UNIVERSIDADE FEDERAL DE SÃO CARLOS
DEPARTAMENTO DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Especificação de perfis e regras, baseada em ontologias, para adaptação de conteúdo na Internet.

Marcos Forte

São Carlos
Agosto / 2006

UNIVERSIDADE FEDERAL DE SÃO CARLOS
DEPARTAMENTO DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Especificação de perfis e regras, baseada em ontologias, para adaptação de conteúdo na Internet.

Marcos Forte

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para obtenção do Título de Mestre em Ciência da Computação.
Orientador: Antonio Francisco do Prado
Co-Orientador: Wanderley Lopes de Souza

São Carlos
2006

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

F737ep

Forte, Marcos.
Especificação de perfis e regras, baseada em ontologias, para adaptação de conteúdo na internet / Marcos Forte. -- São Carlos : UFSCar, 2006.
179 p.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2006.

1. Métodos especiais de computação. 2. Adaptação de conteúdo. 3. Política de adaptação. 4. Ontologia. 5. Serviços da Web. I. Título.

CDD: 006 (20ª)

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

"Especificação de perfis e regras, baseada em ontologias, para adaptação de conteúdo na Internet"

MARCOS FORTE

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Membros da Banca:

Prof. Dr. Antonio Francisco do Prado
(Orientador – DC/UFSCar)

Prof. Dr. Wanderley Lopes de Souza
(Co-Orientador – DC/UFSCar)

Prof. Dr. Paulo César Masiero
(ICMC/USP)

Prof. Dr. José Gonçalves Pereira Filho
(DI-UFES)

São Carlos
Agosto/2006

Aos meus pais...

Que com todo o seu amor escolheram sonhar

os meus sonhos e trilhar o meu caminho.

Por me apoiarem nas minhas decisões

e por me ensinarem os verdadeiros

valores da vida.

AGRADECIMENTOS

Aos meus irmãos, pelo apoio e carinho que sempre me dedicaram.

Ao meu orientador Antonio Francisco do Prado, meu primeiro e único professor de programação, pelos seus ensinamentos que sempre contribuíram para o meu desenvolvimento.

Ao meu co-orientador Wanderley Lopes de Souza pela confiança depositada, pelo apoio em todos os momentos, e pelas críticas e sugestões que foram vitais para a realização deste trabalho.

Aos professores: Célio Estevan Moron, Hélio Crestana Guardia, Luiz Carlos Trevelin e Sérgio Donizetti Zorzo, pelos ensinamentos e apoio durante o desenvolvimento deste trabalho.

Ao Renato, Pedro e Luana pelas colaborações no projeto.

A todos os colegas de pós-graduação, pelo convívio prazeroso, e que sempre se dispuseram a me ajudar.

Às funcionárias da pós-graduação, Cristina e Mirian, pela disponibilidade e atendimento sempre cordial.

Ao Centro Universitário Fundação Santo André, pelo apoio financeiro e de infra-estrutura durante a realização deste trabalho.

À minha noiva, Quelli, pelo amor, carinho, compreensão e paciência.

Agradeço a todos, à vida e a Deus, por tudo o que tenho e por conceder-me a graça de ter saúde e a força de vencer obstáculos e atingir objetivos.

RESUMO

Esta última década foi cenário de grandes transformações, dentre estas destaca-se a revolução social que a Internet está promovendo, alterando a forma como as pessoas encontram informações, comunicam-se e compram produtos e serviços. Além disso, a convergência da telefonia, computação e eletrônica de consumo, está disponibilizando cada vez mais dispositivos móveis de acesso a Internet que se comunicam via redes sem fio.

Essas mudanças causaram um grande problema para o conteúdo disponível na *Web*: como um mesmo conteúdo pode se adequar a diferentes tipos de usuário, que possuem suas próprias preferências, e ser apresentado em dispositivos móveis compostos por uma grande diversidade de características funcionais? Para resolver este problema, muitas pesquisas estão sendo realizadas no campo da adaptação de conteúdo.

Um dos desafios nesse campo é definir, a partir de uma requisição do usuário, quais serviços de adaptação serão necessários e em que ordem estes serão executados. Para que essa decisão seja tomada corretamente são necessárias informações sobre o contexto de entrega e os serviços de adaptação disponíveis. Essas informações são descritas por meio de perfis e regras, sendo que no momento a maioria das soluções propostas se baseia em modelos proprietários e incompatíveis entre si. Além disso, a descoberta de serviços geralmente é amplamente baseada em comparações sintáticas, o que compromete a descoberta e composição de serviços.

Com o objetivo de contribuir para a área de adaptação de conteúdo, esta dissertação define um conjunto de perfis e regras baseado em ontologias. Além disso, demonstra como as regras devem ser integradas aos perfis de serviços, proporcionando

uma solução que seja independente de novos tipos de serviços e regras. Também são apresentados os benefícios que a descoberta e composição de serviços obtêm com o uso de semântica em um ambiente aberto como a *Web*.

A implementação e o estudo de caso dessas propostas foi baseado num framework para adaptação de conteúdo baseado em componentes. Para estender a sua funcionalidade com o propósito de suportar ontologias, e outras funções necessárias, componentes foram reusados, adaptados e criados. Essa extensão viabilizou a realização do estudo de caso e a validar a solução proposta.

Palavras-chave: Adaptação de conteúdo, Ontologias, Perfis, Política de Adaptação, Contexto de Entrega, Serviços Web, *Frameworks*.

ABSTRACT

The last decade was scenery of big transformations, among them the social revolution caused by Internet stand out, changing the way that people find information, communicate and buy products and services. Moreover, the convergence of telephony, informatics and consume electronic, supplies more and more mobile devices equipped with Internet access through wireless network.

This changing caused a big problem to the Internet content available at Web: how the same content should be adjusted to different kinds of users witch have particular preferences, and be presented in mobile devices featured with a big diversity of functionalities? To solve this problem, many research proposals are been developed at content adaptation area.

One challenge of this area is define, from a user request, witch adaptation services are necessary and these execute sequence. In order to have the best decision, delivery context and the available services have to be acknowledged. This information are described at profiles and rules, considering that nowadays the majority of proposed solutions are based in proprietary models and not compatibles among each other. As well, the discovery of services generally is strongly based on syntactic matching, what weak the discovery and composition of services.

Intending to contribute with content adaptation area, this dissertation defines an ontology-based set of profiles and reused. Furthermore, shows how the rules have to be integrated with services profiles, providing a rules and profiles independent solution. Also are presented the benefits use of semantics at match and composition of services in an open environment such as the Web.

The implementation and the case study of this proposition were based at a component-based content adaptation framework. With the purpose of extend its functionality, in order to support ontology, and others necessary functions, components were reused, adapted or created. This extensions made possible achieve the case study and the validation of proposed solution.

Keywords: Content Adaptation, Ontologies, Profiles, Adaptation policy, Delivery context, Web services, Frameworks.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

A2A	Application-to-Application
API	Application Programming Interface
B2B	Business-to-Bussiness
BPEL4WS	Business Process Execution Language for Web Services
CC/PP	Composite Capabilities/Preferences Profile
cHTML	Compact HyperText Markup Language
CIPA	Children's Internet Protection Act
CORBA	Common Object Request Broker Architecture
CSS	Cascading Style Sheets
DAML	DARPA Agent Markup Language
DAML+OIL	DARPA Agent Markup Language + Ontology Interface Layer
DCE	Distributed Computing Environment
DCOM	Distributed Component Object Model
DELI	Delivery Context LIbrary
DI	Device Independence
DL	Description Logic
DTD	Document Type Definition
EODM	EMF Ontology Definition Metamodel
FACI	Framework para Adaptação de Conteúdo na Internet
FOL	First Order Logic
FTP	File Transfer Protocol
GIF	Graphic Image Format
HTML	HyperText Transfer Protocol
IANA	Internet Assigned Numbers Authority

ICAP	Internet Content Adaptation Protocol
IETF	Internet Engineering Task Force
IIOP	Internet Inter-ORB Protocol
IOPE	Inputs Outputs Preconditions Effects
IRML	Intermediary Rule Markup Language
J2EE	Java 2 Platform Enterprise Edition
JPEG	Joint Photographic Experts Group
JSR	Java Specification Request
MIME	Multipurpose Internet Mail Extensions
OASIS	Organization for the Advancement of Structured Information Standards
OCP	OPES Callout Protocol
OPES	Open Pluggable Edge Services
OWL	Web Ontology Language
OWL	Web Ontology Language for Services
PDA	Personal Digital Assistent
PICS	Platform for Internet Content Selection
RDF	Resource Description Framework
RDF-S	Resource Description Framework Schema
RFC	Request for Comments
RMI	Remote Method Invocation
RPC	Remote Procedure Calls
RSTP	Real Time sTreaming Protocol
RTP	Real-Time Transport Protocol
RuleML	Rule Markup Language

RVSA	Remote Variant Selection Algorithm
SADiC	Semantic API for the Delivery Context
SIAP	Safe Internet Action Plan
SLA	Service Level Agreement
SMTP	Simple Mail Transfer Protocol
SOAP	Simple Object Access Protocol
SOUPA	Standard Ontology for Ubiquitous and Pervasive Applications
SVG	Scalable Vector Graphics
SWRL	Semantic Web Rule Language
TCP/IP	Transmission Control Protocol / Internet Protocol
UAPROF	User Agent PROFile
UDDI	Universal Description, Discovery and Integration
UPS	Universal Profiling Schema
URI	Uniform Resource Identifiers
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WAP	Wireless Application Protocol
WBMP	Wireless BitMaP
W-HTTP	Wireless Profiled HTTP
WML	Wireless Markup Language
WSDL	Web Services Description Language
WS-I	Web Services Interoperability Organization
WSP	Wireless Session Protocol
WURFL	Wireless Universal Resource FiLe
XForms	XML Forms

XHTML	Extensible HyperText Markup Language
XML	eXtensible Markup Language
XSL	eXtensible Stylesheet Language
XSL-FO	eXtensible Stylesheet Language Formatting Objects
XSLT	eXtensible Stylesheet Language Transformation

SUMÁRIO

1 INTRODUÇÃO

A *Web* foi introduzida no início dos anos noventa e atualmente tornou-se parte de nossa rotina diária, sendo utilizada por pessoas de todas as culturas e idades. A *Web* promoveu um impacto profundo na busca por informações, nas formas de comunicação, e na aquisição de produtos e serviços. Com o aumento das capacidades dos dispositivos e a cobertura oferecida pelas atuais redes de comunicação, foi possível o acesso à informação a qualquer hora, em qualquer lugar, suportado por ambientes de Computação Ubíqua [1] ou Computação Pervasiva [2].

Nos últimos anos, uma grande diversidade de dispositivos, com capacidade de acessar a *Web*, foi desenvolvida. Esses dispositivos possuem diferentes capacidades de processamento, memória e principalmente diferentes características de tela, incluindo tamanho, capacidade gráfica e número de cores. Além disso, podem acessar a *Web* por meio de redes sem fio, que geralmente disponibilizam uma baixa e variável largura de banda.

De modo a evitar o desenvolvimento de várias versões de um mesmo conteúdo e/ou a fragmentação da *Web* em espaços específicos para determinados dispositivos, estão sendo realizadas pesquisas em busca da independência de dispositivo [3]. O objetivo dessas pesquisas é criar meios de prover-se conteúdo e aplicações *Web,* que possam ser gerados ou adaptados de modo a proporcionar ao usuário uma boa experiência de acesso independentemente do dispositivo utilizado.

Para determinar o conteúdo e/ou adaptação apropriada, é necessário conhecer o contexto de entrega (*delivery context*) [4]. Esse aspecto é expresso por um conjunto de atributos, de modo a caracterizar as capacidades do dispositivo de acesso e as preferências do usuário. A partir do conhecimento dessas capacidades e preferências,

são definidas as adaptações necessárias para a entrega de conteúdo. As capacidades do dispositivo, as modalidades e representações suportadas, as características da rede por onde trafegam os dados e as preferências do usuário, irão afetar a experiência do usuário em acessar determinado conteúdo.

Como existe uma infinidade de adaptações possíveis, quanto maior a quantidade de serviços de adaptação disponíveis, maior a chance de satisfazer as necessidades dos usuários. Neste caso o uso de serviços *Web* e seus padrões, que promovem a independência de tecnologia de software e hardware utilizados, facilitam o desenvolvimento de novos servidores. Isso torna o paradigma de serviços *Web* extremamente útil em ambientes de adaptação de conteúdo, onde a heterogeneidade é um dos principais problemas a ser tratado. Além disso, atualmente existe uma vasta quantidade de ferramentas que suportam e facilitam o desenvolvimento, a distribuição e a invocação de serviços *Web*.

Para se definir as adaptações necessárias a uma determinada requisição, as informações sobre o contexto de entrega e os serviços de adaptação disponíveis devem ser disponibilizadas em um formato comum de representação. Entretanto, um padrão sintático não é suficiente em ambientes abertos (e.g., *Web*), porque não é possível forçar o uso de uma única representação sintática para uma mesma semântica. É necessária à compreensão comum da semântica para a interação entre aplicações nesses ambientes abertos, para que seja possível determinar um *match* entre o contexto de entrega e os serviços de adaptação disponíveis. Caso nenhum *match* exato seja encontrado entre os serviços de adaptação disponíveis, uma composição de serviços pode ser criada, usando-se vários serviços disponíveis, de modo a satisfazer as necessidades do usuário.

Matching pode ser definido como um procedimento para se determinar uma relação entre as necessidades do usuário e os serviços disponíveis no ambiente. Do resultado do procedimento de *matching* espera-se a informação sobre os potenciais serviços disponíveis no ambiente que possam satisfazer a requisição do usuário. Para possibilitar o uso de *matching* em ambientes abertos é necessário identificar o "significado" das entidades. O "significado" de uma entidade particular pode ser definido em um vocabulário chamado ontologia [108]. A fim de se alcançar o objetivo de especificar o significado dos conceitos usados para descrever recursos da *Web*, o *World Wide Web Consortium* (W3C) está propondo a *Web* Semântica [5]. Esta permite a descoberta e a seleção dos recursos da *Web* baseados no conteúdo, em vez de um simples match baseado em palavras chave, que é uma prática comum de matching adotada atualmente. A finalidade da *Web* Semântica é desenvolver linguagens e mecanismos para expressar a informação de uma forma que seja interpretável por computadores.

A definição de ontologias é uma tentativa de modelar domínios do mundo real, e de incluir as entidades que são relevantes a um domínio particular e as relações entre essas entidades definidas. Para especificar estes relacionamentos, algum tipo de correlação lógica entre entidades deve ser suportado. A *Web Ontology Language* (*OWL*) é uma linguagem baseada em *eXtensible Markup Language* (*XML*) que tem a capacidade de especificar essa semântica. A *OWL* suporta Lógica de Descrição (*DL*) [6] e resolve assim o problema de representar logicamente relações entre entidades num modelo particular de domínio. A *DL* é uma família de linguagens de representação do conhecimento que podem ser usadas para representar o conhecimento terminológico de um domínio de aplicação de uma maneira estruturada e formalmente bem-compreendida [6].

Com o objetivo de prover significado ao conteúdo de Serviços *Web*, a coalizão *DARPA Agent Markup Language* (*DAML*) [7] desenvolveu o padrão *Web Ontology Language for Services* (*OWL-S*) [8]. Através desse padrão pode-se definir a "finalidade" de um serviço *Web* atribuindo-se um significado semântico. Os serviços *Web* descritos semanticamente permitem que programas de computador possam decidir, de forma autônoma, se um determinado serviço *Web* satisfaz ou não determinadas exigências. No contexto de adaptação de conteúdo, isso pode resultar numa maior automatização e aumento na eficácia do processo de seleção e composição de serviços, satisfazendo as necessidades do contexto de entrega.

1.1 Caracterização do Problema

Uma das questões fundamentais na adaptação de conteúdo é, a partir de uma requisição de conteúdo, se existe ou não necessidade de adaptação. Em caso afirmativo, novas questões devem ser respondidas:

- Quais adaptações serão necessárias?
- Onde serão executadas essas adaptações?
- Qual será a ordem de execução dessas adaptações?
- Será necessária a composição de serviços para a execução de uma determinada adaptação?

Essas decisões são realizadas com base numa política de adaptação, que a partir das informações do contexto de entrega, armazenadas em perfis, e de um conjunto de regras, são definidas quais ações devem ser executadas.

Atualmente não existem padrões bem definidos para descrição do contexto de entrega. Algumas tentativas de padronização, tais como o *User Agent PROFile* (*UAPROF*) [9] e o *Wireless Universal Resource FiLe* (*WURFL*) [10], só abrangem a descrição de características dos dispositivos de acesso e geralmente são incompatíveis entre si. Além disso, as abordagens para a descrição de serviços de adaptação são geralmente especificas a um determinado tipo de serviço.

A falta dessas informações limita a capacidade da política de adaptação, sobretudo para decidir quais serviços de adaptação serão necessários para o atendimento de uma determina requisição, gerando assim resultados inesperados e/ou indesejados.

1.2 Objetivos da Dissertação

Visando implementar uma política de adaptação flexível, possibilitando que novos contextos e serviços de adaptação podem ser facilmente inseridos, esta dissertação propõe a especificação de perfis e regras, baseada em ontologias, para a adaptação de conteúdo na Internet.

Para proporcionar a independência da política de adaptação em relação a mudanças no contexto de entrega (e.g., criação de novos dispositivos de acesso, novos tipos de conteúdo), as regras para a seleção e a composição de serviços são descritas nos próprios perfis. Ou seja, no perfil de um serviço já estão inseridas as regras que descrevem as condições necessárias para a sua invocação.

Outro objetivo é a inclusão da tecnologia de serviços *Web* na solução proposta. O uso de serviços *Web* facilita o desenvolvimento de novos serviços, além da possibilidade de se adaptar alguns serviços já disponíveis. Para viabilizar essa solução

são utilizados novos padrões que adicionam semântica às tecnologias de serviço *Web*, facilitando a descoberta e a composição de serviços.

Para demonstrar a validade da proposta, a solução é implementada utilizando-se um *Framework* para Adaptação de Conteúdo na Internet (FACI) [11]. Esse framework fornece uma estrutura básica para o desenvolvimento de diferentes aplicações de adaptação de conteúdo. Essa estrutura baseada em componentes combinados facilita a alteração e especialização das funcionalidades do FACI.

Durante a realização deste trabalho, o FACI foi recontruído com base em ontologias e serviços *Web*. Para suportar essas modificações alguns componentes foram adaptados e outros criados. O suporte a vários padrões de linguagens e protocolos foi adicionado a partir do uso de *API's* disponíveis publicamente. A partir dessa implementação e do desenvolvimento de um servidor de adaptação, foi realizado um estudo de caso.

1.3 Organização

Esta dissertação está organizada da seguinte forma:

O capitulo 2 aborda aspectos da adaptação de conteúdo, apresenta as atuais propostas para o desenvolvimento de conteúdo que seja independente de dispositivo, e descreve as características do contexto de entrega. Também são abordados os tipos de adaptação de conteúdo, onde essas adaptações podem ser processadas, e os protocolos de chamada externa. Por fim, são definidos os conceitos de perfis e regras de adaptação e apresentadas algumas abordagens para a descrição de perfis e regras.

O capítulo 3 tem por objetivo descrever conceitos que levem o leitor a compreender três aspectos importantes desta dissertação. O primeiro é o uso de

semântica para a descrição de domínios, onde se versa sobre ontologias e suas principais características, incluindo a integração desta com outras tecnologias (e.g., serviços *Web*). O segundo é a integração de Lógica de Descrição a algumas linguagens, facilitando o desenvolvimento de mecanismos de decisão e descoberta de serviços. O terceiro aborda os conceitos de *matching* e composição de serviços.

O capitulo 4 apresenta os perfis desenvolvidos durante este trabalho, que inclui os perfis de dispositivo, usuário, rede, conteúdo, *Service Level Agreement* (*SLA*), servidores de adaptação, e de serviço. Baseado nesses perfis se descreve o funcionamento do mecanismo de descoberta e composição de serviço, e se demonstra como as regras podem ser integradas aos perfis de serviço. Também destaca as alterações realizadas no FACI com o objetivo de suportar as novas funcionalidades propostas. Por fim, discute os trabalhos relacionados e apresenta as contribuições desta dissertação.

O capitulo 5 versa sobre o estudo de caso realizado para avaliar a proposta deste trabalho. Para viabilizar esse estudo foi desenvolvido um servidor de classificação e filtragem de conteúdo, a partir do qual foram executados testes de eficiência do módulo de classificação. Também são apresentados exemplos sobre o uso dos perfis no processo de adaptação de conteúdo. Para verificar o impacto do uso de ontologias no FACI foram mensurados a capacidade de carga e o tempo de resposta a requisições do usuário.

Finalmente, o capitulo 6 conclui este trabalho apresentando os resultados alcançados e contribuições, limitações e dificuldades encontradas, e possibilidades de trabalhos futuros.

2 ADAPTAÇÃO DE CONTEÚDO

O mercado de dispositivos móveis é caracterizado por uma grande diversidade de tecnologias de hardware e software. Essa diversidade impede que uma única tecnologia de sistemas operacionais, navegadores e linguagens de marcação, possam atender esse mercado. Esse fato não está ligado à defasagem tecnológica de alguns dispositivos, e sim as várias combinações de capacidade, aplicação, e custo desses dispositivos.

Entre os vários tipos de dispositivos móveis os telefones celulares são o que mais se destacam. Apenas no Brasil, em março de 2006, já existiam 89,4 milhões de aparelhos [12], sendo que uma porcentagem crescente desses aparelhos possui capacidade de acesso à internet. Para suportar essa capacidade novos protocolos e linguagens de marcação foram desenvolvidos de modo a atender as várias demandas de mercado.

2.1 Computação Ubíqua

O visionário pesquisador *Mark Weiser* vislumbrou uma forma pouco intrusiva de utilizar dispositivos computacionais no cotidiano das pessoas. Segundo ele, facilidades computacionais devem ser incorporadas aos ambientes de modo a auxiliar as atividades humanas, mudando minimamente a forma como as tarefas são realizadas [13,14]. *Weiser* denominou "computação ubíqua" essa forma transparente de integrar tecnologias às atividades humanas diárias. Para a idealização desse paradigma, Weiser previu o desenvolvimento de novos dispositivos computacionais que permitiriam a computação ser adaptada ao cotidiano do ser - humano e conseqüentemente o apare-

cimento de novas aplicações computacionais para explorar o uso desses dispositivos. Os dispositivos computacionais foram classificados quanto ao seu tamanho em: pequenos e pessoais (*inch-scale*, Figura 1 (a)), de médio porte (*foot-scale*, Figura 1 (b)) e grandes e de uso coletivo (*yard-scale*, Figura 1 (c)). As previsões de Weiser foram comprovadas, fazendo da computação ubíqua uma área de interesse de pesquisa em computação.

Figura 1. Exemplos de dispositivos computacionais ubíquos.

Abowd et al. [16] destacam aplicações de captura e acesso, interfaces naturais e computação ciente de contexto (*context-aware*) como os principais temas de pesquisa em computação ubíqua. Aplicações de captura e acesso visam registrar de modo automático a experiência humana cotidiana. Interfaces naturais permitem ao ser humano interagir com os dispositivos computacionais da mesma maneira como interagem com o mundo físico. Computação ciente de contexto possibilita que aplicações usem o contexto da interação do ser humano com o computador, com o intuito de fornecer serviços dedicados de modo transparente. À medida que a computação não intrusiva passou a fazer parte da vida das pessoas surgiu um novo tema que se associa aos três anteriormente citados, a computação no cotidiano (*everyday computing*) [15,16]. Nesse tipo de computação, devem-se disponibilizar serviços vinte e quatro horas por dia, sete dias por semana, auxiliando atividades informais e não estruturadas comuns no dia-a-dia dos indivíduos.

2.1.1 Independência de Dispositivo

A *Web* está aumentando rapidamente o seu alcance além do *desktop*, atingindo dispositivos que variam de telefones móveis a dispositivos domésticos. Essa rápida expansão de acessibilidade se deve principalmente a utilização de protocolos e linguagens de marcação abertos na *Web*, que disponibiliza a maior infra-estrutura global para acesso a conteúdo e aplicações.

O objetivo original do *HyperText Transfer Protocol (HTML)* era fornecer uma linguagem de marcação independente de dispositivo que fosse baseada na semântica do conteúdo. Identificava elementos do documento tais como títulos, parágrafos, e listas, sem especificar a apresentação. Entretanto, os desenvolvedores de navegadores (*browsers*) introduziram novos elementos e atributos específicos para apresentação no *HTML*, dificultando a distinção entre a semântica e a apresentação. Teoricamente uma apresentação criada para um dispositivo de tela grande pode ser visualizada em um dispositivo de tela pequena, caso ambos usem a mesma linguagem de marcação. Entretanto a visualização geralmente é comprometida porque o autor não criou a apresentação pensando em um usuário que utilize um dispositivo de tela pequena.

A independência de dispositivo é uma tentativa de se resgatar algumas idéias originais da *Web*. Os novos padrões desenvolvidos para *Web* estão incentivando o retorno da distinção entre a semântica e a apresentação por meio de linguagens de estilo tais como o *Cascading Style Sheets* (*CSS*) e o *Extensible Stylesheet Language Formatting Objects* (*XSL-FO*), e as linguagens de marcação de interação, tais como *XML Forms* (*XForms*) para a entrada de dados.

Enquanto o número de dispositivos que acessam a Internet aumenta, o problema em criar apresentações para cada tipo de dispositivo cresce de forma mais grave. O ideal é que os autores criem somente uma versão de seu conteúdo. Então, durante a entrega e o processo de conversão, o software de adaptação criaria uma apresentação compatível com as características do dispositivo de acesso. Para realizar esse processo são necessárias respostas a algumas questões [17]:

- Como definir uma aplicação *Web* que seja independente de dispositivo de acesso?
- Como adaptar aplicações independentes de dispositivo às capacidades do contexto de entrega?
- Como os autores podem reter algum controle sobre a apresentação final de seu conteúdo?

As atuais tecnologias *Web* podem dar respostas parciais a essas perguntas. Pesquisadores e desenvolvedores estão trabalhando em novas tecnologias que irão oferecer soluções mais completas. Aqui, é apresentada uma visão geral de ambas.

Atualmente, os autores tipicamente projetam aplicações *Web* para o navegador e a tela do PC. Para exercer um controle máximo sobre a aparência final, geralmente baseiam o layout da página *Web* em tabelas de posicionamento absoluto das informações. Por causa dessa opção, adaptar tal página para uma tela pequena é eficazmente impossível, fazendo com que os autores criem um site paralelo para acomodar esses dispositivos. A Figura 2 mostra um conteúdo similar adaptado para diferentes dispositivos. Como a variedade de dispositivos com capacidade de acesso a *Web* aumenta a cada dia, criar um site separado para cada tipo de dispositivo é economicamente e administrativamente inviável.

Figura 2. Conteúdo adaptado para diferentes dispositivos [17].

O objetivo da independência de dispositivo é possibilitar que os conteúdos desenvolvidos para Web sejam visualizados por dispositivos com diferentes características. De fato, geralmente é desnecessário que os autores criem um conteúdo completamente diferente para cada dispositivo. As capacidades do visor dos telefones celulares e dos *Personal Digital Assistent (PDAs)*, por exemplo, se sobrepõem em alguns casos - embora as capacidades de entrada de texto possam diferir. O conceito chave para se obter uma independência de dispositivo baseia-se em combinar a formatação do conteúdo com as capacidades dos dispositivos.

Além de diminuir a sobrecarga do autor, a independência de dispositivo oferece aos usuários diversos benefícios. A acessibilidade, por exemplo, é um interesse fundamental, e em alguns países uma necessidade legal. Os usuários podem interagir com a *Web* de um modo adaptado a suas capacidades. Oferecer opções que possibilitem aos usuários substituir imagens por texto, converter o texto para áudio, ou interagir

usando a voz ou dispositivos de entrada especiais podem beneficiar um grande número de usuários. Fazendo a adaptação de conteúdo um aspecto fundamental do projeto, os designers de sites *Web* aumentam o número de opções ao usuário e também se ajustam melhor as preferências de apresentação.

2.1.2 Contexto de Entrega

Segundo o *W3C*, o termo contexto de entrega deve ser interpretado como: "Um conjunto de atributos que caracteriza as capacidades do dispositivo de acesso, as preferências do usuário e outros aspectos do contexto que envolva a entrega de um conteúdo *Web*." [4].

Há vários aspectos do contexto de entrega, que podem influenciar na escolha da melhor representação de algum conteúdo *Web,* em resposta a uma requisição. Nesta seção, são apresentadas algumas características que podem expressar aspectos do contexto de entrega. No entanto, o conjunto de características potenciais é flexível, não se restringindo as apresentadas a seguir [4]:

- Interação
    - modalidades de apresentação (saída) e seus parâmetros.
    - modalidades de captura (entrada) e seus parâmetros.
- Capacidades dos dispositivos de acesso.
    - capacidades de processamento (e.g., *scripting*).
    - formatos aceitos e gerados (e.g. tipos *Multipurpose Internet Mail Extensions* (*MIME*)).
- Conexões
    - largura de banda, latência.

  - redes e protocolos.
  - informações associadas com as operadoras de telecomunicação.
- Situação
  - coordenadas geográficas.
  - proximidade de outros recursos
  - hora do dia.
- Local
  - idioma local.
  - fuso horário local.
- Ambiente
  - Temperatura.
  - Luz.
  - Ruído.
- Confiança
  - privacidade e segurança.
  - restrições de conteúdo.

As características mais relevantes para se alcançar a independência de dispositivos são as que descrevem as capacidades do mecanismo de acesso, as capacidades da rede e as preferências do usuário. Em particular, um usuário pode especificar preferências de adaptação que afetem a sua experiência em relação ao conteúdo entregue.

2.2 Negociação e Adaptação de Conteúdo

Para se atingir a independência de dispositivo são necessárias soluções que adaptem o conteúdo original as características do contexto de entrega. Tecnologias de negociação e adaptação, que podem trabalhar conjuntamente, foram desenvolvidas para atender essas necessidades.

2.2.1 Negociação de Conteúdo

Um dos métodos mais utilizados para se apresentar um mesmo conteúdo em diferentes dispositivos é a negociação. A partir de algumas características obtidas do cabeçalho *HTTP* de requisição, o servidor de conteúdo decide qual das opções de formatação do conteúdo irá se adequar melhor ao dispositivo de acesso.

Apesar de simples esse método apresenta várias limitações, sendo mais utilizado para adaptação de idioma, não suportando todas as necessidades de adaptação requeridas pelos dispositivos móveis. Mas, ao mesmo tempo, a sua utilização pode facilitar e acelerar a adaptação de conteúdo, quando utilizado juntamente com outros métodos.

Multipurpose Internet Mail Extension (MIME) - No contexto da *Web*, o *MIME* [18] informa ao *browser* que tipo de conteúdo foi recebido do servidor *Web*. Por meio dessa informação o *browser* consegue diferenciar se os dados recebidos é uma página *HTML*, uma imagem, um arquivo de áudio ou vídeo e que aplicação suportará este formato. Cada formato tem o seu tipo especifico (*MIME types*) regulamentado pela *Internet Assigned Numbers Authority (IANA)* [19]. Os tipos mais comuns são apresentados na Tabela 1.

Tipo de Documento	Tipo *MIME*
HTML	*text/html*
Graphic Image Format (GIF)	*image/gif*
Joint Photographic Experts Group (JPEG)	*image/jpg*
Wireless Markup Language (WML)	*text/vnd.wap.wml*
Compiled WML	*application/vnd.wap.wmlc*
WMLScript	*text/vnd.wap.wmlscript*
Compiled WMLScript	*application/vnd.wap.wmlscriptc*
Wireless BitMaP (WBMP)	*image/vnd.wap.wbmp*
Extensible HTML (XHTML)	*application/xhtml+xml*
Cascading Style Sheets (CSS)	*text/css*
PALM	*application/vnd.palm*

Tabela 1. Tipos *MIME* mais comuns.

No cabeçalho *HTTP* o campo *content-type* contém a descrição do *MIME*, além de outros parâmetros opcionais (e.g*., Content-Type = text/html*; *charset=ISO-*8859-4).

***HTTP Accept Headers* e *User-Agent* -** Os campos *Accept* definem quais tipos de documentos (*MIME types*) são suportados, incluindo algumas preferências do usuário (e.g., idioma) de modo que o servidor possa selecionar o conteúdo mais adequado para a respectiva requisição. Caso o servidor não satisfaça esses campos ele deve retornar uma mensagem de erro 406 (*not acceptable*). Os campos *Accept* padrões do *HTTP* 1.1 [20] são:

- *Accept: media type* (*MIME type*) - e.g., *Accept* : *text*/*
- *Accept-Charset*: cjto de carateres - e.g., *Accept-Charset* : *iso*-8859-5
- *Accept- Encoding*: codificação do conteúdo – e.g., *Accept-Encoding* : *gzip*.
- *Accept-Language*: idioma – e.g., *Accept-Language* : br, en.

Em adição aos campos *Accept*, o campo *User-Agent* geralmente contém um string com informações sobre o *browser*. Tipicamente inclui o nome do fabricante, a versão e o nome do produto. Em dispositivos móveis é comum incluir informações sobre o dispositivo. Não existe um padrão sobre as informações contidas no campo

User-Agent. Apesar de alguns *browsers* poderem simular as informações de produtos concorrentes (e.g., O *browser* Opera pode identificar-se como sendo o Internet Explorer) as informações contidas nesses campos devem ser utilizadas no processo de negociação e adaptação de conteúdo.

HTTP Content Negotiation - A especificação *HTTP* 1.1 [20] permite que o servidor de conteúdo possa determinar, entre várias alternativas, que conteúdo enviará ao requisitante. Apenas os campos *Accept* são utilizados nessa negociação. De modo a facilitar o processo de negociação é associado, opcionalmente, aos campos *Accept* um valor de qualidade. Por exemplo, o usuário tem preferência pelo conteúdo em português, mas também aceita, com uma menor satisfação, um conteúdo em inglês: *Accept-Language*: br; q=1.0, en; q=0.5. Nesse caso o servidor tentará enviar uma versão da página em português e caso não exista enviará em inglês.

Existem algumas desvantagens associadas a esse método: As informações contidas nos campos *Accept* não possuem uma descrição completa das capacidades do cliente. Por exemplo, no caso de uma imagem pode-se definir o seu tipo *mime*, mas não a resolução ou a quantidade de cores; Também deve se considerar o *overhead* em se enviar todas as possibilidades a cada requisição *HTTP*.

HTTP Transparent Content Negotiation - A escolha do conteúdo também pode ser realizada no dispositivo de acesso, na *Request for Comments* (*RFC*) 2295 foi especificado um protocolo que informa ao *browser* todas as opções de conteúdo oferecidas por um determinado servidor, permitindo que se escolha a melhor opção. Nesse método o servidor envia uma lista com todas as variantes de uma determinada página e suas propriedades ao *user agent*. Um exemplo de uma lista com três variantes é apresentado a seguir:

```
{"paper.1" 0.9 {type text/html} {language en}},
{"paper.2" 0.7 {type text/html} {language fr}},
("paper.3" 1.0 {type application/postscript} {language en}}
```

Quando a lista é recebida, o *browser* pode escolher a melhor variante e requisita-la. O fluxo de mensagens é apresentado na Figura 3.

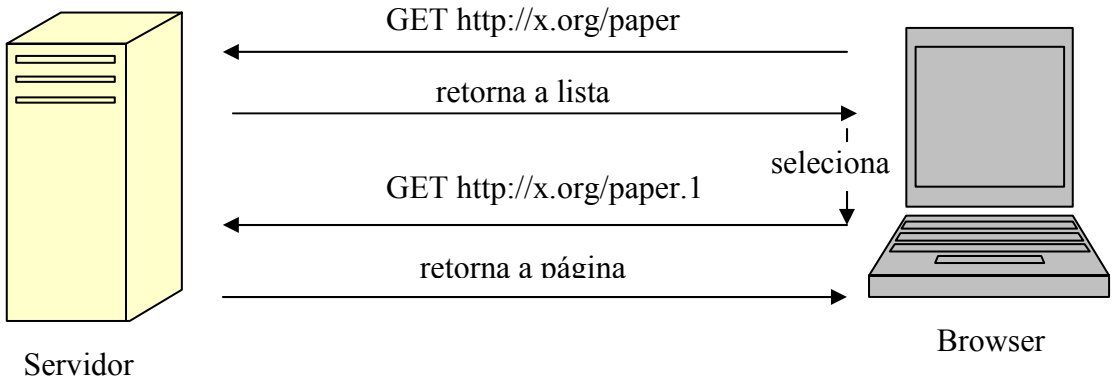


Figura 3. Fluxo de mensagem de uma negociação de conteúdo.

Como nesse método as informações sobre capacidades e preferências são utilizadas apenas pelo *browser*, o envio de um longo cabeçalho *Accept* é desnecessário. O envio de curtos cabeçalhos *Accept* pode acelerar o processo de negociação quando a escolha utilizar um algoritmo remoto de seleção executado no servidor. Este algoritmo utiliza a lista de variantes e os cabeçalhos *Accept* como variáveis de entrada e escolhe a melhor variante.

Um dos algoritmos mais utilizados é o *HTTP Remote Variant Selection Algorithm* – (RVSA/1.0) [21]. Ele calcula a qualidade geral de uma variante por meio da equação:

Q = round5 (qs * qt * qc * ql * qf)

onde round5 é o produto, em ponto flutuante e precisão de 5 casas, dos fatores qs, qt, qc,ql e qf, descritos a seguir:

- qs – fator de qualidade da variante da página (predefinida no servidor).
- qt – fator de qualidade do tipo de mídia. Caso não exista este campo será adotado o valor um; e se o tipo de mídia não for compatível com nenhuma das opções suportadas pelo *browser* será utilizado o valor 0; estas regras também valem para os dois fatores abaixo.
- qc – fator de qualidade do conjunto de caracteres.
- ql – fator de qualidade do idioma.
- qf – fator de qualidade das características.

Para exemplificar o funcionamento do algoritmo são utilizadas a lista de variantes e os cabeçalhos *Accept* descritos a seguir:

```
{"paper.html.en" 0.9 {type text/html} {language en}},
{"paper.html.fr" 0.7 {type text/html} {language fr}},
{"paper.ps.en"   1.0 {type application/postscript} {language en}}

Accept: text/html:q=1.0, */*:q=0.8
Accept-Language: en;q=1.0, fr;q=0.5
```

Com esses dados obteremos os seguintes fatores Q, sendo selecionada a página paper.html.en por atingir o maior valor de qualidade.

```
          round5 ( qs  * qt  * qc  * ql  * qf  ) =   Q
                  ---   ---   ---   ---   ---     -------
paper.html.en: 0.9 * 1.0 * 1.0 * 1.0 * 1.0   = 0.90000
paper.html.fr: 0.7 * 1.0 * 1.0 * 0.5 * 1.0   = 0.35000
paper.ps.en:   1.0 * 0.8 * 1.0 * 1.0 * 1.0   = 0.80000
```

2.2.2 Adaptação de Conteúdo

A adaptação de conteúdo permite que um único conteúdo original seja convertido para um grande número de formatos, compatíveis com o contexto de entrega. Essa adaptação pode ser uma transformação, onde a linguagem de marcação é adaptada e/ou uma transcodificação, onde imagens são convertidas ou redimensionadas.

Apesar de ser um processo mais lento e complexo, a adaptação é mais flexível que a negociação e permite que conteúdos originais que possuam apenas um formato disponível possam ser acessados através de dispositivos móveis.

Transformações - De modo a converter uma página *HTML* em uma linguagem de marcação aceita pelo *browser* de um dispositivo móvel (e.g. *WML*, *XHTML*, *Compact HTML* (*cHTML*)) é utilizado o processo chamado transformação. Dentre os métodos existentes o *Extensible Stylesheet Language Transformation (XSLT)* [22] converte um conteúdo *XML* para diferentes formatos.

Sendo uma meta linguagem, o *XML* é utilizado para representar dados em um formato padrão, definir linguagens de marcação e armazenar informação estruturada. É um padrão aberto desenvolvido pelo *W3C*, exercendo um papel importante na troca de uma grande variedade de dados na *Web* e em vários outros campos.

XML é uma linguagem bem formada que respeita as seguintes regras:

- Todo *tag* de inicio deve possuir um *tag* de fim correspondente.
- *Tags* não podem se sobrepor porque o *XML* é estritamente hierárquico.
- O elemento de maior nível em um documento *XML* é o único elemento raiz, e todos os outros serão seus filhos ou descendentes.
- Os nomes dos elementos/*tags* devem obedecer a certas regras, como não possuir espaços, e a primeira letra não pode ser um número.
- *Tags* são *case-sensitives*.
- Os espaços em branco no texto serão mantidos.

Quando um arquivo *XML* é criado e o conteúdo é incluído nesse arquivo sem nenhuma informação adicional sobre formatação, é necessário um processo para

converter o dado puro em um formato legível, o *XSTL* é utilizado para esse propósito. De acordo com diferentes *templates* de linguagem de marcação expressos em *eXtensible Stylesheet Language* (*XSL*), o conteúdo em *XML* será convertido para o formato desejado, a Figura 4 ilustra essa transformação.

Infelizmente o *XSLT* é aplicado no nível de estrutura do documento e não pode executar adaptações em imagens e outros tipos de mídias. Consequentemente outros mecanismos de adaptação (e.g., transcodificação) devem ser utilizados de modo a complementar as transformações.

