[111] WANT, R.; PERING, T.; SUD, S.; ROSARIO, B. Dynamic composable computing. In: **Proceedings of the 9th workshop on Mobile computing systems and applications**. [S.l.]: ACM, 2008. p. 17–21.

[112] TREVOR, J. Doing the mobile mash. **Computer**, IEEE Computer Society, Los Alamitos, CA, USA, v. 41, n. 2, p. 104–106, 2008. ISSN 0018-9162.

[113] AMUNDSEN, S. L.; ELIASSEN, F. A resource and context model for mobile middleware. **Personal Ubiquitous Comput.**, Springer-Verlag, v. 12, n. 2, p. 143–153, 2008.

[114] PALUSKA, J. M. et al. Structured decomposition of adaptive applications. **Pervasive Mob. Comput.**, Elsevier Science Publishers B. V., v. 4, n. 6, p. 791–806, 2008.

[115] BEAUCHE, S.; POIZAT, P. Automated service composition with adaptive planning. In: **Proceedings of the 6th International Conference on Service-Oriented Computing**. [S.l.]: Springer-Verlag, 2008. p. 530–537.

[116] BERTOLI, P. et al. Continuous orchestration of web services via planning. In: **International Conference on Automated Planning and Scheduling ICAPS**. [S.l.: s.n.], 2009.

[117] IBRAHIM, N.; INFORMATICS, G.; MOUëL, F. L.; FRéNOT, S. Mysim: A spontaneous service integration middleware for pervasive environments. In: **ACM international conference on pervasive services (ICPS)**. [S.l.: s.n.], 2009.

[118] LAGESSE, B.; KUMAR, M.; WRIGHT, M. Resco: A middleware component for reliable service composition in pervasive systems. In: **Eigth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom)**. [S.l.]: IEEE, 2010. p. 486–491.

[119] SANTOFIMIA, M. J.; FAHLMAN, S. E.; TORO, X. del; MOYA, F.; LOPEZ, J. C. A semantic model for actions and events in ambient intelligence. **Eng. Appl. Artif. Intell.**, Pergamon Press, Inc., v. 24, n. 8, p. 1432–1445, 2011.

[120] LIU, F.; LI, C.; YU, J. Description of web service composition model based on z notation. In: **IEEE International Conference on Computer Science and Automation Engineering (CSAE)**. [S.l.: s.n.], 2011. v. 3, p. 587–591.

[121] DUKE, R.; ROSE, G.; SMITH, G. Object-z: A specification language advocated for the description of standards. **Computer Standards & Interfaces**, v. 17, n. 5?6, p. 511 – 533, 1995. ISSN 0920-5489.

[122] SERRANO, M.; SHI, L.; FOGHLú, M.; DONNELLY, W. Cloud services composition support by using semantic annotation and linked data. In: FRED, A.; DIETZ, J.; LIU, K.; FILIPE, J. (Ed.). **Knowledge Discovery, Knowledge Engineering and Knowledge Management**. [S.l.]: Springer Berlin Heidelberg, 2013, (Communications in Computer and Information Science, v. 348). p. 278–293.

[123] HUANG, J.; BASTANI, F.; YEN, I.-L.; ZHANG, W. A framework for efficient service composition in cyber-physical systems. In: **Service Oriented System Engineering (SOSE), 2010 Fifth IEEE International Symposium on**. [S.l.: s.n.], 2010. p. 291–298.

[124] DIAZ, M.; GONZALO, J.; OIKAWA, L.; RYUGA, A. Big data on the internet of things: An example for the e-health. In: **International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing**. [S.l.: s.n.], 2012. p. 898–900.

[125] GARAY, J. R. B. **CyberSens: Uma Plataforma Para Redes de Sensores em Sistemas Ciber-Físicos**. Tese (Doutorado) — Escola Politécnica da Universidade de São Paulo, 2012.

[126] SHEIKH, K.; WEGDAM, M.; SINDEREN, M. V. Quality-of-context and its use for protecting privacy in context aware systems. **Journal of Software**, v. 3, 2008.

[127] KAHANER, L. **Competitive Intelligence: How to Gather, Analyze and Use Information to Move your Business to the Top**. [S.l.]: Touchstone, New York, 1999. ISBN 978-0-684-84404-6.

[128] CHUNNIAN, L.; DEHUI, Y.; YONGLONG, W. Domain ontology and semantic web applications for study of web competitive intelligence analysis system. v. 1, n. 1, p. 99–113, 2011.

[129] FERREIRA, F. C. O comportamento de procura de informação no processo de decisão de compra na web. v. 1, n. 1, p. 3–26, 2011.

[130] DAN, B.; PENG, S.; ZHANG, X.; RAO, K. Product variants search and retrieval based on the semantically annotated product family using multi-facet domain ontology. v. 403, p. 4114–4118, 2011.

[131] VANDIC, D.; DAM, J. van; FRASINCAR, F. Faceted product search powered by the semantic web. 2012.

[132] HOW Best Buy is Using The Semantic Web. 2010. Disponível em: <http://jay.beweep.com/2009/10/26/example-best-buy-product-rdfa-markup-released-beta/>.

[133] TSAI, W.-T. et al. An approach for service composition and testing for cloud computing. In: **Autonomous Decentralized Systems (ISADS), 2011 10th International Symposium on**. [S.l.: s.n.], 2011. p. 631–636.

[134] MIETZNER, R.; LEYMANN, F. A self-service portal for service-based applications. In: **Service-Oriented Computing and Applications (SOCA), 2010 IEEE International Conference on**. [S.l.: s.n.], 2010. p. 1–8.

[135] GUTIERREZ-GARCIA, J.; SIM, K.-M. Self-organizing agents for service composition in cloud computing. In: **Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on**. [S.l.: s.n.], 2010. p. 59–66.

[136] BAO, H.; DOU, W. A qos-aware service selection method for cloud service composition. In: **Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), 2012 IEEE 26th International**. [S.l.: s.n.], 2012. p. 2254–2261.

[137] LARA, R.; ROMAN, D.; POLLERES, A.; FENSEL, D. **OWLS 1.0 Release - Examples Bravo Air Fictitious airline site**. [S.l.], 2003.

[138] DIMITROV, M.; SIMOV, A.; KONSTANTINOV, M.; CEKOV, L.; NORTON, B. **WSMO Studio**. [S.l.], 2009.

[139] SOUZA, K. S. de. **Estudo e implementação de uma ferramenta de mapeamento de WSMO para WSDL (Mecanismo de Grounding)**. [S.l.], 2009.

[140] SUN, X. Osln: An object-oriented semantic link network language for complex object description and operation. **Future Gener. Comput. Syst.**, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 26, n. 3, p. 389–399, mar. 2010. ISSN 0167-739X. Disponível em: <http://dx.doi.org/10.1016/j.future.2009.07.007>.