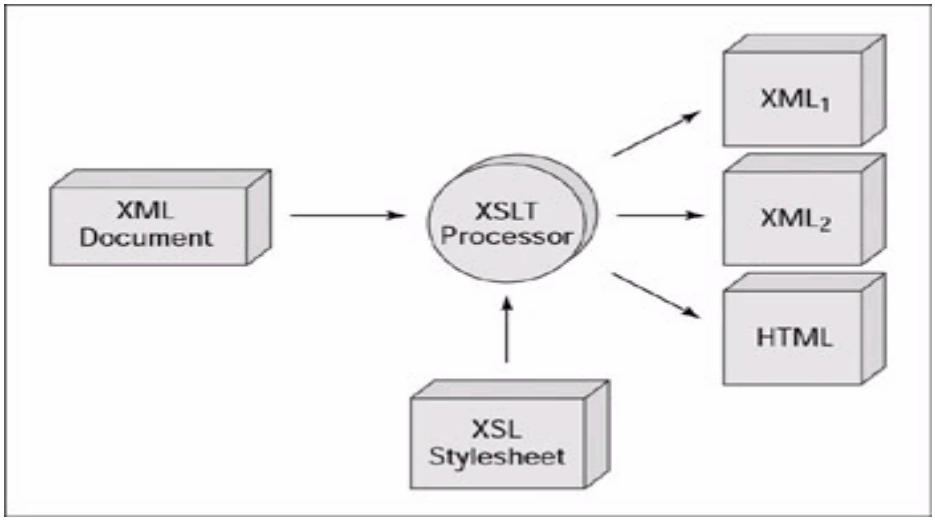


Figura 4. *XSLT* transforma *XML* em uma variedade de formatos.

Adaptação de imagens - Vários fatores podem demandar a adaptação de imagens: tamanho de tela, resolução, conversão de formatos (e.g., *jpeg* para *wbmp*), número de cores, compactação e redução de tamanho do arquivo, necessários para redes de baixa largura de banda. As definições de quais adaptações serão necessárias são baseadas nas capacidades dos dispositivos e preferências dos usuários. Uma das ferramentas mais utilizadas para adaptação de imagens é a *ImageMagick* [23].

Serviços de Valor Agregado - As restrições de capacidade de processamento e memória, presentes na maioria dos dispositivos móveis, impedem a execução de vários aplicativos. Aplicações como antivírus, filtragem de conteúdo, tradução de idiomas, entre outras, podem ser executadas em servidores de adaptação externos, estendendo os benefícios dessas aos dispositivos móveis.

2.2.3 Aplicações de Adaptação de Conteúdo

Dentre as aplicações de adaptação de conteúdo na Internet [99,100], destacam-se:

- Varredura de vírus: capacidade de desempenhar dinamicamente a busca por vírus sobre o conteúdo requisitado antes de entregá-lo ao usuário. Com esse serviço o usuário acessa a Internet sem o risco de receber um conteúdo infectado.
- Inserção de propaganda: adiciona informações publicitárias num conteúdo (e.g., páginas *Web*) baseado nos interesses do usuário ou na sua localização geográfica. Essa adaptação permite, por exemplo, que as propagandas originais das páginas *Web* sejam substituídas por outras regionais ou focadas no perfil do usuário.
- Tradução de linguagens de marcação: permite aos dispositivos que não suportam páginas *HTML* (e.g., telefones celulares), receberem essas páginas em outro formato (e.g., *WML*). Esse serviço além de disponibilizar um número maior de conteúdo para dispositivos limitados, alivia o trabalho que o servidor de origem teria para desenvolver duas apresentações de um mesmo conteúdo.
- Compressão de dados: permite que o cliente receba seus conteúdos compactados, economizando a largura de banda utilizada nessa comunicação.

- Filtragem de conteúdo: capacidade de redirecionar uma requisição não autorizada ou bloquear uma resposta também não permitida. Esse serviço permite, por exemplo, aos pais controlarem o acesso de seus filhos a conteúdos impróprios.
- Transcodificação de imagens: trata-se da capacidade de manipular os arquivos de imagem, assim como: transformar seu formato; reduzir seu tamanho, resolução e sua escala de cores; ou remover imagens de uma página *Web*.
- Tradução de idiomas: traduz uma página *Web* de um idioma para outro, por exemplo, de português para chinês. Pode ser útil quando o servidor de origem não contiver todas as traduções necessárias para entregar um mesmo conteúdo em múltiplos idiomas.

2.3 Processadores de adaptação

Um processador de adaptação [26] é um mecanismo que obtêm uma forma de conteúdo como entrada e produz uma forma alternativa como saída. Processadores de adaptação normalmente são baseados em software, embora existam processadores baseados em hardware.

Adaptações podem ser simples, como realizar pequenas alterações na linguagem de marcação para contornar pequenas diferenças entre *browsers*, ou podem ser complexas quanto mover um conteúdo de uma modalidade (e.g., texto) para outra (e.g., voz sintetizada).

São identificados três locais, onde os processadores de adaptação podem ser implementados: no servidor de conteúdo, no dispositivo de acesso e nos sistemas intermediários (e.g. *proxy*). Processadores de adaptação podem cooperar entre si, sendo

que o conteúdo final adaptado pode ser o resultado de uma composição de adaptações aplicadas em diferentes níveis.

2.3.1 Adaptação no Servidor de Conteúdo

A adaptação no servidor de conteúdo é uma solução largamente utilizada. Essa abordagem apresenta as seguintes vantagens:

- Os desenvolvedores de conteúdo tem o total controle sobre o conteúdo armazenado no servidor, podendo desenvolver um conteúdo com estilo, navegação e layout otimizados para dispositivos móveis.
- O conteúdo pode ser estruturado utilizando *XML,* o que facilita o processo de transformação em outras linguagens de marcação.
- Geralmente possui grande capacidade computacional e de armazenamento, e facilidade para acesso a banco de dados.
- Por meio de negociação e informações sobre o contexto de entrega, o servidor poderá enviar um conteúdo que o dispositivo de acesso terá capacidade de apresentar.

Também existem desvantagens no uso dessa abordagem, e as mais importantes são:

- Um grande volume de adaptações pode comprometer a capacidade de processamento, causando lentidão.
- Os perfis de usuário e de dispositivo devem ser enviados a cada servidor de conteúdo acessado, aumentando a sobrecarga da rede e diminuindo a privacidade das informações.

- A complexidade e o alto custo em se implementar serviços de adaptação, em um único local, que suportem a grande diversidade de dispositivos de acesso, *browser* e conteúdo.
- O usuário só poderá acessar os servidores que possuam uma solução de adaptação de conteúdo.

Um exemplo de implementação é o servidor *Apache Cocoon* - que utiliza o *XSLT* para gerar a marcação apropriada a partir de um conteúdo baseado em *XML*. O autor deve fornecer diversas folhas do estilo (*XSL*) de modo a gerar a marcação apropriada para os diferentes dispositivos de acesso.

2.3.2 Adaptação no Dispositivo de Acesso

A adaptação no dispositivo de acesso geralmente está limitada à capacidade do *browser* em utilizar o *HTML/CSS*. Desenvolvedores de conteúdo podem inserir diretivas no conteúdo de modo a auxiliar a adaptação no dispositivo de acesso. Utilizando as folhas de estilo o *browser* poderá criar uma apresentação satisfatória ao usuário. O *CSS* é igualmente aplicável, em navegadores que o suportam, para definir o estilo do *XHTML*, do Scalable Vector Graphics (*SVG*), ou mesmo de um conteúdo *XML*. O uso de diferentes folhas de estilo permite adaptar um único conteúdo a diferentes dispositivos.

Os tipos de mídia *CSS* (não confundir com os tipos de mídia *MIME* da Internet) são nomes que identificam diferentes tipos de dispositivo, tais como tela, *handheld*, tevê, impressão, projeção, e Braille. No CSS, os autores podem definir diferentes regras de estilo para diferentes tipos de mídia. Em dispositivos menores, os

autores podem usar tipos de mídia CSS para omitir a exposição de partes de uma pagina *Web*.

Uma das vantagens deste método é a facilidade do aplicativo em acessar as capacidades do dispositivo. Mas as limitações de capacidade de processamento, memória e energia da maioria dos dispositivos móveis restringem a execução de adaptações mais complexas. Outra restrição é a grande variedade de hardware e sistemas operacionais disponíveis nesses dispositivos, o que impossibilita o desenvolvimento de uma única solução que possa ser executada em todas essas plataformas.

2.3.3 Adaptação em sistemas intermediários (*Proxy*)

A adaptação também pode ser realizada externamente em um *proxy*. Essa abordagem apresenta as seguintes vantagens:

- Elimina a instalação de software no dispositivo de acesso e/ou servidor de conteúdo, e independe do sistema operacional e características do dispositivo de acesso.
- Como todas as requisições do usuário passam pelo *proxy*, qualquer servidor de conteúdo poderá ser acessado pelo usuário, pois a adaptação será realizada no *proxy*.
- O *proxy* poderá armazenar conteúdo adaptado em seu cache aumentando a eficiência do sistema.
- O *cache* pode ser utilizado como um local único para armazenamento de perfis e regras, evitando redundâncias, melhorando a consistência e aumentando a privacidade das informações.

- A adaptação no sistema intermediário pode ajudar a reduzir a sobrecarga no servidor *Web*, mas somente é bem sucedida quando é baseada no conhecimento das capacidades do dispositivo de acesso e meta dados com informações providas pelo autor da página.

Também existem desvantagens em se usar esta abordagem, e as mais importantes são:

- O excesso de adaptações pode sobrecarregar o *proxy* gerando lentidão.
- A Complexidade e o alto custo em se implementar serviços de adaptação, em um único local, que suportem a grande diversidade de dispositivos de acesso, *browser* e conteúdo.

2.4 Protocolos de Chamada Externa - *Callout Protocols*

Os protocolos de chamada externa possibilitam que um dispositivo intermediário (e.g. *proxy*) invoque serviços de adaptação a servidores distribuídos pela internet. Essa capacidade permite um balanceamento de carga e que um maior número de adaptações possa ser oferecido.

2.4.1 *Internet Content Adaptation Protocol (ICAP)*

O *ICAP* tem como objetivo possibilitar a comunicação entre equipamentos de borda e servidores de adaptação de modo a se adaptar um conteúdo o mais próximo do cliente, o que propicia uma melhor capacidade de personalização aliada a ganhos de velocidade. É um projeto, nascido em 1999, conduzido por um grupo

de empresas, composto por aproximadamente 70 participantes, chamado Fórum *ICAP* [27].

Na atual especificação o *ICAP* é um protocolo de chamada externa remota (*remote callout protocol*) usada entre *proxy cache* e servidores de adaptação onde o conteúdo será modificado. Em um ambiente *ICAP*, o *proxy* cache exerce o papel de um cliente *ICAP* habilitado a enviar requisições *ICAP* para o servidor de adaptação baseado em uma requisição do cliente (usuário). A partir deste ponto o *proxy* cache será chamado de Cliente *ICAP* e o servidor de adaptação de Servidor *ICAP*.

O cliente *ICAP* gera uma requisição *ICAP* através de um *URI* (*Uniform Resource Identifiers – RFC* 2396) *ICAP* [28]. A requisição *ICAP* encapsula cabeçalhos *HTTP* e o conteúdo *HTTP*, se o conteúdo estiver disponível.

Modificação de Requisição (*REQMOD*) - Neste modo o cliente *ICAP* encapsula uma requisição http em uma requisição *ICAP* e a envia para o servidor *ICAP* para adaptação [28]. O servidor *ICAP* pode responder as requisições da seguinte forma:

1) Devolve a versão adaptada da requisição. O cliente *ICAP* pode então executar a requisição modificada contatando o servidor de origem, ou enviar a requisição modificada para outro servidor de adaptação para a realização de outras modificações.
2) Envia uma resposta http ao requisitante, provendo informação útil ao usuário em caso de erro.
3) Responde com uma mensagem de erro.

Um exemplo de uso para este método é a restrição de acesso a certos conteúdos, especialmente, controle dos pais sobre o conteúdo que os filhos acessam. A figura 5 ilustra o fluxo de informações envolvido em uma modificação de requisição (*reqmod*).

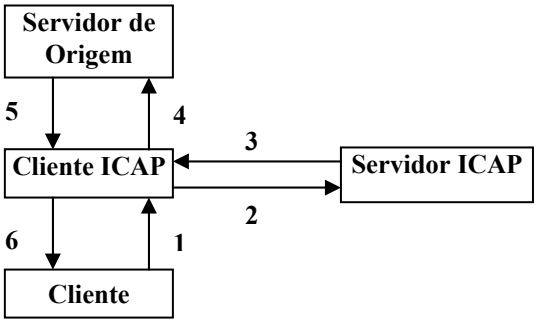


Figura 5. ICAP – REQMOD

Os passos numerados explicam o fluxo de mensagens:

1) O cliente faz uma requisição de algum conteúdo via http.
2) O cliente *ICAP* verifica a requisição *HTTP* do cliente e cria uma modificação de requisição, que inclui o tipo de serviço a ser executado e a envia para o servidor *ICAP*.
3) O servidor *ICAP* executa a solicitação e retorna uma resposta *ICAP*.
4) Se a resposta do item anterior for uma resposta de erro, o cliente *ICAP* encaminha a mensagem de erro ao cliente (6). Caso contrário, o cliente *ICAP* encaminha a requisição adaptada para o servidor de origem no formato de uma requisição http.
5) O servidor de origem envia uma resposta ao cliente *ICAP*.
6) O cliente *ICAP* envia a resposta ao cliente.

Modificação de Resposta (*RESPMOD*) - Neste modo o cliente *ICAP* encapsula a resposta *HTTP* recebida do servidor de origem em uma requisição *ICAP* e a envia para o servidor *ICAP* para adaptação [28]. O servidor *ICAP* irá responder ao cliente *ICAP* com uma resposta: adaptada ou retorna um erro. Finalizando o processo o cliente *ICAP* envia a resposta http para o cliente. Um exemplo de aplicação para este modo é a inserção de propaganda em um conteúdo obtido de um servidor de origem. A Figura 6 ilustra o fluxo de informações envolvido em uma modificação de resposta (*respmod*).

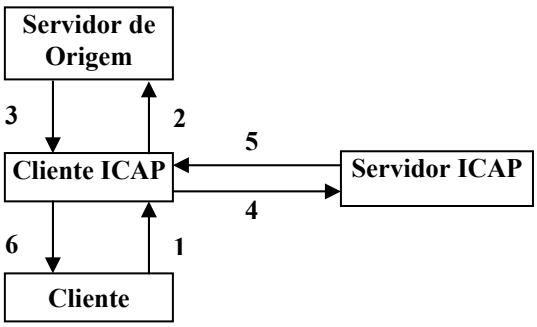


Figura 6. *ICAP – RESPMOD*

Os passos numerados explicam o fluxo de mensagens:

1) O cliente faz uma requisição de algum conteúdo via *http*.
2) O cliente *ICAP* envia a requisição para o servidor de origem.
3) O servidor de origem envia a resposta ao cliente *ICAP*.
4) O cliente *ICAP* identifica o serviço de adaptação a ser realizado na resposta, e envia uma modificação de resposta (*RESPMOD*) para o servidor *ICAP*.
5) O servidor *ICAP* executa o serviço solicitado e pode responder com uma mensagem de erro ou a resposta adaptada no formato *ICAP*.
6) O cliente *ICAP* envia a mensagem de erro apropriada ou a resposta adaptada para o cliente no formato http.

A principal vantagem do ICAP é a simplicidade em termos de cabeçalho, gerando uma pequena sobrecarga (*overhead*) no transporte do conteúdo adaptado entre o *proxy* e o servidor de adaptação. Essa característica também influi no parser desse protocolo, que demanda poucos recursos computacionais.

Como desvantagem pode-se apontar a baixa adoção desse procoloco e o pequeno número de ferramentas e API's desenvolvidas. No caso deste trabalho o suporte ao protocolo ICAP foi contruído a partir da especificação, incluindo a implementação do parser.

2.4.2 Outros Protocolos de Chamada Externa

O grupo de trabalho do *Open Pluggable Edge Services* (*OPES*) [29] desenvolveu um modelo de infra-estrutura para redes de serviços onde novos protocolos foram definidos. O *OPES Callout Protocol (OCP)*, que faz parte desses protocolos, visa desempenhar a comunicação entre o dispositivo de borda e o servidor de adaptação. O seu diferencial em relação ao *ICAP* é que além do *HTTP* ele também suporta o *Simple Mail Transfer Protocol* (*SMTP*), *Real-Time Transport Protocol (RTP)* e *File Transfer Protocol* (*FTP*).

Outro protocolo que pode operar como *Callout Protocol* é o *Simple Object Access Protocol* (*SOAP*) [30]. Ele é um dos componentes da arquitetura de serviços *Web,* sendo utilizado para o transporte de mensagens. Apesar das desvantagens em relação ao tamanho das mensagens, ele é o protocolo mais documentado, um padrão da *W3C* amplamente adotado pelo mercado, e possui uma grande quantidade de bibliotecas disponíveis.

Neste trabalho, além do *ICAP*, também é utilizado o protocolo *SOAP* devido a sua compatibilidade com serviços *Web*. A utilização desses serviços possibilita uma rápida expansão no número de adaptações (e.g., tradução de idiomas) aumentando a probabilidade de satisfazer as necessidades do usuário. Maiores detalhes sobre o *SOAP* são apresentados no tópico de arquitetura de serviços *Web* desta dissertação.

2.5 Perfis e Regras de Adaptação

Um dos aspectos fundamentais de uma arquitetura de adaptação de conteúdo é a definição da política de adaptação, ou seja, quais serviços de adaptação serão oferecidos, quais adaptadores locais ou remotos executarão essas adaptações e quando deverão ser solicitadas. Para possibilitar essas decisões são necessárias informações sobre as características do contexto de entrega, armazenadas em perfis, e as condições que devem ser atendidas para que determinada adaptação seja realizada e as ações a serem executadas, definidas pelas regras.

Informação de contexto [31] é adquirida, armazenada, e interpretada em diferentes partes do sistema. Porém, os mecanismos de adaptação de conteúdo requerem que essa informação pertinente esteja disponível. Uma representação da informação de contexto deve ser aplicável ao longo de todo processo de adquirir, transferir, armazenar, e interpretar. Então, há um conjunto de exigências relativo ao formato de representação. Uma representação de perfil de contexto deve ser:

- **Estruturada**: perfis de contexto podem representar um grande número de informações de diferentes contextos. Uma representação estruturada provê meios para filtrar a informação necessária eficientemente. Além disso, elimina nomes de atributos ambíguos, pois a interpretação será sensível ao contexto.
- **Intercambiável:** perfis de contexto devem ser intercambiáveis entre os diferentes componentes do sistema (e.g., dispositivo móvel, *proxy*). Além de transferir um perfil inteiro, a implementação deve possibilitar se transferir uma sub-árvore do perfil ou um único atributo. Isto significa que um perfil não precisa ser completamente re-transferido caso se altere um único atributo (por exemplo, uma mudança na latência da rede).

- **Composta/Decomposta**: a composição/decomposição de perfis permite que esses sejam armazenados e mantidos de um modo distribuído. Por exemplo, um perfil de dispositivo padrão pode ser armazenado no intermediário ou no servidor *Web*, sendo que as divergências de um dispositivo particular do padrão são armazenadas no próprio dispositivo. Desse modo apenas as divergências devem ser transferidas.
- **Uniforme**: uma representação uniforme de todas as variantes de perfis de contexto (perfis de rede e dispositivo, perfis do usuário, e dados de contexto adicionais) facilita a interpretação durante o processo de mediação de serviço e adaptação de conteúdo.
- **Extensível**: nenhum conjunto de atributos atualmente identificados será suficiente para todas as aplicações futuras. Portanto, um formato de representação de perfil deve suportar futuras extensões.
- **Padronizada**: perfis de contexto são trocados entre diferentes entidades do sistema. Essas entidades geralmente não pertencem ao mesmo domínio administrativo. Portanto, existe uma grande necessidade de uma representação padronizada da informação de contexto.

2.5.1 *Composite Capabilities/Preferences Profile (CC/PP)*

O *Composite Capability / Preferences Profile* (*CC/PP*) [32,33], uma recomendação do *W3C*, é um método para comunicar as capacidades de dispositivos como clientes, *proxies*, *gateways* e *caches*, como também recursos como documentos. O *CC/PP* é baseado no *Resource Description Framework* (*RDF*), outro padrão criado pelo *W3C*.

No *RDF* todas as entidades são descritas como recursos e consistem em um nome de recurso, uma propriedade de recurso e um atributo. Estes recursos são organizados em componentes de modo a formar um *schema*. O CC/PP não define um vocabulário para especificar capacidades de dispositivos, mas uma linguagem genérica para construir tais vocabulários.

Atualmente *CC/PP* não endereça dois problemas fundamentais relativos à independência de dispositivo: ele não provê um vocabulário padrão para descrição do contexto de entrega. Também não descreve os tipos de transformações e customizações que podem ser executadas pelos servidores de adaptação em nome dos dispositivos, baseados nas capacidades desses. Em 2004 o grupo de trabalho do *CC/PP* foi extinto e a continuidade dos trabalhos ficou a cargo do grupo *Device Independence* (*DI*) que está trabalhando para eliminar essas deficiências.

2.5.2 *User Agent Profile (UAProf)*

A *Open Mobile Alliance (OMA)* definiu o *UAProf* [9], atualmente na versão 2.0, como uma implementação do *CC/PP* para terminais moveis que utilizam tecnologia *WAP*. Essa última versão corrigiu alguns problemas de compatibilidade com o *CC/PP*. O *WAP* definiu o protocolo *Wireless Session Protocol (WSP)* para suportar a comunicação entre o dispositivo móvel e o *gateway WAP*, uma de suas funções é transportar os perfis *UAProf*. O *WAP* 2.0 define uma nova extensão do *HTTP 1.1*, chamada *Wireless Profiled HTTP (W-HTTP)* que utiliza cabeçalhos customizados para transportar os perfis, sendo utilizado quando a conexão entre o dispositivo móvel e o servidor de origem é direta. O *UAProf*, ao contrário do *CC/PP*, define um vocabulário

de perfis próprio, que inclui propriedades sobre características de *hardware* e *software*, recursos específicos do *WAP* e capacidades da rede de comunicação.

Atualmente existem duas implementações que suportam *CC/PP* e *UAProf* [34]. A primeira, *Delivery Context LIbrary* (*DELI*), provê uma *Application Program Interface (API)*, desenvolvida pela *HP Labs*, que permite a manipulação de informações de perfis especificados em *CC/PP* ou *UAProf*. Essa *API* também provê suporte a dispositivos legados de forma que descrições de contexto de entrega proprietárias, usadas por algumas aplicações, possam ser substituídas por descrições *CC/PP*.

A segunda é o CC/PP *Processing Specification* (*Java Specification Request* (*JSR*)188) que define um conjunto de *API´s* para processar informações sobre contexto de entrega baseadas em *CC/PP*, incluindo o *UAPROF*. A *SUN* definiu essas *API´s* como uma extensão da plataforma *Java 2 Platform Enterprise Edition* (*J2EE*).

2.5.3 Outras Propostas

Apesar do *CC/PP* e o *UAPROF* serem os padrões para descrição de características de contexto de entrega, eles ainda apresentam algumas deficiências. A especificação *CC/PP* ainda não definiu nenhum vocabulário padronizado, o que pode provocar incompatibilidades entre as soluções de diferentes fabricantes de dispositivos. O *UAPROF* que já está na versão 2.0, ainda não é totalmente compatível com o *CC/PP* e o mercado dispõe de dispositivos que trabalham apenas com a versão 1.0. Além disso, alguns dispositivos móveis não suportam o *UAPROF*, especificando parte de suas características no cabeçalho *HTTP*. Por causa desses problemas algumas soluções alternativas foram disponibilizadas no mercado com o objetivo de eliminar essas falhas.

O *Universal Profiling Schema (UPS)* [35] provê negociação de conteúdo e adaptação de serviços multimídia em ambientes heterogêneos. *UPS* não depende de um tipo particular de dispositivo, protocolos ou formato de conteúdo. Ele define três categorias principais de componentes: perfil de rede, perfil de cliente e perfil de servidor. O diferencial do *UPS* é que ele não considera somente as características do cliente, mas também as características do conteúdo, métodos de transformação e adaptação, e capacidades do servidor.

O *Wireless Universal Resource File (WURFL)* [10] é um projeto de código aberto, mantido por uma comunidade de programadores, que provê uma fonte alternativa de informação sobre dispositivos móveis, contando atualmente com uma base de dados de 4500 dispositivos. Por se tratar de um produto de código aberto qualquer um pode inserir e/ou corrigir informações sobre dispositivos. *WURFL* possui um formato proprietário, formatado em *XML*, para descrever características de dispositivos.

No decorrer dos últimos tópicos é perceptível que existem várias formas de se representar às informações sobre o contexto de entrega. Apesar do domínio do *CC/PP* e do *UAPROF* não existe nenhum formato que tenha suporte para todos os dispositivos existentes. Além disso, existem incompatibilidades entre versões e não foi definido um repositório confiável que suporte a demanda necessária.

O uso de ontologias, além de outras vantagens que serão apresentadas nos próximos capítulos, provê uma facilidade no mapeamento de informações entre os diversos formatos e uma descrição, não ambígua, das características do contexto.

Um exemplo de utilização de ontologias é o projeto *Semantic API for the Delivery Context* (*SADiC*) [36]. *SADiC* é uma *API* utilizada para processar e interrogar

perfis *CC/PP* e *UAProf* que explora as idéias da *Web* Semântica. A aproximação semântica de *SADiC* se apóia em ontologias para alcançar a convergência semântica entre o *CC/PP* e o *UAProf*, utilizando a *OWL* para descrever ontologias.

2.5.4 Regras de Adaptação

A definição da política de adaptação é obtida através das regras de adaptação, que são compostas por um conjunto de condições e ações relacionadas. Essas condições são referentes aos perfis de adaptação e precisam refletir os desejos das seguintes entidades envolvidas no processo: usuário (perfil de usuário e dispositivo), provedor de acesso (perfil de rede de acesso) e provedor de conteúdo (perfil de conteúdo). A combinação dessas condições determina a ação a ser executada e quais servidores de adaptação serão utilizados.

O uso de servidores de adaptação externos, que podem ser desenvolvidos por diferentes administradores, gerou a necessidade de padronizar o formato das regras de adaptação. Um exemplo de especificação é a *Intermediary Rule Markup Language* (*IRML*), uma linguagem baseada na *XML*, que foi criada pelo grupo *OPES*.

Uma das características dessa linguagem é possibilitar que o processamento da regra seja realizado em determinados estágios do fluxo de dados, utilizando-se o atributo *processing-point*, conforme ilustra a Figura 7. Foram especificados quatro pontos de execução sendo que nos pontos 1 e 2 a regra é processada sobre a requisição do usuário antes ou após a busca do conteúdo no *cache*, respectivamente. Para execução durante o estágio de resposta são utilizados os pontos 3 e 4, ocorrendo antes ou após o armazenamento do conteúdo no *cache*, respectivamente.

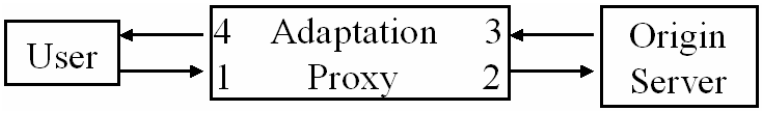


Figura 7. Pontos de processamento das regras de adaptação

A *IRML* também provê suporte às informações fornecidas pelos perfis através de atributos. No atributo *property* são descritas as condições que definem a execução, ou não, de determinada adaptação. As ações de cada regra são descritas no atributo *service*, que é composto por *uri*, que indica o endereço do servidor de adaptação a ser solicitado, e por *parameter*, que descreve os parâmetros a serem enviados ao servidor de adaptação. Esses parâmetros podem ser estáticos (constantes pré-definidas) ou dinâmicos, quando informações obtidas dos perfis são enviadas aos servidores de adaptação (e.g., tamanho de tela).

A Figura 8 mostra um exemplo de regra de adaptação especificada através da *IRML*. No caso, a regra de adaptação é aplicada sobre documentos *HTML* no *processing-point* 4 e contém três condições: *Content-type = html* (a), *Network-traffic = full* (b) e *Image-user-preference = removal* (c), que são referentes aos perfis de conteúdo, de rede de acesso e de usuário, respectivamente. Se essas condições são satisfeitas a ação do parâmetro *removal-images* (e) é solicitada ao servidor de adaptação de endereço *icap://www.htmladapt.com* (d).

```
<rulemodule xmlns="http://...">
 <author type="delegate">... </author>
 <ruleset>
  <authorized-by class="content-consumer" type="group"> ... </authorized-by>
  <protocol>HTTP</protocol>
  <rule processing-point="4">
   <property name="Content-type" context="res-hdr" matches="html/">          (a)
    <property name="Network-traffic" context="system" matches="full">        (b)
     <property name="Image-user-preference" context="system" matches="removal">(c)
      <execute>
       <service name="HtmlRemoval">
        <uri>icap://www.htmladapt.com</uri>                                  (d)
        <parameter name="action" type="static">
         <value>removal-images</value>                                       (e)
        </parameter>
       </service> ...
   </rule> ...
</rulemodule>
```

Figura 8. Exemplo de especificação de regras de adaptação utilizando *IRML*.

O grupo de trabalho *OPES* está desenvolvendo uma nova linguagem para especificação de regras denominada P [37], uma linguagem de programação do tipo funcional semelhante ao C++, que tem como principal vantagem em relação ao *IRML*, de suportar uma grande variedade de serviços e protocolos (e.g., *SMTP, HTTP, Real Time sTreaming Protocol* (*RSTP*)). Porém, essa linguagem ainda não foi completamente definida.

Apesar de não serem desenvolvidas especialmente para o domínio de adaptação de conteúdo. Outras linguagens para especificação de regras estão sendo adotadas pelo mercado. Entre elas pode-se destacar a *Rule Markup Language* (*RuleML*) [38] e a *Semantic Web Rule Language* (*SWRL*) [39], ambas baseadas em *XML* e em soluções de *Web* semântica. Essas linguagens são apresentadas com mais detalhes no tópico de *Web* Semântica desta dissertação.

2.6 Conclusão

O enorme crescimento de dispositivos móveis, com destaque para os telefones celulares, está gerando um grande desafio para os provedores de conteúdo. Como disponibilizar um único conteúdo para um contexto de entrega com características tão diversas.

Uma das respostas para esse problema é a adaptação de conteúdo. Para que essa adaptação seja realizada de modo satisfatório, a política de adaptação deve levar em conta as características do contexto de entrega, que inclui as preferências do usuário, e os serviços de adaptação disponíveis.

Para prover uma solução flexível esta dissertação especificou vários perfis para a descrição do contexto de entrega, baseados em ontologias. Essa escolha proporciona uma descrição não ambígua e de fácil extensão e reuso.

Também se definiu que a adaptação deve ocorrer em sistemas intermediários (e.g., *proxies*), proporcionando uma independência do hardware e software do dispositivo de acesso. Nesse caso, o usuário não fica restrito a utilizar apenas servidores *Web* que possuam adaptadores de conteúdo. Para não sobrecarregar o *proxy* a adaptação pode ser feita internamente ou externamente, por meio de protocolos de chamada externa. Por ser um protocolo "leve" e já possuir alguns servidores de adaptação desenvolvidos foi adotado o protocolo *ICAP*. Como é proposta a integração de serviços *Web* também foi utilizado o protocolo *SOAP* neste trabalho.

O uso de ontologias também facilita o reuso de perfis definidos por outras especificações, sendo que as características dos perfis *UAPROF* e *WURFL* serviram de referência para a criação dos perfis deste trabalho. A escolha da linguagem de regras também foi influenciada pelo uso de semântica na descrição dos perfis, sendo que por esse motivo, entre outros que serão descritos nos próximos capítulos, foi escolhida a *SWRL*.

3 Ontologias e *Web* Semântica

Para alcançar a interoperabilidade entre sistemas heterogêneos, envolvidos na execução de um determinado domínio de aplicações, é fundamental que se compartilhem informações de maneira automática, com um entendimento comum e não ambíguo dos termos e conceitos usados nessas aplicações. Nesse sentido as ontologias constituem-se em artefatos importantes na viabilização do tratamento dessa heterogeneidade.

3.1 Ontologias

Nesta dissertação o termo ontologia refere-se a uma descrição explícita de conceitos e relações de um domínio de aplicação, incluindo um vocabulário de termos empregado nesse domínio e um conjunto de axiomas que expressam as restrições para a interpretação desse vocabulário. Ontologias podem ser usadas para diversos fins, variando do apoio aos processos de desenvolvimento de software, ou qualquer outra atividade executada por equipes geograficamente distribuídas, ao apoio em tempo de execução em sistemas de informação [40].

É importante realçar que, de posse dessa base de conhecimento formalizada como uma teoria lógica, a ontologia não descreve apenas conhecimento imediato, isto é, conhecimento factual que pode ser obtido diretamente a partir da observação do domínio, mas também conhecimento derivado, ou seja, conhecimento obtido através de inferência sobre o conhecimento imediato disponível.

Com base em seu conteúdo as ontologias podem ser classificadas nas categorias, ilustradas na Figura 9, descritas a seguir [40,41]:

- **Ontologias genéricas:** expressam teorias básicas do mundo, de caráter bastante abstrato, aplicáveis a qualquer domínio, conhecimento de senso comum. Tipicamente, ontologias genéricas definem conceitos tais como coisa, estado, evento, processo, ação, etc., com o intuito de serem especializados na definição de conceitos em uma ontologia de domínio.
- **Ontologias de domínio:** expressam conceituações de domínios particulares, descrevendo o vocabulário relacionado a um domínio genérico, tal como medicina e direito.
- **Ontologias de tarefas:** expressam conceituações sobre a resolução de problemas, independentemente do domínio em que ocorram, isto é, descrevem o vocabulário relacionado a uma atividade ou tarefa genérica, tal como vendas.
- **Ontologias de aplicação:** expressam conceitos dependentes do domínio e da tarefa particulares. Esses conceitos freqüentemente correspondem a papéis desempenhados por entidades do domínio quando da realização de uma determinada atividade.
- **Ontologias de representação:** expressam os compromissos ontológicos embutidos em formalismos de representação de conhecimento [42].

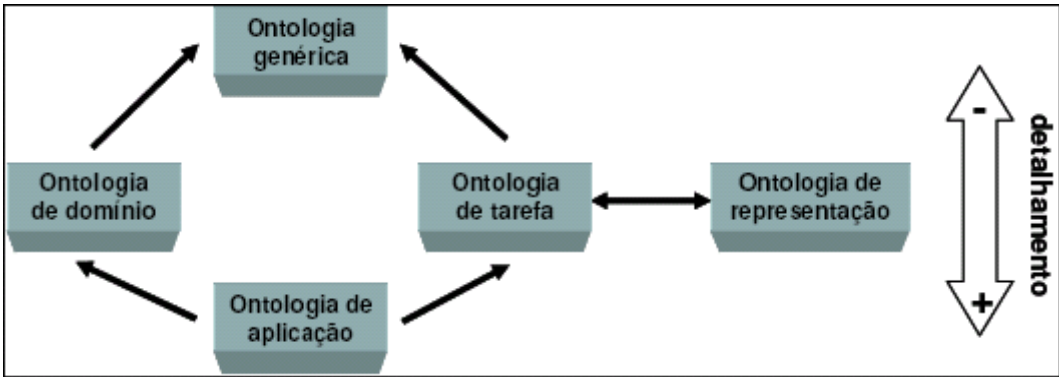


Figura 9. Classificação de ontologias com base em seu conteúdo [45].

A profundidade ontológica pode ser classificada em quatro níveis [43]:

- **Vocabulário:** em sua forma mais simples, uma ontologia é apenas um vocabulário. Nesse sentido, uma *Document Type Definition (DTD)* ou um *XML-Schema* pode definir uma ontologia.
- **Taxonomia:** o significado dos termos é estabelecido pela definição de relacionamentos entre objetos e classes, subclasses e superclasses. Esse tipo de ontologia é muito semelhante à estabelecida por sistemas orientados a objetos. Muitas ontologias existentes são definidas usando-se apenas esses relacionamentos hierárquicos.
- **Sistema relacional:** as ontologias também podem incluir relacionamentos não hierárquicos como nos diagramas de relacionamento de entidades e nos bancos de dados relacionais e, por conseguinte, cada esquema de banco de dados relacional define sua própria ontologia.
- **Teoria axiomática:** além de descrever relacionamentos, as ontologias também podem impor restrições. As restrições são definidas como axiomas. Um axioma é uma afirmação lógica que não pode ser provada a partir de outras afirmações, mas da qual, outras afirmações podem ser derivadas.

Para a construção de ontologias é essencial dispor-se de uma linguagem apropriada. Essa linguagem deve possuir uma semântica bem-definida, ser expressiva o suficiente para descrever inter-relacionamentos complexos e restrições entre os objetos, ser capaz de manipular e inferir automaticamente. Tudo isso dentro de limites aceitáveis de tempo e recursos [44].

3.2 *Web* Semântica

Berners-Lee propôs a *Web*-Semântica [45,46] como uma evolução natural da *Web* vigente, a fim de viabilizar a manipulação de conteúdo por parte de

aplicações com capacidade de interpretar a semântica das informações. O conteúdo da *Web*, que não passa de informação sem significado para os computadores, pode então ser interpretado por máquinas através do uso de ontologias, tornando a recuperação da informação na *Web* menos ambígua e fornecendo respostas mais precisas às demandas dos usuários.

A *Web* Semântica não abrange somente a *Web.* Ela representa um conjunto de tecnologias necessárias também às atividades desenvolvidas em redes coorporativas. Nesse sentido, a *Web* Semântica tem por finalidade solucionar vários problemas das atuais arquiteturas de tecnologia da informação como [47]:

- **Sobrecarga de informação:** na *Web* há uma quantidade imensa de informações não pertinentes que estão disponíveis e são fornecidas pelos processos de busca. As ferramentas de busca enfrentam a dificuldade de executar pesquisas entre documentos que não estão diferenciados em termos de assunto, qualidade e relevância. A tecnologia atual não é capaz de diferenciar uma informação comercial de uma educacional, ou informação entre idiomas, culturas e mídia. É necessário haver informações de qualificação da própria informação para ser possível classificá-las e tornar os processos de recuperação de informações mais eficazes.
- **Integração de informações:** a integração de informações na *Web* é um assunto muito discutido pelos estudiosos da área. A variedade de fontes de informação distintas com diferenças sintáticas, semânticas e estruturais entre elas é muito grande, tornando o compartilhamento, a integração e a resolução de conflitos entre essas informações um problema de difícil solução. A heterogeneidade estrutural e semântica da informação na *Web* atualmente é imensa, e a maioria das propostas de integração ainda adota soluções com alto índice de

centralização, tornando o seu uso na *Web* inviável. Para tratar esses problemas é necessário considerar questões relevantes como à utilização de meta dados e ontologias, visando à busca de uma linguagem única, capaz de estruturar e representar conhecimento e regras.

- **Conteúdo não estruturado:** um dos motivos do grande sucesso da *Web* é sua liberdade de publicação de informação. Devido a isso, existe uma enorme quantidade de todos os tipos de documentos e recursos disseminados na *Web*, tais como bancos de dados, artigos, programas, arquivos, etc. Por serem criados de forma autônoma, sem preocupação com regras de estruturação, catalogação e descrições de suas propriedades, essas informações são difíceis de serem abrangidas pelos mecanismos de pesquisa, ocasionando demora e ineficácia na sua localização. A efetividade dos mecanismos de busca depende principalmente da maneira pela qual as informações foram estruturadas e catalogadas na *Web*.

Na proposta de desenvolvimento da *Web* Semântica [48] é sugerida uma arquitetura de três camadas, conforme apresentada na Figura 10:

- **Esquema:** que estrutura os dados.
- **Ontologia:** que define as relações entre os dados;
- **Lógica:** que define mecanismos para execução de inferências sobre os dados.

A tecnologia da *Web* Semântica é uma proposta de extensão da *Web* baseada no uso de ontologias para descrever relacionamentos entre objetos, formados com informações semânticas, para automatizar o processamento pelas máquinas.

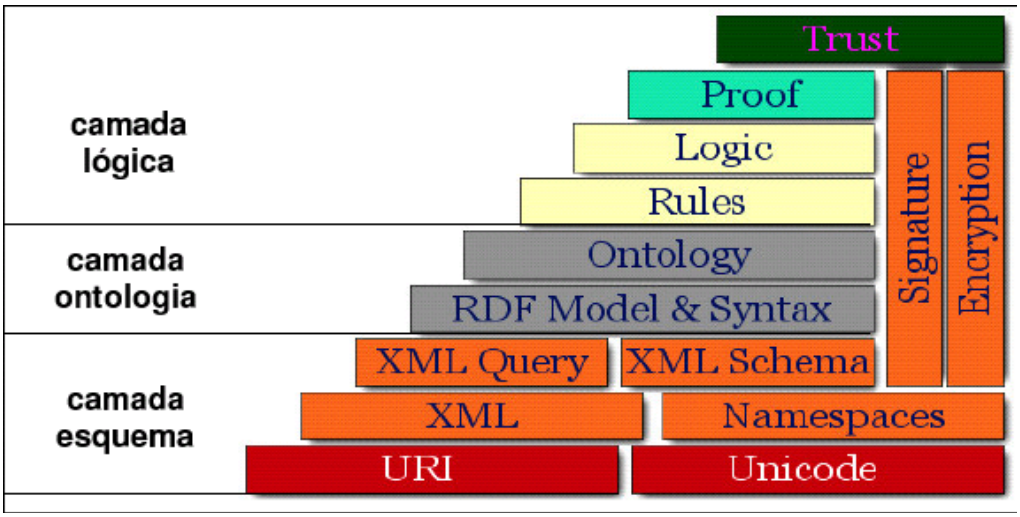


Figura 10. Arquitetura da *Web* Semântica [45].

O *W3C* orienta o desenvolvimento, a organização e a padronização de linguagens para promover a interoperabilidade entre aplicações na *Web*. Dentre essas linguagens destacam-se: *Resource Description Framework (RDF)* [49] e mais recentemente a *Web Ontology Language (OWL)* [50] que, entre outras características a serem apresentadas, são elementos importantes da *Web* Semântica.

Para prover semântica a um recurso, um conhecimento adicional precisa ser apresentado junto ao recurso. Isso cria a necessidade de se utilizar um formalismo de representação de conhecimento. No contexto da *Web* Semântica, o formalismo utilizado é conhecido como Lógica de Descrição (*DL*) e uma das linguagens que integra esse formalismo é a *OWL-DL* [51].

3.2.1 Lógica de Descrição

Pesquisas no campo de representação de conhecimento e raciocínio (*reasoning*) conduziram ao desenvolvimento de sistemas inteligentes que têm a habilidade de encontrar conseqüências implícitas do conhecimento explicitamente representado nesses sistemas. Esses sistemas são caracterizados como sistemas

baseados em conhecimento (*knowledgebased*). Uma vertente desses sistemas é atualmente chamada de Lógica de Descrição.

Lógica de descrição, que daqui em diante serão chamadas apenas pela sigla *DL*, é uma família de linguagens de representação de conhecimento que pode ser usada para representar o conhecimento terminológico de um domínio de aplicação, de uma maneira estruturada e formalmente bem-compreendida [52]. Lógica de descrição pode ser caracterizada pelas seguintes propriedades:

- Os blocos básicos de construção sintática são conceitos atômicos, papéis atômicos e indivíduos.
- O poder expressivo do idioma se restringe ao uso de um pequeno conjunto de construtores, mas permite criar conceitos e funções complexas a partir dos conceitos e funções existentes.
- O conhecimento implícito sobre conceitos e indivíduos pode ser encontrado automaticamente com a utilização de técnicas de raciocínio.

3.2.2 Redes Semânticas e DL

DL são à base das redes semânticas [53]. Como sugere o nome, esses sistemas são organizados em estruturas de rede, e representam por meio de estruturas cognitivas o conhecimento de um sistema. Em geral, os elementos de uma rede semântica são nodos e ligações. Tipicamente são usados nodos para representar conceitos ou classes, e são usadas ligações para representar propriedades Uma classe define um grupo de indivíduos que compartilham algumas propriedades comuns. Indivíduos são instâncias de classes e propriedades são usadas para se relacionar um individuo a outro.

Considere o exemplo de uma pequena rede exibida na Figura 11. Essa rede representa o conhecimento sobre pessoas, pais, etc. A ligação entre os conceitos, Mulher e Pessoa declara que uma mulher é uma pessoa, tal relação é freqüentemente definida como uma relação "*IS-A*". Essa relação define uma hierarquia em cima dos conceitos, no exemplo, Pessoa é um conceito mais genérico que Mulher. O conceito mais genérico é denominado super-conceito, considerando que o conceito mais específico é chamado de sub-conceito. A relação "*IS-A*" também provê a base para a herança de propriedades; quando um conceito for mais específico que outro conceito, herda as propriedades do mais genérico. Por exemplo, se o conceito Pessoa possui a propriedade idade então o conceito Mulher também a terá. É importante mencionar que um conceito pode ter múltiplos super-conceitos, em relação ao exemplo apresentado o conceito Mulher possui dois super-conceitos, Pessoa e Fêmea. Na Figura 11 a propriedade temFilho pode ser usada para relacionar o individuo Y como uma instância da classe Pessoa.

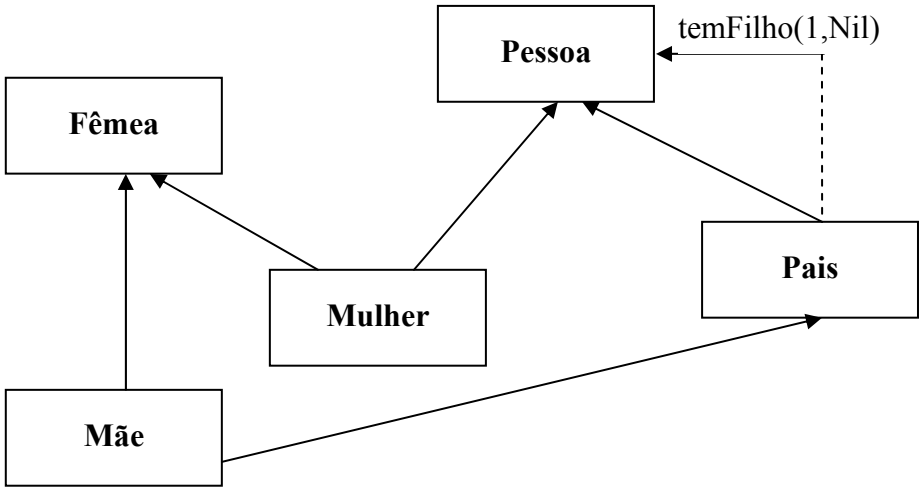


Figura 11. Um pequeno exemplo de rede semântica [53].

É possível criar novos conceitos a partir de construtores, como união, interseção, etc. O uso do construtor de interseção, entre outros, é apresentado na Figura

11, onde o conceito Mulher é definido em termos dos conceitos atômicos Pessoa e Fêmea. A definição de um conceito deve respeitar duas restrições: o nome do conceito só pode ser usado uma vez, e sua definição tem que ser acíclica.

A habilidade para representar outras relações entre conceitos além da relação "*IS-A*" é uma característica que diferencia a DL em relação a outros sistemas semânticos. No exemplo apresentado, a ligação pontilhada de Pai para Pessoa é uma relação denominada papel. Papéis são relações binárias entre dois conceitos que podem ter uma restrição de valor, limitando os tipos de objetos que podem assumir aquele papel. Também pode existir uma restrição de número onde o primeiro número denota um valor mínimo e o segundo número um valor máximo. No exemplo da Figura 11 é expresso que um pai tem pelo menos uma criança. Também são herdadas relações de papel de conceitos mais genéricos. O conceito Mãe herda o papel temFilho do conceito Pais. A estrutura apresentada de conceitos e suas relações também são denominadas como a terminologia do domínio do problema, e representa a generalidade e especificidade dos conceitos envolvidos.

Note que há relações implícitas entre conceitos. No exemplo apresentado, há uma "relação escondida" entre os conceitos Mãe e Mulher. Por se definir o conceito Mãe como uma Pessoa que é uma Fêmea e tem pelo menos uma criança, se define a relação implícita que todas as Mães são Mulheres. Desde que a Mãe é um sub-conceito de Fêmea e Pais, e Pais é um sub-conceito de Pessoa, o conceito Mulher é um super-conceito de Mãe, como o conceito Mulher é um sub-conceito de Pessoa e Fêmea. A habilidade de encontrar essas relações implícitas é característica de sistemas baseados em conhecimento. Essas tarefas de raciocínio, também chamadas de mecanismos de inferência, serão discutidas nos próximos tópicos.

Dentro de uma base de conhecimento há uma distinção clara entre conhecimento intencional que é o conhecimento mais geral sobre o domínio do problema e o conhecimento extensional, o conhecimento mais específico para um problema particular. Em *DL*, esses dois conceitos são incluídos tipicamente através de dois componentes - *TBox* e *ABox*. O *TBox* contém o conhecimento intencional na forma de terminologia e é construído por declarações que definem conceitos gerais e suas propriedades. A representação dos conceitos e relações da rede exemplo abrangeria o *TBox*, pois eles representam o conhecimento geral sobre o domínio família. O *ABox* contém o conhecimento extensional sobre o domínio de interesse. E são introduzidos os indivíduos, dando lhes nomes, e se afirma propriedades desses indivíduos. Um indivíduo definido no *ABox* é um exemplo de um conceito da terminologia.

3.2.3 Sintaxe da Lógica de Descrição

Os elementos básicos de construção da DL são conceitos atômicos e papéis atômicos. A seguir serão denotados conceitos atômicos com as letras A e B, e as letras R e S para papéis atômicos. Podem ser definidos conceitos e papéis atômicos mais complexos por meio de construtores, denotados pelas letras C e D. Como DL define uma família de lógicas, diferentes linguagens de descrição provêem diferentes construtores. Uma DL básica é a linguagem $\mathcal{AL}$ (linguagem atributiva). A linguagem $\mathcal{AL}$ possui os seguintes construtores:

C, D → A | (conceito atômico)
⊤ | (conceito universal)
⊥ | (conceito inferior)
¬ A | (negação atômica)
C ⊓ D | (intersecção)
∀ R.C | (restrição de valor)
∃R.⊤ | (quantificação existencial limitada)

Em $\mathcal{AL}$ a negação só pode ser aplicada a conceitos atômicos, e só o conceito universal é permitido na definição de um construtor de quantificação existencial limitada. A linguagem de DL $\mathcal{ALEN}$ é uma extensão da $\mathcal{AL}$ que inclui negação em conceitos arbitrários (complexos), restrições de cardinalidade, papéis inversos, papéis transitivos e tipos de dados. Essa extensão adiciona os seguintes construtores:

$C, D \rightarrow \exists R.C$ | (quantificador existencial completo)
$\neg C$ | (negação em conceitos arbitrários)
$\leq nR$ | (restrição de cardinalidade – no máximo)
$\geq nR$ | (restrição de cardinalidade – no mínimo)
$= nR$ | (restrição de cardinalidade – igual a)
$\leq nR.C$ | (restrições qualificadas de cardinalidade – no máximo)
$\geq nR.C$ | (restrições qualificadas de cardinalidade – no mínimo)
$= nR.C$ | (restrições qualificadas de cardinalidade – igual a)
$\leq_n R$ | (restrições de domínio concreto máximo)
$\geq_n R$ | (restrições de domínio concreto mínimo)
$=_n R$ | (restrições de domínio concreto igual)

3.2.4 Semântica de DL - Conceitos e Papéis

No tópico anterior foram apresentadas informações relativas à sintaxe de algumas famílias de linguagens de Lógica de Descrição, neste tópico será apresentada uma definição formal da semântica dessas linguagens. A interpretação *I* consiste em um conjunto não-vazio Δ^I (o domínio da interpretação) e uma função da interpretação que mapeia cada conceito atômico *A* sobre um conjunto $A^I \subseteq \Delta^I$ e todo papel atômico R sobre uma relação binária $R^I \subseteq \Delta^I \times \Delta^I$. Para a linguagem $\mathcal{AL}$ a função da interpretação estende as descrições de conceito na seguinte forma:

$$\top^{I} = \Delta^{I}$$
$$\square^{I} = \Phi$$
$$(\neg A)^{I} = \Delta^{I} \setminus A^{I}$$
$$(C \sqcap D)^{I} = C^{I} \cap D^{I}$$
$$(\square R.C)^{I} = \{a \square \Delta^{I} | \square b.(a, b) \square R^{I} \rightarrow b \square C^{I}\}$$
$$(\square R.\top)^{I} = \{a \square \Delta^{I} | \square b.(a, b) \square R^{I}\}$$

Axiomas Terminológicos e Definições. Axiomas Terminológicos fazem declarações sobre como são inter-relacionados conceitos e papéis. Axiomas terminológicos têm a seguinte forma:

$C \square D$ ($R \square S$) para inclusões e
$C \equiv D$ ($R \equiv S$) para igualdades.

A semântica em axiomas está definida do seguinte modo; uma interpretação I satisfaz uma inclusão $C \square D$ se $C^{I} \square D^{I}$, e satisfaz uma igualdade $C \equiv D$ *se* $C^{I} = D^{I}$. Considerando que T denota um conjunto de axiomas; então I satisfaz T *se, e somente se, I* satisfaz cada elemento de T. Se I satisfaz um axioma pode-se dizer que ele é o *modelo* desse axioma. Uma definição é um conceito atômico do lado esquerdo de uma igualdade. São usadas definições para introduzir nomes simbólicos para descrições complexas. Por exemplo, ao axioma

$$\text{Mãe} \equiv \text{Mulher} \sqcap \square \text{temFilho.Pessoa}$$

foi associado o nome Mãe no lado esquerdo da igualdade.

Um conjunto finito de definições T é chamado de *terminologia* ou *TBox*. Em uma terminologia nenhum nome simbólico é definido mais de uma vez. Um exemplo de *TBox* com definições de conceito e papel sobre relações de família pode ser visto na Figura 12.

Mulher ≡ Pessoa ⊓ Fêmea
Homem ≡ Pessoa ⊓ ¬Mulher
Mãe ≡ Mulher ⊓ ∃temFilho.Pessoa
Pai ≡ Homem ⊓ ∃temFilho.Pessoa
Pais ≡ Pai ⊔ Mãe
Avó ≡ Mãe ⊓ ∃temFilho.Pais
MãeComMuitosFilhos ≡ Mãe ⊓ ≥ 3 temFilho
MãeSemFilha ≡ Mãe ⊓ ∃temFilho ¬Mulher

Figura 12. *TBox* com conceitos sobre o domínio da família.

Afirmações (*Assertions*) Considerando que *a*, *b e c* denotam o nome de indivíduos. Há dois tipos diferentes de afirmações para indivíduos:

$C(a)$ Afirmação de conceito.

$R(b, c)$ Afirmação de papel.

A primeira afirmação declara que o indivíduo a pertence à interpretação *C*, na segunda afirmação c é um complemento do papel *R* para *b*. Um conjunto finito *A* de afirmações é denominado *ABox*. A semântica para *ABoxes* é definida se estendendo as interpretações de modo a incluir nomes de indivíduos. Uma interpretação *I* agora mapeia cada nome individual para um elemento $a^I \in \Delta^I$. A interpretação *I satisfaz a afirmação de conceito* $C(a)$ se $a^I \in C^I$, e satisfaz a afirmação de $R(b, c)$ if $(b^I, c^I) \in R^I$. Uma interpretação *I satisfaz* uma *ABox A se ela satisfazer cada afirmação em A*. Nesse caso *I é um modelo para A*. Um exemplo de *ABox* com algumas afirmações correspondentes a *TBox* da Figura 12 é apresentada na Figura 13.

Pai(PAULO)
temFilho(ANA,PAULO)

Figura 13. *ABox* com afimações para o domínio da família

Uma interpretação *I* satisfaz uma *ABox A* com respeito a um *TBox T* se além de ser um modelo para *A*, também seja um modelo para *T*.

3.2.5 Raciocinadores de Lógicas de Descrição

Raciocinadores (*Reasoners*) de Lógica de Descrição são ferramentas de software que, baseadas no conhecimento provido, tentam inferir conclusões adicionais transformando o conhecimento implícito em explícito. O conhecimento apresentado a um raciocinador está na forma de uma base de conhecimento. Uma base de conhecimento é uma coleção interpretável por computadores de conceitos, relações entre conceitos, fatos, regras, heurística, modelos e procedimentos, que podem ser usados na resolução de um problema. O conjunto de toda informação e conhecimento em uma base de conhecimento pertence a um campo específico de interesse. Uma base de conhecimento manualmente construída na qual não são providas todas as relações entre os vários conceitos é chamada de hierarquia de classe declarada (*asserted class hierarchy*). Para determinar relações entre conceitos, é necessário que sejam mencionadas todas as relações entre os conceitos explicitamente. Para inferir essas relações, a base de conhecimento precisa ser classificada, que é a tarefa de computar a hierarquia de classe inferida [54]. Essa representação explícita da base de conhecimento em que todas as relações entre todos os conceitos estão presentes na base de conhecimento é chamada hierarquia de classe inferida. Além de inferir informação a partir do conhecimento provido, os raciocinadores DL também têm capacidade para responder a vários tipos de consultas, baseada no conhecimento explícito provido e nas inferências que computou. Foram desenvolvidos vários raciocinadores de *DL* na comunidade acadêmica (e.g., FaCT++ [55], RACER [56], Pellet [57]).

Na *DL* os raciocinadores formam uma parte integrante de qualquer sistema que realize algum aspecto do paradigma da *Web* Semântica. A *Web* Semântica tem o potencial para resolver problemas de interoperabilidade e de representação de conhecimento, devido ao fato de que as aplicações baseadas nesse paradigma

compartilham uma semântica comum na forma de vocabulários comuns, evitando a ambigüidade sintática. Como as entidades da *Web* Semântica especificaram o uso de algumas linguagens semânticas, como a *OWL* que utiliza a semântica de DL, os raciocinadores de DL podem ser utilizados para extrair informações entre tais entidades.

Como especificado em [58], o suporte a família de *DL* $\mathcal{SHOQ}$(D) é o mínimo necessário para que um raciocinador seja utilizado em serviços de *matchmaking* semântico. Outras exigências são descritas a seguir [58]:

a) Dinâmico - o raciocinador deve ser dinâmico permitindo que ontologias possam ser adicionadas, removidas e modificadas, resultando em uma reclassificação da base de conhecimento.
b) Habilidade para trabalhar com múltiplos *TBoxes* interconectados – pois diferentes ontologias podem ser usadas e conceitos podem estar baseados em conceitos e papeis externos.
c) Escalabilidade - o raciocinador deve ser capaz de trabalhar com grandes volumes de informação de uma maneira eficiente.
d) Suporte a sintaxe da *OWL* - que evitaria traduções desnecessárias.

3.2.6 Web Ontology Language (*OWL*)

A *OWL* é utilizada quando as informações contidas em recursos *Web* precisam ser processadas através de aplicações, ao contrário de situações onde o conteúdo só precisa ser apresentado a humanos. *OWL* pode ser usada para representar explicitamente o significado de termos nos vocabulários e as relações entre esses termos. *OWL* possui uma estrutura para expressar significado e semântica melhor do

que o *RDF*, e o *RDF-S* [59]. A *OWL* é uma revisão da *DAML+OIL* [60] incorporando lições aprendidas durante o projeto e implementação desse ultimo.

A *OWL* provê três sub-linguagens de expressividade crescente, projetadas para uso por comunidades específicas de desenvolvedores e usuários:

- ***OWL Lite*** é indicada para usuários que precisam de uma hierarquia de classificação e restrições simples. Por exemplo, existe suporte a restrições de cardinalidade, mas só permite valores de cardinalidade 0 ou 1. *OWL Lite* tem uma complexidade formal menor em relação as outras sub-linguagens da *OWL*.
- ***OWL DL*** é indicada para usuários que desejam o máximo da expressividade enquanto mantém a integralidade computacional (garante que todas as conclusões são computáveis) e decidibilidade (todas as computações terminarão em tempo finito). *OWL DL* inclui todos os construtores da *OWL*, mas eles só podem ser usados sob certas restrições (por exemplo, enquanto uma classe pode ser uma subclasse de várias classes, uma classe não pode ser uma instância de outra classe). *OWL DL* é nomeado assim devido a sua correspondência com Lógica de Descrição que formam a base formal da *OWL*.
- ***OWL Full*** é indicada para usuários que desejam a máxima expressividade e a liberdade sintática do *RDF* sem garantias computacionais. Por exemplo, na *OWL Full* uma classe pode ser tratada simultaneamente como uma coleção de indivíduos. *OWL Full* permite que uma ontologia amplie o significado predefinido de um vocabulário (*RDF* ou *OWL*).

Cada uma dessas sub-linguagens é uma extensão de sua antecessora mais simples. A seguir é apresentado o conjunto de relações válidas, sendo que o inverso não é verdadeiro:

- Toda ontologia *OWL Lite* válida também é uma ontologia *OWL DL* válida.
- Toda ontologia *OWL DL* válida também é uma ontologia *OWL Full* válida.
- Toda conclusão *OWL Lite* válida também é uma conclusão *OWL DL* válida.
- Toda conclusão *OWL DL* válida também é uma conclusão *OWL Full* válida.

Desenvolvedores de ontologias, que adotam *OWL*, precisam considerar qual sub-linguagem melhor se adapta a suas necessidades. A escolha entre *OWL Lite* e *OWL DL* depende até que ponto os usuários necessitam de uma construção mais expressiva. A escolha entre *OWL DL* e *OWL Full* depende se os usuários requerem facilidades dos meta-modelos do *RDF Schema*. Ao usar *OWL Full* o suporte ao raciocínio é menos previsível e implementações que suportem o *OWL Full* não existem atualmente [50].

3.2.7 Características da *OWL*

Como a *OWL* é baseada em *RDF* e *RDF-S* certas construções são diretamente relacionadas a esses formalismos. As várias construções básicas, suportadas pela *OWL*, são apresentadas a seguir:

- ***owl:Class*:** Uma classe define um grupo de indivíduos que compartilham algumas propriedades. Por exemplo, a Mariana e o Roberto são membros da classe Pessoa. Podem ser organizadas classes em uma hierarquia de especialização usando *subClassOf*. Também existem duas classes gerais pré-definidas, *Thing* que é a classe de todos os indivíduos e a super-classe de todas as classes da *OWL*, e *Nothing* que é uma classe que não possui nenhum individuo e uma subclasse de todas as classes da *OWL*.

- ***rdfs:subClassOf***: Hierarquias de classe podem ser criadas fazendo uma ou mais declarações que uma classe é uma subclasse de outra classe. Por exemplo, a classe Pessoa poderia ser declarada como uma subclasse da classe Mamífero.
- ***rdfs:Property*:** Propriedades podem ser usadas para declarar relações entre indivíduos ou de indivíduos para valores de dados. Exemplos de propriedades incluem temFilho, temParente, e temIdade. Os primeiros dois podem ser usados para relacionar uma instância da classe Pessoa a outra instância da classe Pessoa (e são ocorrências da *ObjectProperty*), e o último temIdade pode relacionar uma instância da classe Pessoa a uma instancia do tipo de dado *Integer* (sendo uma ocorrência da *DatatypeProperty*). A o*wl:ObjectProperty* e a *owl:DatatypeProperty* são subclasses da classe *rdf:Property*.
- ***rdfs:subPropertyOf*:** Hierarquias de propriedade podem ser criadas fazendo uma ou mais declarações de que uma propriedade é uma sub-propriedade de uma ou mais propriedades. Por exemplo, temIrmão pode ser declarada como uma sub-propriedade de temParente.
- ***rdfs:domain***: Um domínio de uma propriedade limita os indivíduos para os quais a propriedade pode ser aplicada. Se uma propriedade relaciona um indivíduo a outro individuo, e a propriedade tiver uma classe como um de seus domínios, então o indivíduo tem que pertencer a essa classe. Por exemplo, a propriedade temFilho pode ter declarada como domínio a classe Pais.
- ***rdfs:range*:** A escala de uma propriedade limita os indivíduos que a propriedade pode ter como seu valor**.** Se uma propriedade relaciona um indivíduo a outro individuo, e a propriedade tem uma classe como sua escala, então o outro indivíduo tem que pertencer à classe da escala. Por exemplo, a propriedade temFilho pode ter declarada como escala a classe Pessoa.

- **Indivíduos:** Indivíduos são instâncias de classes, e podem ser usadas propriedades para relacionar um individuo com outro. Por exemplo, o indivíduo Débora pode ser descrito como uma instância da classe Pessoa e a propriedade temFilho pode ser usada para relacionar o individuo Débora ao individuo Ricardo.

***OWL* Igualdade e Desigualdade -** As seguintes características da *OWL* são relacionadas à igualdade ou desigualdade.

- ***equivalentClass*** : As classes equivalentes possuem as mesmas instâncias. A igualdade pode ser usada para criar classes sinônimas. Por exemplo, a classe Carro pode ser declarada como uma *equivalentClass* da classe Automóvel.
- ***equivalentProperty***: As propriedades equivalentes relacionam um indivíduo ao mesmo conjunto de outros indivíduos. A igualdade pode ser usada para criar propriedades sinônimas.
- ***sameAs***: Essas construções podem ser usadas para criar um número de nomes diferentes que se referem ao mesmo indivíduo.
- ***differentFrom***: Um indivíduo pode ser declarado como sendo diferente de outros indivíduos.
- ***AllDifferent***: Um número de indivíduos podem ser declarados como sendo mutuamente distintos. Usa-se conjuntamente com *distinctMembers* que indica que todos os membros de uma lista são distintos.

Características de Propriedade da *OWL* - Há identificadores especiais na *OWL* que são usados para fornecer informações a respeito das propriedades e de seus valores.

- ***InverseOf***: Uma propriedade pode ser declarada como sendo o inverso de outra propriedade. Por exemplo, se temFilho for o inverso de temPai e Ricardo temPai Marcelo, então um raciocinador pode deduzir que Marcelo temFilho Ricardo.

- ***TransitiveProperty***: Se uma propriedade é transitiva, então se o par (x,y) é um exemplo da propriedade transitiva P, e o par (y,z) é uma instância de P, então o par (x,z) também é uma instância de P.
- ***SymmetricProperty***: Se uma propriedade é simétrica, e o par (x,y) é uma instância da propriedade simétrica P, então o par (y,x) também é uma instância de P.
- ***FunctionalProperty***: Se uma propriedade for uma *FunctionalProperty*, não pode ser atribuído mais de um valor para cada individuo (ou nenhum valor atribuído). Por exemplo, temMãe pode ser declarada como uma *FunctionalProperty*.
- ***InverseFunctionalProperty***: Se uma propriedade é inversa funcional então o inverso da propriedade também é funcional.

Restrições de Propriedade da *OWL* - *OWL* permite que sejam colocadas restrições em como as propriedades podem ser usadas por instâncias de uma classe. Esse tipo é usado dentro do contexto do *owl:Restriction*. O elemento *owl:onProperty* indica a propriedade restringida. As duas restrições seguintes limitam que valores podem ser usados enquanto na próxima seção as restrições limitam quantos valores podem ser usados.

- ***allValuesFrom*** : A restrição *allValuesFrom* é declarada em uma propriedade relativa a uma classe. Significa que a propriedade desta classe particular tem uma restrição de escala local associada a ela.
- ***someValuesFrom*** : A restrição *someValuesFrom* é declarada em uma propriedade relativa a uma classe. Uma classe particular pode ter uma restrição em uma propriedade que pelo menos um valor para essa propriedade é de algum tipo.

Restrições de Cardinalidade da *OWL* - *OWL* inclui uma forma limitada de restrições de cardinalidade. A cardinalidade é declarada em uma propriedade em relação a uma classe particular.

- ***minCardinality***: Se uma *minCardinality* 1 é declarada em uma propriedade em relação a uma classe, então qualquer instância daquela classe será relacionada a pelo menos um indivíduo por aquela propriedade. Essa restrição é outro modo de se dizer que a propriedade requer um valor para todas as instancias da classe.
- ***maxCardinality***: Se uma *maxCardinality* 1 é declarada em uma propriedade em relação a uma classe, então qualquer instância daquela classe será relacionado a no máximo um individuo por aquela propriedade. Uma restrição maxCardinality 1 às vezes é chamada de propriedade funcional ou única.
- ***cardinality***: É provida como uma conveniência quando for necessário definir em uma propriedade de uma classe *minCardinality* x e *maxCardinality* x. Por exemplo, a classe Pessoa tem exatamente um valor da propriedade temAniversárioMãe.
- ***intersectionOf***: *OWL* permite intersecções de classes e restrições. Por exemplo, a classe PessoaEmpregada pode ser descrita como a intersecção de Pessoa e Empregos.

Há muitas outras características e construtores que são suportados pela *OWL*, mas as suas descrições estão além do escopo deste trabalho. Uma especificação completa da linguagem pode ser encontrada em [51].

3.2.8 Lógica de Descrição e OWL

Após uma introdução à *Lógica de Descrição* (*DL*) e as características da *OWL*, agora o foco será direcionado ao mapeamento de construções *OWL* para termos

de *DL*. Como *DL* representam uma família de lógicas, a correspondência entre várias famílias de lógicas e a *OWL* será abordada. Como já mencionado, a família de *DL* básica é o idioma $\mathcal{AL}$ (idioma atributivo). A linguagem de *DL* $\mathcal{ALEN}$ é uma extensão da $\mathcal{AL}$ e incluem conceitos de negação, restrições de cardinalidade, inversão e tipos de dados. Com a adição de propriedades transitivas a linguagem $\mathcal{ALEN}$ é estendida a $\mathcal{ALCR}$+ que pode ser representada por $\mathcal{S}$. Mais linguagens de DL são obtidas a partir do agrupamento de várias construções de *DL*, por exemplo, $\mathcal{SHIF}(D)$, $\mathcal{SHOIN}(D)$, etc. A Tabela 2 apresenta detalhes do agrupamento de vários construtores. Também provê um mapeamento de sintaxe de DL à sintaxe da OWL demonstrando a relação de correspondência entre elas.

As duas sub-linguages *OWL-Lite* e *OWL DL* suportam diferentes famílias de *DL*. *OWL-Lite* é semelhante ao *DL* $\mathcal{SHIF}(\mathcal{D})$. *OWL DL* é muito próxima da $\mathcal{SHOIN}(\mathcal{D})$ que é uma extensão da $\mathcal{SHOQ}(\mathcal{D})$ [52].

Expressividade DL	Sintaxe DL	Sintaxe OWL	Descrição
$\mathcal{ALCR}_+$, também chamada de $\mathcal{S}$	A	owl: Class	Concept
	⊤	owl: Thing	Thing
	⊥	owl: Nothing	Nothing
	(C ⊑ D)	owl: subClassOf	Subsumption
	(C≡ D)	owl: equivalentClass	Equivalence
	R	owl: Property	Properties
	R	owl: ObjectProperty	Object Properties
	(C ⊓ D)	owl: intersectionOf	Conjunction
	¬C	owl: complementOf	Negation
	⊑R.C	owl: transitiveProperty	Transitive Property
$\mathcal{F}$	T ⊑ ≤ 1 R⁻	owl: FunctionalProperty	Functional Property
$\mathcal{N}$	≤nR.⊤	owl: maxCardinality	Non-Qualified Cardinality
	≥nR.⊤	owl: minCardinality	
	= nR.⊤	owl: cardinality	
$\mathcal{I}$	R	owl: inverseOf	Inverse Roles

$\mathcal{H}$	(R □ S)	rdfs: subPropertyOf	Subsumption of Roles
	(R≡ S)	owl: samePropertyAs	Equivalence of Roles
$\mathcal{O}$	{o}	owl:individual	Indivuduals
	□T. {o}	owl: hasValue	Value Restrictions
(D)	D	owl: dataTypeProperty	Datatype Property

Tabela 2. Expressividade da *DL* mapeada às construções da *OWL DL* [52].

3.2.9 Estendendo a *OWL* com Regras

A *OWL* acrescenta poder expressivo considerável à *Web* semântica. Porém, por uma variedade de razões, incluindo manter a decidibilidade dos mecanismos de inferência na *OWL DL e OWL Lite, a OWL* possui limitações expressivas. Essas restrições podem ser onerosas em alguma aplicação de domínio (e.g., descrição de serviços *Web*), onde seja necessário relacionar entradas e saídas de processos compostos às entradas e saídas demandadas [61].

Muitas das limitações da *OWL* se devem ao fato que, embora a linguagem inclua um conjunto relativamente rico de construtores de classe, a linguagem provida para tratar propriedades é pobre. Em particular, não há nenhum construtor de composição, assim é impossível capturar relações entre uma propriedade composta e outra propriedade (possivelmente composta). Um exemplo é a relação obvia entre a composição das propriedades "pai" e "irmão", e a propriedade "tio". Um modo para superar algumas das restrições expressivas da *OWL* é estendê-la com alguma linguagem de regras. O *W3C* definiu o *SWRL*, derivado do *RuleML*, como uma extensão do *OWL*.

***Rule Markup Language* (*RuleML*) -** A *RuleML* é uma iniciativa para a padronização de uma linguagem de marcação de regras, iniciada em agosto de 2000. “A Iniciativa de *RuleML* está desenvolvendo um padrão aberto, independente de fornecedor e baseada na

linguagem XML/RDF. Isso permitirá a troca de regras entre vários sistemas que incluem componentes de software distribuídos na *Web*" [62].

RuleML provê uma hierarquia modular de doze sub-linguagens para os vários tipos de usuários de regras. Todas essas sub-linguagens correspondem a sistemas de regras bem conhecidos sendo que cada sub-linguagem possui uma caracterização semântica correspondente. A atual versão, 0.8 é definida por descrições *DTD* e *XML Schema*. Dentro da *RuleML* são descritos quatro tipos diferentes de regras, organizados em uma hierarquia apresentada na Figura 14.

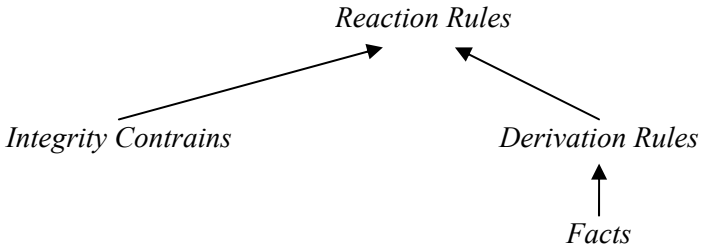


Figura 14. A hierarquia da *RuleML* [62] (modificada).

Reactionrules tratam a invocação de ações em resposta a eventos. *Integrityrule* é uma regra de reação especial, sendo que a única ação possível é a sinalização de inconsistência quando são cumpridas certas condições. Uma *derivationrule* é uma declaração de um conhecimento que é derivado de outro conhecimento. Regras de derivação são consideradas regras de reação especiais cuja ação acontece somente para somar ou 'afirmar' uma conclusão quando certas condições (premissas) forem cumpridas. E *Facts* (fatos) são regras de derivação especiais que possuem uma conjunção de premissas vazia (conseqüentemente 'verdadeiro').

Embora existam vários participantes no desenvolvimento do *RuleML* e várias ferramentas disponíveis, também existem algumas desvantagens no *RuleML*. Por exemplo, a atual versão 0.8 do *RuleML* só abrange uma forma limitada de regras de derivação e não oferece nenhuma forma de regra de reação [62]. Wagner [63] conclui:

“RuleML, em sua versão 0.8 não é suficiente como uma linguagem de regras genérica para *Web*. . . porém, é um bom ponto de partida.”

Semantic Web Rule Language (SWRL) - A SWRL [39] está em fase de desenvolvimento sendo uma combinação da *OWL DL* com as sub-linguagens *Unary/Binary Datalog da RuleML* [64]. *SWRL* pode ser considerada como *DL* + regras *FO Horn* livres de funções.

A Figura 15 apresenta uma visão geral da sintaxe abstrata do *SWRL*, que estende o OWL adicionando conceitos de variáveis e átomos. Na *SWRL*, nomes de variáveis são referências URI. Considerando que variáveis são o escopo para expressões de regras, elas não precisam ser globalmente únicas.

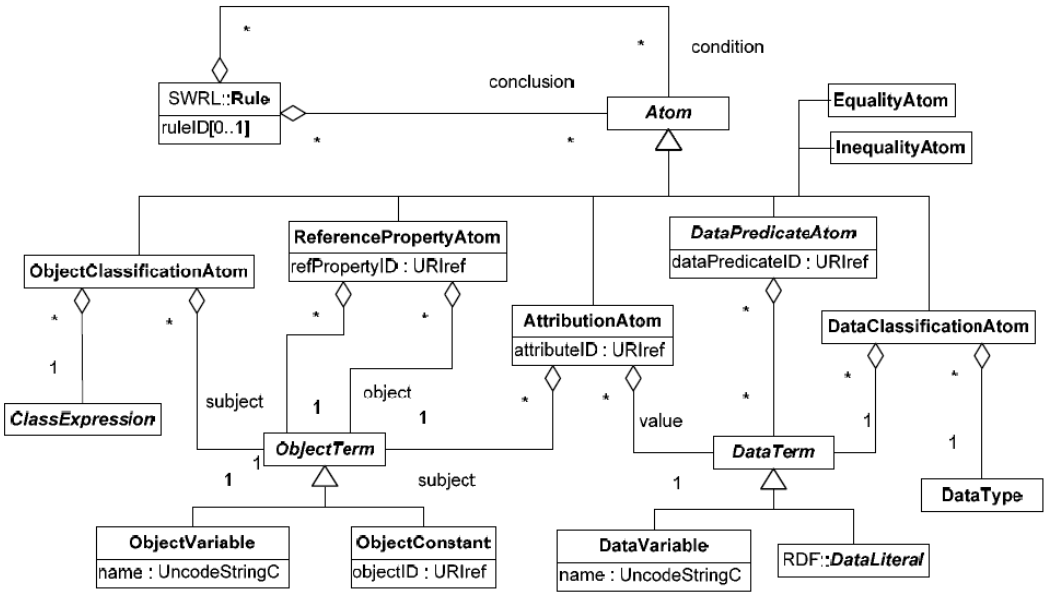


Figura 15. Sintaxe abstrata de regras SWRL [65].

A adição de regras *SWRL* provê um maior poder de expressividade a Lógica de Descrição da *OWL DL*. Por exemplo, você pode usar *First Order Logic* (*FOL*) para definir, representada na sintaxe da *SWRL,* o conceito Tia.

Pai(?x, ?y) ^ Irmã(?y, ?z) □ Tia (?x, ?z)

Nesse exemplo o “Pai(?x, ?y) ^ Irmã(?y, ?z)” é chamado de antecedente (corpo) e “Tia(?x, ?z)” é chamado de conseqüente (cabeça) da regra. Sempre que o

antecedente for verdadeiro, o conseqüente também será. A *OWL* não pode definir relações como essas. A aplicação de regras também pode estender a capacidade de composição da *OWL*.

Embora o campo da *SWRL* seja relativamente novo, *FOL* já foi amplamente estudada. Também, a combinação com *FOL* faz com que o *SWRL* tenha facilidade para operar com bancos de dados relacionais. Esta característica é muito atraente, pois a maioria da informação ainda é armazenada em bancos de dados relacionais.

A desvantagem da *SWRL* é sua complexidade computacional. Embora ainda esteja em construção, os desenvolvedores concordaram que por causa de sua indecidibilidade [40], ela suportará parcialmente a maioria das características da *DL* e programas de Lógica *Horn*. Uma boa solução para esse problema é utilizar idéias do projeto da *OWL*, desenvolvendo várias sub-linguagens de *SWRL* com complexidade decrescente. Essa tarefa ainda não começou, mas já foi considerada [66].

3.3 Arquitetura de Serviços *Web*

A arquitetura de Serviços *Web* definida pelo *W3C* é baseada no *XML* de modo a prover interoperabilidade entre diferentes plataformas. As mensagens *XML* são transmitidas utilizando *HTTP* o que facilita o trafego de mensagens pela Internet.

Essa arquitetura permite a construção de sistemas distribuídos com baixo acoplamento, utilizando tecnologias baseadas em padrões abertos, independentes da linguagem de programação utilizada e da plataforma computacional. Por conta dessas características a arquitetura apresenta vantagens sobre outros padrões de computação distribuída, onde se pode destacar:

- *Distributed Component Object Model* (*DCOM*) é proprietário, o que compromete a interoperabilidade.
- *Remote Method Invocation* (*RMI*) é baseado em *JAVA* e não opera bem em outras linguagens.
- *Common Object Request Broker Architecture* (*CORBA*) é o que mais se aproxima por ser baseado em padrões, independente de fornecedor e linguagem. Sua limitação está no tráfego de suas mensagens através da Internet e de *firewalls* corporativos.

A Tabela 3 fornece uma comparação das principais tecnologias utilizadas para descrever as características de um serviço, incluindo a opção de descrição semântica de serviços *Web* (*OWL-S*). Essas características, que são importantes ao selecionar uma tecnologia para descoberta de serviços, são descritas em [67].

Característica	*CORBA*	*RMI*	Serviços Web	*OWL-S*
Desenvolvedor	*OMG*	*JAVA (SUN)*	*W3C/OASIS*	*DARPA*
Protocolo de transporte	*TCP/IP*	*IP*	*TCP/IP*	Independente
Linguagem de Programação	Independente	*Java*	Independente	Independente
Atributos extensíveis	Não	No	Sim (*WSDL*)	Sim
Atributos não ambíguos	Não	Não	Não	Sim
Pode ser modificado em tempo de execução	Não	Não	Não	Sim

Tabela 3. Tecnologias para descoberta de serviços.

A arquitetura é composta por várias tecnologias dispostas em camadas como ilustra a Figura 16. Dessas tecnologias se destacam o *Simple Object Access Protocol (SOAP)* na camada de mensagem e o *Web Services Description Language (WSDL)* na camada de descrição. E na camada de processos, o *Universal Description, Discovery and Integration (UDDI)* e o *Business Process Execution Language for Web*

Services (BPEL4WS). As tecnologias relacionadas à segurança e gerenciamento não serão abordadas nesta dissertação.

3.3.1 *Simple Object Access Protocol* (*SOAP*)

SOAP [69] é o protocolo padrão para transporte de mensagens de serviços *Web*. A aplicação primária do *SOAP* é a comunicação entre aplicações, sendo geralmente utilizado em plataformas de *Business-to-Bussiness* (*B2B*) onde o foco é integrar aplicações de software e compartilhamento de dados. Para ser efetivo nesse ambiente um protocolo deve ser independente de plataforma, flexível e baseado em padrões. O *SOAP* satisfaz estas exigências, possui uso difundido, e foi endossado pela maioria dos fabricantes de software empresarial e pelas principais organizações de padronização (e.g. *W3C, Web Services Interoperability Organization* (*WS-I*), *Organization for the Advancement of Structured Information Standards* (*OASIS*)).

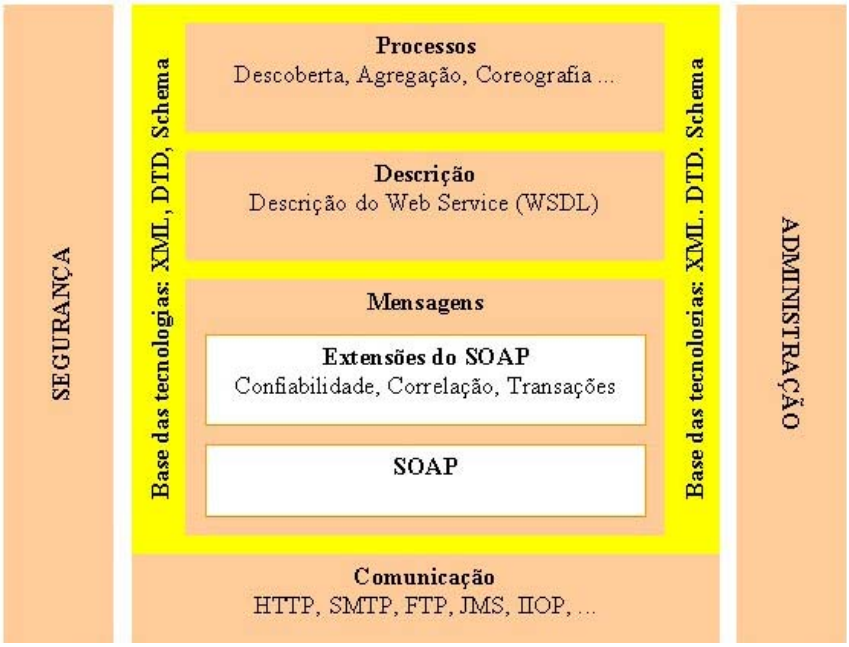


Figura 16. A arquitetura de Serviços *Web* [68]

Tecnicamente *SOAP* é simplesmente uma linguagem de marcação baseada em *XML* acompanhada por regras que ditam seu uso. Uma instância de um documento *SOAP XML*, chamada de mensagem *SOAP*, normalmente é levada como carga útil de algum outro protocolo de rede. O modo mais comum para trocar mensagens de *SOAP* é via *HTTP,* usado pelos navegadores para acessar páginas *Web*. Mensagens *SOAP* também podem ser transportadas por e-mail usando *SMTP* ou através de outros protocolos de rede, como *FTP* e *TCP/IP* puro, a Figura 17 ilustra esta característica.

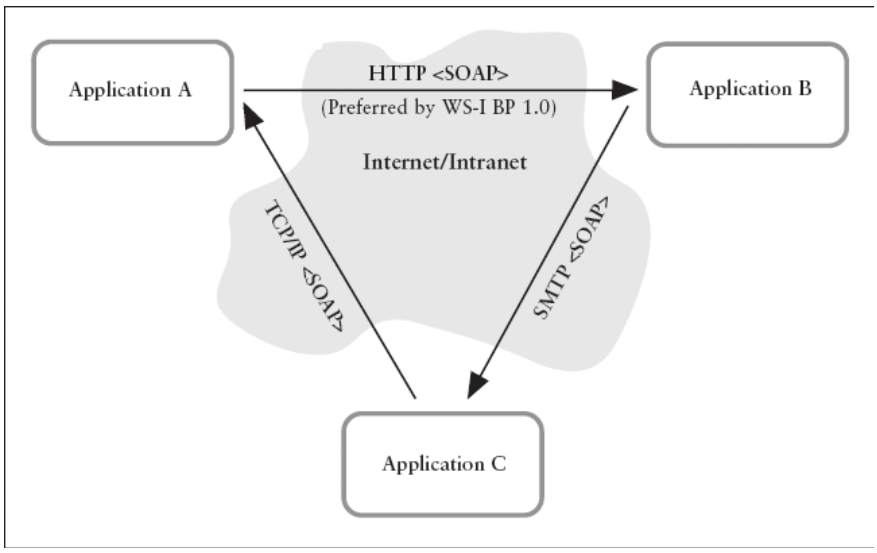


Figura 17. Uma mensagem *SOAP* pode ser transportada através de vários protocolos

Uma das principais razões do sucesso de *SOAP* é ter sido baseado em *XML*. O formato de uma mensagem *SOAP* está definido por um *XML schema* que explora *namespaces XML* tornando *SOAP* extensível. Outra vantagem do SOAP é sua definição explícita de como se anexar em um cabeçalho *HTTP*, um método padrão para *HTTP tunneling*. *HTTP tunneling* é o processo de esconder outro protocolo dentro de mensagens *HTTP* de modo a atravessar *firewalls*. *Firewalls* normalmente permitem trafego de *HTTP* pela porta 80, mas restringem ou proíbem o uso de outros protocolos e portas.

Mensagem *SOAP* - Uma mensagem *SOAP* é um documento *XML* que contém os seguintes elementos:

- Um elemento *envelope* que identifica o documento *XML* como um *SOAP*.
- Um elemento *header* (opcional) que contém informações de chamadas e repostas;
- Um elemento *fault* (opcional) que provê informação sobre erros que ocorrem durante o processamento da mensagem.
- Um *elemento body* – que contém o corpo da mensagem *SOAP* que será interpretado pelo destino final da mensagem

Esqueleto de mensagens *SOAP* - A Figura 18 apresenta a estrutura de uma mensagem *SOAP*, cujos componentes serão detalhados nos próximos tópicos.

```
<?xml version="1.0"?>
<soap:Envelope
    xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
    soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
    <soap:Header>
        ...
    </soap:Header>
    <soap:Body>
        ...
        <soap:Fault>
            ...
        </soap:Fault>
    </soap:Body>
</soap:Envelope>
```

Figura 18. Esqueleto de uma mensagem *SOAP*.

Envelope *SOAP* - O envelope *SOAP* é o elemento mais importante, o elemento raiz de uma mensagem *SOAP*, definindo o documento *XML* como uma mensagem *SOAP*. Um exemplo de envelope é apresentado na Figura 19.

```
<?xml version="1.0"?>
<soap:Envelope
   xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
   soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
   ...
   /* Informações da mensagem são inseridas neste ponto */
   ...
</soap:Envelope>
```

Figura 19. Exemplo de envelope *SOAP*

Header SOAP - O elemento *header* contém informação específica da aplicação, associada à mensagem *SOAP*. Se o *header* estiver presente, esse deve ser o primeiro elemento do envelope. Um exemplo de *header* é apresentado na Figura 20.

```
<?xml version="1.0"?>
<soap:Envelope
   xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
   soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
   <soap:Header>
      <m:Trans
      xmlns:m="http://www.w3schools.com/transaction/"
      soap:mustUnderstand="1">234</m:Trans>
   </soap:Header>
   ...
</soap:Envelope>
```

Figura 20. Exemplo de *header SOAP*

SOAP Body - O elemento requerido *SOAP Body* contém o corpo da mensagem *SOAP* que será interpretado pelo destino final da mensagem. Na Figura 21 se requisita o preço de maças, note que o *m:GetPrice* e os elementos Item são específicos da aplicação e não fazem parte da especificação do *SOAP*.

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body>
   <m:GetPrice xmlns:m="http://www.w3schools.com/prices">
      <m:Item>Apples</m:Item>
   </m:GetPrice>
</soap:Body>

</soap:Envelope>
```

Figura 21. Exemplo de requisição *SOAP*

A resposta *SOAP* a requisição do preço das maças é apresentada na Figura 22.

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body>
   <m:GetPriceResponse xmlns:m="http://www.w3schools.com/prices">
      <m:Price>1.90</m:Price>
   </m:GetPriceResponse>
</soap:Body>

</soap:Envelope>
```

Figura 22. Exemplo de reposta *SOAP*

3.3.2 *Web Services Description Language* (*WSDL*)

WSDL é uma linguagem baseada em *XML* que descreve as informações necessárias para se conectar a um serviço *Web*. Assim um arquivo *WSDL* inclui o nome da rede, do servidor que provê o serviço, junto com o tipo de conexão, como *HTTP GET* ou *SOAP*. *WSDL* também pode descrever os tipos de mensagens a serem enviadas ou recebidas, incluindo as suas estruturas e tipos de dados.

WSDL pode ser estendido para cobrir novos tipos de serviço e conexão porque ele usa um modelo abstrato de mensagens, portas de entrada e saída, e permite operações para essas portas. Na realidade, um serviço é considerado uma coleção de portas.

Na prática, essas abstrações precisam ser conectadas a redes reais e a computadores em seus atuais endereços. Ligações *(bindings*) provêem essas conexões. Há uma ligação para *SOAP*, e outras para *HTTP GET e POST*. Novas ligações podem ser adicionadas, pois o *WSDL* é extensível para cobrir novos protocolos e tipos de mensagem. A Figura 23 descreve esse modelo de serviço baseado em *WSDL*. Um documento *WSDL* contém partes que descrevem cada um desses componentes.

Pode-se notar que o *WSDL* descreve serviços de um modo limitado: só descreve os pontos técnicos necessários para fazer requisições de um serviço e receber respostas. Mesmo onde há operações, elas só são conhecidas pelos seus nomes. *WSDL*

não possui nenhuma descrição sobre a funcionalidade, em termos globais, dos serviços *Web*.

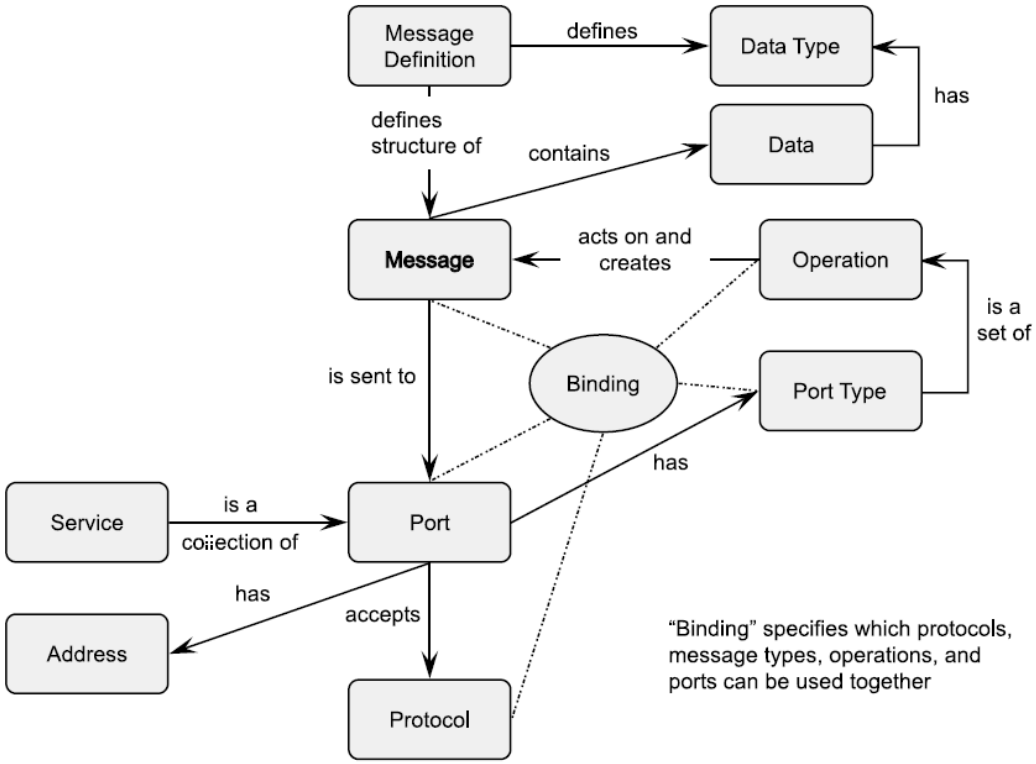


Figura 23. Modelo *WSDL* de um serviço *Web* que define quatro abstrações: mensagem, porta, protocolo, e operação [70].

Um exemplo comentado de uma descrição de serviço baseada em *WSDL* [70], onde são destacadas as principais características, é apresentado na Figura 24.

```
<definitions>
<!-- Namespace declarations have been omitted to shorten this example -->

 <types>    1 ← Define os tipos de dados.
  <s:schema elementFormDefault="qualified"
    targetNamespace="http://www.serviceobjects.com/">

    <s:complexType name="ATMLocations"> 2 ← Define o tipo ATMLocations
      <s:sequence>
        <s:element minOccurs="0"
          maxOccurs="unbounded" name="ATM" type="s0:ATM" />
        <s:element minOccurs="0"
          maxOccurs="1" name="Error" type="s:string" />
      </s:sequence>
    </s:complexType>

    <s:complexType name="ATM">    3 ← Define o tipo complexo ATM
      <s:sequence>
        <s:element minOccurs="0" maxOccurs="1"
            name="Bank" type="s:string" />
        <s:element minOccurs="0" maxOccurs="1"
            name="Address" type="s:string" />
        <s:element minOccurs="0" maxOccurs="1"
            name="Location" type="s:string" />
```

```
        <s:element minOccurs="0" maxOccurs="1"
            name="City" type="s:string" />
        <s:element minOccurs="0" maxOccurs="1"
            name="State" type="s:string" />
        <s:element minOccurs="0" maxOccurs="1"
            name="Zip" type="s:string" />
        <s:element minOccurs="0" maxOccurs="1"
            name="Latitude" type="s:string" />
        <s:element minOccurs="0" maxOccurs="1"
            name="Longitude" type="s:string" />
        <s:element minOccurs="0" maxOccurs="1"
            name="Notes" type="s:string" />
      </s:sequence>
    </s:complexType>

    <s:element name="ATMLocations" nillable="true"    4 ← Define o elemento
ATMLocations
        type="s0:ATMLocations" />
  </s:schema>
 </types>

 <message name="GetATMLocationsHttpGetIn">   5 ← Define a mensagem de entrada.
   <part name="strInput" type="s:string" />
   <part name="strLicenseKey" type="s:string" />
 </message>
 <message name="GetATMLocationsHttpGetOut">  6 ← Define a mensagem de saída.
   <part name="Body" element="s0:ATMLocations" />
 </message>


 <portType name="GeoCashHttpGet">    7 ← A Porta
   <operation name="GetATMLocations">
     <documentation>Please use our services at
         http://ws.serviceobjects.net</documentation>
     <input message="s0:GetATMLocationsHttpGetIn" />
     <output message="s0:GetATMLocationsHttpGetOut" />
   </operation>
 </portType>

 <binding name="GeoCashHttpGet" type="s0:GeoCashHttpGet">  8 ← Ligação
   <http:binding verb="GET" />     9 ← Operação HTTP Get
   <operation name="GetATMLocations">
     <http:operation location="/GetATMLocations" />
     <input>    10 ← Define o estilo de codificação da mensagem de entrada.
       <http:urlEncoded />
     </input>  11 ← Define o estilo de codificação da mensagem de saída
     <output>
       <mime:mimeXml part="Body" />
     </output>
   </operation>
 </binding>

 <service name="GeoCash"> 12 ← Define o Serviço
   <port name="GeoCashHttpGet"
     binding="s0:GeoCashHttpGet">
     <http:addresslocation="http://ws.serviceobjects.net/gc/GeoCash.asmx"/>
   </port>
  </service>
 </definitions>
```

Figura 24. Exemplo de descrição de serviço baseada em *WSDL* [70].

(1) - Os tipos de dados são definidos utilizando o *XML Schema*.

(2) - O tipo *ATMLocations* é definido. Ele pode conter qualquer número de elementos *ATM* e também um elemento *Error* que, presumidamente, contém qualquer informação de erro.

(3) - A definição do tipo complexo *ATM*, que contém o nome do banco, endereço, e assim por diante.

(4) - O *schema* também define um elemento chamado *ATMLocations*, e seu tipo é declarado como *ATMLocations* (é legal utilizar um mesmo nome para um elemento e um tipo). Esse elemento representará a resposta de retorno a um pedido.

(5),(6) – Duas mensagens são definidas, uma para a mensagem de entrada (o pedido) e a outra para a mensagem de saída (a resposta). Não existe uma regra para nomear as mensagens, desde que os nomes só sejam usados para referência dentro do documento *WSDL*. Os nomes são normalmente escolhidos de modo a descrever o propósito da mensagem. A mensagem de pedido é chamada *GetATMLocationsHttpGetIn*; a parte *HttpGetIn* do nome sugere que esta mensagem se aplica a requisições do tipo *HTTP GET*. A mensagem de saída definida em (6) usa o tipo *ATMLocations* definido em (2).

O nome de cada elemento de *part* se refere ao nome do parâmetro que será provido como parte da requisição *GET*. Note que há um *strInput,* para o cep, e um *strLicenseKey*, para um código de autorização. Estes parâmetros serão anexados à *URL* do serviço.

A *URL* para essa operação particular de serviço é construída a partir de valores fornecidos em (9) e (12) O usuário poderia invocar o serviço para o CEP 12398 e chave de licença XXX-YYY-ZZ, com essa *URL*:

```
http://ws.serviceobjects.net/gc/GeoCash.asmx/
/GetATMLocations?strInput=12398&strLicenseKey=XXX-YYY-ZZ
```

Note que o tipo de dados do pedido é uma *string*, que é a única opção de se enviar com um pedido do tipo *GET*. É suposto que a resposta seja um documento *XML* com um elemento *ATMLocations* que contenha várias localizações de *ATMs* em elementos *ATM*.

(7) – Uma porta é um ponto de conexão, e em *WSDL* é uma conexão abstrata em vez de uma fisicamente real. Note que a porta consiste em uma operação (poderia ser mais que uma) e uma mensagem de entrada e saída. Essas mensagens estão relacionadas com as definidas em (4) e (5).

(8) – Nesse ponto são conectadas as mensagens e portas abstratas com as reais entidades de rede. A ligação é conectada à operação definida anteriormente e a operação *HTTP GET* definida em (9).

(10),(11) – Onde são especificados os estilos de codificação das mensagens de entrada e saída. Nesse exemplo, se espera receber uma mensagem *XML* de resposta.

(12) Essa é uma definição de serviço (documentos *WSDL* podem definir múltiplos serviços). Note que o serviço é representado como uma coleção de portas. O .asmx é uma das assinaturas de um serviço *Web* desenvolvido com a plataforma *.NET* da Microsoft, mas o arquivo *WSDL*, e o serviço, podem ser utilizados por outras plataformas e linguagens.

3.3.3 *Universal Description, Discovery and Integration (UDDI)*

Há dois modos de se achar um *site Web* na Internet: digitando a *URL* diretamente, ou usando um mecanismo de busca para localizar um *site* que satisfaça os critérios providos para seleção. Como os serviços *Web* operam em cima da *Internet* ou *intranet* corporativa, a localização pode ser feita de modo idêntico aos *sites Web*. Deve-se conhecer o endereço de rede do serviço, ou usar um utilitário de pesquisa para serviços *Web*.

Se já é conhecido o endereço do serviço *Web* que se deseja invocar, ou o endereço de sua interface, então se pode contatar o serviço diretamente. Isso acontece

tipicamente em uma *intranet* corporativa, ou uma *extranet* de *B2B* no qual se compartilham os endereços de serviços importantes com os seus parceiros empresariais. Porém, se o endereço do serviço não é conhecido, ou sua interface, então tem que se procurar o serviço examinando um registro de serviços *Web*.

Antes de um parceiro, cliente potencial, ou aplicação interna invocar um serviço, deve-se localizar uma aplicação empresarial ou interna que possua o serviço pretendido, descobrir a interface de chamada e a semântica, e então escrever ou configurar o software de modo a colaborar com o serviço *Web*. Para realizar isso, é necessário um mecanismo unificado para publicação, localização e ligação de serviços *Web*.

O projeto *UDDI,* encabeçado pela *IBM, Microsoft, e Ariba* no início de 2000, é uma solução desenvolvida para prover um *framework* aberto para registro e localização de serviços empresariais. O termo "*UDDI*" está associado a vários contextos, entre eles: o projeto *UDDI*, a especificação *UDDI*, a *API UDDI*, ou até mesmo um servidor de registro *UDDI* particular.

Registro *UDDI* - O *UDDI* provê um mecanismo neutro para publicação e localização de descrições de serviço. Ele suporta diferentes tipos de descrição de serviços, incluindo documentos *WSDL*. A especificação define uma *API* para registro que cumpre duas tarefas essenciais, registro de um negócio e seus serviços, e localização (e ligação) a um serviço registrado. O nó de um registro *UDDI* serve como um provedor de serviços, divulgador de serviços; um registro de serviço, expondo um diretório de serviços *Web* navegável; e um requisitante de serviço, localizando o serviço solicitado e ligando o cliente àquele serviço. Registro e localização são realizados enviando comandos *UDDI* no corpo de uma mensagem *SOAP* para um registro de *UDDI*.

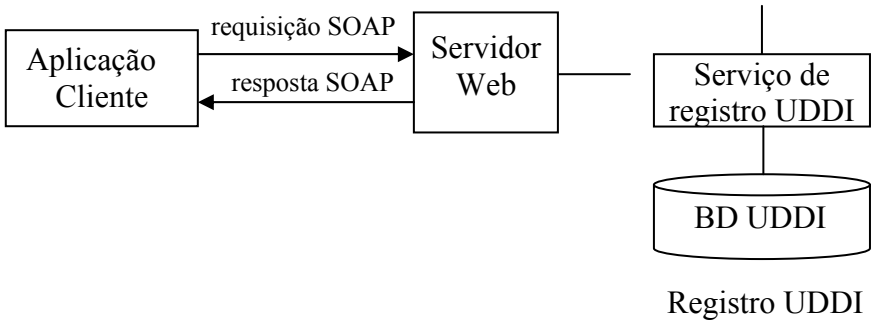


Figura 25. Exemplo de consulta a um registro UDDI

Informação	**Operações**	**Informações Detalhadas** (suportadas pela *API* de baixo nível)
White Pages: Informações sobre nome, endereço, número de telefone, e qualquer outra informação de contato de um determinado negócio,	***Publish***: Como o provedor de um serviço *Web* se registra.	***Business information***: Contida em um objeto *BusinessEntity* , que a partir do seu relacionamento com outros objetos pode-se obter informações sobre serviços, categorias, contatos, *URLs*, e outros dados necessários para se interagir com um determinado negócio.
Yellow Pages: Informação que categoriza negócios. Esta categorização está baseada em padrões existentes (não-eletrônicos).	***Find***: Como encontrar um serviço *Web* em particular.	***Service information***: Descreve um grupo de serviços *Web*. Esses estão contidos em um objeto *BusinessService.*
Green pages: informações técnicas sobre os serviços *Web* de um determinado negócio.	***Bind***: Como a aplicação se conecta e interage com um serviço *Web*, após ele ter sido localizado. ***BindingTemplate***	***Binding information*:** Se obtêm os detalhes técnicos necessários para se invocar um serviço *Web*. Isso inclui *URLs*, informações sobre os nomes dos métodos, tipos de argumento, e assim por diante. O objeto *BindingTemplate* representa esses dados. ***Service Specification Detail***: Descreve meta dados sobre as várias especificações implementadas por um determinado serviço *Web*. Na especificação *UDDI* são chamados de *tModels*.

Tabela 4. Operações *UDDI* e suas descrições.

A arquitetura básica para se consultar um registro *UDDI* pode ser vista na figura 25. Uma requisição *SOAP* é enviada ao servidor e de-serializada pelo processador *SOAP* do registro *UDDI*. As chamadas *UDDI* são enviadas ao serviço de registro que irá consultar a base de dados do *UDDI*. Uma resposta é devolvida por meio

do processador *SOAP*, que serializa a informação e despacha ao servidor *Web* que irá enviá-la à aplicação cliente.

Registrando serviços com o *UDDI* - Os registros *XML* do *UDDI* armazenam informações sobre organizações, seus serviços, e como esses serviços podem ser acessados. Há um modelo distinto de dados e uma *API UDDI* que facilitam a publicação de entidades de negócio e seus serviços, como também a consulta a essas informações. Esse sistema pode ser comparado a uma lista telefônica eletrônica na qual são indexados diferentes tipos de informação e organizadas de diferentes modos. Especificamente, é dito que o *UDDI* suporta três tipos de dados de registro: Páginas Brancas (*White Pages*) organizando negócios através de nome; Páginas Amarelas (*Yellow Pages*) organizando negócios através de categoria; e Páginas Verdes (*Green Pages*) organizando negócios através de serviço. A Tabela 4 apresenta um resumo das informações e operações suportadas pelo registro *UDDI*.

3.4 Serviços *Web* Semânticos

Os serviços *Web*, que há anos são a base das arquiteturas orientadas a serviços, começam a apresentar deficiências principalmente nas áreas de descrição, descoberta e composição de serviços. O problema está na falta de suporte semântico na linguagem de descrição de serviços *WSDL* e no mecanismo de armazenamento e pesquisa de serviços *UDDI*.

Com o propósito de integrar a *Web* semântica aos serviços *Web*, foram desenvolvidas novas linguagens para descrição e composição de serviços apresentadas na Figura 26. Essa integração ainda está em curso, e as atuais extensões semânticas do *UDDI* [71,72] não garantem a compatibilidade e performance necessárias.

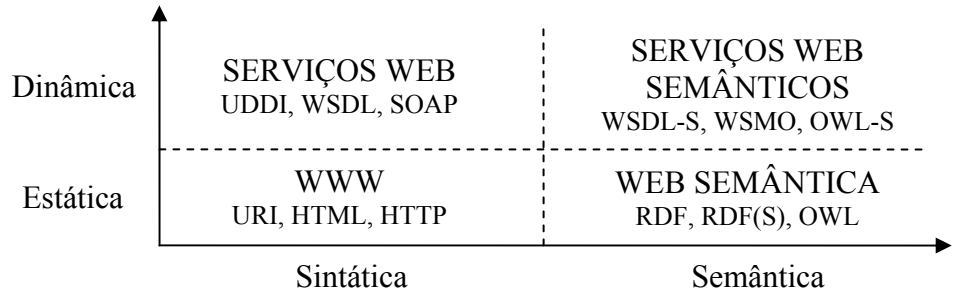


Figura 26. Evolução das tecnologias *Web*

3.4.1 *Web Ontology Language for Services* (*OWL-S*)

OWL-S é uma ontologia de serviços *Web* baseada na *OWL* que disponibiliza aos provedores de serviços *Web* um conjunto de linguagens de marcação construído para descrição das propriedades e capacidades de serviços *Web* de modo não ambíguo e interpretável por computadores. A *OWL-S* facilita a automatização das tarefas de serviços *Web*, incluindo automatização da descoberta, execução, composição e inter-operação, de serviços [8].

O *OWL-S* pode ser subdividido em três partes, cada uma responsável por uma tarefa particular, que descrevem as seguintes características de um serviço *Web*: o que o serviço faz é apresentado no *Service Profile*, como o serviço trabalha é descrito pelo *Service Model*, e o acesso ao serviço é suportado pelo *Service Grounding*.

O Service Profile é uma descrição de alto-nível do provedor de serviços e das funcionalidades do serviço. O *Service Profile* descreve as seguintes características: informações de contato de um determinado provedor/serviço (e.g., *serviceName, textDescription, contactInformation*); atributos para categorização do serviço oferecido (e.g., *serviceParameter, serviceCategory*); e as representações funcionais do serviço em forma de *Inputs Outputs Preconditions Effects (IOPEs*). Os *IOPEs* são descritos pelas propriedades *hasParameter, hasInput, hasOutput, hasPrecondition, hasEffect*.

Três componentes são necessários para se usar um serviço *Web*: o requisitante que busca um serviço, o provedor que oferece um serviço, e os componentes de infra-estrutura que agem como registros para auxiliar o requisitante na localização do provedor de serviço apropriado. O papel de um Service Profile é ser publicado em um registro, porque ele provê as informações necessárias para a descoberta de um serviço. Pode ser visto como um cartão de visita, descrevendo o que o serviço faz sem mencionar todos os detalhes que são capturados no *Service Model*.

O *Service Model* é uma descrição do fluxo de trabalho definindo como o serviço opera. Uma subclasse do *Service Model* é o *ProcessModel,* onde os serviços são representados através de processos. Cada processo é visto como uma transformação de dados de um conjunto de entradas para um conjunto de saídas e como uma transição no contexto de um estado para outro, onde essa transição é descrita por pré-condições e efeitos.

O *Service Grounding* especifica os detalhes de como acessar o serviço. É a única parte de um serviço *OWL-S* em um nível concreto de especificação, considerando que o *Service Model* e o *Service Profile* são mais abstratos. Está altamente relacionado com *WSDL*: um processo atômico do *OWL-S* corresponde a uma operação do *WSDL*.

OWL-S se assemelha a outras tecnologias de vários modos:

- O *Service Profile* é análogo aos anúncios das páginas amarelas do *UDDI*.
- O *ProcessModel* é semelhante ao modelo de processo empresarial da *BPEL4WS*.
- O *Grounding* é o mapeamento da *OWL-S* para a *WSDL*.

Descrições de serviço *OWL-S* podem se apoiar em outras ontologias que descrevem tipos particulares de serviço e suas características. Por exemplo, suponha que exista uma ontologia em *OWL* para descrever sensores. Essa ontologia contém uma classe

principal Sensor para definir o conceito de sensor. Sensor tem subclasses como *AcousticSensor* e *InfraRedSensor*. Na semântica da *OWL*, subclasses herdam as propriedades de suas superclasses e podem estender esses atributos com a adição de novos. Como descrições de serviço *OWL-S* são nada mais do que documentos *OWL*, pode-se utilizar todas as características do modelo de domínio da *OWL* para estruturar as descrições de serviço, como também usar livremente conceitos de outras ontologias.

3.4.2 *Matching de Serviços*

Matching de serviços significa comparar a descrição de uma requisição de serviço com as descrições de serviços anunciados. O objetivo dessa comparação é obter a informação sobre a similaridade que elas possuem. Esse grau de similaridade é usado para determinar se o serviço anunciado satisfaz às potencialidades requisitadas [73].

Comparar a descrição da requisição de serviço com as descrições dos serviços anunciados deve levar em conta todas as entradas e as saídas. As descrições das entradas da requisição de serviço são comparadas às descrições das entradas de serviços anunciados. E as descrições das saídas da requisição de serviço são comparadas às descrições das saídas dos serviços anunciados.

A fim de encontrar o serviço que melhor atenda as potencialidades requisitadas é necessário distinguir graus de similaridade. Usando métodos de comparação da *DL* quatro diferentes graus de similaridade entre conceitos podem ser identificados:

- ***Exact*** – dois conceitos são semanticamente iguais.

• ***PlugIn*** – a saída anunciada contém mais características do que a saída pedida ou a entrada pedida contém mais características do que a entrada anunciada.

• ***Subsumes*** – inverso do *PlugIn* – as saídas pedidas contém mais características do que é fornecido pelas saídas anunciadas ou a entrada anunciada necessita mais do que é pedido como a entrada.

• ***Fail*** – nenhuma das anteriores.

Esses tipos de *matching* têm efeitos diferentes em entradas e em saídas. Para auxiliar na compreensão dessas diferenças será apresentado um exemplo onde os efeitos dos tipos de *matching* são esclarecidos. O exemplo consiste em comparar os parâmetros de uma requisição de serviço com os parâmetros de um serviço anunciado. Os serviços são anotados com os três conceitos mostrados na Figura 27, e listados na Tabela 5.

O match exato significa que os conceitos comparados são iguais. Isto é verdadeiro para a entrada 1 e a saída 1 do exemplo.

PlugIn é usado para identificar que a saída do serviço anunciado é mais geral do que a saída do serviço requisitado. A saída pode ser anotada como cães e gatos, enquanto a saída do serviço requisitado requer somente cães. A saída do serviço que fornece cães e gatos pode então ser utilizada para satisfazer a saída da requisição e a informação extra que é obtida, gatos, pode ser ignorada.

O *match* exato é considerado como um *match* melhor do que o *PlugIn* porque um *match* exato está mais perto do que foi requisitado. Isso é verdadeiro mesmo que os *matches PlugIn* forneçam o mesmo, ou mais, do que os *matches* exatos, e é baseado na definição que um conceito que represente algo geral é menos especializado do que um conceito que represente algo menos genérico, que é o mesmo que dizer que um serviço que fornece somente cães é mais especializado do que um serviço que

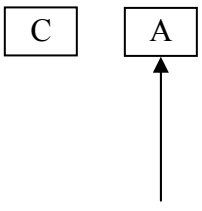
forneça cães e gatos. Isso é verdadeiro no termo da especialização, mas não necessariamente em termos de qualidade. Um serviço que fornece cães não obrigatoriamente oferece mais tipos de cães ou uma qualidade melhor de cães do que um serviço que forneça cães e gatos.

No exemplo é descrito que a saída anunciada 2 fornece o conceito A, enquanto B foi requisitado. A Figura 27 mostra que o conceito A contém o B. A é conseqüentemente uma substituição válida, onde A pode ser "plugado" no lugar do B. A saída fornece mais capacidades do que foi requisitado.

No caso das entradas a lógica é invertida. A entrada anunciada necessita ser mais específica do que a entrada pedida. Isso é porque a entrada tem que ser satisfeita. No exemplo, pode-se ver que a entrada anunciada 2 necessita B, quando o que foi requisitado é A. O pedido está fornecendo conseqüentemente algo mais geral do que é necessário e satisfaz à entrada como um *PlugIn*.

O terceiro *match*, *subsumes*, significa satisfação parcial. Neste caso o serviço anunciado não consegue suprir todas as funcionalidades requisitadas. A entrada anunciada 3 requer A, mas o serviço requisitado fornece somente B, que é uma subclasse de A. A entrada do serviço anunciado conseqüentemente não é satisfeita.

A saída 3 mostra como a saída anunciada B *subsumes* a saída requisitada. A saída do serviço anunciado não pode satisfazer completamente à saída requisitada. Entretanto, a satisfação parcial pode ser útil na composição dos serviços. Dois serviços que fornecem saídas complementares podem, por exemplo, ser combinados para satisfazer a alguma saída requisitada. Do mesmo modo, serviços podem ser combinados de modo a satisfazer as entradas.



B □ A
B *subsumes* A
A *is subsumed by* B

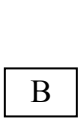


Figura 27. Ontologia representada com conceitos, e vários modos de se descrever relações de subordinação.

Serviço Requisitado	Serviço Anunciado	Tipo de *Match*
Entrada 1: A	Entrada 1: A	*Exact*
Entrada 2: A	Entrada 2: B	*PlugIn*
Entrada 3: B	Entrada 3: A	*Subsumes*
Entrada 4: A	Entrada 4: C	*Fail*
Saída 1: A	Saída 1: A	*Exact*
Saída 2: B	Saída 2: A	*PlugIn*
Saída 3: A	Saída 3: B	*Subsumes*
Saída 4: A	Saída 4: C	*Fail*

Tabela 5. Classificação dos vários tipos de *matches*.

O quarto *match, fail*, significa que os conceitos comparados não possuem nada em comum, são disjuntos.

Distinguir esses tipos de *matches* é necessário para se encontrar serviços similares. Essa escala discreta é usada para dar a cada serviço anunciado um grau de *match*, que torna possível requisitar os serviços anunciados de acordo com sua similaridade em relação ao serviço requisitado.

3.4.3 Composição de Serviços

Uma funcionalidade chave é automatizar a composição de serviços a partir da definição, pelo cliente, dos estados iniciais e finais. Esses estados são descrições semânticas da informação, e a composição significa encontrar uma seqüência de serviços combinados que forneça uma transformação da informação do estado inicial ao estado final. Essas composições são definidas a partir da resolução das dependências entre serviços, baseada em suas entradas e as saídas. Um serviço que, por exemplo,

tenha entradas que *match* com as entradas iniciais pedidas pode ter as saídas que combinam com as entradas de mais de um serviço.

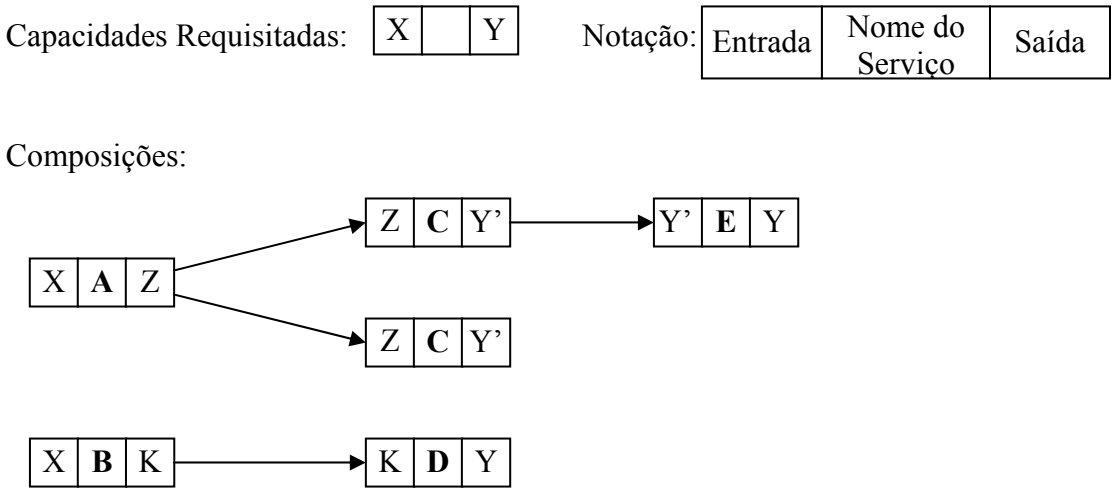


Figura 28. Exemplo de composições de serviços.

A Figura 28 ilustra composições na forma de um grafo onde os nós A e B são serviços que combinam com as entradas iniciais requisitadas. As saídas desses serviços são combinadas subseqüentemente com as entradas dos serviços anunciados restantes, onde cada serviço-nó mantém uma lista dos serviços não utilizados. Os ramos dos nós são expandidos exaustivamente com todas as composições possíveis, mas sem permitir que o mesmo serviço seja usado mais de uma vez, evitando laços infinitos. No gráfico pode-se ver que os serviços A e B têm as entradas que combinam com as entradas requisitadas X. O serviço C tem as entradas que combinam com as saídas do serviço A, e o serviço D tem as entradas que combinam com as saídas do serviço B. Os serviços que são introduzidos no grafo têm suas saídas combinadas com as saídas finais requisitadas e com as entradas dos serviços disponíveis restantes. A razão para isto é que se podem encontrar múltiplas composições diferentes, e mesmo que uma das composições tenha alcançado o estado final, a outra composição pode, por alguma razão, ser preferida. Como um exemplo, considerar a Figura 28 onde o serviço A ramifica em dois nós do mesmo serviço C. A razão para esta ramificação é que o

parâmetro de saída Y' do serviço C combina como um *PlugIn* com Y, e é conseqüentemente um *match* com a saída final pedida, e ao mesmo tempo combina com as entradas de um serviço anunciado não utilizado E. Desse modo duas composições diferentes são criadas.

Composição em termos funcionais – Como apresentado, pode ser necessário selecionar um conjunto de serviços, compostos na forma de um fluxo de execuções, de modo a satisfazer a requisição de um serviço. O processo de decomposição da requisição e seleção de serviço é comumente chamado de composição de serviço a nível funcional.

A composição de serviço a nível funcional trata o problema de selecionar um conjunto de serviços que, combinados de um modo satisfatório, podem satisfazer uma determinada requisição. Cada serviço existente está definido em termos de uma interação atômica, (e.g., parâmetros de entrada e saída e possivelmente também em termos de pré-condições e efeitos). Composição de serviço a nível funcional explora a informação que é provida por um perfil de serviço baseado em *OWL-S*.

Inicialmente é necessário identificar os serviços *Web*, que farão parte da composição, assumindo que os mesmos são definidos pelos seguintes elementos: Entradas *I*, Saídas *O*, Pré-Condições *P* e Efeitos *E*. Seja S = {S_1, S_2,..., S_n} um conjunto de serviços, onde cada S_i é definida por uma quádrupla $\left(I_s^i, O_s^i, P_s^i, E_s^i\right)$, e seja M = {$M_1$, M_2,...,M_n} um conjunto de metas, onde cada meta M_j é definida por uma quádrupla $\left(I_m^j, O_m^j, P_m^j, E_m^j\right)$. Se existe um S_i tal que:

1) $\bigcup_{i=1}^{m} I_s^i \subseteq \bigcup_{j=1}^{n} I_m^n$ e $\bigcup_{i=1}^{m} P_s^i \subseteq \bigcup_{j=1}^{n} P_m^n$; e,

2) $\bigcup_{i=1}^{m} O_s^i \supseteq \bigcup_{j=1}^{n} O_m^n$ e $\bigcup_{i=1}^{m} E_s^i \supseteq \bigcup_{j=1}^{n} E_m^n$

são incluídos na lista de possíveis serviços a serem aplicados na adaptação de um conteúdo. Caso somente a primeira parte seja satisfeita, significando que não foi encontrado um serviço que realize sozinho a adaptação necessária, é ativado o algoritmo de composição. Uma das técnicas utilizadas para a composição de serviços é a *forward chaining* [74]. Essa técnica baseia-se na seleção de um possível serviço *S*, que contenha as entradas requeridas por *M* e verifica se os elementos de saída são satisfeitos. Caso *S* não atinja o objetivo, um novo objetivo *M'* será definido a partir de *M* e das saídas e efeitos geradas por *S* e todo o processo se repete.

Caso a composição de serviço a nível funcional atinja a meta, os serviços selecionados serão escalonados em um fluxo de execução que deverá respeitar a dependências de dados entre os serviços (e.g., restrições na ordem de execução dos serviços).

Composição em termos de processo - Dado um conjunto de serviços *Web* existentes W_1,. . . , W_n, o problema de construir uma composição em termos de processo consiste em achar um programa que interaja com esses serviços *Web* de um modo satisfatório, com o objetivo de alcançar uma determinada exigência de composição (ou meta de composição). É chamado composição de serviço em termos de processo - o processo de gerar esse programa. Considerando o caso de uma agência de viagens virtual e assumindo que um conjunto de provedores de serviços de turismo foi identificado para resolver um pedido de cliente. Esses serviços podem consistir, por exemplo, em um serviço de reserva de vôo (ou um serviço de reserva de viagem de trem) e um serviço de reserva de hotel que sejam adequados ao pedido específico do cliente, nesse caso um destino específico (a seleção de tais serviços *Web* pode ser o resultado de uma composição em termos funcionais). A meta da composição em termos de processo é

obter um código executável que invoque esses serviços *Web* de modo a obter uma oferta para o pedido do cliente.

Na definição da aplicação de código executável a composição, é necessário levar em conta o fato que, em casos reais, a reserva de um hotel não é um passo atômico, mas requer uma sucessão de operações, incluindo autenticação, submissão de uma requisição específica, negociação de uma oferta, aceitação (ou recusa) da oferta, e reserva do quarto. Quer dizer, os serviços Web W_1. . . , Wn são normalmente compostos, ou seja, a interação com eles não consiste em um único passo de requisição-resposta, necessitando seguir um protocolo complexo para alcançar o resultado exigido. Além disso, os passos que definem a interação complexa não seguem obrigatoriamente uma sequência. Realmente, esses passos podem ter condições, ou resultados não nominais (e.g., autenticação pode falhar; pode não existir nenhuma oferta disponível de um determinado serviço) isso afeta os passos seguintes (e.g., nenhum pedido pode ser submetido se a autenticação falhar; se não houver nenhuma oferta disponível, uma ordem não poder ser submetida). Também pode ser o caso que a mesma operação possa ser repetida iterativamente, por exemplo, de modo a refinar um pedido ou negociar as condições de oferta.

Os detalhes da exata seqüência de operações exigidas para se interagir com um determinado serviço não são essenciais na fase de descoberta. Levar em conta esses detalhes se torna imprescindível quando se gerar o código executável da aplicação de composição. Por isso, a composição de processo de serviços *Web* existentes precisa ser descrita em termos de processos complexos e compostos. Esses processos consistem em arbitrar (condicional e iterativo) combinações de interações atômicas, e essas interações atômicas podem ter resultados condicionais (composição em nível de processo explora a informação que é provida pelo perfil de serviço do *OWL-S*).

Também como uma conseqüência, o código executável gerado tem que ser um programa complexo, sendo que ele tem que levar em conta todas as possíveis contingências que acontecem na interação com os serviços *Web*.

Composição automatizada parte de um conjunto de serviços *Web*, e de uma requisição de composição, e gera um serviço *Web* executável que implementa o serviço composto. A síntese de um serviço Web composto não é limitada a um componente atômico de serviços *Web*. A função desse componente é definir um protocolo de interação com os serviços selecionados, de forma que uma implementação executável da composição é obtida. Desse ponto de vista o serviço *Web* é definido como um fluxo de atividades ou um protocolo de interação.

3.5 *Web* Semântica e Computação Pervasiva

Atualmente, existem vários projetos que utilizam ontologias para a representação de contextos que refletem a situação do usuário no mundo real. Essas ontologias podem ser construídas utilizando a linguagem *OWL* e servirem aos propósitos da *Web* Semântica. Dessa forma as instâncias dos conceitos podem referenciar uma pessoa, um dispositivo computacional ou mesmo um local. Um projeto bastante conhecido nessa área é o *SOUPA (Standard Ontology for Ubiquitous and Pervasive Applications)* [75] que utiliza a *OWL* na criação de suas ontologias. O objetivo do projeto é definir ontologias para suportar aplicações destinadas à computação pervasiva. O desenvolvimento do *SOUPA* é conduzido por um conjunto de casos de uso. O vocabulário do *SOUPA* utiliza o vocabulário de algumas ontologias existentes e seu mérito está em providenciar aos desenvolvedores de aplicações uma ontologia que combina muitos vocabulários úteis de diferentes ontologias consensuais.

As ontologias que são referenciadas pelo *SOUPA* são: *Friend-Of-A-Friend* (*FOAF*) [76], *DAML-Time* [77], as ontologias sobre espaço da *OpenCYC* [78], Regional Connection Calculus *(RCC*) [79], *COBRA-ONT* [80], *MoGATU BDI* [81] e a ontologia de políticas Rei [82]. Dentre as características de cada uma pode-se destacar.

- ***FOAF*:** essa ontologia permite a expressão de informações pessoais e relacionamentos. É útil para a criação de sistemas de informação que suportam comunidades online. Aplicações de computação pervasiva (CP) usam essa informação para raciocinar sobre o perfil e conexões sociais da pessoa em relação aos seus vizinhos.
- ***DAML - Time*:** os vocabulários dessa ontologia são desenvolvidos para expressar conceitos temporais e propriedades comuns a qualquer formalização de tempo. Aplicações de CP podem usar essa ontologia para compartilhar representações comuns de tempo e para raciocinar sobre as ordens temporais de diferentes eventos.
- **Ontologias espaciais *OpenCYC* e *RCC*:** as ontologias do *OpenCYC* definem um conjunto compreensivo de vocabulários para representação simbólica de espaço. A ontologia *RCC* consiste em vocabulários para expressar relações espaciais para raciocínio qualitativo de espaço. Em aplicações de CP essas ontologias podem ser exploradas para descrever e raciocinar sobre contexto de localização.
- ***COBRA-ONT* e *MoGATU BDI*:** ambas ontologias são destinadas a suportar a representação de conhecimento e inferências em ambientes de CP. Enquanto o projeto do COBRA-ONT foca na modelagem do contexto em salas de reuniões inteligentes o projeto do *MoGATU BDI* foca na modelagem de

crenças, desejos e intenções de usuários humanos e agentes.

- **Ontologia de política Rei:** a ontologia Rei define um conjunto de conceitos de direitos e deveres (e.g., direitos, proibições, obrigações e desobrigações) para especificar e inferir sobre regras de segurança e controle de acesso. Em ambientes de CP os usuários podem usar essa ontologia de política para especificar regras de alto nível para garantir e revogar os direitos de acesso para/de diferentes serviços.
- **Ontologias *SOUPA***: consiste em dois conjuntos distintos, porém relacionados, de ontologias: *SOUPA Core* e *SOUPA Extension*. As ontologias do *SOUPA Core* pretendem definir os vocabulários genéricos que são universais para diferentes aplicações em computação pervasiva. As ontologias do *SOUPA Extension* são extensões de ontologias básicas onde são definidos vocabulários adicionais para suportar tipos específicos de aplicações e providenciar exemplos para futuras extensões de ontologias.

3.6 Conclusão

A proposta da *Web* semântica se baseia em um conjunto de padrões que possibilita uma interpretação semântica de informações disponíveis na *Web*. Para expressar a semântica de um determinado conteúdo são incorporadas informações que descrevem e relacionam os seus componentes, essas informações são baseadas em ontologias.

O uso de ontologias possibilita uma interpretação não ambígua da informação e geralmente está integrada a lógica de descrição. Essa integração possibilita o uso de mecanismos de raciocínio (*reasoner*), facilitando a descoberta e

composição de conhecimento. No caso deste trabalho a descrição semântica do contexto de entrega e dos serviços de adaptação disponíveis facilita o desenvolvimento de uma política de adaptação mais genérica e precisa.

Essas características da *Web* semântica começam a ser integradas na arquitetura de serviços *Web*. Arquitetura que possui uma grande adoção pelo mercado devido a sua independência de fornecedor, linguagem de programação, e plataforma. Essa integração está sendo desenvolvida através de novas linguagens como a *OWL-S* e a *WSDL-S*.

Nesta dissertação foi adotada a *OWL-S* para descrever os serviços de adaptação e a *OWL-DL* para descrever os perfis de contexto de entrega. A *OWL-S* propicia uma rápida integração entre ontologias e os padrões de serviços *Web* (e.g., *WSDL*), além disso integra uma linguagem de regras, a *SWRL*, que será utilizada pelo mecanismo de descoberta e composição de serviços.

4 Especificação de Perfis e Regras

O principal componente de uma arquitetura de adaptação de conteúdo é o que define: os serviços de adaptação que serão executados; os adaptadores locais ou remotos que realizarão essas adaptações; quando as adaptações poderão ser solicitadas; e a ordem de suas execuções. Para que essas decisões, que compõe a política de adaptação, sejam tomadas com eficiência é necessário conhecer o conjunto de informações relativas ao ambiente de adaptação, que é constituído pelo contexto de entrega. Além disso, devem ser definidas condições para se invocar, ou não, determinado serviço, que são expressas através de regras.

Neste capitulo são descritos os perfis propostos por este trabalho. Esses perfis são o resultado de pesquisas nos temas de contexto de entrega e adaptação de conteúdo, sendo baseadas em trabalhos correlatos e outras especificações (e.g., *UAPROF*, *WURFL, OPES*). Também são propostas soluções para a integração de regras aos perfis de serviço e a integração de serviços Web, facilitando a descoberta e composição de serviços. Por fim, são descritas as alteração realizadas no *framework* FACI com o objetivo de implementar as soluções propostas.

4.1 Perfis

Para descrever o contexto de entrega, visando à eficácia da política de adaptação, as seguintes informações são necessárias: características e capacidades dos dispositivos de acesso; dados pessoais e preferências dos usuários; condições da rede de comunicação; características dos conteúdos requisitados; resoluções contratuais entre o provedor de serviços e o usuário final; e características dos servidores de adaptação.

Essas informações estão contidas, respectivamente, nos seguintes perfis: *dispositivo*, *usuário*, *rede*, *conteúdo, Service Level Agreement (SLA) e servidores de adaptação.* Esses perfis são descritos em *OWL*.

Também são necessárias informações sobre os serviços de adaptação disponíveis. Neste caso, além das características dos serviços oferecidos, devem ser incluídas informações sobre comunicação (e.g., protocolos, endereçamento) e as condições para a execução de determinada adaptação. Essas condições são baseadas em regras e utilizadas pela política de adaptação. Os perfis de serviço são descritos em *OWL-S.*

4.1.1 Perfis de dispositivo, usuário, rede, conteúdo, *SLA* e Servidores de Adaptação.

O perfil de dispositivo contém características de hardware e software do dispositivo. O conhecimento das capacidades de um dispositivo orienta o processo de adaptação, a fim de que apenas as adaptações necessárias sejam aplicadas ao conteúdo (e.g., remoção de som de um conteúdo requisitado por um dispositivo incapaz de reproduzir som). O campo *UserAgent*, do cabeçalho *HTTP*, é empregado para determinar o dispositivo de acesso utilizado pelo usuário, permitindo assim consultar o perfil desse dispositivo que está armazenado na base de dados. A Figura 29 apresenta um dos modelos de ontologia, baseado no *EMF Ontology Definition Metamodel* (*EODM*) [83], para a descrição de perfis de dispositivo. De modo a apresentar as vantagens do uso de ontologias para a descrição de perfis, o modelo permite a inserção de restrições que auxiliam na consistência e validação do perfil. Por exemplo, a restrição *Supported_ImageRestriction* define que a classe *Supported_Image* só possa ser instanciada com os indivíduos declarados na classe enumerada *Image_Format*.

Atualmente os padrões de perfis de dispositivos móveis mais adotados pelos fabricantes são o *UAPROF* [9] e o *CC/PP* [33], sendo que ambos deveriam produzir especificações equivalentes, já que o *UAPROF* seria um vocabulário específico com base na estrutura do *CC/PP* [36]. Infelizmente os desenvolvedores do *UAPROF* acabaram criando um novo esquema (www.openmobilealliance.org/tech/profiles/UAPROF/ccppschema-20030226), que redefiniu alguns elementos do *CC/PP* (www.w3.org/2002/11/08-ccpp-schema). Por esse motivo a equivalência lógica entre elementos das diferentes especificações não pode ser reconhecida no nível de *RDF*. Para resolver esse problema, o uso de ontologias, baseadas em *OWL*, permite o mapeamento dos elementos de diferentes padrões, possibilitando o reuso de perfis que utilizem essas especificações.

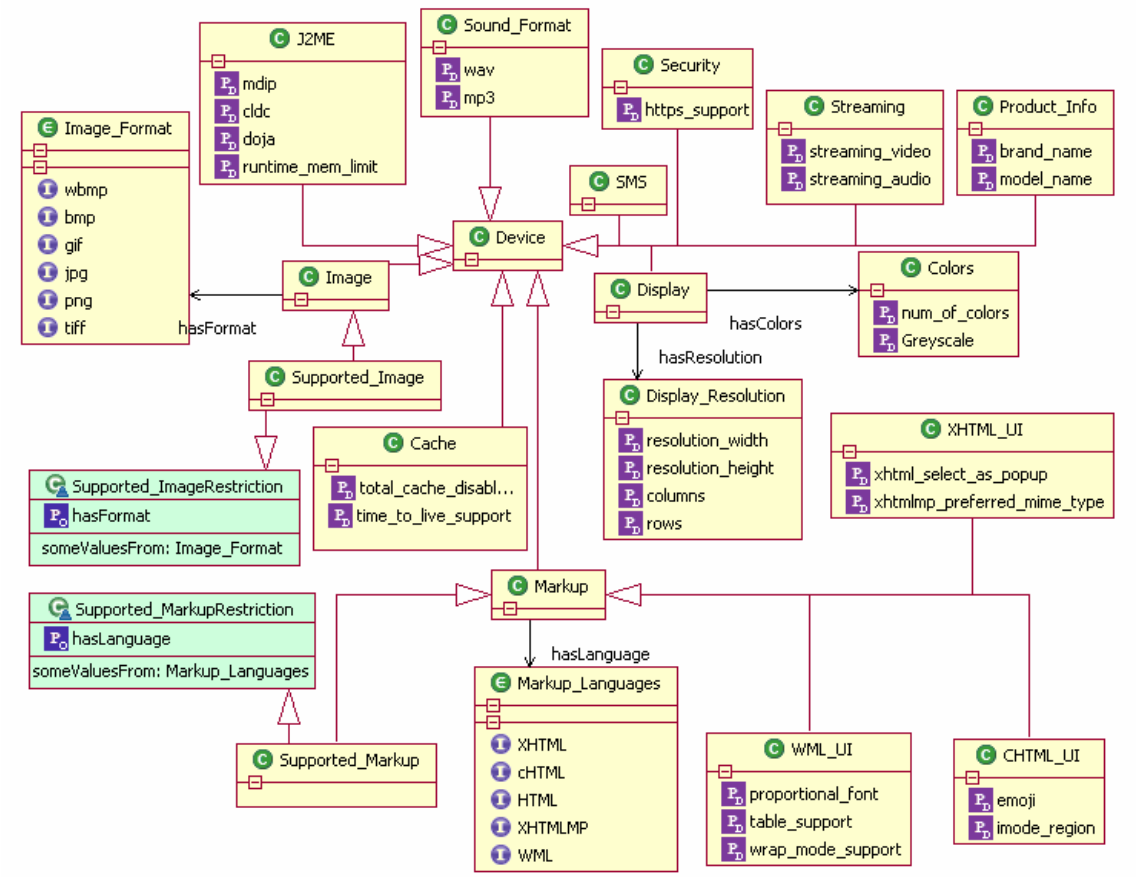


Figura 29. Modelo de ontologia para a descrição de perfis de dispositivo.

O perfil de usuário contém informações pessoais deste e de suas preferências de adaptação de conteúdo. Diferentes usuários podem desejar que diferentes adaptações sejam aplicadas a um conteúdo requerido (e.g., enquanto um usuário tem preferência pela redução da resolução de imagens, um outro pode preferir pela redução do número de cores). Adaptações, não baseadas nas preferências dos usuários, podem ser inconvenientes ou até mesmo indesejadas.

A Figura 30 apresenta um dos modelos de ontologia para a descrição de perfis de usuário. Esse modelo representa a classe *USER* com as propriedades *ID*, que é usada para recuperar as informações do usuário na base de dados, e *Name,* que identifica o nome do usuário. Nas subclasses de *Services* são descritas as propriedades dos serviços de adaptação, que podem ser configuradas pelo usuário de acordo com as suas preferências. Para exemplificar, na classe *Classification_and_Filtering* há a propriedade *Filter_Profile*, onde são definidos quais tipos de conteúdo serão bloqueados (e.g., sexo, shopping, jogos, violência).

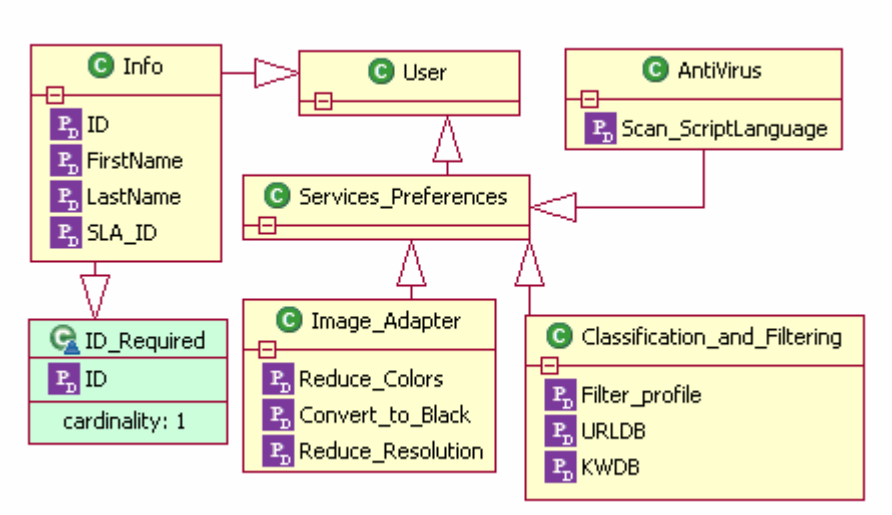


Figura 30. Modelo de ontologia para descrição de perfis de usuário.

O perfil de rede é obtido dinamicamente através de agentes, que monitoram parâmetros da rede de comunicação entre o provedor e o usuário. As

informações contidas nesse perfil orientam alguns processos de adaptação (e.g., imagens, vídeo e áudio sob demanda), a fim de que o conteúdo adaptado esteja otimizado para as condições da rede num determinado momento. De modo a evitar que breves oscilações na rede interfiram o processo de tomada de decisão, as informações deste perfil não são geradas a partir de dados absolutos, e sim por um histórico de comportamento da rede.

A Figura 31 apresenta um dos modelos de ontologia para a descrição de perfis de rede. Esse modelo representa a classe *General* com as propriedades *connection_time,* que armazena informação sobre o tempo de conexão, e *address*, que armazena o endereço *IP* do dispositivo de acesso. A classe *QoS* (*Quality of Service*), uma subclasse de *Network*, possui propriedades relativas à rede de acesso, tais como atraso (*delay*), a velocidade efetiva de transferência de dados (*throughput*) e a confiabilidade de transmissão (*frame_error_rate* e *Bit Error Rate* (*BER*)). Essas orientam alguns processos de adaptação (e.g., imagens, vídeo e áudio sob demanda), a fim de que o conteúdo adaptado esteja otimizado para as condições da rede.

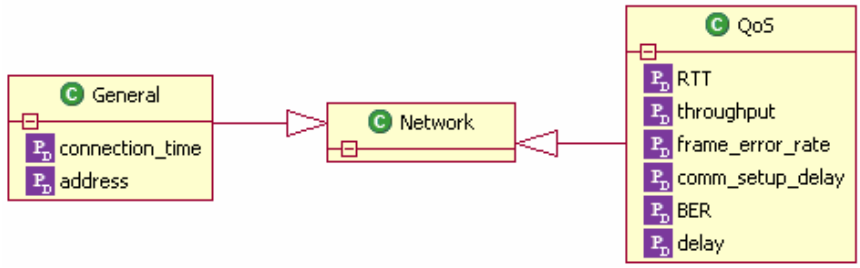


Figura 31. Modelo de ontologia para descrição de perfis de rede.

O perfil de conteúdo também é obtido dinamicamente, sendo baseado em características do próprio conteúdo requisitado. A partir de informações extraídas do cabeçalho *HTTP* (e.g., se o conteúdo dispõe de texto e/ou imagem, idioma) e do conjunto de meta dados do conteúdo, se disponível, são determinadas as alterações

necessárias e aplicáveis ao conteúdo. A Figura 32 apresenta um dos modelos de ontologia para a descrição de perfis de conteúdo.

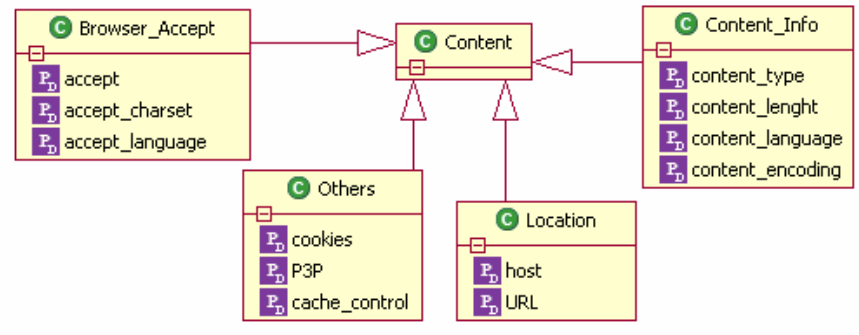


Figura 32. Modelo de ontologia para descrição de perfis de conteúdo.

No perfil *SLA* estão descritas as resoluções contratuais entre o provedor de acesso e o usuário. Atualmente esses provedores oferecem diferentes planos aos seus usuários, incluindo largura de banda, tempo de conexão e vários serviços de valor agregado, permitindo assim que os usuários escolham o plano mais adequado as suas necessidades. A Figura 33 apresenta um dos modelos de ontologia para a descrição de perfis de *SLA*. Na classe *SLA*, e suas subclasses, são descritos os serviços oferecidos pelo provedor, e quais opções de serviço foram contratadas pelo usuário. O *SLA* está relacionado ao perfil do usuário por meio do parâmetro *SLA_ID*.

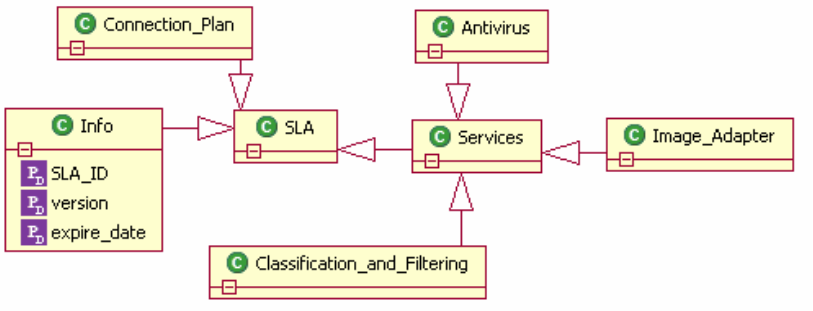


Figura 33. Modelo de ontologia para descrição de perfis de *SLA*.

O perfil de Servidores de Adaptação tem como objetivo descrever características dos servidores de adaptação facilitando a criação dos perfis de serviços. Dentre as características definidas estão as relacionadas à qualidade de serviço (*QoS*)

onde são descritas a disponibilidade do servidor (*Availability*), e a confiabilidade (*Reliability*) onde é avaliada a taxa de execuções bem sucedidas. Em conjunto com as características de tempo máximo de execução (*MaxProcessTime*), largura de banda mínima requerida (*RequiredBandwidth*) e Custo (*Cost*), essas informações auxiliam o processo de decisão caso o mecanismo de descoberta de serviços encontre mais de um provedor de serviço que satisfaça as necessidades do contexto de entrega. Também é definido nesse perfil que protocolo será utilizado para comunicação entre o *proxy* e o servidor de adaptação: *SOAP* ou *ICAP*.

A Figura 34 apresenta um dos modelos de ontologia para a descrição de perfis de Servidores de Adaptação. A classe *Supported_Execution_Points* descreve os quatro pontos para invocação de serviços de adaptação disponibilizada pela arquitetura. A invocação de serviços de adaptação pode ser realizada em vários estágios do fluxo de dados, sendo que foram especificados quatro pontos de execução junto ao *Adaptation Proxy*, conforme ilustrado na Figura 35.

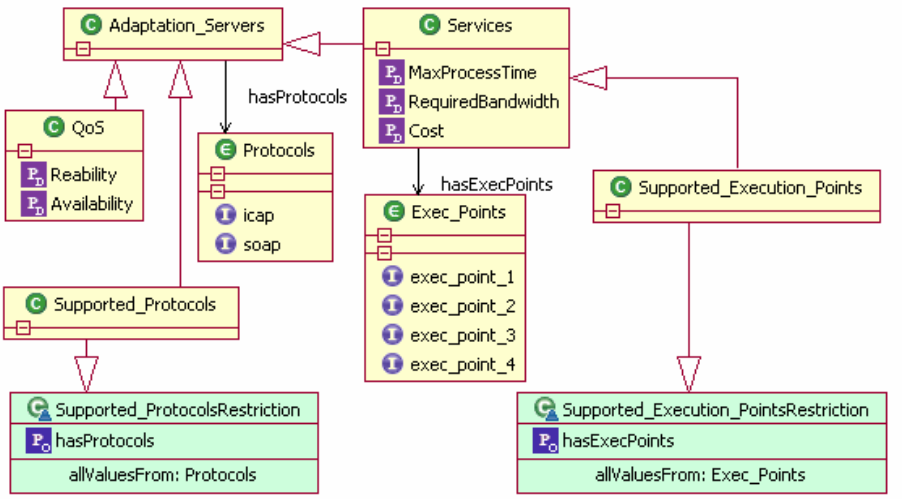


Figura 34. Modelo de ontologia para descrição de perfis de Servidores de Adaptação.

Nos pontos 1 e 2 a regra é processada sobre a requisição do usuário, respectivamente antes ou após o conteúdo ser buscado no *cache*, enquanto nos pontos 3 e 4 a regra é processada sobre a resposta do servidor de origem, respectivamente antes

ou após o conteúdo ser armazenado no *cache*. A definição do ponto de execução de cada regra depende do serviço de adaptação requerido. Por exemplo, um serviço de antivírus deve ser executado no ponto 3, evitando assim que um conteúdo contaminado seja armazenado no *cache*.

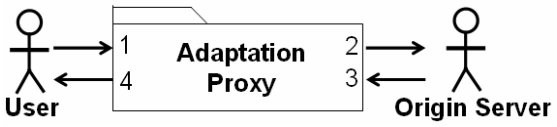


Figura 35. Pontos de invocação de serviços de adaptação.

4.1.2 Perfis de Serviços

O perfil de serviço descreve as características de um serviço de adaptação, incluindo as condições necessárias para a sua execução. Essas características podem ser divididas em duas classes: as diretamente associadas ao processo de adaptação, representadas pelas Entradas e Saídas; e as de apoio, representadas pelas Pré-condições e Efeitos, que auxiliam a decisão da política de adaptação em executar, ou não, determinado serviço, conforme ilustra a Figura 36.

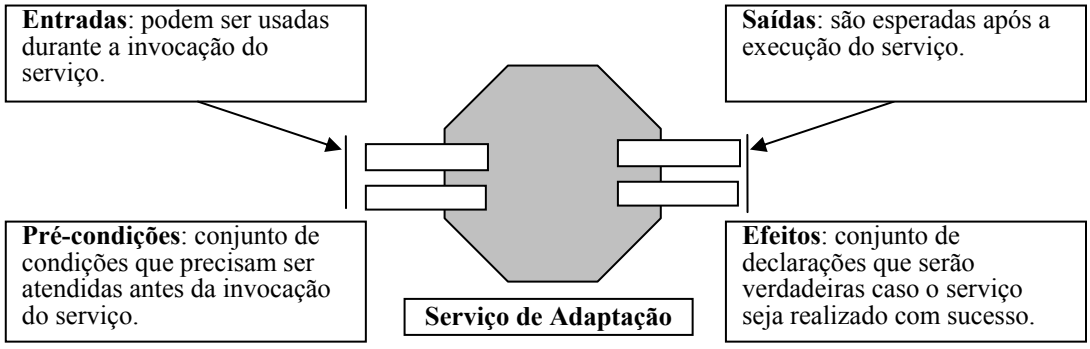


Figura 36. Características de perfis de serviço.

Existem outras funções que destacam esse perfil em relação aos outros, e que determinaram o uso do *OWL-S* para sua descrição. A principal delas, é que nesse perfil é realizado o mapeamento das descrições semânticas (*OWL*) dos perfis, com as

descrições sintáticas (*WSDL*) dos serviços *Web,* permitindo a integração de ontologias com serviços *Web.* Além disso, ele também descreve as características de comunicação (e.g., protocolos, endereços) dos serviços *Web*, facilitando o acesso remoto a esses serviços. A Figura 37 ilustra que o perfil de serviço importa descrições semânticas dos perfis, baseadas em *OWL*, e descrições de serviços *Web*, baseadas em *WSDL*.

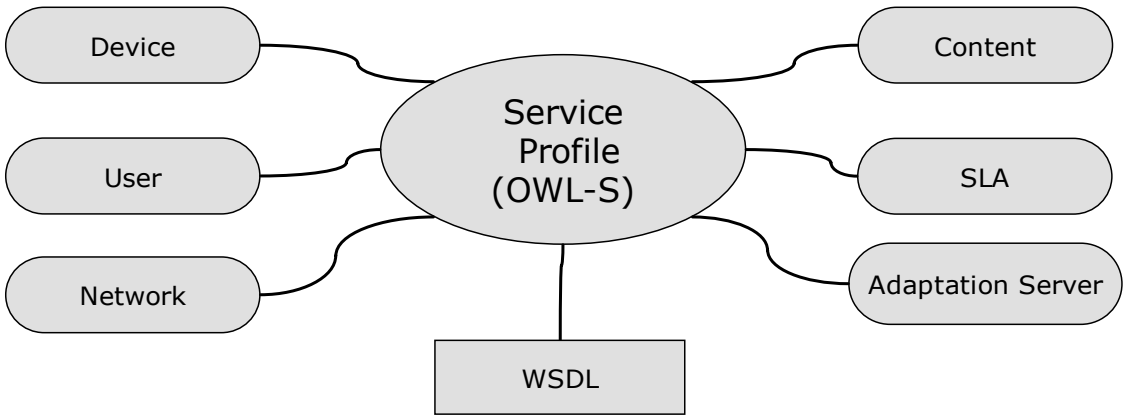


Figura 37. Descrições de perfis e serviços que são importadas pelo perfil de serviço.

A partir da combinação das informações do ambiente ubíquo, descritas nos perfis de *dispositivo*, *usuário*, *rede*, *conteúdo, SLA* e servidores de adaptação, e das informações dos perfis de serviço, são definidos os serviços necessários para a execução de uma determinada adaptação. Caso seja necessário mais de um serviço para realizar a adaptação necessária, deve-se gerenciar a ordem de suas execuções.

Para auxiliar na determinação dos serviços necessários, os perfis de serviço podem conter regras especificadas em *SWRL* [39]. Essa abordagem permite que os componentes responsáveis pela política de adaptação, sejam independentes das variações provocadas pelas mudanças e inclusões de novos serviços e regras.

Com o propósito de exemplificar a especificação, baseada em *OWL-S*, dos perfis de serviço, é apresentado na Figura 38 o esqueleto do perfil de um serviço de conversão de linguagens de marcação. Inicialmente são descritas as ontologias que serão importadas, de modo a incluir as informações semânticas já definidas (1). Nesse

caso são reutilizados: uma ontologia pública, *semwebglossary.owl,* que define termos da *Web* semântica com o propósito de evitar ambigüidade de definições; os perfis de conteúdo (*content.owl*) e dispositivo (*device.owl*).

```
...
<!ENTITY gloss "http://www.personal-reader.de/rdf/semwebglossary.owl">
<!ENTITY device "http://www.adaptationsrv.org/device.owl">
<!ENTITY content "http://www.adaptationsrv.org/content.owl">
...
<owl:Class rdf:ID="SupportedMarkupLanguage">
  <owl:oneOf rdf:parseType="Collection">
     <factbook:Language rdf:about="&glossary;#HTML"/>
     <factbook:Language rdf:about="&glossary;#XHTML"/>
     <factbook:Language rdf:about="&glossary;#XML"/>
     <factbook:Language rdf:about="&glossary;#CHTML"/>
     <factbook:Language rdf:about="&glossary;#WML"/>
  </owl:oneOf>
</owl:Class>
<owl:ObjectProperty rdf:ID="canBeConvertedTo">
  <rdfs:domain rdf:resource="#SupportedMarkupLanguage"/>
  <rdfs:range rdf:resource="#SupportedMarkupLanguage"/>
</owl:ObjectProperty>
<rdf:Description rdf:about="&glos;#HTML"><canBeConvertedTo rdf:resource="&glos;#WML"/></rdf:Description>
<rdf:Description rdf:about="&glos;#HTML"><canBeConvertedTo rdf:resource="&glos;#XHTML"/></rdf:Description>
<rdf:Description rdf:about="&glos;#HTML"><canBeConvertedTo rdf:resource="&glos;#XML"/></rdf:Description>
<rdf:Description rdf:about="&glos;#HTML"><canBeConvertedTo rdf:resource="&glos;#cHTML"/></rdf:Description>
<rdf:Description rdf:about="&glos;#XML"><canBeConvertedTo rdf:resource="&glos;#XHTML"/></rdf:Description>
...
<process:Input rdf:ID="InputMarkupLanguage">
 <process:parameterType  rdf:datatype="&xsd;#anyURI">&content;#Content_Type
 </process:parameterType>
</process:Input>
<process:Input rdf:ID="OutputMarkupLanguage">
 <process:parameterType  rdf:datatype="&xsd;#anyURI">&device;#Supported_Markup
 </process:parameterType>
</process:Input>
...
<process:AtomicProcess rdf:ID="MarkupConverterProcess">
 <process:hasInput rdf:resource="#InputMarkupLanguage"/>
 <process:hasInput rdf:resource="#OutputMarkupLanguage"/>
 <process:hasInput rdf:resource="#InputString"/>
 <process:hasOutput rdf:resource="#OutputString"/>
 <process:hasPrecondition rdf:resource="#SupportedConvertion"/>
 <process:hasEffect rdf:resource= ="#MarkupLanguageConverted"/>
</process:AtomicProcess>
...
<expr:SWRL-Condition rdf:ID="SupportedConvertion">
  <rdfs:label>canBeConvertedTo(InputMarkupLanguage, OutputMarkupLanguage)</rdfs:label>
  <expr:expressionLanguage rdf:resource="&expr;#SWRL"/>
  <expr:expressionObject>
    <swrl:AtomList>
      <rdf:first>
        <swrl:IndividualPropertyAtom>
          <swrl:propertyPredicate rdf:resource="#canBeConvertedTo"/>
          <swrl:argument1 rdf:resource="#InputMarkupLanguage"/>
          <swrl:argument2 rdf:resource="#OutputMarkupLanguage"/>
        </swrl:IndividualPropertyAtom>
      </rdf:first>
      <rdf:rest rdf:resource="&rdf;#nil"/>
    </swrl:AtomList>
  </expr:expressionObject>
</expr:SWRL-Condition>
...
```

Figura 38. Esqueleto do perfil de serviço de conversão de linguagens de marcação.

Na seqüência é definida uma classe, que descreve as linguagens de marcação suportadas (2), e uma propriedade *canBeConvertedTo* (3), que será utilizada na definição das possíveis conversões realizadas pelo serviço (4).

Em (5) é declarado o parâmetro que descreve a linguagem de marcação a ser convertida. Essa informação é obtida da classe *Content_Type* do perfil de conteúdo, que é previamente instanciada com informações sobre o conteúdo requisitado. Do mesmo modo, a linguagem de marcação desejada é obtida através do parâmetro, definido na classe Supported_*Markup* do perfil de dispositivo, que informa as linguagens suportadas (6). As entradas, as saídas, as precondições e os efeitos são definidos na descrição do processo (7). A pré-condição baseia-se em uma regra em SWRL (8), que retorna verdadeiro quando a conversão necessária está entre as definidas em (4). A seção de *Grounding*, entre outras que não são apresentadas neste exemplo por limitações de espaço, é gerada automaticamente a partir da descrição *WSDL* do serviço.

4.2 Suporte a ontologias no FACI

Para o domínio de aplicações alvo desta pesquisa, os casos de uso foram modelados e implementados no *Framework para Adaptação de Conteúdo na Internet (FACI)* [11], estendido de modo a suportar ontologias e serviços Web, aproveitando a sua estrutura básica para o desenvolvimento de diferentes aplicações de adaptação de conteúdo. Esse *framework* foi organizado em dois pacotes principais: *Adaptation Proxy*, que desempenha o papel do dispositivo de borda (e.g., proxy); e *Adaptation Server*, que desempenha o papel do módulo de serviços de adaptação. No *FACI,* os atores, usuário da Internet (*User*) e servidor de origem (*Origin Server*) interagem com o *Adaptation Proxy* através das requisições de conteúdo e suas respostas. O *Adaptation Proxy* oferece

recursos para analisar essas requisições e respostas, implementa a política de adaptação e pode realizar adaptações localmente ou solicitá-las aos *Adaptation Servers*, que as realizarão remotamente.

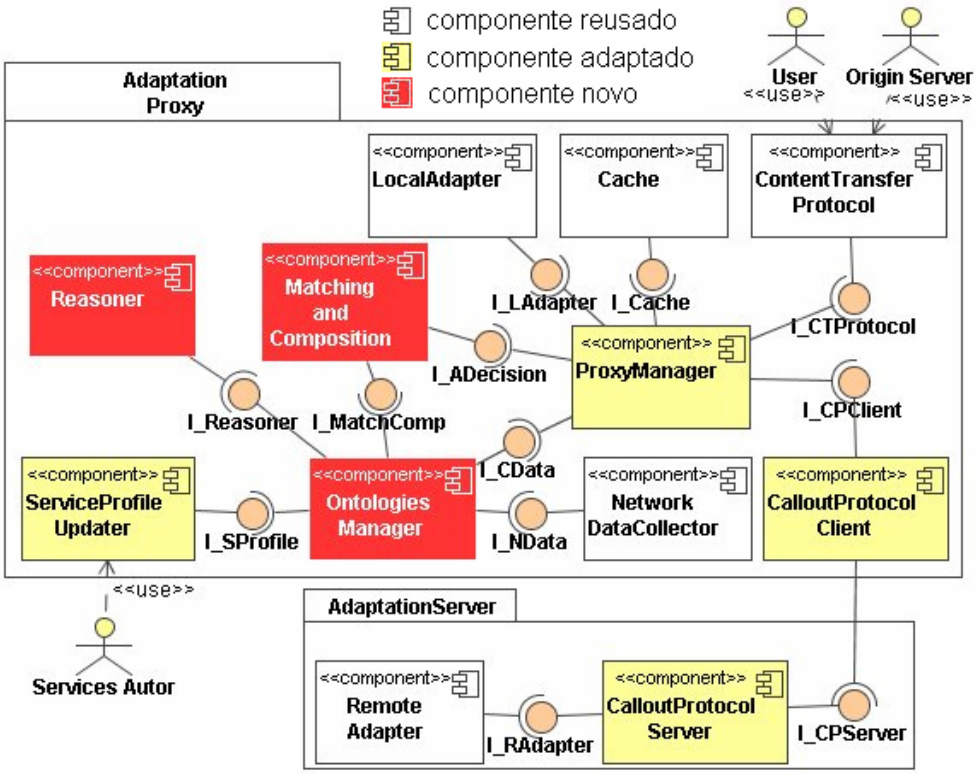


Figura 39. Componentes do *Framework para Adaptação de Conteúdo na Internet*.

No nível de componente a preocupação é com a estrutura interna do sistema para atender as suas funcionalidades. Para tal o *FACI* foi estruturado num conjunto de componentes combinados, que disponibilizam seus serviços através de suas interfaces. A Figura 39 apresenta os componentes do *FACI*, reconstruído com base em ontologias e serviços *Web*. No modelo são ressaltados os componentes adaptados e adicionados com a extensão do *FACI*.

O *Callout Protocol Client*, que é responsável pela comunicação com o *Adaptation Server*, foi adaptado com a adição do *SOAP* [30], proporcionando a compatibilidade com serviços *Web*. O *ICAP* [28] foi mantido de modo a suportar servidores já desenvolvidos. O perfil de serviço descreve qual protocolo será utilizado para a comunicação com o respectivo serviço.

Ontologies Manager, um novo componente que foi introduzido para gerenciar o armazenamento, a recuperação e o processamento das ontologias estáticas (perfis de dispositivo, usuário, *SLA,* servidores de adaptação e serviços) numa base de dados relacional. Esse componente também trata os perfis de serviços, inseridos através do *Service Profile Updater*. As ontologias dinâmicas, armazenadas em memória, são geradas sob demanda a partir das informações obtidas pela interface *I_NData*, para a geração do perfil de rede, e *I_CData*, para a geração do perfil de conteúdo. A interface *I_CData* foi adicionada a partir da adaptação do componente ProxyManager.

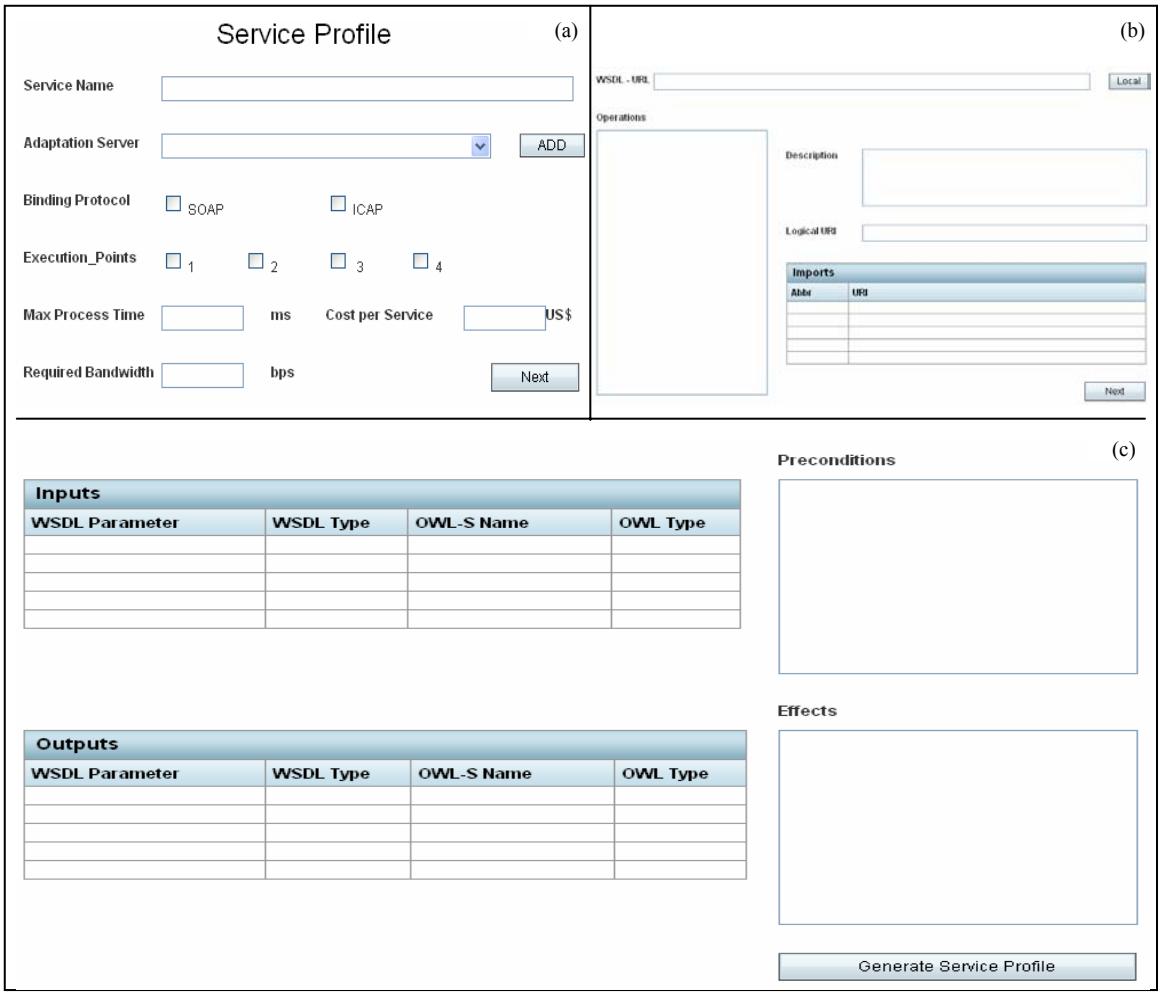


Figura 40. Páginas da interface *Web* para inserção de perfis de serviços.

De modo a facilitar a inserção de novos perfis de serviço, o *Service Profile Updater* disponibiliza uma interface *Web* para a inserção dos perfis de serviços,

onde o provedor de serviço deverá inserir informações sobre os serviços de adaptação oferecidos, que serão convertidos para a especificação *OWL-S*. A Figura 40 ilustra algumas páginas desta interface. Na página inicial (a) são inseridas informações gerais sobre o serviço incluindo: o protocolo utilizado (*SOAP* ou *ICAP*), os pontos de execução onde o serviço poderá ser invocado, e outros requisitos do serviço. Caso o serviço utilize *SOAP* para a comunicação entre o *proxy* e o servidor de adaptação, ao clicar *Next* a página (b) será carregada. Nessa página a partir da inserção da *URI* da descrição *WSDL* do serviço, cujas informações serão convertidas em um modulo Grounding da *OWL-S*, são listadas as operações disponíveis. Nessa tela também são inseridas informações sobre a *URI* do serviço e se define quais ontologias (incluindo os perfis) serão importadas. Ao se clicar novamente *Next* o autor de serviços irá completar a tabela de mapeamento entre os parâmetros *WSDL* (que serão inseridos automaticamente a partir das informações obtidas do arquivo *WSDL*) e as descrições semânticas *OWL* (c). Nas caixas de texto Preconditions e Effects deverão ser inseridas as regras especificadas em *SWRL*. Ao finalizar a inserção das informações é gerado o perfil de serviço ao se clicar em *Generate Service Profile*

Matching and Composition, outro componente adicionado, realiza a função da política de adaptação no *Framework* a partir do cruzamento das informações do contexto de entrega com as características dos perfis de serviços. Para executar essa tarefa o algoritmo de *matching,* auxiliado pelo *reasoner,* procura por serviços que possuam entradas e saídas compatíveis com o contexto de entrega, que é descrito pelas informações armazenadas nos perfis. Os resultados obtidos passam por uma segunda filtragem que verifica se as pré-condições e os efeitos também são compatíveis. Caso o algoritmo de matching não encontre um único serviço capaz de realizar a adaptação necessária, será invocado o algoritmo de composição. Esse algoritmo, juntamente com o

de *matching*, tentará compor um conjunto de serviços que satisfaça a adaptação necessária. O gerenciamento da ordem de execução dos serviços de adaptação, no caso de uma composição de serviços, também será realizado nesse componente.

Através da interface *I_ADecision* são enviadas as informações, sobre o serviço de adaptação a ser invocado, ao *Proxy Manager* que irá, através do *Callout Protocol Client,* requisitar o serviço ao respectivo *Adaptation Server***.** A invocação de serviços é executada recursivamente até que o último serviço da composição seja realizado, liberando o *Proxy Manager* para o envio da resposta adaptada ao *User*.

No pacote *Adaptation Server*, o componente *Callout Protocol Server* foi adaptado, para suportar a tecnologia de serviços *Web, com a adição do SOAP*. Porém, com as alterações feitas no *Adaptation Proxy*, qualquer servidor de aplicações compatível com os padrões de serviços *Web* pode assumir o papel de Adaptation Server.

Para que se tenha uma melhor compreensão do emprego de ontologias e serviços *Web* na política de adaptação, a Tabela 6 apresenta um contexto de adaptação com perfis de uma aplicação, utilizando a notação perfil.classe, onde um usuário acessa a *Web* via celular. Quando o usuário conecta-se a *Web*, o provedor de acesso envia seu *User_ID*, conjuntamente com a requisição, para o *Adaptation Proxy*. A partir do *User_ID* o *Ontologies Manager* busca, na sua base de dados, os perfis de usuário e *SLA*. Da requisição do usuário é extraído o *User Agent (UA)*, que identifica o modelo do dispositivo de acesso, possibilitando assim a busca do perfil de dispositivo. O perfil de rede é criado dinamicamente a partir das informações obtidas pelo *Network Data Collector* e o perfil de conteúdo é gerado a partir de informações extraídas dos campos do cabeçalho da mensagem de resposta. Neste exemplo o usuário contratou o serviço pago de classificação e filtragem de conteúdo, tendo a propriedade *Pay_for_Filtering*, do perfil *SLA*, configurada como *true*.

User.Filter_Profile – 001	**Network**.WAN – GPRS
SLA.Services – Pay_for_Filtering	**Network**.RTT – 10
Device.DisplayResolution – 320x200	**Content**.Type – HTML
Device.Supported_Markup – XHTML	**Content**.URL – www.test.com

Tabela 6. Exemplo de um contexto de adaptação.

Uma vez coletadas as informações dos perfis, é definido o objetivo, *O = (HTML, XHTML, Pay_for_Filtering*, _), e são iniciadas as buscas de serviços para a realização das adaptações necessárias. O componente *Matching and Composition* descobre que o serviço de classificação e filtragem de conteúdo, S_x = *(HTML, HTML, Pay_for_Filtering*, _), atende às entradas e pré-condições, mas não suporta a saída em *XHTML*. Como todos objetivos pretendidos não foram atingidos, é criado um novo objetivo, *O' = (HTML, XHTML*, _ , _), sendo que em uma nova busca o serviço de tradução de linguagem de marcação, S_y = *(HTML, XHTML*, _ , _), atendeu aos requisitos e a composição é concluída.

4.2.1 Alterações na implementação do FACI

Várias alterações foram realizadas no FACI para prover as funcionalidades propostas neste trabalho. Considerando que o framework FACI é do tipo caixa branca, o seu reuso é provido por herança, ou seja, a partir das classes abstratas contidas no framework são criadas subclasses especializadas. Essa seção descreve como foram implementadas essas funcionalidades e quais ferramentas e *API's* foram utilizadas.

Para adaptar os componentes *Callout Protocol Client* e *Callout Protocol Server*, foi adicionado o suporte ao protocolo *SOAP* através da criação de uma classe

que invoca a implementação desse protocolo na *API Apache AXIS*. Como esse protocolo é utilizado na comunicação cliente-servidor de serviços *Web*, qualquer servidor de aplicações compatível com essa tecnologia pode substituir o *Adaptation Server* (e.g., *Tomcat* + *AXIS*).

Nos componentes *Ontologies Manager* e *Matching and Composition* foram utilizadas as *API´s OWL-S* [84] e *JENA* [85]. A primeira possui métodos para o tratamento de ontologias baseadas em *OWL-S, que* descreve o perfil de serviços, e também provê suporte para a linguagem de regras *SWRL*, o *matching* e a composição de serviços. A segunda oferece suporte a ontologias *OWL-DL* e a persistência de dados onde foi utilizado o banco de dados relacional *MySQL. O Ontologies Manager* também faz interface com o componente *Reasoner* onde foi usada a *API Pellet* [86], escolha que foi motivada pela sua integração com a *API OWL-S*.

O componente *Service Profile Update* foi alterado de modo a fornecer uma interface *Web* para a inserção de novos perfis de serviço. Para implementar essa funcionalidade foram utilizados *Servlets* para implementação da lógica e *Java Server Pages* para o desenvolvimento das páginas da interface. Os *Servlets* invocam métodos da classe *WSDL2OWL* da *API OWL-S* quando for necessária a conversão de descrições de serviços Web baseada em *WSDL,* para um perfil de serviços baseado em *OWL-S*. Outros métodos da *API OWL-S* também são invocados durante a criação do perfil de serviços.

Durante este trabalho foi utilizada a plataforma de desenvolvimento *Eclipse*, juntamente com os plug-ins: *Web Tools Platform (WTP)* para o suporte a serviços Web, e o *IBM Integrated Ontology Development Toolkit* para a modelagem e validação das ontologias. Também foi utilizado o editor *Protégé* com o plug-in para a linguagem *OWL-S* para o desenvolvimento e validação dos perfis de serviço.

4.2.2 Seqüência de Adaptação de Conteúdo

Para exemplificar a utilização de perfis e regras, é apresentada na Figura 41 a seqüência de uma adaptação de conteúdo. A partir de uma requisição *HTTP* ou *WAP* do usuário (1), o provedor de acesso (*ISP*) envia ao *proxy* de adaptação a requisição *HTTP* juntamente com a identificação do usuário (2).

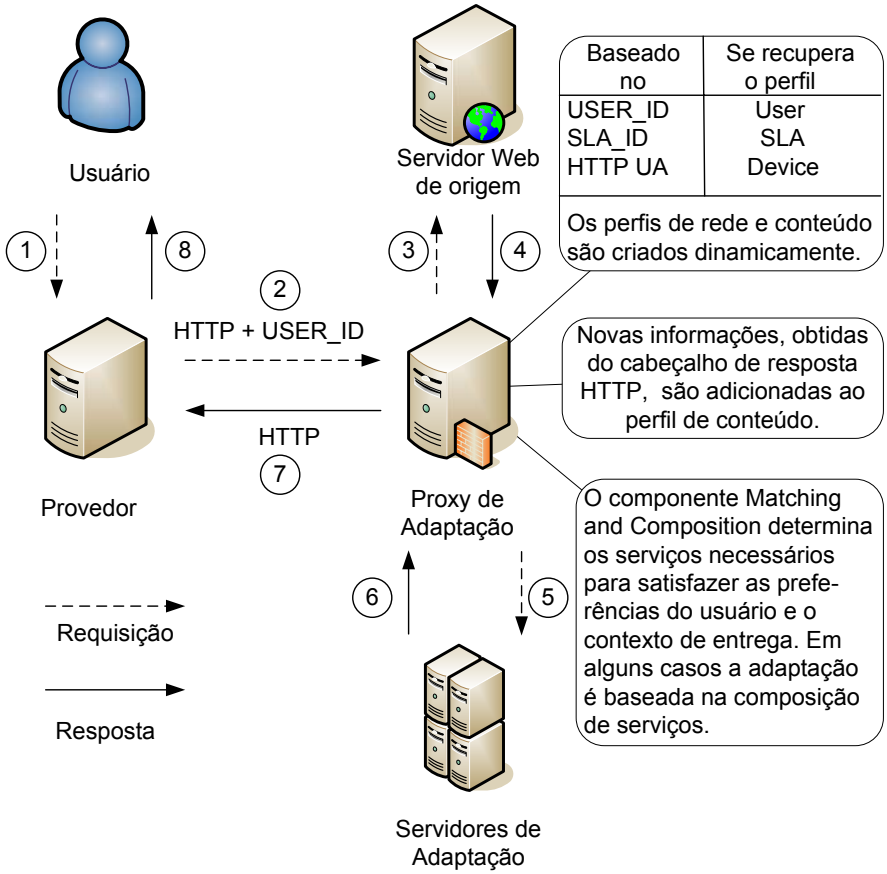


Figura 41. Seqüência de uma adaptação de conteúdo.

O *Ontology Manager* obtém de sua base de dados os perfis de usuário (baseado no *USER_ID*), *SLA* (baseado no *SLA_ID* do perfil do usuário) e o perfil de dispositivo, baseado no *User_Agent* (*UA*) obtido pela interface *I_CData* (que extraiu essa informação do cabeçalho de requisição HTTP). Nesse ponto já é possível verificar se existem serviços de adaptação a serem invocados (5)(6) nos pontos de execução 1 e

HTTP ou WAP

2. Caso o *proxy*, não tendo localizado o conteúdo solicitado em seu *cache*, é enviada uma requisição ao servidor *Web* de origem (3) e, ao receber a resposta (4), novas informações são adicionadas ao perfil de conteúdo. Caso seja necessária adaptação de conteúdo nos pontos de execução 3 e 4, outros serviços de adaptação poderão ser invocados (5)(6). Ao termino das adaptações necessárias o conteúdo adaptado é enviado ao provedor de acesso, via HTTP (7), que por sua vez o encaminha ao usuário (8).

4.3 Trabalhos Relacionados

O uso de ontologias, para a descrição de contexto e de serviços de adaptação, é uma linha de pesquisa recente, devendo receber um grande impulso quando novas especificações dos padrões, que suportam as tecnologias de Serviços *Web* (e.g., *UDDI*, *WSDL*), proverem suporte nativo para as descrições semânticas. Além desse tópico principal, esta dissertação também aborda tópicos sobre descoberta de serviço (*matching*) e componentes. Dentre esses três tópicos destacam-se os seguintes trabalhos correlatos.

Inicialmente a descrição de contexto utilizava padrões baseados em RDF (e.g., *UAPROF e CC/PP*). As fragilidades desses padrões foram abordadas em [87] onde se destaca a incompatibilidade entre múltiplas versões e os problemas de ambigüidade quando se trabalha com múltiplos vocabulários. Tratando do mesmo tema [88] apresenta as características necessárias para se representar informação de contexto, e faz uma abordagem crítica sobre as propostas para descrição de contexto, incluindo *CC/PP*, *UAPROF* e *Internet Engineering Task Force (IETF) Media Features Sets*. Por fim, é proposta uma nova abordagem, baseada em *RDF*, chamada *Comprehensive Structured Context Profiles* (*CSCP*) que, segundo o autor, possui uma melhor estrutura

para descrição de perfis complexos. [89] apresenta uma abordagem para a criação de perfis baseados em *CC/PP* para a adaptação de conteúdo, mas não demonstra como esses perfis são utilizados para a tomada de decisão, e como os serviços de adaptação são descritos. [90], trata da especificação formal do processo de adaptação e da utilização de *proxies* de adaptação de conteúdo, que incorporam o protocolo *ICAP*, mas que não empregam ontologias.

Em 2003 foi apresentado um dos primeiro trabalhos a propor o uso de ontologias para descrição de contexto [80]. Nele é proposto um conjunto de ontologias para descrição de ambientes de computação pervasiva, sensível a contexto, que é integrado a uma arquitetura denominada *Context Broker Architecture* (*CoBrA*). Em [91] são apresentadas ontologias para descrição de características de rede, incluindo *QoS*, e para interação entre agentes de software que operem em ambientes com comunicação sem fio. Apesar da definição de várias ontologias de contexto ambos os artigos não demonstram como são descritas as características dos dispositivos e como esses perfis são utilizados na política de adaptação.

Com a proposta de uma ontologia para descrição de serviços *Web* (*OWL-S)* viabilizou-se a integração da tecnologia de serviços *Web,* com a descrição de contexto e adaptação de conteúdo. Em [92] são apresentados comparativos entre a utilização do *UDDI* sintático e o *UDDI* semântico que trabalha em conjunto com o *OWL-S*, demonstrando como é integrado o suporte semântico a descoberta de serviços. Um ponto interessante do artigo é a definição das cinco fases do processo de inserção de novas ontologias. Também em [93] são apresentados os problemas de ambigüidade relativos à descoberta por palavra chave do *UDDI* e destaca as vantagens do uso de ontologias e *OWL-S*. [94] faz uma analise para verificar se as linguagens para descrição de serviços *Web* são adequadas para o contexto da computação móvel. A partir dessa

análise são recomendadas algumas adições de classes *OWL* na estrutura do *OWL-S*, essa nova especificação é chamada de *MobiOWL-S*. Em [95] é apresentada uma solução baseada em *OWL-S* e *WSDL*, para adaptação de conteúdo, com foco no desenvolvimento de um módulo que substitui o *UDDI*, visando ganhos de desempenho. Em [72] é proposta uma *API*, que proporciona um mapeamento entre as informações de registro da *UDDI* e as descrições semânticas do *OWL-S*. Apesar das vantagens proporcionadas por essa *API* que permite o uso, de semântica pelos servidores *UDDI* distribuídos pela Internet, o estudo de caso apresenta valores de tempo de resposta que inviabilizam, no atual estágio, o seu uso no domínio da adaptação de conteúdo. Em [96] é proposto um modelo, baseado em redes de tarefa hierárquica, para a composição e a descoberta de serviços, e uma nova especificação para a descrição de regras muito similar às definidas pelo *SWRL* e *OWL-S*.

Em [97] é apresentada uma abordagem para o gerenciamento de informações de contexto baseada em componentes. O destaque dessa proposta é que o próprio sistema se adapta ao contexto ativando e desativando módulos de acordo com a necessidade. O artigo [98] propõe uma ontologia genérica para troca de informações entre agentes de serviços *Web*, utilizando *OWL-S* para descrição desses serviços. Adicionalmente é proposto um agente *matchmaker* que age como um coordenador entre o agente de busca e os agentes provedores de serviço. A solução é baseada no *DECAF* (*Distributed, Environment-Centered Agent Framework*). Em ambos os artigos não se descreve como é especificado o contexto de entrega e como essa informação de contexto é utilizada pela arquitetura.

Apesar da qualidade desses trabalhos correlatos, a pesquisa apresentada nesta dissertação destaca-se por apresentar uma solução, de ponta a ponta, para a adaptação de conteúdo na Internet. Inicia pela utilização de ontologias para a descrição

de perfis, proporcionando uma definição mais precisa do contexto, que resulta em uma busca de serviços de adaptação mais precisa. O uso de serviços *Web* simplifica o desenvolvimento de servidores de adaptação e facilita a migração da solução proposta para os futuros padrões da *Web* semântica. Ambos, ontologias e serviços *Web*, foram incorporados a uma abordagem baseada em um *framework* de componentes. A sua característica modular oferece uma solução flexível, previamente testada, para o domínio de adaptação de conteúdo na Internet, facilitando assim a reutilização e a manutenção das aplicações.

4.4. Conclusão

Para a especificação dos perfis deste trabalho foram pesquisadas as principais características de um contexto de entrega. Como referência, várias propostas de descrição de contexto foram avaliadas (e.g., *UAPROF*, *WURFL*) concentrando-se nas qualidades e defeitos de cada uma. A partir dessas informações foram especificados os vários perfis descritos neste capítulo.

A base em ontologias possibilita a extensão do vocabulário dos perfis, possibilitando que novas características sejam facilmente adicionadas. Além disso, as bases de perfis descritos por outras especificações podem ser mapeadas para os perfis deste trabalho, proporcionando o reuso dessas bases.

O perfil de serviço é o principal destaque desta proposta. A sua principal função é integrar as descrições de contexto, especificadas nos outros perfis, e as regras de adaptação, facilitando a execução da política de adaptação. Essa integração também evita que a criação de novas regras e/ou características de contexto demandem alterações no código do *proxy* de adaptação. A escolha da *OWL-S* para especificar esse

perfil proporcionou o suporte a serviços *Web* por meio do mapeamento entre o perfil de serviços e o *WSDL*. A *OWL-S* também possui características que auxiliam o processo de composição de serviços.

A utilização do FACI, que fornece uma estrutura básica para o desenvolvimento de diferentes aplicações de adaptação de conteúdo, viabilizou uma rápida implementação do protótipo para testes. A sua estrutura baseada em componentes foi alterada através do reuso, adaptação e criação de componentes para suportar novas funcionalidades. Entre essas funcionalidades pode-se destacar o suporte a ontologias, o mecanismo de raciocínio, a descoberta e composição de serviços, e a interface para inserção de perfis de serviço.

Apesar de já existirem vários trabalhos relacionados à descrição de contexto de entrega, incluindo uso de ontologias para o domínio da adaptação de conteúdo, este trabalho apresenta contribuições que o diferencia dos demais. Nenhum outro trabalho relacionado descreveu tantas características do contexto de entrega, abrangendo características do dispositivo, usuário, rede de acesso, conteúdo, resoluções contratuais, servidores de adaptação e de serviço. Vários exemplos ilustraram a relação das regras com o perfil de serviços, o funcionamento do mecanismo de composição de serviços, e como é realizada uma adaptação de conteúdo. Também foi proposta inserção de serviços *Web* no domínio da adaptação de conteúdo.

5 Estudo de Caso

Para poder avaliar a proposta desta dissertação foi desenvolvido um servidor de adaptação de conteúdo que ira exercer o papel de *Adaptation Server* na nova implementação do FACI. Dentre os possíveis serviços de adaptação de conteúdo, foi escolhido o serviço de classificação e filtragem de conteúdo.

A pesquisa e desenvolvimento de ferramentas de classificação e filtragem de conteúdo tiveram um grande impulso nos últimos 5 anos. Não só pelo crescimento da demanda no ambiente corporativo, mas também pelo investimento governamental na área. Em 1999 a União Européia lançou o *Safe Internet Action Plan (SIAP)* [99] que tem como base o desenvolvimento de ferramentas de classificação e filtragem de conteúdo baseadas na cultura local.

Um dos diferenciais do servidor de adaptação desenvolvido é a sua capacidade de utilizar informações obtidas dos perfis para personalizar a adaptação do conteúdo requisitado. Demonstrando que o uso dos perfis não está limitado à política de adaptação.

Baseado nesse servidor de adaptação foram realizados testes de eficiência do serviço de classificação de conteúdo, em um cenário real de acesso a páginas *Web*. Para verificar o impacto do uso de ontologias no tempo de resposta do FACI foram realizados testes de carga nesse *framework*, cuja metodologia e resultados são apresentados na seção 5.3.

5.1 Serviço de Filtragem e Adaptação de Conteúdo

Como o acesso à informação disponível na *Web* tornou-se um fator competitivo primordial, empresas e escolas passaram a disponibilizá-lo aos seus empregados e alunos respectivamente. Graças à convergência das tecnologias de computadores, comunicação e eletrônica de consumo esse acesso passou também a ser realizado via uma grande variedade de dispositivos móveis (e.g., celulares, computadores de mão). Embora esse acesso ubíquo à Internet seja uma fonte para benefícios inegáveis, este pode ser também uma fonte para a distração dos empregados de suas tarefas profissionais e pode disponibilizar conteúdos inapropriados e/ou ofensivos, o que gera a necessidade do seu controle.

Para controlar o acesso ao conteúdo indesejado foram desenvolvidos sistemas para classificação e filtragem de conteúdo. Como base para o estudo desses sistemas é necessária a definição de três termos inter-relacionados [100]:

- *rotular* (*labelling*) é o processo que visa descrever um conteúdo associado a um rótulo, sem que seja necessário ao usuário abrir o recipiente para examinar esse conteúdo. Esse rótulo pode ser gerado pelo próprio criador do conteúdo ou por um terceiro;
- *classificar* (*rating*) é o processo que visa atribuir valores a um conteúdo baseado em certas suposições/critérios. Caso o conteúdo disponha de um rótulo, esse já possui uma pré-qualificação que pode ser (ou não) aceita pelo filtro;
- *filtrar* (*filtering*) é o processo que visa bloquear (*blocking*) o acesso a um conteúdo a partir da comparação da classificação deste com as definições de conteúdo indesejado pelo sistema.

É importante ressaltar que a classificação e filtragem de conteúdo não se restringem apenas a conteúdos ilegais (e.g., racismos, apologia à violência, pedofilia) ou inapropriados (e.g., pornografia), mas também a conteúdos indesejados numa corporação (e.g., *shopping*, *chats*, *blogs*). Entre as vantagens de se implantar este serviço num ambiente corporativo destacam-se [101]: proteção contra a exposição a conteúdo inapropriado, ofensivo ou ilegal que pode levar a uma responsabilidade legal; garantia de obediência às políticas internas de trabalho e sustentação de um ambiente positivo de trabalho; aumento de produtividade, preservação da capacidade de fluxo de dados da rede e melhoria do tempo de resposta, na medida em que restringe o acesso à Internet a conteúdos relativos ao trabalho.

5.1.1 Métodos de classificação de conteúdo

O método de classificação mais antigo e utilizado baseia-se em coleções proprietárias de *Uniform Resource Locator (URL)*, onde se associa cada *URL* a uma categoria específica de conteúdo. Quando uma página é solicitada, o classificador verifica o seu endereço no banco de dados em busca de sua categoria. Com a definição da categoria o filtro pode bloquear ou liberar o acesso ao site, de acordo com a política de uso da Internet configurada pela organização ou individuo [102]. *URLs* não localizadas no banco de dados geralmente são liberadas, sendo que os filtros podem ser configurados para bloquear o tráfego de sites não classificados.

Esses bancos são regularmente atualizados por pesquisadores, que auxiliados ou não por algoritmos de classificação revisam e categorizam manualmente cada *URL*, sendo que os usuários devem pagar uma taxa periódica para manter esse serviço ativo. Manter esses bancos de dados atualizados é um desafio para os

fornecedores de serviços de classificação e filtragem de conteúdo, uma vez que a taxa de criação de novas páginas na Internet é muito maior do que a capacidade destes em classificá-las.

Uma segunda geração de classificadores executa sob demanda a análise e classificação de todo o tráfego *Web* solicitado pelo usuário. Ao ser recebida uma página é analisada e categorizada de acordo com o seu conteúdo, sendo que em função da política de filtragem estabelecida o sistema bloqueia ou libera a página. Entre as diferentes técnicas para essa análise dinâmica de conteúdo destacam-se:

- *palavras chave*: a página tem o seu conteúdo rastreado e comparado com palavras chave pré-definidas e pré-classificadas por categoria. Quando o resultado de uma comparação é positivo a categoria da palavra chave é associada ao conteúdo rastreado. Apesar de sua fácil implementação, esse modelo leva a uma alta taxa de bloqueios indevidos de conteúdo [102];

- *análise textual*: é realizada uma análise do contexto no qual estão inseridas as palavras chave encontradas numa página. Geralmente há uma fase de aprendizado, em que o sistema é alimentado com exemplos e contra-exemplos da categoria a ser classificada, e uma fase de classificação, quando o sistema usa a base de conhecimento adquirida para classificar um novo conteúdo. Com essa técnica reduzem-se os erros de classificação nos casos em que uma palavra chave pertence a duas ou mais categorias distintas (e.g., *breast* – pornografia e medicina). As abordagens mais utilizadas são *Perceptron, Naive-Bayes, MC4, Nearest-Neighbor, Rochio Centroid e Support Vector Machine* [103];

- *rótulos*, que são lidos pelo sistema de análise e classificação e onde estão inseridas, pelo produtor de conteúdo, as características do conteúdo da página solicitada. O *W3C* criou a *Platform for Internet Content Selection (PICS)* [104], estabelecendo padrões para formatos de rótulos e métodos de distribuição. Essa plataforma possui uma parte destinada ao produtor de conteúdo *Web*, que deseja ou necessita que seu conteúdo seja visto por um público específico, e uma parte destinada aos produtores de software, que implementam sistemas de classificação baseados em *PICS*, no *browser* utilizado, softwares adicionais, ou no servidor de adaptação de conteúdo.
- *análise de imagens*, onde características genéricas das imagens (e.g., cor, textura, formato) são extraídas e comparadas com imagens pornográficas armazenadas num banco de dados. Atualmente essa técnica ainda está em maturação, consome um grande volume de processamento e apresenta um alto grau de erros de classificação [105].

Devido à complexidade da análise de conteúdo, que se agravou nos últimos anos com o aumento da diversidade de formatos nas páginas *Web*, incluindo o uso de áudio e vídeo de fluxo contínuo, o processo de classificação, independente dos algoritmos utilizados, é passível dos seguintes problemas:

- *under-blocking*, quando o filtro não bloqueia algum conteúdo indesejado. Os motivos podem ser: uma base de dados de *URL* desatualizada ou, no caso das abordagens dinâmicas, uma classificação errada do conteúdo;
- *over-blocking*, quando o filtro bloqueia indevidamente um conteúdo. Geralmente ligado à classificação dinâmica de conteúdo, ocorre sobretudo quando palavras chave são usadas sem análise de contexto. Páginas de educação sexual e de medicina são as mais afetadas por esse problema [106].

5.2 Desenvolvimento do Servidor de Adaptação de Conteúdo

O servidor de classificação e filtragem de conteúdo [107] faz parte de uma arquitetura de adaptação de conteúdo que engloba um conjunto de servidores de adaptação de conteúdo, e um *proxy* de adaptação de conteúdo que estão em desenvolvimento.

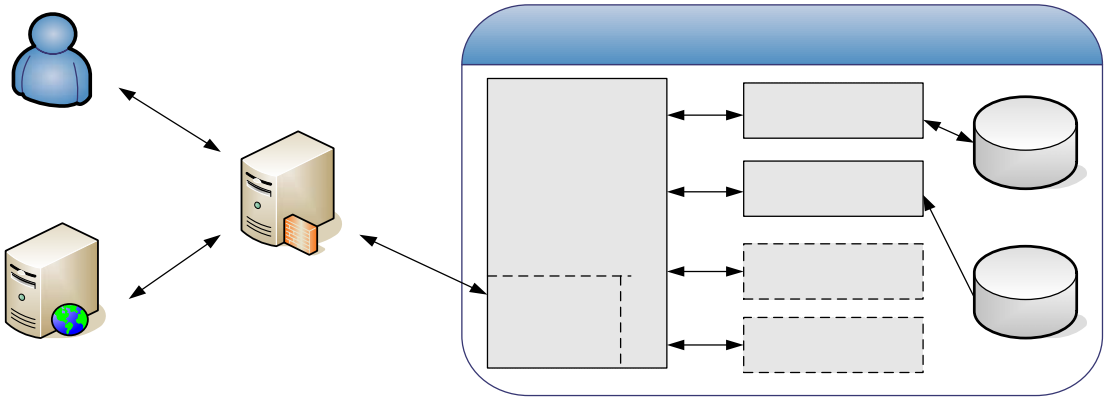


Figura 42. Arquitetura para classificação e filtragem de conteúdo

A Figura 42 ilustra a arquitetura de adaptação de conteúdo, nesse caso destacando o servidor de classificação e filtragem.

Caso a política de adaptação defina a necessidade de classificação e filtragem de conteúdo o *proxy* envia uma requisição *ICAP* ao servidor de classificação e filtragem. Para otimizar a performance e aumentar as possibilidades de utilização, o servidor pode atuar nos dois modos de operação do protocolo *ICAP*.

O servidor de classificação e filtragem de conteúdo foi projetado de forma modular, permitindo a fácil integração de novos módulos de classificação. O modulo gerenciador de classificação e filtragem gerencia todos os módulos de classificação, a comunicação com o *proxy* de adaptação de conteúdo, utilizando o *ICAP*, incluindo o *parser* dos cabeçalhos *ICAP* e *HTTP*, e filtra o conteúdo a partir das informações enviadas pelos módulos de classificação. Os módulos de classificação

PICS e de imagem não fazem parte da atual implementação podendo ser objetos de trabalhos futuros.

5.2.1 Utilização das informações dos Perfis pelos servidores de Adaptação de Conteúdo.

O uso das informações armazenadas nos perfis não se limita à política de adaptação, elas também podem ser utilizadas pelos servidores de adaptação para customizar a adaptação de acordo com o contexto de entrega e preferências do usuário. No perfil de serviço são definidas quais informações serão enviadas ao servidor de adaptação.

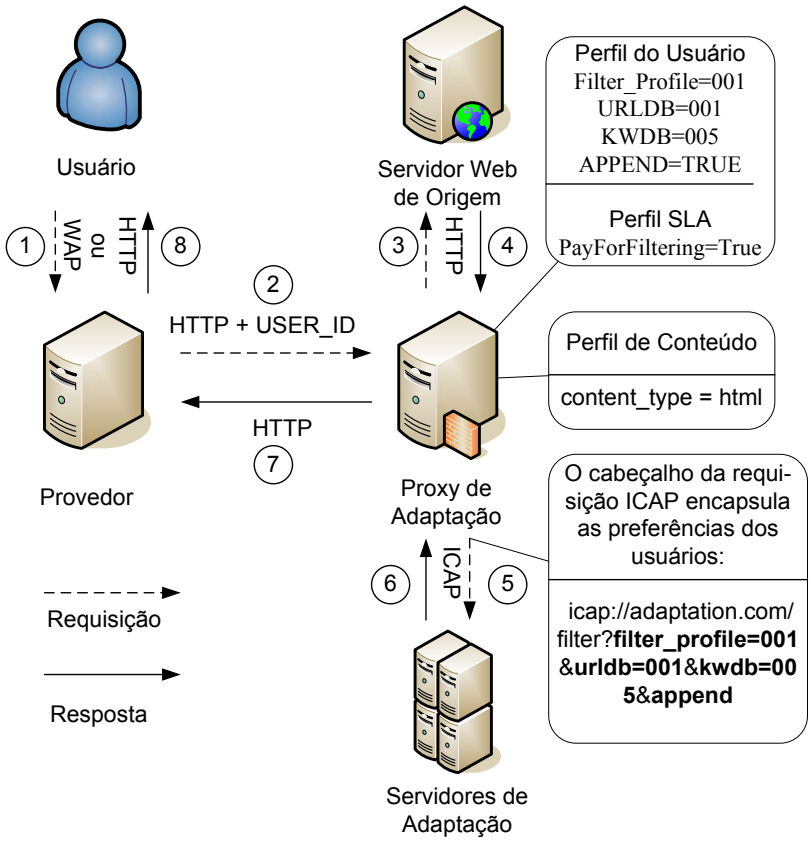


Figura 43. Seqüência de adaptação de conteúdo

A Figura 43 apresenta um exemplo de envio de informações dos perfis para o servidor de classificação e filtragem de conteúdo. Nesse exemplo o usuário

contratou do provedor de acesso o serviço de classificação e filtragem de conteúdo e definiu as suas preferências, em relação a esse serviço, que foram armazenadas no perfil do usuário. Caso o serviço seja executado (a pré-condição *PayForFiltering* é atendida) as preferências do usuário relativas a este serviço são encapsuladas no cabeçalho da requisição *ICAP* (5) (ou *SOAP*) e enviada ao servidor de adaptação.

Uma requisição *ICAP* encapsula o cabeçalho *ICAP*, o cabeçalho de requisição *HTTP*, o cabeçalho de resposta *HTTP* e o corpo da página solicitada, sendo que estes dois últimos apenas quando se opera em *respmod*. Quando essa requisição *ICAP* chega ao servidor de classificação e filtragem de conteúdo, são extraídas informações: do cabeçalho *ICAP*, que definem as categorias de conteúdo a serem bloqueadas (e.g., filter_profile=001), a bases de dados de *URLs* e domínios categorizados que será utilizada (e.g., urldb=001) e, caso a adaptação esteja ocorrendo em *respmod*, a base de dados de palavras chave (e.g., kwdb=005). O domínio (*domain*) e a *URL* da página requisitada pelo usuário são extraídos do cabeçalho *HTTP* de requisição. Caso se esteja operando em *respmod* será extraído o conteúdo requisitado (*body*) possibilitando a classificação por palavras chave. A partir dessas informações é realizado o processo de classificação e filtragem de conteúdo cujo modelo de estados é apresentado na Figura 44.

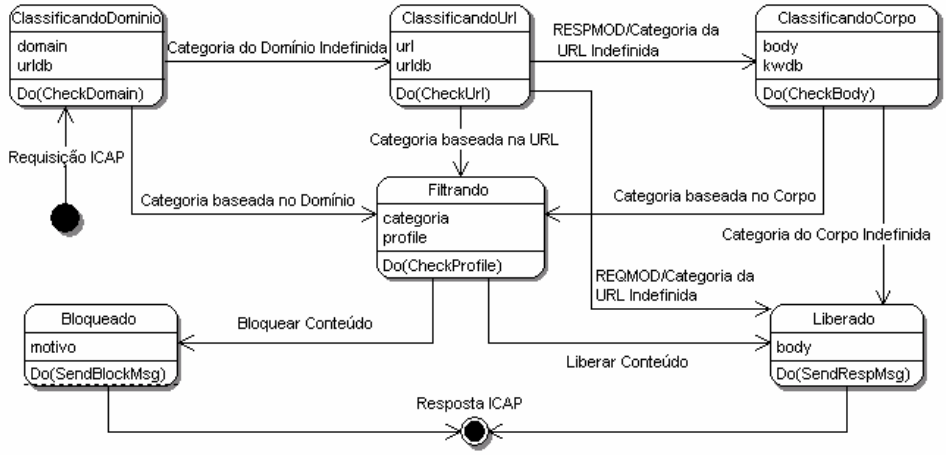


Figura 44. Modelo de estado do servidor de classificação e filtragem de conteúdo

5.2.2 Avaliação da eficiência do classificador de conteúdo.

Para realizar o teste de eficiência do classificador de conteúdo, o servidor de adaptação foi instalado em um centro universitário da grande São Paulo. Foram realizados testes em dois ambientes independentes: a rede administrativa, contendo 203 computadores; os laboratórios de informática de uma das faculdades, contendo 135 computadores. Durante um período de 48 horas foram verificadas 1215854 requisições (6,8 GB) da rede administrativa e 409428 requisições (2,8GB) da rede de laboratórios. Essas requisições foram geradas pelos próprios usuários da rede, não havendo nenhuma limitação ou controle, de modo a refletir um cenário real de utilização. De modo a causar o menor impacto possível no tempo de resposta dos acessos de funcionários e alunos, incluindo a já saturada rede da instituição, o servidor de adaptação utilizou o protocolo *ICAP* e operou apenas no modo *reqmod.* Esse modo gera um fluxo de dados menor na rede, pois somente as requisições são enviadas ao servidor de adaptação. Em contrapartida não é possível classificar o conteúdo por palavras chave, que funciona apenas em *respmod*. Os resultados obtidos são apresentados na Figura 45.

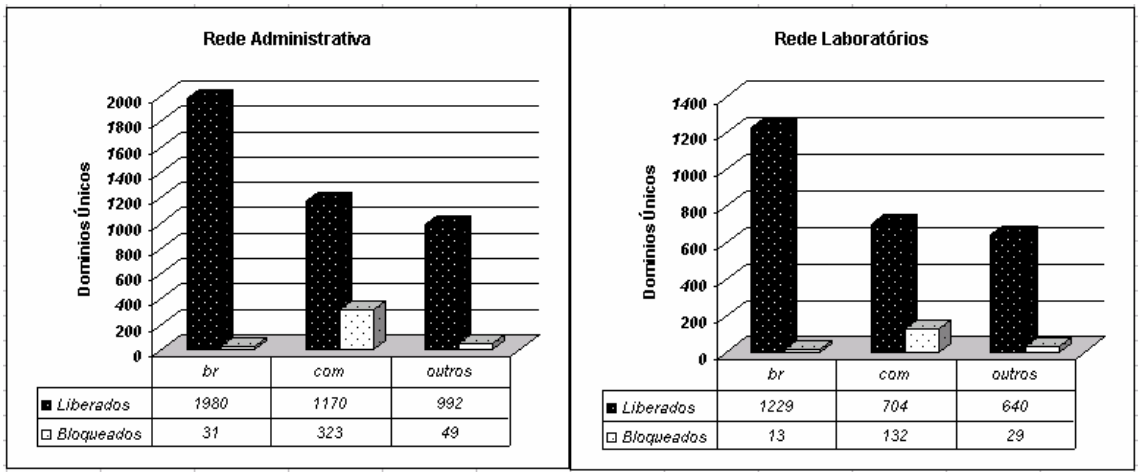


Figura 45. Resultados da Classificação e Filtragem de conteúdo do estudo de caso.

Do total de domínios classificados em ambas as redes 3 foram classificados indevidamente em categorias restritas (*over-blocking*) e 22 não foram classificados e passaram pelo filtro indevidamente (*under-blocking*). Esta verificação foi realizada manualmente, ou seja, acessando cada um dos domínios únicos. Dentre os que passaram pelo filtro, 19 pertenciam a domínios .br, o que demonstra uma menor eficiência das *blacklists* produzidas em outros países. Estes 22 domínios foram testados separadamente com o servidor de classificação e filtragem operando no modo *respmod*. Nessa nova análise de conteúdo utilizando palavras chave, 16 foram categorizados corretamente diminuindo assim sensivelmente a margem de erro.

5.3 Influência do uso de ontologias no desempenho do FACI

Para testar a performance e a capacidade de carga do FACI com suporte a ontologias e serviços *Web*, foi realizado um estudo de caso envolvendo dois computadores. Um com sistema operacional *Linux Fedora Core 2* (2Ghz – 256MB) e o outro com *Windows XP* (3Ghz – 1GB). Para evitar que alterações no tempo de resposta dos servidores de origem (incluindo variações no *throughput* da Internet) afetassem o resultado final, um servidor *Apache 2* foi instalado, empregando-se o recurso *hosts virtuais*. Isto permitiu a clonagem das páginas testadas, restringindo o fluxo de dados aos computadores envolvidos no estudo de caso. No *Linux* também foi instalado um servidor *Tomcat/Axis*, com os respectivos *remote adapters*, que exerceu o papel de *adaptation server*. Na plataforma *Windows XP* foram instalados o *adaptation proxy* com os perfis de serviços de classificação e filtragem de conteúdo e de conversão de linguagens de marcação.

A experiência consistiu em acessar 5 *links* pré-definidos durante 5 minutos. Desses *links* 3 eram bloqueados pelos seus endereços de domínio. Aleatoriamente foram atribuídos aos usuários dispositivos de acesso de mesa (e.g., *desktop*) e celulares, de modo a demandar, em alguns casos, a conversão da linguagem de marcação. Foi aplicada uma carga progressiva de usuários até o limite de 20 usuários, com cada usuário clicando num link a cada 5 segundos.

Foram definidos dois cenários de teste: o primeiro, utilizando-se ontologias, com a descoberta e a composição de serviços realizados sob demanda, onde se obteve um tempo médio de resposta de 453 ms; o segundo, sem a utilização de ontologias, com a descoberta e composição de serviços definidos manualmente, com um tempo médio de 39 ms.

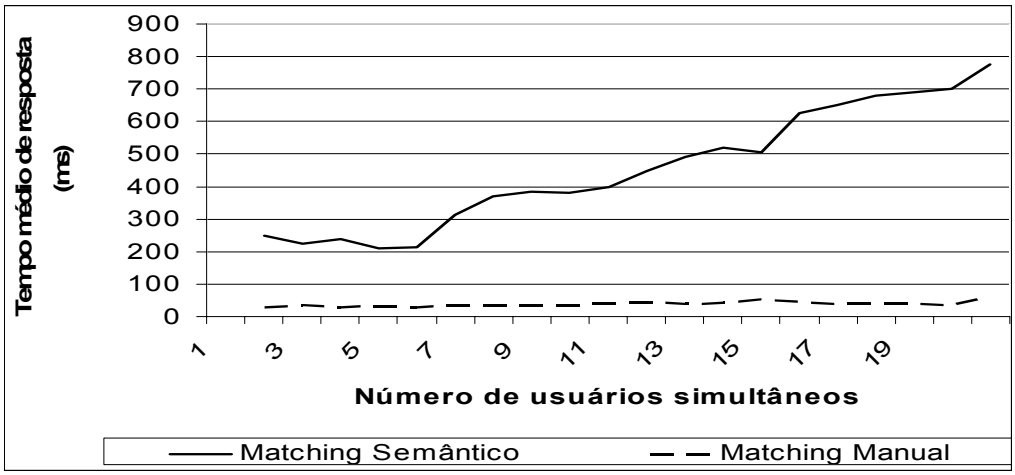


Figura 46. Tempo Médio de Resposta x Número de Usuários

Do ponto de vista quantitativo, analisando os resultados da Figura 46, fica evidente o aumento no tempo de resposta e na demanda por processamento, que foram causados pelo uso de ontologias no FACI reconstruído. Esse tempo excedente é composto pelas seguintes tarefas: inicialização das bibliotecas relacionadas ao tratamento de ontologias e raciocínio, que corresponde a 13,5 % do tempo; carga das ontologias e suas dependências (*imports*), que estavam armazenadas em disco, que corresponde a 12,3% do tempo; e o *matching* dos serviços que corresponde a 74,1% do

tempo. Esses valores foram obtidos a partir da ferramenta de *profiler* do *IDE NetBeans* 5.0. Essas informações serão utilizadas para guiar a otimização do código das próximas versões do FACI.

Entretanto, do ponto de vista qualitativo, fica evidente também a flexibilidade e precisão, que o uso de semântica no mecanismo de descoberta e composição proporciona ao engenheiro de software. É necessário destacar que esse tempo de resposta tende a diminuir com o amadurecimento das *API's* semânticas. Além disso, esse estudo de caso demonstra que a opção, de realizar a pesquisa e a composição de serviços internamente, apresentou um ganho significativo no tempo de resposta (de 40 a 50%) em relação a outros estudos de caso que usam *UDDI's* adaptadas [71,72].

5.4 Conclusão

Vários aspectos das prospostas desta dissertação foram avaliados durante o estudo de caso. Para possibilitar essa avaliação foi desenvolvido um *Adaptation Server,* sendo que a função escolhida foi a de classificação e filtragem de conteúdo. E os testes foram realizados em ambientes controlados e não controlados.

A escolha do servidor de classificação e filtragem de conteúdo foi motivada pelo grande interesse nesse campo e pelo crescimento do uso de dispositivos móveis para acesso a conteúdos impróprios. Apesar da pornografia ser o principal fator para o uso de filtros, outros tipos de conteúdos (e.g., *shopping, blogs*) podem ser indesejados quando acessados de um ambiente corporativo.

Para suportar diferentes necessidades este estudo de caso demonstrou como as informações armazenadas nos perfis podem ser utilizadas pelos servidores de adaptação, com o objetivo de personalizar os serviços. Neste caso, informações

armazenadas no perfil de usuário, e transferidas ao servidor de adaptação, definem quais categorias de conteúdo deveriam ser bloqueadas e quais bases de dados de *URL's* e palavras chave seriam utilizadas.

O teste de eficiência do classificador de conteúdo foi realizado em um ambiente real de uso, que era composto por duas classes de usuários: estudantes universitários e funcionários da instituição. Inicialmente o teste foi realizado com a classificação por *URL*, onde se concluiu que as listas de *URL´s* pré-classificadas possuem uma menor eficiência em páginas brasileiras. Esse fato demonstra a necessidade da elaboração de listas de páginas hospedadas no Brasil. Quando adicionada à classificação por palavras chave, os erros de classificação foram reduzidos sensivelmente.

O impacto que o uso de ontologias causou no FACI foi mensurado no segundo estudo de caso. Esse estudo foi realizado em um ambiente controlado para evitar qualquer influência de variações externas nos resultados obtidos durante as várias fases de implementação. Esses resultados demonstraram que a adição de ontologias causou um grande aumento no uso de recursos computacionais, o que diminui a capacidade de carga do *adaptation proxy* e aumenta o tempo de resposta ao usuário.

Um novo estudo de caso deverá ser realizado para se verificar, em termos qualitativos, o impacto causado pelo uso de ontologias. Na teoria são claras as vantagens obtidas pelo uso de semântica nos processos de descoberta e composição de serviços no contexto da adaptação de conteúdo. Será necessário desenvolver novos serviços de adaptação, incluindo os seus respectivos perfis de serviço, para se realizar uma avaliação dos ganhos obtidos.

6. Conclusão

Esta dissertação propôs a especificação de perfis e regras para adaptação de conteúdo, utilizando-se de ontologias, com o objetivo de oferecer uma solução flexível que viabilize o uso de uma política de adaptação em ambientes abertos como a *Web*.

Além das vantagens, apresentadas ao longo desta dissertação, do uso de semântica, a escolha das linguagens utilizadas para descrever as ontologias, *OWL-DL* e *OWL-S*, possibilitou a adição de novos recursos ao projeto. Inicialmente pode-se citar a inclusão de Lógica de Descrição, auxiliando o funcionamento dos mecanismos de inferência e raciocínio, que são componentes importantes do modulo de *matching*. Do *OWL-S* foram aproveitadas as características de suporte a regras (*SWRL*) e a sua integração com serviços *Web*, possibilitando uma melhor descrição dos serviços de adaptação, incluindo o uso de pré-condições e efeitos, e provendo mecanismos para a composição de serviços.

A partir das alterações propostas nesta dissertação a nova arquitetura de adaptação de conteúdo apresenta as seguintes vantagens em relação à anterior:

- Novas características podem ser adicionadas aos perfis, incluindo novos serviços de adaptação, sem que os componentes do FACI tenham que ser alterados. Essa flexibilidade da arquitetura se deve a inserção das regras dentro dos próprios perfis de serviço, cabendo ao FACI apenas executá-las.
- O uso de Serviços *Web* facilitou o desenvolvimento de novos servidores de adaptação devido a grande disponibilidade de ferramentas e servidores (e.g., *Tomcat*) para essa tecnologia. Também poderão ser utilizados serviços *Web* já disponíveis para a adaptação de conteúdo (e.g., tradução de idiomas).

- A inserção de novos perfis de serviços foi facilitada por meio de uma interface *Web* que permite uma configuração intuitiva e a importação de descrições *WSDL*.
- O uso de ontologias facilita a importação de perfis, principalmente de dispositivos (e.g., *UAPROF*, *WURFL*) onde poderão ser declaradas similaridades entre características com diferentes nomes.
- A atual arquitetura poderá ser facilmente alterada quando as novas versões de *UDDI* e *WSDL* proverem suporte semântico.

Para demonstrar a viabilidade desta proposta foi desenvolvida uma extensão do FACI, a partir da adição, adaptação e reuso de componentes. Essa extensão resultou em um *framework* mais genérico, facilitando o desenvolvimento de novas soluções no domínio da adaptação de conteúdo. A partir desta extensão foi desenvolvido um estudo de caso inicial para verificar a performance do sistema.

Por fim, o uso de ontologias na *Web, incluindo* as ferramentas e *APIs* disponíveis, ainda tem muito que amadurecer, mas a flexibilidade obtida, e os esforços do *W3C* e outras organizações no desenvolvimento da *Web* Semântica, demonstram a viabilidade do uso de ontologias em uma arquitetura de adaptação de conteúdo.

6.1. Limitações e Dificuldades Encontradas

Como limitação à proposta apresenta problemas de desempenho, resultando em uma baixa capacidade de carga do novo *FACI*. Nesta seção são descritos alguns motivos que causam este problema além de outras dificuldades encontradas.

O pouco tempo de vida do *OWL-S* gerou inúmeros problemas no seu uso e implementação. O primeiro desses é a escassez de ferramentas *OWL-S* disponíveis.

Por exemplo, apenas no ano de 2004 foi disponibilizada a primeira *API* de *OWL-S* [84]. Como é uma ferramenta desenvolvida academicamente, ela não possui um ritmo contínuo de desenvolvimento, sendo pouco confiável devido a sua baixa maturidade. Um segundo problema é a escassez de serviços *OWL-S*. A falta de exemplos de OWL-S dificulta o desenvolvimento e testes de ferramentas *OWL-S*, e criar os próprios serviços *OWL-S* pode ser um processo proibitivamente prolongado devido à verbosidade do formato do *XML*. Outras dificuldades com a marcação semântica de Serviços *Web* incluem:

- A falta de suporte a um serviço com múltiplas interfaces, isso é, um que possa aceitar diferentes tipos de parâmetros de entradas.
- Não há uma correspondência clara entre *OWL-S* e conceitos tradicionais de engenharia de software.
- O mapeamento de *OWL-S* para *WSDL* limita a expressividade.

No entanto, talvez a principal razão do *OWL-S* não ter sido largamente adotado foi a sua instabilidade. Nos últimos anos o padrão para marcação semântica de serviços *Web* mudou de *DAML-S 0.9* para *OWL-S 1.0*, e uma versão de *OWL-S 1.1* está atualmente disponível. Mudanças extensas foram propostas pelo último, particularmente ao *ServiceModel*.

Naturalmente, a cada alteração do padrão, mudanças correspondentes devem ser feitas às ferramentas dependentes tal como o *API* da *OWL-S*. É compreensível que muitos preferem esperar até que o padrão estabilize antes de desenvolver qualquer aplicação que dependa dele.

Outra dificuldade encontrada foi obter informações sobre características dos dispositivos de acesso. Os fabricantes de dispositivos móveis desenvolveram o padrão *UAPROF*, mas não disponibilizam um repositório comum para o

armazenamento dos perfis. Desse modo cada fabricante possui o seu próprio repositório, que nem sempre está ativo. Por esse motivo algumas outras soluções, como o *WURFL*, ganharam força por centralizar toda a base de informações em um único local.

6.2 Recomendações para Trabalhos Futuros

A área de adaptação de conteúdo é complexa e envolve várias tecnologias, desse modo ainda existem muitos aspectos a serem desenvolvidos. A partir das experiências obtidas nesta dissertação são sugeridos os seguintes trabalhos futuros:

- Pesquisa e implementação de um novo modulo de composição de serviços que tire melhor proveito dos recursos de composição do *OWL-S*;
- Desenvolvimento de um estudo de caso qualitativo demonstrando a eficiência do mecanismo de descoberta e composição de serviços em relação a outras abordagens (e.g.; sintáticas).
- Expandir as ontologias de modo a suportar outras informações de contexto (e.g., localização, tempo).
- Implementar o gerenciamento de composição de serviços em termos de processo.
- Aprimoramento na performance do *Adaptation Proxy*, com ênfase nos módulos semânticos, de modo a promover um aumento na capacidade de carga;
- Possibilitar que a tarefa de descoberta e composição possa ser realizada externamente, utilizando-se das novas especificações de serviços *Web* Semânticos que estão sendo desenvolvidas (e.g., *OWL-S/UDDI*, *WSDL-S*);
- Adição de um componente que gerencie a troca automática de perfis de serviços entre *Adaptation Proxies*;

- Redução do fluxo de dados entre o *Adaptation Proxy* e os servidores de Adaptação, com destaque aos que utilizam *SOAP*. Para aumentar a eficiência do *SOAP* novas soluções de compressão estão sendo desenvolvidas;
- Desenvolvimento de novos *local* e *remote adapters* de modo a ampliar a variedade de adaptações disponíveis;

7 Referências

1 - WEISER, M. **Ubiquitous computing**. Disponível em: <http://sandbox.xerox.com/hypertext/weiser/UbiHome.html>. Acesso em: 15 jan. 2006.

2 - TANG, B. A. **Pervasive computing**. SearchNetworking.com. Disponível em: <http://searchnetworking.techtarget.com/sDefinition/0,,sid7_gci759337,00.html >. Acesso em: 15 jan. 2006.

3 - GIMSON, R. **Device independence principles:** W3C working group note. Disponível em: <http://www.w3.org/TR/2003/NOTE-di-princ-20030901/>. Acesso em: 18 jan. 2006.

4 - GIMSON, R. et al. **Delivery Context Overview for Device Independence:** W3C working group note. Disponível em: <http://www.w3.org/TR/di-dco/>. Acesso em: 10 abr. 2006.

5 - MILLER, E. **Semantic web activity statement:** technical report. Disponível em: <http://www.w3.org/2001/sw/Activity>. Acesso em: 05 mar. 2006.

6 - **DESCRIPTION logics**. Disponível em: < http://dl.kr.org/>. Acesso em: 10 out. 2005.

7 - **THE DAML coalition**. Disponível em: < http://www.daml.org>. Acesso em: 05 ago. 2005.

8 - MARTIN, D. et al. **OWL-S: semantic markup for web services 1.0.** Disponível em: <http://www.daml.org/services/owl-s/1.0/>. Acesso em: 12 abr. 2005.

9 - OPEN MOBILE ALLIANCE LTD. **OMA User Agent PROFile (UAPROF) V2.0.** Disponível em: <www.openmobilealliance.org /release_program/docs/UAProf/OMA-UAProf-V2_0-20030520-C.PDF>. Acesso em: 15 set. 2004.

10 - PASSANI, L. **Wireless Universal Resource FiLe (WURFL).** Disponível em: <http://wurfl.sourceforge.net/>. Acesso em: 21 nov. 2004.

11 - CLAUDINO, R. A. T.; SOUZA, W. L.; PRADO, A. F. Um framework baseado em componentes para o domínio de adaptação de conteúdo na Internet. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE, 19., 2005, Uberlândia. **Anais...** Uberlandia: SBC, 2005, p. 88-103.

12 - MORAIS, M. **Março é melhor mês do ano para telefonia móvel, que se aproxima dos 90 milhões de assinantes.** Disponível em: <http://www.anatel.gov.br/biblioteca/releases/2006/release_18_04_2006lf.pdf>. Acesso em: 01 mai. 2006.

13 - WIESER, M. The computer for the 21st century. In:____.**Human-computer interaction**: toward the year 2000. San Francisco: Morgan Kaufmann, 1995. p. 933-940.

14 - WEISER, M. *Some computer science issues in ubiquitous computing*. **Communications of the ACM,** New York: ACM Press, v.36, n. 1, p. 75-84, 1993.

15 - ABOWD, G. D.; MYNATT, E. D. Charting past, present, and future research in ubiquitous computing. **ACM Transactions On Computer-Human Interaction (TOCHI),** New York: ACM Press, v.7, n.1, p-29-28, 2000.

16 - ABOWD, G. D.; MYNATT, E. D.; RODDEN, T. The human experience. ***Pervasive Computing***, New Jersey: IEEE Press, v. 1, n. 1, p. 48-57, 2002.

17 - BUTLER, et al. Device independence and the web, **IEEE Internet Computing**, New Jersey: IEEE Press, v. 6, n. 5, p. 81-86, 2002.

18 - FREED, N.; BORENSTEIN, N. **Multipurpose Internet Mail Extensions (MIME) part one: format of internet message bodies.** Disponível em: <http://www.ietf.org/rfc/rfc2045.txt?number=2045>. Acesso em: 10 mai. 2005.

19 - IANA. **MIME media types**. Disponível em:< http://www.iana.org/assignments /media-types/>. Acesso em: 10 mai. 2005.

20 - FIELDING, R. et al. **HyperText Transfer Protocol HTTP/1.1:** request for comments: 2616, Disponível em: <ftp://ftp.isi.edu/in-notes/rfc2616.txt>. Acesso em: 5 mai. 2005.

21 - HOLTMAN, K.; MUTZ, A. **HTTP remote variant selection algorithm. RVSA/1.0:** request for comments: 2296. Disponível em: <http://www.faqs.org/rfcs/rfc2296.html>. Acesso em: 5 mai. 2005.

22 - CLARK, J. **XSL Transformations (XSLT) Version 1.0 :** W3C Recommendation. W3C, 1999. Disponível em: <http://www.w3.org/TR/xslt> Acesso em : 18 out. 2005.

23 - **Image Magick Studio LLC**. Disponível em: <http://www.imagemagick.org>. Acesso em : 05 jan. 2005.

24 - BECK, A.; HOFMANN, M. **Example Services for Network Edge Proxies**. InternetDraft. 2000. Disponível em: <standards.nortelnetworks.com/opes/non-wg-doc/draft-beck-opes-esfnep-01.txt>. Acesso em: 10 jan 2005.

25 - MCHENRY, S. et al. **OPES Use Cases and Deployment Scenarios**. IETF Internet Draft. 2001. Disponível em: <standards.nortelnetworks.com/opes/non-wg.htm>. Acesso em: 10 jan 2005.

26 - HANRAHAN, R. **Authoring Techniques for Device Independence**: W3C Working Group Note. W3C, 2004. Disponível em: <http://www.w3.org/ TR/2004/NOTE-di-atdi-20040218/>. Acesso em: 15 dez. 2004.

27 - **Fórum ICAP**. Disponível em: <www.i-cap.org>. Acesso em: 15 out. 2004.

28 - ELSON J.; CERPA A. **Internet Content Adaptation Protocol (ICAP):** request for comments 3507, Network Working Group, IETF, 2003. Disponível em: <http://www.isi.edu/in-notes/rfc3507.txt>. Acesso em: 15 out. 2004.

29 - HANSEN, T. et al. **Open Pluggable Edge Services (OPES):** Charter. IETF, 2004. Disponível em: <http://www.ietf.org/html.charters/opes-charter.html>. Acesso em: 18 nov. 2005.

30 - MITRA, N. **SOAP Version 1.2 Part 0: Primer**: W3C Recommendation. W3C, 2003. Disponível em: <http://www.w3.org/TR/2003/REC-soap12-part0-20030624/>. Acesso em: 05 jan. 2006.

31 - HELD, A.; BUCHHOLZ, S.; SCHILL, A. **Modeling of Context Information for Pervasive Computing Applications.** In: WORLD MULTICONFERENCE ON SYSTEMICS, 6., 2002, Orlando, FL, USA. **Proceedings…** p. 60-69.

32 - BUTLER, M. H. **Current technologies for device independence.** HP Laboratories Bristol. 2001. Disponível em: <http://www.hpl.hp.com/techreports /2001/HPL-2001-83.pdf>. Acesso em: 11 nov. 2005.

33 - KLYNE, Graham et al. **Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0:** W3C Recommendation. W3C, 2004. Disponível em: <http://www.w3.org/TR/2004/REC-CCPP-struct-vocab-20040115/>. Acesso em: 14 out 2004.

34 - MIKHALENKO, P. **Introduction to Device Independence, Part 2.** O'Reilly Media. 2004. Disponível em: <http://www.xml.com/lpt/a/2004/10/06/di2.html>. Acesso em: 05 abr. 2005.

35 - LEMLOUMA, T.; LAYAÏDA, N. **Universal Profiling Schema for Content Negotiation**. OPERA Project, INRIA Rhône Alpes, 2002. Disponível em: <http://www.inrialpes.fr/opera/people/Tayeb.Lemlouma/Papers/Universal ProfilingSchema.pdf>. Acesso em: 10 jun. 2005.

36 - CANNISTRÀ, Francesco. Exploiting Ontologies to Achieve Semantic Convergence Between Different CC/PP-like RDF Schemes for Representing Device´s Capabilities: the SADiC Approach. In: INTERNATIONAL SEMANTIC WEB CONFERENCE, 2., 2003, Sanibel Island, Florida, 2003.

37 - BECK, A; ROUSSKOV, A. ***P: Message Processing Language:*** Open Pluggable Edge Services, IETF Internet-Draft, IETF, 2003. Disponível em: <http://www.ietf.org/proceedings/03nov/I-D/draft-ietf-opes-rules-p-02.txt>. Acesso em: 13 nov. 2004.

38 - HIRTLE, David et al. **Schema Specification of RuleML 0.9.** RuleML Initiative, 2005. Disponível em: <http://www.ruleml.org/spec/>. Acesso em: 05 jan. 2006.

39 - HORROCKS, Ian. ***SWRL:A Semantic Web Rule Language Combining OWL and RuleML*** : W3C Member Submission. W3C, 2004. Disponível em: < http://www. w3.org/Submission/2004/SUBM-SWRL-20040521/>. Acesso em: 05 jul 2005.

40 - GUARINO, N. formal ontology and information systems. In: Formal Ontology in Information Systems FOIS'98, 1., 1998, Amsterdam. **Proceedings…** Amsterdam: IOS Press, 1998. p. 3-15

41 - GUARINO, N., Understanding, building and using ontologies, **Int. Journal Human-Computer Studies**, USA: Academic Press, v. 46, n. 2-3, p. 293-310, 1997.

42 - GRUBER, R., Toward principles for the design of ontologies used for knowledge sharing, **Int. J. Human-Computer Studies**, Duluth- MN-USA: Academic Press, v43, n. 5-6, p. 907-928, 1995.

43 - GUARINO, N., WELTY, C., **Conceptual modeling and ontological analysis**, LADSEB-CNR, Padova, 1998.

44 - MARTIN, D. L.; MCLLRAITH, S. A. **Bringing semantics to web services**. IEEE intelligent systems. USA: IEEE Press, 2003. p. 90-93.

45 - ALBUQUERQUE, N. D. B.; KERN, V. M. Uma arquitetura para o compartilhamento do conhecimento em bibliotecas digitais**.** In: SEMINCO - SEMINÁRIO DE COMPUTAÇÃO, 13., 2004, Blumenau-SC. **Anais...** Blumenau-SC: FURB, 2004. p. 149-160.

46 - BERNERS-LEE, T.; HENDLER, J.; LASSILA, O. **The semantic web**. Disponível em: < http://www.scientificamerican.com/article.cfm ?articleID= 00048144-10D2-1C70-84A9809EC588EF21&catID=2>. Acesso em: 05 jan. 2005.

47 - DACONTA, M.C.; OBRST, L.J.; SMITH, K.T. **The semantic web**, Indianapolis: Wiley Publishing, 2003.

48 - BERNERS-LEE, T. et al., **Semantic web development proposal**. Disponível em: <http://w3c.org/2001/sw/>. Acesso em: 15 jan 2005.

49 - MANOLA, F.; MILLER, E. **RDF Primer**: W3C recommendation. Disponível em: <http://www.w3.org/TR/2004/REC-rdf-primer-20040210/>. Acesso em: 07 jan. 2005.

50 - MCGUINNESS, D. L.; HARMELEN, F. V. **OWL web ontology language overview:** W3C recommendation. Disponível em: <http://www.w3.org/ TR/2004/REC-owl-features-20040210/>. Acesso em: 15 mar. 2005.

51 - DEAN, M. et al. **OWL web ontology language reference**: W3C recommendation. Disponível em: <http://www.w3.org/TR/owl-ref/>. Acesso em: 15 mar. 2005.

52 - HORROCKS, I.; PATEL-SCHNEIDER, P.; HARMELEN F. van. *From SHIQ and RDF to OWL:The Making of a Web Ontology Language.* **Journal of Web Semantics,** v. 1, n. 1, p. 7-26, 2003.

53 - BAADER, et al. **The description logic handbook:** theory, implementation and applications. USA: Cambridge University Press, 2003.

54 - HORRIDGE, M. et al. **A practical guide to building OWL ontologies using the Protege-OWL plugin and CO-ODE tools.** Disponível em: <http://www.co-ode.org/resources/tutorials/ ProtegeOWLTutorial.pdf>. Acesso em: 15 dez 2004.

55 - HORROCKS, I.; TSARKOV, D. **The FaCT++ system**. Disponível em: <http://owl.man.ac.uk/factplusplus/>. Acesso em 20 mar. 2006.

56 - RACER-SYSTEMS. **Renamed Abox and Concept Expression Reasoner - RACER**. Disponível em: <http://www.racer-systems.com/index.phtml>. Acesso em 10 mai. 2006.

57 - MIND LAB. **Pellet OWL reasoner**. University of Maryland Institute for Advanced Computer Studies. 2003. Disponível em: <http://www.mindswap.org/2003/pellet/>. Acesso em: 14 set. 2005.

58 - GONZALES-CASTILHO, J.; TRASTOUR D.; BARTOLINI C. Description logics for matchmaking of services. In: WORKSHOP ON APPLICATION OF DESCRIPTION LOGICS AT KI-2001, 2., 2001, Vienna, Austria, **Proceedings…**, 2001.

59 - BRICKLEY, D.; GUHA, R.V. **RDF vocabulary description language 1.0: RDF Schema:** W3C Recommendation. W3C, 2004. Disponível em: <http://www.w3.org/TR/2004/REC-rdf-schema-20040210/>. Acesso em: 15 nov. 2004.

60 - HORROCKS, I. et. al. **DAML + OIL**. DARPA, 2001. Disponível em: <http://www.daml.org/2001/03/daml+oil-index.html>. Acesso em: 20 abr. 2005.

61 - ANTONIOU, G. et al. **Combining Rules and Ontologies:** A survey**.** REWERSE, 2005. Disponível em: <http://rewerse.net/deliverables/m12/i3-d3.pdf>. Acesso em: 12 dez. 2005.

62 - BOLEY, H.; TABET, S.; WAGNER, G. Design Rationale of RuleML: A Markup Language for Semantic Web Rules. In: INTERNATIONAL SEMANTIC WEB WORKING SYMPOSIUM (SWWS), 2001.

63 - WAGNER, G. The Semantic Web Needs Two Kinds of Negation. In: WORKSHOP ON PRINCIPLES AND PRACTICE OF SEMANTIC WEB REASONING, Mumbai, India, 2003. p. 33-50

64 - BOLEY, H. et al. **FOL RuleML: The First-Order Logic Web Language**. Disponível em: <http://www.ruleml.org/fol/>. Acesso em: 17 set. 2005.

65 - WAGNER, G.; DAMÁSIO, C. V.; LUKICHEV, S. **First-Version Rule Markup Languages.** REWERSE, 2005. Disponível em: <http://rewerse.net/deliverables/m12/i1-d3-1.pdf>. Acesso em: 12 dez. 2005.

66 - RUCKHAUS, E**.** Efficiently Answering Queries to DL and Rules Web Ontologies. In: CONFERENCE ON ADVANCED INFORMATION SYSTEMS ENGINEERING - CAISE'05, 2005.

67 - HORAN B. **The Use of Capability Descriptions in a Wireless Transducer Network**. Technical Report TR-2005-131, Sun Microsystems, 2005.

68 - BOOTH, D. et al. **Web Services Architecture**: W3C working group, W3C, 2004. Disponível em: <http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/>. Acesso em: 30 jun. 2005.

69 - GUDGIN, M. et al. **SOAP Version 1.2 Part 1: Messaging Framework:** W3C recommendation, W3C, 2003. Disponível em: <http://www.w3.org/TR/2003/REC-soap12-part1-20030624/>. Acesso em: 30 jun. 2005.

70 - PASSIN, T. B. *Semantic Web services*. In:___. ***Explorer's Guide to the Semantic Web.*** Connecticut: Manning, 2004, Cap. 8, p. 187-192.

71 - NAUMENKO, A. et al. Using UDDI for Publishing Metadata of the Semantic Web, In: INTERNATIONAL IFIP/WG12.5 WORKING CONFERENCE IASW-2005, 1., 2005, Jyväskylä. **Proceedings…** Finland: Springer, 2005. p. 141-159.

72 - SRİNİVASAN, N.; PAOLUCCİ, M.; SYCARA, K.; Adding OWL-S to UDDI, implementation and throughput. In: INTERNATIONAL WORKSHOP ON SEMANTIC WEB SERVICES AND WEB PROCESS COMPOSITION, 1. 2004, California.

73 - PAOLUCCI, M. et al. Semantic Matching of Web Services Capabilities. In: INTERNATIONAL SEMANTIC WEB CONFERENCE ON THE SEMANTIC WEB (ICWC), 1., 2002, Londres. **Proceedings…** Londres: Springer-Verlag, 2002. p. 333-347.

74 - LARA, R. et al. **Definition of semantics for web service discovery and composition**. Knowledge Web Deliverable D2.4.2, 2004. Disponível em: <http://knowledgeweb.semanticweb.org/semanticportal/servlet/download?ontology=Documentation+Ontology&concept=Deliverable&instanceSet=kweb&instance=D2.4.2%3A+Definition+of+semantics+for+web+service+discovery+and+co

mposition&attribute=On-line+PDF+Version&value=d2.4.2.pdf>. Acesso em: 15 nov. 2005.

75 - CHEN, H. et al. SOUPA: Standard Ontology for Ubiquitous and Pervasive Application. In: INTERNATIONAL CONFERENCE ON MOBILE AND UBIQUITOUS SYSTEMS: NETWORKING AND SERVICES, Boston, 2004.

76 - BRICKLEY, D.; MILLER, L. **FOAF vocabulary specification**. Disponível em: <http://xmlns.com/foaf/0.1/>. Acesso em: 15 jan 2006.

77 - PAN, F.; HOBBS, J. R. Time in OWL-S. In: *PROCEEDINGS OF AAAI-04 SPRING SYMPOSIUM ON SEMANTICWEB SERVICES*, Stanford University, California, 2004.

78 - LENAT, D. B.; GUHA, R. V. ***Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project***. Addison-Wesley, 1990.

79 - RANDELL, D. A.; CUI, Z.; COHN, A. **A spatial logic based on regions and connection.** In: B. Nebel, C. Rich, and W. Swartout, (eds), *KR'92. Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference*, California: Morgan Kaufmann, 1992. p. 165–176.

80 - CHEN, H.; FININ, T.; JOSHI, A. An ontology for context aware pervasive computing environments. Special Issue on Ontologies for Distributed Systems, **Knowledge Engineering Review**, 2003.

81 - PERICH. F. **MoGATU BDI Ontology.** University of Maryland, 2004. Disponível em: <http://mogatu.umbc.edu/bdi/20040131/>. Acesso em: 15 dez. 2005.

82 - KAGAL, L.; FININ, T.; JOSHI, A. **A Policy Based Approach to Security for the Semantic Web.** In: *International Semantic Web Conference (ISWC2003)*, 2. , 2003, Florida.

83 - EODM. **IBM Integrated Ontology Development Toolkit.** Disponível em: <http://www.alphaworks.ibm.com/tech/semanticstk>. Acesso em: 01 fev. 2006.

84 - **OWL-S API.** Maryland Information and Network Dynamics Lab Semantic Web Agents Project (Mindswap), University of Maryland. Disponível em: <http://www.mindswap.org/2004/owl-s/api/>. Acesso em: 10 fev. 2005.

85 - **Jena API**: A semantic web framework for Java, HP Labs Semantic Web Research. Disponível em: <http://jena.sourceforge.net/>. Acesso em: 12 dez. 2004.

86 - **Pellet API**. Maryland Information and Network Dynamics Lab Semantic Web Agents Project (Mindswap), University of Maryland. Disponível em: <http://www.mindswap.org/2003/pellet/>. Acesso em: 14 mar. 2005.

87 - NUTJER, M. H. CC/PP and UAProf: Issues, Improvements and Future Directions. In: W3C WORKSHOP ON DELIVERY CONTEXT, France: INRIA Sophia-Antipolis, 2002.

88 - HELD, A.; BUCHHOLZ, S.; SCHILL, A**.** Modeling of Context Information for Pervasive Computing Applications. In: WORLD MULTICONFERENCE ON SYSTEMICS, CYBERNETICS AND INFORMATICS (SCI), 6., 2002, Orlando, **Proceedings…** p 14-18.

89 - VIANA, W. et al. Mobile Adapter: Uma abordagem para a construção de Mobile Application Servers adaptativos utilizando as especificações CC/PP e UAProf. In: SEMINÁRIO INTEGRADO DE SOFTWARE E HARDWARE, 32., 2005, São Leopoldo, RS :Unisinos, 2005.

90 - BERHE, G.; BRUNIE, L.; PIERSON, J. Modeling Service-Based Multimedia Content Adaptation in Pervasive Computing. In: CONFERENCE OF COMPUTING FRONTIERS, 2004, NY, **Proceedings…**New York: ACM Press, 2004. p 60-69.

91 - TOIVONEN, S. et al. Context-Sensitive Conversation Patterns for Agents in Wireless Environments. In: INTERNATIONAL WORKSHOP ON UBIQUITOUS COMPUTING, IWUC 2004, IN CONJUNCTION WITH ICEIS 2004, 1., 2004, Porto - Portugal, **Proceedings…** Porto: INSTICC Press, 2004. p. 11-17.

92 - SRINIVASAN, N.; PAOLUCCI, M.; SYCARA, K. P. An Efficient Algorithm for OWL-S Based Semantic Search in UDDI. In: SEMANTIC WEB SERVICES AND WEB PROCESS COMPOSITION - SWSWPC 2004, 1., 2004, CA - USA, **Proceedings…** p. 96-110.

93 - YAO, Y.; SU, S.; YANG, F. Service Matching Based on Semantic Descriptions. In: ADVANCED INTERNATIONAL CONFERENCE ON TELECOMMUNICATIONS AND INTERNATIONAL CONFERENCE ON INTERNET AND WEB APPLICATIONS AND SERVICES (AICT/ICIW 2006), 2006, Guadeloupe - French Caribbean. **Proceedings…**Washington: IEEE Computer Society, 2006. p. 126.

94 - PAOLUCCI, M. et al. Representing Services for Mobile Computing using OWL and OWL-S**.** In: WWW SERVICE COMPOSITION WITH SEMANTIC WEB SERVICES (WSCOMP), Compiègne, France, 2005.

95 - ZAHREDDINE, W.; MAHMOUD Q. H. A Framework for Automatic and Dynamic Composition of Personalized Web Services. In: INTERNATIONAL CONFERENCE ON ADVANCED INFORMATION NETWORKING AND APPLICATIONS (AINA 2005), 19., 2005, USA. **Proceedings…** Washington: IEEE Computer Society, 2005. p. 513-518.

96 - GEYTER, M.; SOETENS, P**.** A Planning Approach to Media Adaptation within the Semantic Web. In: INTERNATIONAL CONFERENCE ON

DISTRIBUTED MULTIMEDIA SYSTEMS (DMS 2005), 11., 2005, Canada. **Proceedings…**Canada, 2005.

97 - PREUVENEERS, D.; BERBERS, Y. Adaptive context management using a component-based approach. **Lecture Notes in Computer Science (LNCS)**, v. 3543, London: Springer Verlag, 2005. p. 1-12.

98 - SHABAN, A. ; HAARSLEV, V. Applying Semantic Web technologies to matchmaking and web service descriptions. In: MONTREAL CONFERENCE ON ETECHNOLOGIES (MCETECH2005), 2005, Montreal – Canada. **Proceedings...** 2005, p. 97-104.

99 - **Safe Internet Action Plan (SIAP)** – Disponível em: <http://europa.eu.int/information_society/programmes/iap/index_en.htm>. Acesso em : 8 jan 2004 .

100 - **OII Guide to labelling, rating and filtering**. 2002. Disponível em: <http://www.difuse.org/oii/en/labels.html>. Acesso em: 15 jan 2004.

101 - N2H2 CORP. **Managing the workplace Internet**. 2001. Disponível em: <http://www.n2h2.com>. Acesso em: 20 jan 2004.

102 - ICOGNITO Technologies Ltd. **Dynamic filtering of Internet content: An overview of next generation filtering technology**. 2002. Disponível em: <http://www.icognito.com>. Acesso em: 30 jan 2004.

103 - GOLLER, C. **Automatic document classification: A thorough evaluation of various methods**. Disponível em: <http://citeseer.ist.psu.edu/ context/2496479/0>. Acesso em: 12 jan 2004.

104 - **Platform for Internet Content Selection - PICS.** Disponível em: **<**http://www.w3.org/ PICS/>. Acesso em: 20 dez 2003.

105 - NETPROTECT Consortium. **Report on filtering techniques and approaches**. 2001. Disponível em: <http://www.net-protect.org/en/OPT-WP2-D2.3-v1.0.pdf>. Acesso em: 6 jan 2004.

106 - The Henry J. Kaiser Family Foundation. **See no evil: How Internet filters affect the search for online health information.** Disponível em: <http://www. kff.org>. Acesso em: 15 dez 2003.

107 - FORTE, M.; SOUZA, W. L.; PRADO, A. F. Um servidor para a classificação e filtragem de conteúdo na Internet. In: SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES, 23., 2005, Fortaleza. **Anais…** Fortaleza, CE: II/UFC, 2005. p 191-204.

108 - GRUBER, T. R. A translation approach to portable ontology specifications. **Knowledge Acquisition.** Special issue: Current issues in knowledge modeling. London, UK: Academic Press Ltd, 1993, p. 199-220.

Apêndices

Perfil de dispositivo:

```
<?xml version="1.0" encoding="GB2312"?>
<!DOCTYPE rdf:RDF [
	<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
	<!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
	<!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
	<!ENTITY owl "http://www.w3.org/2002/07/owl#">
	<!ENTITY owl_editor_base "file:/D:/eclipse/workspace/Ontologias/Device.owl/">
	<!ENTITY ns_1 "file:///">
]>
<rdf:RDF
	 xmlns:xsd="&xsd;"
	 xmlns:rdf="&rdf;"
	 xmlns:rdfs="&rdfs;"
	 xmlns:owl="&owl;"
	 xmlns:owl_editor_base="&owl_editor_base;"
	 xmlns:ns_1="&ns_1;"
>

	<owl:Ontology rdf:about="file:///OWLInputStream">
		<rdfs:comment>IBM OWL Editor</rdfs:comment>
	</owl:Ontology>
	<owl:Class rdf:about="&owl;Thing">
	</owl:Class>
	<owl:Class rdf:about="&ns_1;Device">
	</owl:Class>
	<owl:Class rdf:about="&ns_1;Streaming">
		<rdfs:subClassOf rdf:resource="&ns_1;Device"/>
	</owl:Class>
	<owl:Class rdf:about="&ns_1;Network">
		<rdfs:subClassOf rdf:resource="&ns_1;Device"/>
	</owl:Class>
	<owl:Class rdf:about="&ns_1;Image">
		<rdfs:subClassOf rdf:resource="&ns_1;Device"/>
	</owl:Class>
	<owl:Class rdf:about="&ns_1;Sound_Format">
		<rdfs:subClassOf rdf:resource="&ns_1;Device"/>
	</owl:Class>
	<owl:Class rdf:about="&ns_1;J2ME">
		<rdfs:subClassOf rdf:resource="&ns_1;Device"/>
	</owl:Class>
	<owl:Class rdf:about="&ns_1;Markup">
		<rdfs:subClassOf rdf:resource="&ns_1;Device"/>
	</owl:Class>
	<owl:Class rdf:about="&ns_1;Storage">
		<rdfs:subClassOf rdf:resource="&ns_1;Device"/>
	</owl:Class>
	<owl:Class rdf:about="&ns_1;SMS">
		<rdfs:subClassOf rdf:resource="&ns_1;Device"/>
	</owl:Class>
	<owl:Class rdf:about="&ns_1;Display">
		<rdfs:subClassOf rdf:resource="&ns_1;Device"/>
	</owl:Class>
	<owl:Class rdf:about="&ns_1;Product_Info">
		<rdfs:subClassOf rdf:resource="&ns_1;Device"/>
	</owl:Class>
```

```
<owl:DatatypeProperty rdf:about="&ns_1;brand_name">
        <rdfs:domain rdf:resource="&ns_1;Product_Info"/>
        <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="&ns_1;model_name">
        <rdfs:domain rdf:resource="&ns_1;Product_Info"/>
        <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:Class rdf:about="&ns_1;Display_Resolution">
</owl:Class>
<owl:DatatypeProperty rdf:about="&ns_1;resolution_width">
        <rdfs:domain rdf:resource="&ns_1;Display_Resolution"/>
        <rdfs:range rdf:resource="&xsd;unsignedInt"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="&ns_1;resolution_height">
        <rdfs:domain rdf:resource="&ns_1;Display_Resolution"/>
        <rdfs:range rdf:resource="&xsd;unsignedInt"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="&ns_1;columns">
        <rdfs:domain rdf:resource="&ns_1;Display_Resolution"/>
        <rdfs:range rdf:resource="&xsd;unsignedInt"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="&ns_1;rows">
        <rdfs:domain rdf:resource="&ns_1;Display_Resolution"/>
        <rdfs:range rdf:resource="&xsd;unsignedInt"/>
</owl:DatatypeProperty>
<owl:ObjectProperty rdf:about="&ns_1;hasResolution">
        <rdfs:domain rdf:resource="&ns_1;Display"/>
        <rdfs:range rdf:resource="&ns_1;Display_Resolution"/>
</owl:ObjectProperty>
<owl:Class rdf:about="&ns_1;Colors">
</owl:Class>
<owl:ObjectProperty rdf:about="&ns_1;hasColors">
        <rdfs:domain rdf:resource="&ns_1;Display"/>
        <rdfs:range rdf:resource="&ns_1;Colors"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:about="&ns_1;num_of_colors">
        <rdfs:domain rdf:resource="&ns_1;Colors"/>
        <rdfs:range rdf:resource="&xsd;unsignedInt"/>
</owl:DatatypeProperty>
<owl:ObjectProperty rdf:about="&ns_1;hasFormat">
        <rdfs:domain rdf:resource="&ns_1;Image"/>
        <rdfs:range rdf:resource="&ns_1;Image_Format"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="&ns_1;hasLanguage">
        <rdfs:domain rdf:resource="&ns_1;Markup"/>
        <rdfs:range rdf:resource="&ns_1;Markup_Languages"/>
</owl:ObjectProperty>
<owl:Class rdf:about="&ns_1;Supported_Markup">
        <rdfs:subClassOf rdf:resource="&ns_1;Markup"/>
        <rdfs:subClassOf>
                <owl:Restriction rdf:about="&ns_1;Supported_MarkupRestriction">
                        <owl:onProperty rdf:resource="&ns_1;hasLanguage"/>
                        <owl:someValuesFrom rdf:resource="&ns_1;Markup_Languages"/>
                </owl:Restriction>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="&ns_1;Supported_Image">
        <rdfs:subClassOf rdf:resource="&ns_1;Image"/>
        <rdfs:subClassOf>
```

```
                    <owl:Restriction rdf:about="&ns_1;Supported_ImageRestriction">
                        <owl:onProperty rdf:resource="&ns_1;hasFormat"/>
                        <owl:someValuesFrom rdf:resource="&ns_1;Image_Format"/>
                    </owl:Restriction>
            </rdfs:subClassOf>
    </owl:Class>
    <owl:DatatypeProperty rdf:about="&ns_1;Greyscale">
            <rdfs:domain rdf:resource="&ns_1;Colors"/>
            <rdfs:range rdf:resource="&xsd;boolean"/>
    </owl:DatatypeProperty>
    <owl:Class rdf:about="&ns_1;WML_UI">
            <rdfs:subClassOf rdf:resource="&ns_1;Markup"/>
    </owl:Class>
    <owl:DatatypeProperty rdf:about="&ns_1;proportional_font">
            <rdfs:domain rdf:resource="&ns_1;WML_UI"/>
            <rdfs:range rdf:resource="&xsd;boolean"/>
    </owl:DatatypeProperty>
    <owl:DatatypeProperty rdf:about="&ns_1;table_support">
            <rdfs:domain rdf:resource="&ns_1;WML_UI"/>
            <rdfs:range rdf:resource="&xsd;boolean"/>
    </owl:DatatypeProperty>
    <owl:DatatypeProperty rdf:about="&ns_1;wrap_mode_support">
            <rdfs:domain rdf:resource="&ns_1;WML_UI"/>
            <rdfs:range rdf:resource="&xsd;boolean"/>
    </owl:DatatypeProperty>
    <owl:Class rdf:about="&ns_1;CHTML_UI">
            <rdfs:subClassOf rdf:resource="&ns_1;Markup"/>
    </owl:Class>
    <owl:DatatypeProperty rdf:about="&ns_1;emoji">
            <rdfs:domain rdf:resource="&ns_1;CHTML_UI"/>
            <rdfs:range rdf:resource="&xsd;boolean"/>
    </owl:DatatypeProperty>
    <owl:DatatypeProperty rdf:about="&ns_1;imode_region">
            <rdfs:domain rdf:resource="&ns_1;CHTML_UI"/>
            <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>
    <owl:Class rdf:about="&ns_1;XHTML_UI">
            <rdfs:subClassOf rdf:resource="&ns_1;Markup"/>
    </owl:Class>
    <owl:DatatypeProperty rdf:about="&ns_1;xhtml_select_as_popup">
            <rdfs:domain rdf:resource="&ns_1;XHTML_UI"/>
            <rdfs:range rdf:resource="&xsd;boolean"/>
    </owl:DatatypeProperty>
    <owl:DatatypeProperty rdf:about="&ns_1;xhtmlmp_preferred_mime_type">
            <rdfs:domain rdf:resource="&ns_1;XHTML_UI"/>
            <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>
    <owl:Class rdf:about="&ns_1;Cache">
            <rdfs:subClassOf rdf:resource="&ns_1;Device"/>
    </owl:Class>
    <owl:DatatypeProperty rdf:about="&ns_1;total_cache_disable_support">
            <rdfs:domain rdf:resource="&ns_1;Cache"/>
            <rdfs:range rdf:resource="&xsd;boolean"/>
    </owl:DatatypeProperty>
    <owl:DatatypeProperty rdf:about="&ns_1;time_to_live_support">
            <rdfs:domain rdf:resource="&ns_1;Cache"/>
            <rdfs:range rdf:resource="&xsd;boolean"/>
    </owl:DatatypeProperty>
    <owl:Class rdf:about="&ns_1;Security">
            <rdfs:subClassOf rdf:resource="&ns_1;Device"/>
```

```
</owl:Class>
<owl:DatatypeProperty rdf:about="&ns_1;https_support">
        <rdfs:domain rdf:resource="&ns_1;Security"/>
        <rdfs:range rdf:resource="&xsd;boolean"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="&ns_1;streaming_video">
        <rdfs:domain rdf:resource="&ns_1;Streaming"/>
        <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="&ns_1;streaming_audio">
        <rdfs:domain rdf:resource="&ns_1;Streaming"/>
        <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="&ns_1;wav">
        <rdfs:domain rdf:resource="&ns_1;Sound_Format"/>
        <rdfs:range rdf:resource="&xsd;boolean"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="&ns_1;mp3">
        <rdfs:domain rdf:resource="&ns_1;Sound_Format"/>
        <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="&ns_1;mdip">
        <rdfs:domain rdf:resource="&ns_1;J2ME"/>
        <rdfs:range rdf:resource="&xsd;boolean"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="&ns_1;cldc">
        <rdfs:domain rdf:resource="&ns_1;J2ME"/>
        <rdfs:range rdf:resource="&xsd;boolean"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="&ns_1;doja">
        <rdfs:domain rdf:resource="&ns_1;J2ME"/>
        <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="&ns_1;runtime_mem_limit">
        <rdfs:domain rdf:resource="&ns_1;J2ME"/>
        <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:Class rdf:about="&ns_1;Image_Format">
        <owl:oneOf rdf:parseType="Collection">
                <ns_1:Image_Format rdf:about="&ns_1;wbmp">
                </ns_1:Image_Format>
                <ns_1:Image_Format rdf:about="&ns_1;bmp">
                </ns_1:Image_Format>
                <ns_1:Image_Format rdf:about="&ns_1;gif">
                </ns_1:Image_Format>
                <ns_1:Image_Format rdf:about="&ns_1;jpg">
                </ns_1:Image_Format>
                <ns_1:Image_Format rdf:about="&ns_1;png">
                </ns_1:Image_Format>
                <ns_1:Image_Format rdf:about="&ns_1;tiff">
                </ns_1:Image_Format>
        </owl:oneOf>
</owl:Class>
<ns_1:Image_Format rdf:about="&ns_1;wbmp">
</ns_1:Image_Format>
<ns_1:Image_Format rdf:about="&ns_1;bmp">
</ns_1:Image_Format>
<ns_1:Image_Format rdf:about="&ns_1;gif">
</ns_1:Image_Format>
<ns_1:Image_Format rdf:about="&ns_1;jpg">
```

```
        </ns_1:Image_Format>
        <ns_1:Image_Format rdf:about="&ns_1;png">
        </ns_1:Image_Format>
        <ns_1:Image_Format rdf:about="&ns_1;tiff">
        </ns_1:Image_Format>
        <owl:Class rdf:about="&ns_1;Markup_Languages">
                <owl:oneOf rdf:parseType="Collection">
                        <ns_1:Markup_Languages rdf:about="&ns_1;XHTML">
                        </ns_1:Markup_Languages>
                        <ns_1:Markup_Languages rdf:about="&ns_1;cHTML">
                        </ns_1:Markup_Languages>
                        <ns_1:Markup_Languages rdf:about="&ns_1;HTML">
                        </ns_1:Markup_Languages>
                        <ns_1:Markup_Languages rdf:about="&ns_1;XHTMLMP">
                        </ns_1:Markup_Languages>
                        <ns_1:Markup_Languages rdf:about="&ns_1;WML">
                        </ns_1:Markup_Languages>
                </owl:oneOf>
        </owl:Class>
        <ns_1:Markup_Languages rdf:about="&ns_1;XHTML">
        </ns_1:Markup_Languages>
        <ns_1:Markup_Languages rdf:about="&ns_1;cHTML">
        </ns_1:Markup_Languages>
        <ns_1:Markup_Languages rdf:about="&ns_1;HTML">
        </ns_1:Markup_Languages>
        <ns_1:Markup_Languages rdf:about="&ns_1;XHTMLMP">
        </ns_1:Markup_Languages>
        <ns_1:Markup_Languages rdf:about="&ns_1;WML">
        </ns_1:Markup_Languages>
        <owl:Class rdf:nodeID="_4d2e6f56abdaca48-12">
        </owl:Class>
</rdf:RDF>
```

Perfil de Usuário:

```
<?xml version="1.0" encoding="GB2312"?>
<!DOCTYPE rdf:RDF [
	<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
	<!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
	<!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
	<!ENTITY owl "http://www.w3.org/2002/07/owl#">
	<!ENTITY owl_editor_base "file:/D:/eclipse/workspace/Ontologias/user2.owl/">
	<!ENTITY ns_4 "file:///">
]>
<rdf:RDF
	 xmlns:xsd="&xsd;"
	 xmlns:rdf="&rdf;"
	 xmlns:rdfs="&rdfs;"
	 xmlns:owl="&owl;"
	 xmlns:owl_editor_base="&owl_editor_base;"
	 xmlns:ns_4="&ns_4;"
>

	<owl:Ontology rdf:about="file:///OWLInputStream">
		<rdfs:comment>IBM OWL Editor</rdfs:comment>
	</owl:Ontology>
	<owl:Class rdf:about="&owl;Thing">
	</owl:Class>
	<owl:Class rdf:about="&ns_4;User">
	</owl:Class>
	<owl:Class rdf:about="&ns_4;Info">
		<rdfs:subClassOf rdf:resource="&ns_4;User"/>
		<rdfs:subClassOf>
			<owl:Restriction rdf:about="&ns_4;ID_Required">
				<owl:onProperty rdf:resource="&ns_4;ID"/>
				<owl:cardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">1</owl:cardinality>
			</owl:Restriction>
		</rdfs:subClassOf>
	</owl:Class>
	<owl:DatatypeProperty rdf:about="&ns_4;ID">
		<rdfs:domain rdf:resource="&ns_4;Info"/>
		<rdfs:range rdf:resource="&xsd;unsignedInt"/>
	</owl:DatatypeProperty>
	<owl:DatatypeProperty rdf:about="&ns_4;FirstName">
		<rdfs:domain rdf:resource="&ns_4;Info"/>
		<rdfs:range rdf:resource="&xsd;string"/>
	</owl:DatatypeProperty>
	<owl:DatatypeProperty rdf:about="&ns_4;LastName">
		<rdfs:domain rdf:resource="&ns_4;Info"/>
		<rdfs:range rdf:resource="&xsd;string"/>
	</owl:DatatypeProperty>
	<owl:Class rdf:about="&ns_4;Classification_and_Filtering">
		<rdfs:subClassOf rdf:resource="&ns_4;Services_Preferences"/>
	</owl:Class>
	<owl:Class rdf:about="&ns_4;AntiVirus">
		<rdfs:subClassOf rdf:resource="&ns_4;Services_Preferences"/>
	</owl:Class>
	<owl:Class rdf:about="&ns_4;Image_Adapter">
		<rdfs:subClassOf rdf:resource="&ns_4;Services_Preferences"/>
	</owl:Class>
	<owl:DatatypeProperty rdf:about="&ns_4;Filter_profile">
		<rdfs:domain rdf:resource="&ns_4;Classification_and_Filtering"/>
```

```
                <rdfs:range rdf:resource="&xsd;unsignedInt"/>
        </owl:DatatypeProperty>
        <owl:DatatypeProperty rdf:about="&ns_4;Scan_ScriptLanguage">
                <rdfs:domain rdf:resource="&ns_4;AntiVirus"/>
                <rdfs:range rdf:resource="&xsd;boolean"/>
        </owl:DatatypeProperty>
        <owl:DatatypeProperty rdf:about="&ns_4;Reduce_Colors">
                <rdfs:domain rdf:resource="&ns_4;Image_Adapter"/>
                <rdfs:range rdf:resource="&xsd;boolean"/>
        </owl:DatatypeProperty>
        <owl:DatatypeProperty rdf:about="&ns_4;Convert_to_Black">
                <rdfs:domain rdf:resource="&ns_4;Image_Adapter"/>
                <rdfs:range rdf:resource="&xsd;boolean"/>
        </owl:DatatypeProperty>
        <owl:DatatypeProperty rdf:about="&ns_4;Reduce_Resolution">
                <rdfs:domain rdf:resource="&ns_4;Image_Adapter"/>
                <rdfs:range rdf:resource="&xsd;boolean"/>
        </owl:DatatypeProperty>
        <owl:Class rdf:about="&ns_4;Services_Preferences">
                <rdfs:subClassOf rdf:resource="&ns_4;User"/>
        </owl:Class>
        <owl:DatatypeProperty rdf:about="&ns_4;Preferrred">
                <rdfs:domain>
                        <owl:Class >
                        </owl:Class>
                </rdfs:domain>
                <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>
        <owl:DatatypeProperty rdf:about="&ns_4;URLDB">
                <rdfs:domain rdf:resource="&ns_4;Classification_and_Filtering"/>
                <rdfs:range rdf:resource="&xsd;unsignedInt"/>
        </owl:DatatypeProperty>
        <owl:DatatypeProperty rdf:about="&ns_4;KWDB">
                <rdfs:domain rdf:resource="&ns_4;Classification_and_Filtering"/>
                <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>
        <owl:DatatypeProperty rdf:about="&ns_4;SLA_ID">
                <rdfs:domain rdf:resource="&ns_4;Info"/>
                <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>
</rdf:RDF>
```

Perfil de Rede:

```
<?xml version="1.0" encoding="GB2312"?>
<!DOCTYPE rdf:RDF [
	<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
	<!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
	<!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
	<!ENTITY owl "http://www.w3.org/2002/07/owl#">
	<!ENTITY owl_editor_base "file:/D:/eclipse/workspace/Ontologias/network.owl/">
	<!ENTITY ns_0 "file:///">
]>
<rdf:RDF
	 xmlns:xsd="&xsd;"
	 xmlns:rdf="&rdf;"
	 xmlns:rdfs="&rdfs;"
	 xmlns:owl="&owl;"
	 xmlns:owl_editor_base="&owl_editor_base;"
	 xmlns:ns_0="&ns_0;"
>

	<owl:Ontology rdf:about="file:///OWLInputStream">
		<rdfs:comment>IBM OWL Editor</rdfs:comment>
	</owl:Ontology>
	<owl:Class rdf:about="&owl;Thing">
	</owl:Class>
	<owl:Class rdf:about="&ns_0;Network">
	</owl:Class>
	<owl:Class rdf:about="&ns_0;General">
		<rdfs:subClassOf rdf:resource="&ns_0;Network"/>
	</owl:Class>
	<owl:DatatypeProperty rdf:about="&ns_0;connection_time">
		<rdfs:domain rdf:resource="&ns_0;General"/>
		<rdfs:range rdf:resource="&xsd;string"/>
	</owl:DatatypeProperty>
	<owl:DatatypeProperty rdf:about="&ns_0;address">
		<rdfs:domain rdf:resource="&ns_0;General"/>
		<rdfs:range rdf:resource="&xsd;string"/>
	</owl:DatatypeProperty>
	<owl:Class rdf:about="&ns_0;QoS">
		<rdfs:subClassOf rdf:resource="&ns_0;Network"/>
	</owl:Class>
	<owl:DatatypeProperty rdf:about="&ns_0;RTT">
		<rdfs:domain rdf:resource="&ns_0;QoS"/>
		<rdfs:range rdf:resource="&xsd;string"/>
	</owl:DatatypeProperty>
	<owl:DatatypeProperty rdf:about="&ns_0;throughput">
		<rdfs:domain rdf:resource="&ns_0;QoS"/>
		<rdfs:range rdf:resource="&xsd;string"/>
	</owl:DatatypeProperty>
	<owl:DatatypeProperty rdf:about="&ns_0;frame_error_rate">
		<rdfs:domain rdf:resource="&ns_0;QoS"/>
		<rdfs:range rdf:resource="&xsd;string"/>
	</owl:DatatypeProperty>
	<owl:DatatypeProperty rdf:about="&ns_0;comm_setup_delay">
		<rdfs:domain rdf:resource="&ns_0;QoS"/>
		<rdfs:range rdf:resource="&xsd;string"/>
	</owl:DatatypeProperty>
	<owl:DatatypeProperty rdf:about="&ns_0;BER">
		<rdfs:domain rdf:resource="&ns_0;QoS"/>
		<rdfs:range rdf:resource="&xsd;string"/>
```

```
        </owl:DatatypeProperty>
        <owl:DatatypeProperty rdf:about="&ns_0;delay">
                <rdfs:domain rdf:resource="&ns_0;QoS"/>
                <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>
</rdf:RDF>
```

Perfil de conteúdo:

```
<?xml version="1.0" encoding="GB2312"?>
<!DOCTYPE rdf:RDF [
	<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
	<!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
	<!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
	<!ENTITY owl "http://www.w3.org/2002/07/owl#">
	<!ENTITY owl_editor_base "file:/D:/eclipse/workspace/Ontologias/content.owl/">
	<!ENTITY ns_0 "file:///">
]>
<rdf:RDF
	 xmlns:xsd="&xsd;"
	 xmlns:rdf="&rdf;"
	 xmlns:rdfs="&rdfs;"
	 xmlns:owl="&owl;"
	 xmlns:owl_editor_base="&owl_editor_base;"
	 xmlns:ns_0="&ns_0;"
>

	<owl:Ontology rdf:about="file:///OWLInputStream">
		<rdfs:comment>IBM OWL Editor</rdfs:comment>
	</owl:Ontology>
	<owl:Class rdf:about="&owl;Thing">
	</owl:Class>
	<owl:Class rdf:about="&ns_0;Content">
	</owl:Class>
	<owl:Class rdf:about="&ns_0;Browser_Accept">
		<rdfs:subClassOf rdf:resource="&ns_0;Content"/>
	</owl:Class>
	<owl:Class rdf:about="&ns_0;Content_Info">
		<rdfs:subClassOf rdf:resource="&ns_0;Content"/>
	</owl:Class>
	<owl:DatatypeProperty rdf:about="&ns_0;accept">
		<rdfs:domain rdf:resource="&ns_0;Browser_Accept"/>
		<rdfs:range rdf:resource="&xsd;string"/>
	</owl:DatatypeProperty>
	<owl:DatatypeProperty rdf:about="&ns_0;accept_charset">
		<rdfs:domain rdf:resource="&ns_0;Browser_Accept"/>
		<rdfs:range rdf:resource="&xsd;string"/>
	</owl:DatatypeProperty>
	<owl:DatatypeProperty rdf:about="&ns_0;accept_language">
		<rdfs:domain rdf:resource="&ns_0;Browser_Accept"/>
		<rdfs:range rdf:resource="&xsd;string"/>
	</owl:DatatypeProperty>
	<owl:Class rdf:about="&ns_0;Location">
		<rdfs:subClassOf rdf:resource="&ns_0;Content"/>
	</owl:Class>
	<owl:DatatypeProperty rdf:about="&ns_0;host">
		<rdfs:domain rdf:resource="&ns_0;Location"/>
		<rdfs:range rdf:resource="&xsd;string"/>
	</owl:DatatypeProperty>
	<owl:DatatypeProperty rdf:about="&ns_0;URL">
		<rdfs:domain rdf:resource="&ns_0;Location"/>
		<rdfs:range rdf:resource="&xsd;string"/>
	</owl:DatatypeProperty>
	<owl:Class rdf:about="&ns_0;Others">
		<rdfs:subClassOf rdf:resource="&ns_0;Content"/>
	</owl:Class>
	<owl:DatatypeProperty rdf:about="&ns_0;cookies">
```

```
                <rdfs:domain rdf:resource="&ns_0;Others"/>
                <rdfs:range rdf:resource="&xsd;boolean"/>
        </owl:DatatypeProperty>
        <owl:DatatypeProperty rdf:about="&ns_0;P3P">
                <rdfs:domain rdf:resource="&ns_0;Others"/>
                <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>
        <owl:DatatypeProperty rdf:about="&ns_0;cache_control">
                <rdfs:domain rdf:resource="&ns_0;Others"/>
                <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>
        <owl:DatatypeProperty rdf:about="&ns_0;content_type">
                <rdfs:domain rdf:resource="&ns_0;Content_Info"/>
                <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>
        <owl:DatatypeProperty rdf:about="&ns_0;content_lenght">
                <rdfs:domain rdf:resource="&ns_0;Content_Info"/>
                <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>
        <owl:DatatypeProperty rdf:about="&ns_0;content_language">
                <rdfs:domain rdf:resource="&ns_0;Content_Info"/>
                <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>
        <owl:DatatypeProperty rdf:about="&ns_0;content_encoding">
                <rdfs:domain rdf:resource="&ns_0;Content_Info"/>
                <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>
</rdf:RDF>
```

Perfil de SLA:

```
<?xml version="1.0" encoding="GB2312"?>
<!DOCTYPE rdf:RDF [
	<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
	<!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
	<!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
	<!ENTITY owl "http://www.w3.org/2002/07/owl#">
	<!ENTITY owl_editor_base "file:/D:/eclipse/workspace/Ontologias/SLA.owl/">
	<!ENTITY ns_5 "file:///">
]>
<rdf:RDF
	 xmlns:xsd="&xsd;"
	 xmlns:rdf="&rdf;"
	 xmlns:rdfs="&rdfs;"
	 xmlns:owl="&owl;"
	 xmlns:owl_editor_base="&owl_editor_base;"
	 xmlns:ns_5="&ns_5;"
>

	<owl:Ontology rdf:about="file:///OWLInputStream">
		<rdfs:comment>IBM OWL Editor</rdfs:comment>
	</owl:Ontology>
	<owl:Class rdf:about="&owl;Thing">
	</owl:Class>
	<owl:Class rdf:about="&ns_5;SLA">
	</owl:Class>
	<owl:Class rdf:about="&ns_5;Info">
		<rdfs:subClassOf rdf:resource="&ns_5;SLA"/>
	</owl:Class>
	<owl:DatatypeProperty rdf:about="&ns_5;SLA_ID">
		<rdfs:domain rdf:resource="&ns_5;Info"/>
		<rdfs:range rdf:resource="&xsd;string"/>
	</owl:DatatypeProperty>
	<owl:DatatypeProperty rdf:about="&ns_5;version">
		<rdfs:domain rdf:resource="&ns_5;Info"/>
		<rdfs:range rdf:resource="&xsd;string"/>
	</owl:DatatypeProperty>
	<owl:DatatypeProperty rdf:about="&ns_5;expire_date">
		<rdfs:domain rdf:resource="&ns_5;Info"/>
		<rdfs:range rdf:resource="&xsd;string"/>
	</owl:DatatypeProperty>
	<owl:Class rdf:about="&ns_5;Services">
		<rdfs:subClassOf rdf:resource="&ns_5;SLA"/>
	</owl:Class>
	<owl:Class rdf:about="&ns_5;Antivirus">
		<rdfs:subClassOf rdf:resource="&ns_5;Services"/>
	</owl:Class>
	<owl:Class rdf:about="&ns_5;Connection_Plan">
		<rdfs:subClassOf rdf:resource="&ns_5;SLA"/>
	</owl:Class>
	<owl:Class rdf:about="&ns_5;Image_Adapter">
		<rdfs:subClassOf rdf:resource="&ns_5;Services"/>
	</owl:Class>
	<owl:Class rdf:about="&ns_5;Classification_and_Filtering">
		<rdfs:subClassOf rdf:resource="&ns_5;Services"/>
	</owl:Class>
</rdf:RDF>
```

Perfil de Servidores de Adaptação:

```
<?xml version="1.0" encoding="GB2312"?>
<!DOCTYPE rdf:RDF [
	<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
	<!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
	<!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
	<!ENTITY owl "http://www.w3.org/2002/07/owl#">
	<!ENTITY owl_editor_base "file:/D:/eclipse/workspace/Ontologias/QoWS.owl/">
	<!ENTITY ns_0 "file:///">
]>
<rdf:RDF
	 xmlns:xsd="&xsd;"
	 xmlns:rdf="&rdf;"
	 xmlns:rdfs="&rdfs;"
	 xmlns:owl="&owl;"
	 xmlns:owl_editor_base="&owl_editor_base;"
	 xmlns:ns_0="&ns_0;"
>

	<owl:Ontology rdf:about="file:///OWLInputStream">
		<rdfs:comment>IBM OWL Editor Example</rdfs:comment>
	</owl:Ontology>
	<owl:Class rdf:about="&owl;Thing">
	</owl:Class>
	<owl:Class rdf:about="&ns_0;Adaptation_Servers">
	</owl:Class>
	<owl:Class rdf:about="&ns_0;QoS">
		<rdfs:subClassOf rdf:resource="&ns_0;Adaptation_Servers"/>
	</owl:Class>
	<owl:DatatypeProperty rdf:about="&ns_0;Reability">
		<rdfs:domain rdf:resource="&ns_0;QoS"/>
		<rdfs:range rdf:resource="&xsd;unsignedInt"/>
	</owl:DatatypeProperty>
	<owl:DatatypeProperty rdf:about="&ns_0;Availability">
		<rdfs:domain rdf:resource="&ns_0;QoS"/>
		<rdfs:range rdf:resource="&xsd;unsignedInt"/>
	</owl:DatatypeProperty>
	<owl:Class rdf:about="&ns_0;Services">
		<rdfs:subClassOf rdf:resource="&ns_0;Adaptation_Servers"/>
	</owl:Class>
	<owl:DatatypeProperty rdf:about="&ns_0;MaxProcessTime">
		<rdfs:domain rdf:resource="&ns_0;Services"/>
		<rdfs:range rdf:resource="&xsd;unsignedInt"/>
	</owl:DatatypeProperty>
	<owl:DatatypeProperty rdf:about="&ns_0;RequiredBandwidth">
		<rdfs:domain rdf:resource="&ns_0;Services"/>
		<rdfs:range rdf:resource="&xsd;unsignedInt"/>
	</owl:DatatypeProperty>
	<owl:DatatypeProperty rdf:about="&ns_0;Cost">
		<rdfs:domain rdf:resource="&ns_0;Services"/>
		<rdfs:range rdf:resource="&xsd;string"/>
	</owl:DatatypeProperty>
	<owl:ObjectProperty rdf:about="&ns_0;hasExecPoints">
		<rdfs:domain rdf:resource="&ns_0;Services"/>
		<rdfs:range rdf:resource="&ns_0;Exec_Points"/>
	</owl:ObjectProperty>
	<owl:Class rdf:about="&ns_0;Supported_Execution_Points">
		<rdfs:subClassOf rdf:resource="&ns_0;Services"/>
		<rdfs:subClassOf>
```

```
                        <owl:Restriction rdf:about="&ns_0;Supported_Execution_PointsRestriction">
                                <owl:onProperty rdf:resource="&ns_0;hasExecPoints"/>
                                <owl:allValuesFrom rdf:resource="&ns_0;Exec_Points"/>
                        </owl:Restriction>
                </rdfs:subClassOf>
        </owl:Class>
        <owl:Class rdf:about="&ns_0;Exec_Points">
                <owl:oneOf rdf:parseType="Collection">
                        <ns_0:Exec_Points rdf:about="&ns_0;exec_point_1">
                        </ns_0:Exec_Points>
                        <ns_0:Exec_Points rdf:about="&ns_0;exec_point_2">
                        </ns_0:Exec_Points>
                        <ns_0:Exec_Points rdf:about="&ns_0;exec_point_3">
                        </ns_0:Exec_Points>
                        <ns_0:Exec_Points rdf:about="&ns_0;exec_point_4">
                        </ns_0:Exec_Points>
                </owl:oneOf>
        </owl:Class>
        <ns_0:Exec_Points rdf:about="&ns_0;exec_point_1">
        </ns_0:Exec_Points>
        <ns_0:Exec_Points rdf:about="&ns_0;exec_point_2">
        </ns_0:Exec_Points>
        <ns_0:Exec_Points rdf:about="&ns_0;exec_point_3">
        </ns_0:Exec_Points>
        <ns_0:Exec_Points rdf:about="&ns_0;exec_point_4">
        </ns_0:Exec_Points>
        <owl:Class rdf:about="&ns_0;Protocols">
                <owl:oneOf rdf:parseType="Collection">
                        <ns_0:Protocols rdf:about="&ns_0;icap">
                        </ns_0:Protocols>
                        <ns_0:Protocols rdf:about="&ns_0;soap">
                        </ns_0:Protocols>
                </owl:oneOf>
        </owl:Class>
        <ns_0:Protocols rdf:about="&ns_0;icap">
        </ns_0:Protocols>
        <ns_0:Protocols rdf:about="&ns_0;soap">
        </ns_0:Protocols>
        <owl:ObjectProperty rdf:about="&ns_0;hasProtocols">
                <rdfs:domain rdf:resource="&ns_0;Adaptation_Servers"/>
                <rdfs:range rdf:resource="&ns_0;Protocols"/>
        </owl:ObjectProperty>
        <owl:Class rdf:about="&ns_0;Supported_Protocols">
                <rdfs:subClassOf>
                        <owl:Restriction rdf:about="&ns_0;Supported_ProtocolsRestriction">
                                <owl:onProperty rdf:resource="&ns_0;hasProtocols"/>
                                <owl:allValuesFrom rdf:resource="&ns_0;Protocols"/>
                        </owl:Restriction>
                </rdfs:subClassOf>
                <rdfs:subClassOf rdf:resource="&ns_0;Adaptation_Servers"/>
        </owl:Class>
</rdf:RDF>
```

Perfil de serviço de Classificação e Filtragem de Conteúdo

```
<?xml version="1.0"?>
<!DOCTYPE uridef [
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY owl "http://www.w3.org/2002/07/owl">
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema">
  <!ENTITY service "http://www.daml.org/services/owl-s/1.1/Service.owl">
  <!ENTITY profile "http://www.daml.org/services/owl-s/1.1/Profile.owl">
  <!ENTITY process "http://www.daml.org/services/owl-s/1.1/Process.owl">
  <!ENTITY grounding "http://www.daml.org/services/owl-s/1.1/Grounding.owl">
  <!ENTITY expr "http://www.daml.org/services/owl-s/1.1/generic/Expression.owl">
  <!ENTITY swrl "http://www.w3.org/2003/11/swrl">
  <!ENTITY user "http://www.owl-ontologies.com/user.owl">
  <!ENTITY this "http://www.example.org/FilterDomain.owl">
]>
<rdf:RDF
  xmlns:rdf="&rdf;#"
  xmlns:rdfs="&rdfs;#"
  xmlns:owl="&owl;#"
  xmlns:xsd="&xsd;#"
  xmlns:service="&service;#"
  xmlns:profile="&profile;#"
  xmlns:process="&process;#"
  xmlns:grounding="&grounding;#"
  xmlns:expr="&expr;#"
  xmlns:swrl="&swrl;#"
  xmlns:user="&user;#"
  xml:base="&this;"
  xmlns="&this;#"
>
<owl:Ontology rdf:about="">
            <owl:imports rdf:resource="&service;"/>
            <owl:imports rdf:resource="&profile;"/>
            <owl:imports rdf:resource="&process;"/>
            <owl:imports rdf:resource="&grounding;"/>
</owl:Ontology>

<owl:Class rdf:ID="FilterServices"/>

<rdf:Description rdf:about='http://www.example.org/FilterDomain.owl#No_Service'>
    <rdf:type rdf:resource='http://www.example.org/FilterDomain.owl#FilterServices'/>
  </rdf:Description>
  <rdf:Description rdf:about='http://www.example.org/FilterDomain.owl#Full_Filtering'>
    <rdf:type rdf:resource='http://www.example.org/FilterDomain.owl#FilterServices'/>
  </rdf:Description>
  <rdf:Description rdf:about='http://www.example.org/FilterDomain.owl#Domain_Only'>
    <rdf:type rdf:resource='http://www.example.org/FilterDomain.owl#FilterServices'/>
  </rdf:Description>
  <rdf:Description rdf:about='http://www.example.org/FilterDomain.owl#Content_Only'>
    <rdf:type rdf:resource='http://www.example.org/FilterDomain.owl#FilterServices'/>
  </rdf:Description>

 <owl:Class rdf:ID="SupportedFilterSrv">
   <rdfs:comment>Precondition for execute filtering and classification services</rdfs:comment>
   <owl:oneOf rdf:parseType="Collection">
            <FilterServices rdf:about="&this;#Full_Filtering"/>
            <FilterServices  rdf:about="&this;#Domain_Only"/>
            <FilterServices  rdf:about="&this;#Content_Only"/>
   </owl:oneOf>
</owl:Class>

            <!-- Service description -->
  <service:Service rdf:ID="FilterDomainService">
    <service:supports>
      <grounding:WsdlGrounding rdf:ID="FilterDomainGrounding"/>
```

```
  </service:supports>
  <service:describedBy>
   <process:AtomicProcess rdf:ID="FilterDomainProcess"/>
  </service:describedBy>
  <service:presents>
   <profile:Profile rdf:ID="FilterDomainProfile"/>
  </service:presents>
 </service:Service>

          <!-- Profile description -->
 <profile:Profile rdf:about="#FilterDomainProfile">
  <profile:hasInput>
   <process:Input rdf:ID="domain">
    <rdfs:label>domain</rdfs:label>
    <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
    >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
   </process:Input>
  </profile:hasInput>
  <profile:hasOutput>
   <process:Output rdf:ID="FilterDomainReturn">
    <rdfs:label>FilterDomainReturn</rdfs:label>
    <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
    >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
   </process:Output>
  </profile:hasOutput>
  <profile:hasInput>
   <process:Input rdf:ID="URLDB">
    <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">http://www.owl-
ontologies.com/user.owl#URLDB</process:parameterType>
    <rdfs:label>URLDB</rdfs:label>
   </process:Input>
  </profile:hasInput>
  <service:presentedBy rdf:resource="#FilterDomainService"/>
  <profile:textDescription>Auto generated from
http://adaptationsrv:8080/axis/ContentFiltering.jws?wsdl</profile:textDescription>
  <profile:hasInput>
   <process:Input rdf:ID="filter_profile">
    <rdfs:label>filter_profile</rdfs:label>
    <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">http://www.owl-
ontologies.com/user.owl#filter_profile</process:parameterType>
   </process:Input>
         </profile:hasInput>
           <profile:hasInput>
   <process:Input rdf:ID="ServiceLevel">
    <rdfs:label>ServiceLevel</rdfs:label>
    <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
    >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
   </process:Input>
  </profile:hasInput>

          <profile:hasPrecondition rdf:resource="#hasClassification_and_Filtering"/>
  <profile:serviceName>FilterDomain</profile:serviceName>
 </profile:Profile>

 <expr:SWRL-Condition rdf:ID="hasClassification_and_Filtering">
                  <expr:expressionBody rdf:parseType="Literal">
       <swrl:AtomList>
        <rdf:first>
         <swrl:ClassAtom>
          <swrl:classPredicate rdf:resource="#SupportedFilterSrv" />
          <swrl:argument1 rdf:resource="#ServiceLevel" />
         </swrl:ClassAtom>
        </rdf:first>
         <rdf:rest rdf:resource="&rdf;#nil" />
       </swrl:AtomList>
     </expr:expressionBody>
  </expr:SWRL-Condition>
```

```
<!-- Process description -->
<process:AtomicProcess rdf:about="#FilterDomainProcess">
  <process:hasInput rdf:resource="#URLDB"/>
  <process:hasInput rdf:resource="#filter_profile"/>
  <process:hasOutput rdf:resource="#FilterDomainReturn"/>
        <process:hasPrecondition rdf:resource="#hasClassification_and_Filtering"/>
        <rdfs:label>FilterDomainProcess</rdfs:label>
  <process:hasInput rdf:resource="#domain"/>
  <process:hasInput rdf:resource="#ServiceLevel"/>
        <service:describes rdf:resource="#FilterDomainService"/>
</process:AtomicProcess>

<!-- Grounding description -->
<grounding:WsdlGrounding rdf:about="#FilterDomainGrounding">
  <service:supportedBy rdf:resource="#FilterDomainService"/>
  <grounding:hasAtomicProcessGrounding>
    <grounding:WsdlAtomicProcessGrounding rdf:ID="FilterDomainAtomicProcessGrounding"/>
  </grounding:hasAtomicProcessGrounding>
</grounding:WsdlGrounding>
<grounding:WsdlAtomicProcessGrounding rdf:about="#FilterDomainAtomicProcessGrounding">
  <grounding:wsdlInputMessage rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
  >http://adaptationsrv:8080/axis/ContentFiltering.jws#FilterDomainRequest</grounding:wsdlInputMessage>
  <grounding:wsdlOutputMessage rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
  >http://adaptationsrv:8080/axis/ContentFiltering.jws#FilterDomainResponse</grounding:wsdlOutputMessage>
  <grounding:wsdlDocument rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
  >http://adaptationsrv:8080/axis/ContentFiltering.jws?wsdl</grounding:wsdlDocument>
  <grounding:wsdlInput>
    <grounding:WsdlInputMessageMap>
      <grounding:owlsParameter rdf:resource="#filter_profile"/>
      <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://adaptationsrv:8080/axis/ContentFiltering.jws?wsdl#filter_profile</grounding:wsdlMessagePart>
    </grounding:WsdlInputMessageMap>
  </grounding:wsdlInput>
  <grounding:owlsProcess rdf:resource="#FilterDomainProcess"/>
  <grounding:wsdlOutput>
    <grounding:WsdlOutputMessageMap>
      <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://adaptationsrv:8080/axis/ContentFiltering.jws?wsdl#FilterDomainReturn</grounding:wsdlMessagePart>
      <grounding:owlsParameter rdf:resource="#FilterDomainReturn"/>
    </grounding:WsdlOutputMessageMap>
  </grounding:wsdlOutput>
  <grounding:wsdlInput>
    <grounding:WsdlInputMessageMap>
      <grounding:owlsParameter rdf:resource="#URLDB"/>
      <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://adaptationsrv:8080/axis/ContentFiltering.jws?wsdl#URLDB</grounding:wsdlMessagePart>
    </grounding:WsdlInputMessageMap>
  </grounding:wsdlInput>
  <grounding:wsdlOperation>
    <grounding:WsdlOperationRef>
      <grounding:operation rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://adaptationsrv:8080/axis/ContentFiltering.jws?wsdl#FilterDomain</grounding:operation>
      <grounding:portType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://adaptationsrv:8080/axis/ContentFiltering.jws?wsdl#ContentFiltering</grounding:portType>
    </grounding:WsdlOperationRef>
  </grounding:wsdlOperation>
  <grounding:wsdlInput>
    <grounding:WsdlInputMessageMap>
      <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://adaptationsrv:8080/axis/ContentFiltering.jws?wsdl#domain</grounding:wsdlMessagePart>
      <grounding:owlsParameter rdf:resource="#domain"/>
    </grounding:WsdlInputMessageMap>
  </grounding:wsdlInput>
</grounding:WsdlAtomicProcessGrounding>
<owl:Class rdf:about="http://www.w3.org/2001/XMLSchema#int"/>
</rdf:RDF>
```